DϵpS: Delayed ϵ-Shrinking for Faster Once-For-All Training

Aditya Annavajjala[*1], Alind Khare[*1], Animesh Agrawal[1], Igor Fedorov[3], Hugo Latapie[2], Myungjin Lee[2], and Alexey Tumanov[1]

[1] Georgia Institute of Technology, Atlanta, USA
[2] Cisco Research, USA
[3] Meta, USA

Abstract. CNNs are increasingly deployed across different hardware, dynamic environments, and low-power embedded devices. This has led to the design and training of CNN architectures with the goal of maximizing accuracy subject to such variable deployment constraints. As the number of deployment scenarios grows, there is a need to find scalable solutions to design and train specialized CNNs. Once-for-all training has emerged as a scalable approach that jointly co-trains many models (subnets) at once with a constant training cost and finds specialized CNNs later. The scalability is achieved by training the full model and simultaneously reducing it to smaller subnets that share model weights (weight-shared shrinking). However, existing once-for-all training approaches incur huge training costs reaching 1200 GPU hours. We argue this is because they either start the process of shrinking the full model *too* early or *too* late. Hence, we propose Delayed ϵ-Shrinking (DϵpS) that starts the process of shrinking the full model when it is *partially* trained ($\sim 50\%$), which leads to training cost improvement and better in-place knowledge distillation to smaller models. The proposed approach also consists of novel heuristics that dynamically adjust subnet learning rates incrementally (ϵ), leading to improved weight-shared knowledge distillation from larger to smaller subnets as well. As a result, DϵpS outperforms state-of-the-art once-for-all training techniques across different datasets including CIFAR10/100, ImageNet-100, and ImageNet-1k on accuracy and cost. It achieves 1.83% higher ImageNet-1k top1 accuracy or the same accuracy with 1.3x reduction in FLOPs and 2.5x drop in training cost (GPU*hrs). Code is released at https://github.com/gatech-sysml/deps.

Keywords: Scalable vision · NAS · Efficient Inference

1 Introduction

CNNs are pervasive in numerous applications including smart cameras [2], smart surveillance [6], self-driving cars [26], search engines [12], and social media [1]. As a result, they are increasingly deployed across diverse hardware ranging from server-grade GPUs like V100 [19] to edge-GPUs like Nvidia Jetson [18] and

---

[*] Authors contributed equally to this research.

dynamic environments like Autonomous Vehicles [8] that operate under strict latency or power budget constraints. As the diversity in deployment scenarios grows, efficient deployment of CNNs under a myriad of deployment constraints becomes challenging. It calls for developing techniques that find appropriate CNNs suited for different deployment conditions.

Neural Architecture Search (NAS) [4, 32] has emerged as a successful technique that finds CNN architectures specialized for a deployment target. It searches for an appropriate CNN architecture and trains it with the goal of maximizing accuracy subject to the target deployment constraints. However, state-of-the-art NAS techniques remain prohibitively expensive, requiring many GPU hours due to the costly operation of the combined search and training of specialized CNNs. The problem is exacerbated when NAS is employed to satisfy multiple deployment targets, as it must run repeatedly for *each* deployment target. This makes the cost of NAS linear in the number of deployment targets considered ($O(k)$), which is prohibitively expensive and doesn't scale with the growing number of deployment targets. Therefore, there is a need to develop *scalable* NAS solutions able to satisfy multiple deployment targets efficiently.

One such technique is Once-for-all training [3, 29, 40]—a step towards making NAS computationally feasible to satisfy multiple deployment targets by decoupling training from search. It achieves this decoupling by co-training a family of models (weight-shared subnets with varied shapes and sizes) embedded inside a supernet *once*, incurring a constant training cost. After the supernet is trained, NAS can then be performed for any specific deployment target by simply extracting a specialized subnet from the supernet without retraining (*once-for-all*).

This achieves $O(1)$ training cost w.r.t. the number of deployment targets and, therefore, makes NAS scalable. However, the efficiency of this once-for-all training remains limited as it incurs a significant training cost (~ 1200 GPU hours in [3]). This is primarily due to (a) the large number of training epochs required to overcome training interference (OFA [3] in Fig. 1), and (b) the high average time per-epoch caused by shrinking—defined as sampling and adding smaller subnets to the training schedule— per minibatch (BigNAS [40] in Fig. 1). Thus, in order to make once-for-all training more efficient, we must reduce its training time without sacrificing state-of-the-art accuracy across the whole operating latency/FLOP range of the supernet.
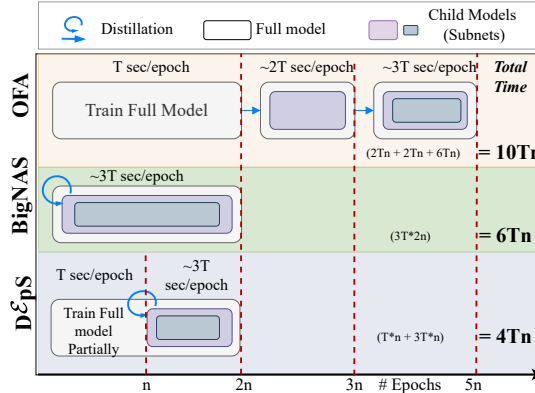

Fig. 1: DϵpS reduces training time compared to existing approaches like OFA [3] & BigNAS [40].

We propose DϵpS, a technique that increases the scalability of once-for-all training. It consists of three key components designed to meet their respective goals — Full Model warmup (FM-Warmup) provides better supernet initialization, ϵ-Shrinking keeps the accuracy of the full model (largest subnet that contains all the supernet parameters) on par with OFA and BigNAS, and IKD-Warmup boosts the accuracy of small subnets with effective knowledge distillation in once-for-all training. Particularly, with better supernet initialization, FM-Warmup (DϵpS in Fig. 1) reduces both the total number of epochs (compared to OFA) and average time per-epoch (compared to BigNAS). In FM-Warmup, the supernet is initialized with the partially trained full model ($\sim$50%) and then subnet sampling (shrinking) is started to train the model family. The partial full model training ensures a lower time per epoch initially. Then, ϵ-Shrinking ensures smooth optimization of the full model. It incrementally warms up the learning rate of subnets using parameter ϵ when the shrinking starts, while keeping the learning rate of the full model higher. Lastly, IKD-Warmup enables knowledge distillation from multiple partially trained full models (that are progressively better) to smaller subnets. The three components, when combined, reduce the training time of once-for-all training and outperform state-of-the-art w.r.t. accuracy of subnets across different datasets and neural network architectures. We summarize the contributions of our work as follows:

- FM-Warmup provides better initialization to the weight shared supernet by training the full model only partially and delaying model shrinking. This leads to reduced time per epoch and lower training cost.
- ϵ-Shrinking ensures smooth and fast optimization of the full model by warming up the learning rate of smaller subnets. This enables it to reach optimal accuracy quickly.
- IKD-Warmup provides rich knowledge transfer to subnets, enabling them to quickly learn good representations.

We extensively evaluate DϵpS against existing once-for-all training baselines [3, 29, 40] on CIFAR10/100 [21], ImageNet-100 [34], and ImageNet-1k [7] datasets. DϵpS outperforms all baselines across all the datasets both w.r.t. accuracy (of subnets) and training cost. It achieves 1.83% ImageNet-1k top1 accuracy improvement or the same accuracy with 1.3x FLOPs reduction while reducing training cost by upto 1.8x w.r.t. OFA and 2.5x w.r.t. BigNAS (in dollars or GPU hours). We also provide a detailed ablation study to demonstrate the benefits of DϵpS components in isolation.

2 Background

Formulation. Let W_o denote the supernet's weights, the objective of once-for-all training is given by —

$$\min_{W_o} \sum_{a \in \mathcal{A}} \mathcal{L}(S(W_o, a)) \tag{1}$$

where $S(W_o, a)$ denotes weights of subnet a selected from the supernet's weight W_o and $\mathcal{A}$ represents the set of all possible neural architectures (subnets). The goal of once-for-all training is to find optimal supernet weights that minimize the loss ($\mathcal{L}$) of all the neural architectures in $\mathcal{A}$ on a given dataset.

Challenges. However, optimizing (1) is non-trivial. On one hand, enumerating gradients of all subnets to optimize the overall objective is computationally infeasible. This is due to the large number of subnets optimized in once-for-all training ($|\mathcal{A}| \approx 22B$ subnets in our case[*]). On the other hand, a naive approximation of objective (1) to make it computationally feasible leads to *interference* (sampling a few subnets in each update step). *Interference* occurs when smaller subnets affect the performance of the larger subnets [3, 40]. Hence, interference causes sub-optimal accuracy of the larger subnets. Existing once-for-all training techniques mitigate interference by increasing the training time significantly (Fig. 1). For instance, OFA [3] mitigates interference by first training the full model (largest subnet) and then progressively increasing the size of $|\mathcal{A}|$. This leads to a large number of training epochs and ≈ 1200 GPU hours to perform once-for-all training. Therefore, the following challenges remain in once-for-all training — **(C1)** training supernet at a lesser training cost than SOTA, *and* **(C2)** mitigating interference. We divide challenge **C2** into two sub-challenges — matching existing once-for-all training techniques [3, 40] w.r.t. accuracy of **(C2a)** the full model (largest subnet), and **(C2b)** child models (smaller subnets).

3 Related Work

Efficient NN-Architectures in Deep Learning. Efficient deep neural networks (NNs) achieve high accuracy at low FLOPs. These neural nets are easy to deploy as they increase hardware efficiency by operating at low FLOPs. Developing such networks is an active research area. Several efficient neural networks include MobileNets [15], SqueezeNets [17], EfficientNets [33], and TinyNets [10].

Neural network compression. Neural network compression reduces the size and computation of neural networks for efficient deployment. The compression occurs after the network is trained. Hence, the performance of compression methods is bounded by the accuracy of the trained neural network. Neural network compression can be broadly divided into two categories — network pruning and quantization. Network pruning removes unimportant units [11, 25, 30] or channels [22, 23, 31]. Network quantization converts the representation of neural weights and activations to low bits [16, 20, 37].

Hardware aware NAS. Neural architecture search (NAS) automates the design of efficient NN architectures. NAS typically involves searching for and training NN architectures that are more accurate than manually designed NNs [27, 42]. Recently, NAS methods are becoming hardware-aware [4, 32, 38] *i.e.* they find NN architectures suited for deployment at target hardware. These methods incorporate deployment constraints of hardware or latency in their search. Then,

---

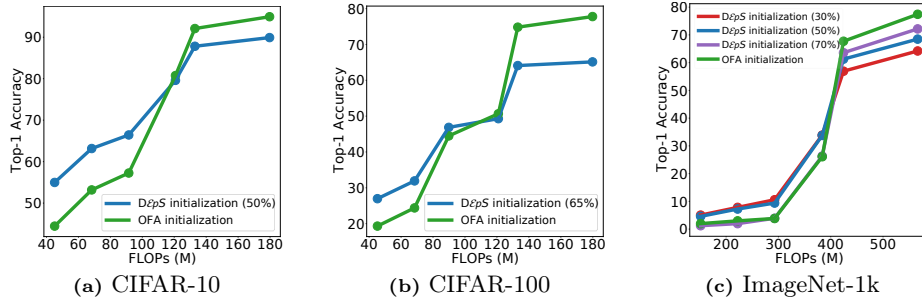[*] 5 stages, 3 depths (2, 3, 4), 3 expands $= (3^2 + 3^3 + 3^4)^5 \approx 22B$ subnets

Fig. 2: Supernetwork initialization. DεpS provides better initialization for the supernetwork for smaller subnets compared to OFA due to FM-Warmup. This validates the hypothesis that the supernet weights become specialized if the full model is trained to completion (OFA), resulting in poorer accuracy of subnetworks with increased training of the full model.

they find and train efficient NNs that meet the constraints. However, these NAS methods only satisfy a single deployment target. They need to run repeatedly for each deployment target that doesn't scale well.

Once-For-All Training. Once-for-all training is a scalable NAS method that satisfies multiple deployment targets. It co-trains models (subnets) that vary in shape and size embedded inside a single weight-shared supernet. NAS is performed later by extracting specialized subnets from the trained supernet for target hardware. OFA [3], BigNAS [40], and CompOFA [29] exemplify once-for-all training methods. OFA proposes Progressive Shrinking (PS) for once-for-all training that trains the full model first and then progressively introduces smaller subnets into training by dividing the training procedure into multiple training jobs (phases). Compared to OFA, DεpS performs once-for-all training as a single training job and starts shrinking from a partially trained full model to reduce the training cost. BigNAS starts the process of shrinking early and samples multiple subnets at every minibatch. In contrast, DεpS initially only trains the full model and delays the shrinking. Finally, CompOFA changes the architecture search space of OFA and performs Progressive Shrinking with reduced phases. DεpS algorithmically changes the shrinking procedure in once-for-all training and is complementary to architecture space changes proposed in CompOFA.

4   Proposed Approach

We present DεpS, a once-for-all training technique that trains supernets in less training time. DεpS consists of three key components that meet the challenges **C1** and **C2**. We describe each component in detail and highlight the core contributions of our work.

4.1   Full-Model Warmup Period (P_{warmup}^{fm}): When to Shrink the Full Model?

Shrinking the full model at an appropriate time is vital for reducing training cost (meet **C1**). Both early or late shrinking isn't sufficient to meet the challenges in once-for-all training. Early shrinking (BigNAS [40] in Fig. 1) doesn't meet the challenge **C1**. It increases the overall training time as multiple subnets are sampled in each update (increasing per-epoch time) to optimize objective (1). Early shrinking also requires a lot of hyper-parameter tuning to meet challenge **C2**. It becomes sensitive to training hyper-parameters due to interference. For instance, training the full model with early shrinking becomes unstable with the standard initialization of the full model [40].

On the other hand, if shrinking happens late after the full model is completely trained (OFA [3] in Fig. 1), the supernet weights become too specialized for the full model architecture and require a large number of training epochs to reduce interference. Hence, late shrinking meets challenge **C2** but not **C1**.
We argue that shrinking should occur after the full model is partially trained (warmed up, trained at least 50%, proposed approach in Fig. 1).

Delayed Shrinking has numerous advantages. It reduces the overall training time to meet challenge **C1**. The initial updates in DεpS are cheap compared to early shrinking as only the full model gets trained and no subnets are sampled. Moreover, since supernet weights are not specialized for the full model, DεpS can meet challenge **C2** in less number of epochs. To validate our hypothesis, we ask whether a partially trained full model serves as a good initialization for the supernet. To do this, we compare the accuracy of small subnets (shrinking) on multiple datasets (CIFAR-10, CIFAR-100, ImageNet-1k) in a mobilenet-based supernet [3] when initialized with a partially trained (50%), and completely trained full model ($\sim$600 MFLOPs) in Fig. 2.
The takeaway from the experiment in Fig. 2 is that a partially-trained full model-based initialization performs better for smaller subnets than the initialization with the full model completely trained. This validates our hypothesis that supernet weights become too specialized if the full model is trained to completion. Hence, warming up the full model helps in meeting challenge **C1**. DεpS introduces a hyperparameter P_{warmup}^{fm} denoting the fraction of total epochs used to warm up the full model. P_{warmup}^{fm} is usually kept $\geq 50\%$ in DεpS.

4.2 ϵ-Shrinking: Learning Rates for Subnets

In addition to the full model warmup, we propose ϵ-Shrinking that enables the full model to reach comparable accuracy with SOTA and meet challenge **C2a**. ϵ-Shrinking ensures that the full model's accuracy doesn't get affected when shrinking is introduced in between its training. When the shrinking starts, the learning rate of subnets is gradually ramped to reach the full model's learning rate (ϵ-Shrinking) as the full model gets sampled with other subnets in each update step.

Without the gradual warmup, the full model becomes prone to an accuracy drop as the supernet weights change rapidly at the start of shrinking. To understand this change, we compare the updates in the supernet with and without shrinking for a minibatch $\mathcal{B}$. Consider supernet weights W_t at iteration t. Without shrinking, the update is given by -

$$W_{t+1}^{noShrink} = W_t - \eta_t \underbrace{\nabla l_{\mathcal{B}}(S(W_t, a_{full}))}_{=G_{noShrink}^{\mathcal{B},t}} \tag{2}$$

where $l_{\mathcal{B}}(S(W_t, a_{full})$ denotes the loss of the full model on minibatch $\mathcal{B}$ and equals $\frac{1}{|\mathcal{B}|} \sum_{x \in \mathcal{B}} l(x, S(W_t, a_{full}))$; x denotes the samples in $\mathcal{B}$. η_t denotes the learning rate at iteration t used to update the weights. Whereas introducing shrinking for the same supernet weights W_t yields the following update -

$$W_{t+1}^{Shrink} = W_t - \eta_t \underbrace{\left(\overbrace{\sum_{a \in \mathcal{U}_k(\mathcal{A})} \nabla l_{\mathcal{B}}\left(S(W_t, a)\right)}^{\text{shrinking}} \right)}_{=G_{Shrink}^{\mathcal{B},t}} \tag{3}$$

where $\mathcal{U}_k(A)$ denotes uniformly sampling k subnets from the architecture space $\mathcal{A}$. This update step is the approximation of the objective (1). Clearly, the updates differ, it is *improbable* that $W_{t+1}^{Shrink} = W_{t+1}^{NoShrink}$. This difference in updates causes the supernet weights to change rapidly when shrinking is introduced. The rapid change in supernet weights causes degradation in the full model's accuracy. To avoid rapid changes in weights, a widely adopted technique is to use less aggressive learning rates via learning rate warmup schedules [9,13].

However, applying such principles in the context of weight-sharing is nontrivial but at the same time important. Our key idea is two-fold to a) always sample the full model with other subnets while shrinking, and b) use less aggressive learning rates for subnets at the start of shrinking. Particularly, it is important to ensure $G_{noShrink}^{\mathcal{B},t} \approx G_{Shrink}^{\mathcal{B},t}$ to make $W_{t+1}^{Shrink} \approx W_{t+1}^{NoShrink}$ initially when the shrinking starts. To do this, we introduce a parameter ϵ that controls the effective learning rate of subnets and makes $G_{noShrink}^{\mathcal{B},t} \approx G_{Shrink}^{\mathcal{B},t}$. The gradient in ϵ-Shrinking is given as follows -

$$G_{Shrink}^{\mathcal{B},t}(\epsilon_t) = G_{noShrink}^{\mathcal{B},t} + \overbrace{\epsilon_t * \sum_{a \in \mathcal{U}_{k-1}(\mathcal{A} \setminus \{a_{full}\})} \nabla l_{\mathcal{B}}\left(S(W_t, a)\right)}^{\epsilon - \text{shrinking}} \tag{4}$$

where $\epsilon_t \in (0, 1]$. Note that the effective learning rate becomes $\eta_t * \epsilon_t$ for subnets and remains η_t for the full model in ϵ-Shrinking. Hence, slowly increasing ϵ_t warms up the effective learning of subnets. We start with a small value of ϵ_t ($=10^{-4}$) and increment it by a constant amount to reach 1. Once ϵ_t reaches 1, it stays constant for the rest of the training. We empirically verify if $G_{noShrink}^{\mathcal{B},t}, G_{Shrink}^{\mathcal{B},t}$ differ in magnitude (l_2-norm) and direction (cosine similarity) and whether ϵ-Shrinking is able to reduce the differences with $G_{noShrink}^{\mathcal{B},t}(\epsilon_t)$.

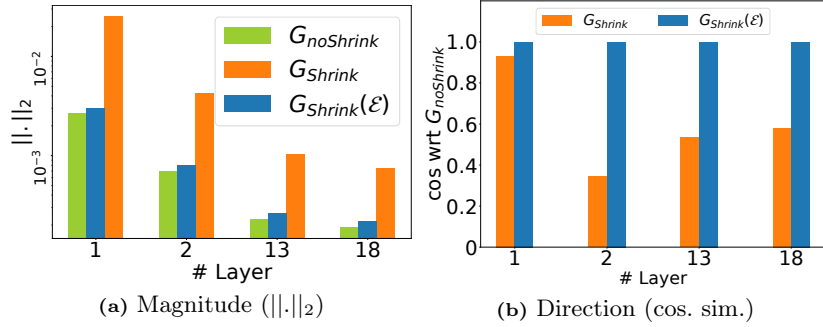(a) Magnitude ($||.||_2$)

(b) Direction (cos. sim.)

Fig. 3: Gradients w/ & w/o Shrinking on Mobilenet-Based Supernet. Delayed Shrinking causes gradients (G_{Shrink}) to differ from the full model gradient ($G_{noShrink}$) leading to rapid changes in the supernet's weights. ϵ-Shrinking's gradient ($G_{Shrink}(\mathcal{E})$) reduces such differences and avoids rapid weight changes.

Fig. 3 compares the magnitude and direction of the gradients of the full model ($G_{noShrink}$), shrinking (G_{Shrink}) and ϵ-Shrinking ($G_{noShrink}(\epsilon)$) ($\epsilon = 0.001$) on the weights of a mobilenet-based supernet [3] for the ImageNet-1k dataset [28]. $G_{noShrink}$ and G_{Shrink} differ both in magnitude and direction across supernet layers.

The magnitude of G_{Shrink} is an order of magnitude higher than $G_{noShrink}$ for early layers. ϵ-Shrinking maintains the low magnitude of gradient throughout the training as shown in Fig. 4. The magnitude of G_{Shrink} is consistently higher than $G_{Shrink}(\epsilon_t)$ when normalized with the magnitude of $G_{noShrink}$. Such differences cause poor convergence at the start of shrinking and often lead to accuracy drops. Whereas, $G_{noShrink}(\epsilon)$ has minimal differences w.r.t. $G_{noShrink}$ enabling healthy convergence and no potential accuracy drops.
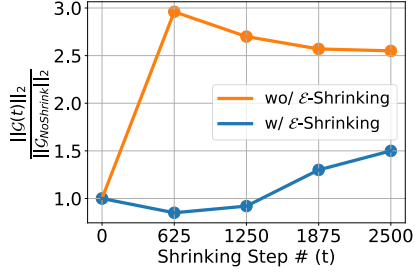

Fig. 4: Gradient Magnitude Over Time. Gradient magnitude with ($\mathcal{G}_{shrink}(\mathcal{E}, t)$) and without ($\mathcal{G}_{shrink}(t)$) $\mathcal{E}$-shrinking is compared w.r.t the initial full model gradient ($\mathcal{G}_{Noshrink}$) over shrinking steps. $\mathcal{E}$-shrinking avoids sudden changes in the supernet parameters by lowering the gradient magnitude.

4.3 IKD-Warmup: In-Place Knowledge Distillation (KD) from Warmed-up Full Model

We now discuss IKD-Warmup that distills knowledge from the full model to subnets and meets challenge **C2b**. Effectively distilling the knowledge from the full model becomes non-trivial due to weight-sharing. On one hand, KD requires the supernet weights biased to the full model to offer meaningful knowledge transfer to subnets. On the other hand, having a large bias in the supernet

weights toward the full model may result in subnets' sub-optimal performance since the weights are shared. To tackle this trade-off, OFA [3] biases the supernet weights to a trained full model and then uses it to perform vanilla-KD [14]. However, this results in a long training time during shrinking as the supernet weights are trained to fit subnets' architectures. Another approach like BigNAS [40] doesn't bias the shared weights to the full model by using inplace-KD [39] but lacks in providing rich knowledge transfer to subnets (initially).

This is because inplace-KD distills the knowledge "on the fly" to other subnets as the full model gets trained from randomly initialized weights. Precisely, the full model predictions become ground truth for other subnets. Hence, when the full model is under-trained initially, it doesn't offer rich knowledge transfer.

We believe that the proposed delayed shrinking has an added advantage w.r.t. KD for once-for-all training — the partially trained full model (50/60% trained) is rich enough to provide meaningful knowledge transfer to the subnets and doesn't bias the supernet weights to the full model. It has been shown that for vanilla-KD [14], partially trained (intermediate) models provide a comparable or at times better knowledge transfer than the completely trained models [5,36]. This is because they provide more information about non-target classes than the trained models [5]. We use this insight in DϵpS that performs inplace-KD from a partially trained full model (IKD-Warmup).

IKD-Warmup offers two advantages, it — a) distills knowledge from multiple progressively better partially trained models as the full model gets trained (unlike a single partially/fully trained model used in vanilla-KD [36]), and b) provides rich knowledge transfer to the subnets at all times (unlike inplace-KD [39] that distills from an under-trained full model initially).

5   Experiments

We establish that DϵpS **a**) reduces training cost w.r.t. SOTA in once-for-all training [3,29,40], **b**) performs at-par or better than SOTA's accuracy across subnets (covering the entire range of architectural space), **c**) generalizes across datasets, **d**) generalizes to different deep neural network (DNN) architecture spaces, and **e**) produces specialized subnets for target hardware without retraining (once-for-all property). We also aim to demonstrate attribution of benefits in DϵpS by providing detailed ablation on **a**) a full model warmup period: empirically demonstrating a sweet spot, **b**) ϵ-Shrinking: showing healthy convergence, and **c**) IKD-Warmup: distilling knowledge better than existing distillation approaches in weight-sharing.

5.1   Setup

Baselines. We first compare DϵpS with the other NAS methods or efficient DNNs [4,15,33,35] w.r.t. accuracy. Then, we compare DϵpS with once-for-all training techniques — OFA [3], BigNAS [40], CompOFA [29] w.r.t. both training cost and accuracy of subnets spanned across supernet's FLOP range. Training

Group	Approach	MACs (M)	Top-1 Test Acc (%)
0-100 (M)	OFA [3]	67	70.5
	DϵpS	**67**	**72.3**
100-200 (M)	OFA [3]	141	71.6
	DϵpS	**141**	**73.7**
200-300 (M)	FBNetv2 [35]	238	76.0
	BigNAS [40]	242	76.5
	OFA [3]	230	76
	DϵpS	**230**	**77.3**
300-400 (M)	MNasnet [32]	315	75.2
	ProxylessNAS [4]	320	74.6
	FBNetv2 [35]	325	77.2
	MobileNetV3 [15]	356	76.6
	EfficientNetB0 [33]	390	77.3

Table 1: Comparison of DϵpS with state of the art neural architecture search approaches on ImageNet-1k. DϵpS consistently outperforms the baselines.

time of all the techniques is measured on NVIDIA A40 GPUs. Accuracy of subnets is evaluated without additional finetuning. As once-for-all training trains multiple subnets, the comparison is done by uniformly dividing the entire FLOP range into 6 buckets and picking the most accurate subnet from each bucket for every baseline. All methods are evaluated on the same architecture space.

Success Metrics. DϵpS is compared against the baselines on the following success metrics — a) Training cost measured in GPU hours or dollars (lower is better), b) *Pareto-frontier*: Accuracy of best-performing subnets as a function of FLOPs/latency. To compare Pareto-frontiers obtained from different baselines, we use a metric called *mean pareto accuracy* that is defined as the area under the curve (AUC) of accuracy and normalized FLOPs/latency. The higher the mean pareto accuracy the better.

Datasets. We evaluate all methods on CIFAR10/100 [21], ImageNet-100 [34] and ImageNet-1k [7] datasets. The complexity of datasets progressively increases from CIFAR10 to ImageNet-1k. The datasets vary in the number of classes, image resolution, and number of train/test samples.

DNN Architecture Space. All methods are trained on the supernets derived from two different DNN architecture spaces — MobilenetV3 [15] and Proxyless-NAS [4] (same as OFA [3]). The base architecture of ProxylessNAS is derived from ProxylessNAS run for the GPU as a target device. To avoid confounding, we evaluate all baselines on the same DNN architecture space.

Training Hyper-parameters. The training hyper-parameters of DϵpS are similar to the hyper-parameters of the full model training. The hyper-parameters for MobilenetV3, and ProxylessNAS training are borrowed from [15] and [4] respectively. Specifically, we use SGD with Nesterov momentum 0.9, a CosineAnnealing LR [24] schedule, and weight decay $3e^{-5}$. Unless specified, the shrinking is introduced in DϵpS after the full model gets $\approx 50\%$ trained ($P_{\text{warmup}}^{fm} \approx 50\%$).

Approach	Smallest Subnet		Largest Subnet		mean pareto acc.	Training Cost			
	Acc(%)	MACs (M)	Acc (%)	MACs (M)		# Epochs	Avg. GPU min. epoch	Total Time /(GPU hours.)	Dollar Cost ($)
OFA [3]	71.8	150	77.2	230	75.77	605	125	1256	2675
CompOFA [29]	🔲	150	🔲	230	- 🔲	330	142	782	1665
BigNAS [40]	70.6	150	74	230	72.51	400	266	1778	3787
DϵpS	**73.8**	150	**77.3**	230	**75.81**	**270**	155	**700**	**1491**

Table 2: Comparison of DϵpS vs SOTA on ImageNet-1k. Accuracy and Training Cost comparison of DϵpS against SOTA approaches are shown for MobilenetV3-based architecture space. DϵpS outperforms SOTA and achieves 2% better accuracy for the smallest subnet and is at-par with the largest subnet (full model) respectively at 1.8x training cost reduction (in $) compared to OFA. Dollar-cost is calculated based on the on-demand prices for A40 GPUs from exoscale.com

5.2   Evaluation

Comparison with NAS methods/Efficient Nets on ImageNet-1k. We compare DϵpS with MobilenetV3 [15], FBNet [35], ProxylessNAS [4], BigNAS [40] and efficient nets [33] on the ImageNet-1k dataset.

Takeaway. Tab. 1 compares accuracy vs MACs of the baselines. DϵpS consistently surpasses the baselines over multiple MAC ranges. Especially in the lower MAC region (0-100M), DϵpS is **1.8%** more accurate. In the larger MAC region (200-300M), DϵpS achieves 77.3% accuracy with upto 1.69x MACs improvement compared to the baselines (efficientNet-B0). DϵpS benefits from supernet initialization and effective knowledge distillation to get superior performance.

Comparison with Once-for-all training methods on ImageNet-1k We now demonstrate the accuracy and training cost benefits of DϵpS on ImageNet-1k dataset [28]. Tab. 2 compares DϵpS with the baselines [*] on a) the upper-bound (largest subnet) and lower-bound (smallest subnet) top1 accuracy, and b) GPU hours and dollar costs.

Takeaway. DϵpS is atleast **2**% more accurate at 150 MACs (smallest subnet) than baselines and at-par w.r.t. accuracy at 230 MACs (largest subnet). DϵpS matches the Pareto-optimality of baselines (with highest mean pareto accuracy) at a reduced training cost (least among all the baselines). It takes 1.8x and 2.5x less dollar cost (or GPU hours) than OFA and BigNAS respectively.

The training cost improvement of DϵpS comes due to FM-Warmup. FM-Warmup allows DϵpS to train subnets in less number of total epochs (lowest among the baselines) and a lower average time per epoch than BigNAS (Tab. 2). The full model's accuracy (largest subnet in Tab. 2) is improved as ϵ-Shrinking enables its smooth convergence. Finally, DϵpS improves accuracy at lower FLOPs (150 MACs) as IKD-Warmup distills knowledge effectively in once-for-all training.

Generalization across datasets. We establish that the accuracy improvements of DϵpS generalize to other vision datasets.

---

[*] Since CompOFA reports matching OFA accuracy, we achieve comparison to both by reporting Pareto frontier results for OFA.

(a) CIFAR-10   **(b)** CIFAR-100   **(c)** ImageNet-100   **(d)** ImageNet-1k
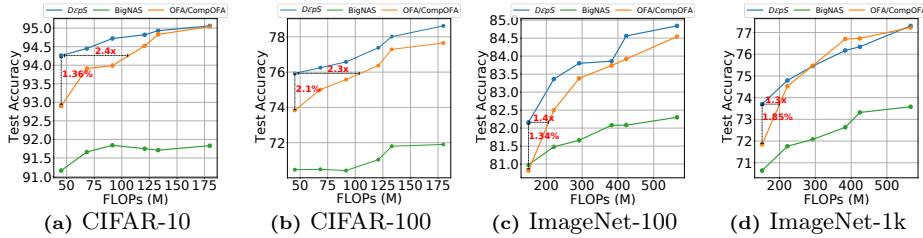
Fig. 5: DεpS's Accuracy Improvement across Datasets. The comparison of DεpS with the baselines is shown w.r.t. accuracy (of subnets) for CIFAR10/100, ImageNet-100, and ImageNet-1k datasets. DεpS consistently outperforms the baselines across all the datasets and achieves upto 2.1% better accuracy for the same FLOPs or upto 2.3x FLOP reduction at same accuracy.

Training Details. DεpS uses the standard hyper-parameters of the MobileNetV3 for all the datasets using SGD with cosine learning rate decay and nestrov momentum, and shrinking is introduced when the full model is 50% trained. For OFA, we first train the largest network independently. Shrinking occurs after the full model is completely trained and vanilla KD is used for distillation. The depth and expand phases are run for 100 epochs each. The initial learning rate of different phases is set as per OFA [3]. BigNAS uses RMSProp optimizer with its proposed hyper-parameters for ImageNet-1k. However, we use SGD optimizer in BigNAS for CIFAR10/100 and ImageNet-100 datasets as we empirically find that SGD performs better than RMSProp on these datasets. Fig. 5 compares the Pareto-frontiers of top1 test accuracy and FLOPs obtained from each baseline across various datasets. The subnets are present in six different FLOP buckets that uniformly divide the supernet's FLOP range. The comparison includes the performance of the smallest and largest subnets to measure the lower-bound and upper-bound test accuracy reached by the baselines.

Takeaway. DεpS outperforms baselines w.r.t. accuracy of smaller subnets ($\leq$ 300 MFLOPs) on all the datasets. It achieves slightly better or at-par accuracy for larger subnets ($\geq$ 300 MFLOPs) than OFA/CompOFA. DεpS outperforms BigNAS and achieves a better Pareto-Frontier across all the datasets.

Generalization across DNN-Architecture Spaces. We demonstrate that DεpS generalizes to other DNN-architecture spaces. We train DεpS on ImageNet-1k dataset using ProxylessNAS-based supernet (DNN-architecture space) with training-hyperparameters borrowed from [4]. Fig. 6 compares Pareto-frontiers obtained from DεpS and OFA on ImageNet-1k dataset. We reiterate that we don't conduct an additional exhaustive hyperparameter search and instead use training-hyperparameters
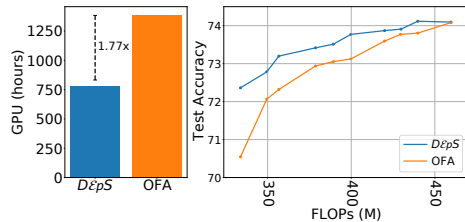

Fig. 6: DεpS on ProxyLessNAS architecture space: superior Pareto-Frontier with a 1.8% improvement in ImageNet-1k test accuracy on the smallest subnet.
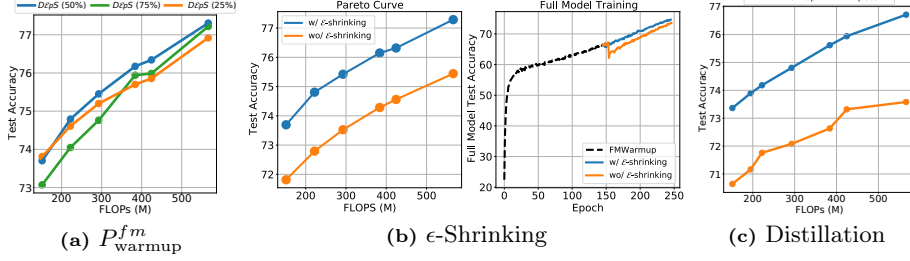
from [4] and a $P_{\text{warmup}}^{fm} \approx 50\%$.

Fig. 7: DϵpS Ablations. Three ablations are shown for DϵpS— Full model warmup period (P^{fm}_{warmup}), ϵ-Shrinking, and Distillation. a) There exists a sweet spot w.r.t. accuracy (of subnets) in P^{fm}_{warmup} (=50%), b) ϵ-Shrinking improves the entire pareto front (left) and prevents drop in accuracy of the full model (right), c) IKD-Warmup performs better than Inplace KD as it uses more information from non-target classes (further details are provided in supplementary material).

Takeaway. DϵpS outperforms OFA w.r.t. ImageNet-1k test accuracy (with 0.5% better mean pareto accuracy). It improves the accuracy of the smallest subnet by **1.8%**. The accuracy improvements come with **1.8x** training cost reduction compared to OFA.

5.3 Ablation Study

We provide detailed ablation on DϵpS components — FM-Warmup, ϵ-Shrinking, and IKD-Warmup to attribute their benefits.

Full Model Warmup Period (P^{fm}_{warmup}). In this ablation, we establish the benefits of delayed shrinking as opposed to early or late shrinking. To do this, we configure DϵpS to run with different full model warmup periods (P^{fm}_{warmup}) – the time at which shrinking starts in DϵpS. Our goal is to empirically demonstrate the existence of a sweet spot in P^{fm}_{warmup} w.r.t. accuracy (of subnets). Fig. 7a compares the accuracy of best-performing subnets in six different FLOP buckets of three P^{fm}_{warmup} periods {25%, 50%, 75%} on ImageNet-1k dataset. P^{fm}_{warmup} =25%, 75% represents early and late shrinking respectively.

Takeaway. DϵpS with $P^{fm}_{\text{warmup}} = 50\%$ achieves the best test accuracy across subnets compared to DϵpS configured to run with $P^{fm}_{\text{warmup}} = 25\%, 75\%$. Hence, a sweet spot exists in P^{fm}_{warmup}. The existence of a sweet spot demonstrates that both early (25%) or late (75%) shrinking is sub-optimal in training the model family (discussed in §4.1). Early shrinking results in sub-optimal accuracy of the larger subnets as training interference occurs very early in the training. While late shrinking causes the specialization of supernet weights to the full model architecture that results in sub-optimal accuracy of smaller subnets ($\approx$ 1% accuracy degradation around 200 MFLOPs for P^{fm}_{warmup} =75% compared to P^{fm}_{warmup} =50%).

$\mathcal{E}$**-Shrinking.** We investigate whether an accuracy drop occurs in the full model's accuracy when shrinking is introduced in DϵpS and if $\mathcal{E}$-Shrinking prevents it. In this ablation, we run DϵpS with and without $\mathcal{E}$-Shrinking and introduce shrinking at 150th epoch while keeping all other training-hyperparameters constant.

Fig. 7b (right) compares DϵpS with and without $\mathcal{E}$-Shrinking on ImageNet-1k top1 test accuracy of the full model over training epochs. Fig. 7b (left) compares subnets for six different FLOP buckets with and without ϵ-Shrinking.

Takeaway. DϵpS without $\mathcal{E}$-Shrinking observes a **2%** drop in full model's accuracy at 150th epoch when the shrinking starts. And, DϵpS with $\mathcal{E}$-Shrinking prevents this huge accuracy drop at the start of shrinking that leads to better full model accuracy overall. The prevention of the drop in full model's accuracy demonstrates that $\mathcal{E}$-Shrinking leads to smooth optimization of the full model. $\mathcal{E}$-Shrinking achieves this by incrementally warming up subnets' learning rate at the start of shrinking to avoid sudden changes in the supernet weight (Fig. 7b, right). $\mathcal{E}$-Shrinking also achieves superior accuracy across the entire FLOP range when compared to the supernet trained without $\mathcal{E}$-Shrinking (Fig. 7b, left).

IKD-Warmup. We assess the benefits of IKD-Warmup in this ablation. IKD-Warmup performs inplace knowledge distillation from a partially trained full model instead of a randomly initialized full model (inplace KD) as proposed in [41]. Hence, to show benefits of IKD-Warmup, we run DϵpS with inplace KD and our proposed IKD-Warmup. Fig. 7c compares DϵpS run with IKD-Warmup (blue) and inplace KD (orange) on the ImageNet-1k top1 test accuracy of best-performing subnets in seven different FLOP buckets.

Takeaway. IKD-Warmup outperforms inplace KD across all the subnets that cover the supernet's FLOP range on the ImageNet-1k dataset. It is **3.5%** and **2%** more accurate at 560 MFLOPs and 150 MFLOPs respectively. This shows that IKD-Warmup distills knowledge effectively in once-for-all training as multiple progressively better partially trained full model transfer their knowledge to smaller subnets (§4.3). Inplace KD is not able to provide meaningful knowledge transfer as the full model is under-trained initially.

6   Conclusion

DϵpS is a training technique that increases the scalability of once-for-all training. DϵpS consists of three key components — FM-Warmup that decreases training costs, ϵ-Shrinking that maintains full model accuracy on par with existing works, and IKD-Warmup that enables effective knowledge distillation in once-for-all training. FM-Warmup's key insight is to delay the process of shrinking (introducing smaller weight-shared subnets) until the full model is partially trained ($\sim$50%) to reduce training cost. ϵ-Shrinking averts accuracy degradation in the full model by avoiding rapid changes in the supernet weights and, instead, enables smooth optimization by progressively warming up subnets' learning rates. IKD-Warmup provides rich knowledge transfer to subnets from multiple partially trained full models that are progressively better w.r.t. accuracy. DϵpS generalizes to different datasets and DNN architecture spaces. It improves the accuracy of smaller subnets, achieves on-par Pareto-optimality, and reduces training cost by upto 2.5x when compared with existing once-for-all weight-shared training techniques.

Acknowledgments

References

1. Bai, S., Kolter, J.Z., Koltun, V.: An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. CoRR **abs/1803.01271** (2018), http://arxiv.org/abs/1803.01271
2. Bonnard, J., Abdelouahab, K., Pelcat, M., Berry, F.: On building a cnn-based multi-view smart camera for real-time object detection. Microprocessors and Microsystems **77**, 103177 (2020)
3. Cai, H., Gan, C., Wang, T., Zhang, Z., Han, S.: Once-for-all: Train one network and specialize it for efficient deployment. In: International Conference on Learning Representations (2020), https://openreview.net/forum?id=HylxE1HKwS
4. Cai, H., Zhu, L., Han, S.: Proxylessnas: Direct neural architecture search on target task and hardware. arXiv preprint arXiv:1812.00332 (2018)
5. Cho, J.H., Hariharan, B.: On the efficacy of knowledge distillation. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 4794–4802 (2019)
6. Cob-Parro, A.C., Losada-Gutiérrez, C., Marrón-Romera, M., Gardel-Vicente, A., Bravo-Muñoz, I.: Smart video surveillance system based on edge computing. Sensors **21**(9), 2958 (2021)
7. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)
8. Gog, I., Kalra, S., Schafhalter, P., Wright, M.A., Gonzalez, J.E., Stoica, I.: Pylot: A modular platform for exploring latency-accuracy tradeoffs in autonomous vehicles. In: 2021 IEEE International Conference on Robotics and Automation (ICRA). pp. 8806–8813. IEEE (2021)
9. Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., He, K.: Accurate, large minibatch sgd: Training imagenet in 1 hour. arXiv preprint arXiv:1706.02677 (2017)
10. Han, K., Wang, Y., Zhang, Q., Zhang, W., Xu, C., Zhang, T.: Model rubik's cube: Twisting resolution, depth and width for tinynets. Advances in Neural Information Processing Systems **33**, 19353–19364 (2020)
11. Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149 (2015)
12. Hashemi, H.B., Asiaee, A., Kraft, R.: Query intent detection using convolutional neural networks. In: International conference on web search and data mining, workshop on query understanding (2016)

13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
14. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
15. Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al.: Searching for mobilenetv3. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 1314–1324 (2019)
16. Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., Bengio, Y.: Binarized neural networks. Advances in neural information processing systems **29** (2016)
17. Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., Keutzer, K.: Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. arXiv preprint arXiv:1602.07360 (2016)
18. Inc., N.: Nvidia jetson. https://www.nvidia.com/en-in/autonomous-machines/embedded-systems/, [Accessed 13-May-2023]
19. Inc, N.: Nvidia v100. https://www.nvidia.com/en-in/data-center/v100/, [Accessed 13-May-2023]
20. Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., Kalenichenko, D.: Quantization and training of neural networks for efficient integer-arithmetic-only inference. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2704–2713 (2018)
21. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
22. Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P.: Pruning filters for efficient convnets. arXiv preprint arXiv:1608.08710 (2016)
23. Lin, M., Ji, R., Wang, Y., Zhang, Y., Zhang, B., Tian, Y., Shao, L.: Hrank: Filter pruning using high-rank feature map. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 1529–1538 (2020)
24. Loshchilov, I., Hutter, F.: Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983 (2016)
25. Luo, J.H., Wu, J., Lin, W.: Thinet: A filter level pruning method for deep neural network compression. In: Proceedings of the IEEE international conference on computer vision. pp. 5058–5066 (2017)
26. Ouyang, Z., Niu, J., Liu, Y., Guizani, M.: Deep cnn-based real-time traffic light detector for self-driving vehicles. IEEE transactions on Mobile Computing **19**(2), 300–313 (2019)
27. Real, E., Aggarwal, A., Huang, Y., Le, Q.V.: Regularized evolution for image classifier architecture search. In: Proceedings of the aaai conference on artificial intelligence. vol. 33, pp. 4780–4789 (2019)
28. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) **115**(3), 211–252 (2015). https://doi.org/10.1007/s11263-015-0816-y
29. Sahni, M., Varshini, S., Khare, A., Tumanov, A.: Comp{ofa} – compound once-for-all networks for faster multi-platform deployment. In: International Conference on Learning Representations (2021), https://openreview.net/forum?id=IgIk8RRT-Z
30. Sanh, V., Wolf, T., Rush, A.: Movement pruning: Adaptive sparsity by fine-tuning. Advances in Neural Information Processing Systems **33**, 20378–20389 (2020)

31. Sun, W., Zhou, A., Stuijk, S., Wijnhoven, R., Nelson, A.O., Corporaal, H., et al.: Dominosearch: Find layer-wise fine-grained n: M sparse schemes from dense neural networks. Advances in neural information processing systems **34**, 20721–20732 (2021)
32. Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., Le, Q.V.: Mnasnet: Platform-aware neural architecture search for mobile. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 2820–2828 (2019)
33. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: International conference on machine learning. pp. 6105–6114. PMLR (2019)
34. Tian, Y., Krishnan, D., Isola, P.: Contrastive multiview coding. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16. pp. 776–794. Springer (2020)
35. Wan, A., Dai, X., Zhang, P., He, Z., Tian, Y., Xie, S., Wu, B., Yu, M., Xu, T., Chen, K., et al.: Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 12965–12974 (2020)
36. Wang, C., Yang, Q., Huang, R., Song, S., Huang, G.: Efficient knowledge distillation from model checkpoints. In: Oh, A.H., Agarwal, A., Belgrave, D., Cho, K. (eds.) Advances in Neural Information Processing Systems (2022), `https://openreview.net/forum?id=OltDq6SjrfW`
37. Wang, L., Dong, X., Wang, Y., Liu, L., An, W., Guo, Y.: Learnable lookup table for neural network quantization. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 12423–12433 (2022)
38. Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., Keutzer, K.: Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10734–10742 (2019)
39. Yu, J., Huang, T.S.: Universally slimmable networks and improved training techniques. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 1803–1811 (2019)
40. Yu, J., Jin, P., Liu, H., Bender, G., Kindermans, P.J., Tan, M., Huang, T., Song, X., Pang, R., Le, Q.: Bignas: Scaling up neural architecture search with big single-stage models. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M. (eds.) Computer Vision – ECCV 2020. pp. 702–717. Springer International Publishing, Cham (2020)
41. Yu, J., Yang, L., Xu, N., Yang, J., Huang, T.: Slimmable neural networks. arXiv preprint arXiv:1812.08928 (2018)
42. Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V.: Learning transferable architectures for scalable image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 8697–8710 (2018)