Layer-Wisely Supervised Learning For One-Shot Neural Architecture Search

Yifei Chen*†, Zhourui Guo*†, Qiyue Yin*†, Hao Chen*†, Kaiqi Huang*†
*School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China
†CRISE, Institute of Automation, Chinese Academy of Sciences, Beijing, China
chenyifei2019@ia.ac.cn,guozhourui2021@ia.ac.cn,qyyin@nlpr.ia.ac.cn,
chenhao2019@ia.ac.cn,kqhuang@nlpr.ia.ac.cn

2022 International Joint Conference on Neural Networks (IJCNN) | 978-1-7281-8671-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/IJCNN55064.2022.9892632

***Abstract*—Neural architecture search aims to automatically discover both efficient and effective neural architectures. Recently, one-shot neural architecture search (one-shot NAS) has drawn great attention due to its high efficiency and competitive performance. One of the most important problems in one-shot NAS is to evaluate the capabilities of architecture candidates. In particular, a pre-trained super-net is served as an evaluator. Due to the large weight-sharing space, current one-shot methods suffer from the ranking disorder issue, that is, the ranking correlation between estimated capabilities and true capabilities of candidates is incorrect. Moreover, the super-net in search is dense thus it is inefficient to train with end-to-end back-propagation. In this paper, we propose to modularize the large weight-sharing space of one-shot NAS into layers by introducing layer-wisely supervised learning. But we discover that greedy layer-wise learning that learns each layer separately with a local objective hurts super-net performance as well as ranking correlation. Instead, we learn each layer by using the gradients propagated from the objective associated with the adjacent upper layer. The simple proposal reduces the representation shift and improves the ranking correlation. In addition, it reduces 47.4% memory footprint and gets a faster convergence of super-net training compared with the strong baseline. Extensive experiments on ImageNet with both supervised and self-supervised objectives demonstrate the effectiveness of our proposal.**

***Index Terms*—Deep Learning, Neural Architecture Search, Layer-Wise Pretraining.**

I. Introduction

Deep Neural Networks (DNNs) [1]–[3] have made significant progress in various domains. However, designing an effective neural architecture for a specific task often requires substantial effort from human experts. To automatically seek effective architectures, Neural Architecture Search (NAS) [4]–[24] has been proposed. Recently, one-shot neural architecture search (one-shot NAS) [14]–[17] has drawn great attention due to its high efficiency and competitive performance. Rather than training thousands of separate networks from scratch, one-shot NAS only trains a single large super-net and takes the pre-trained super-net as an evaluator for final searching. Particularly, the large super-net takes each candidate in the predefined search space as a subnet and trainable weights are shared across architecture candidates in the pre-defined search space. Such a weight-sharing strategy can greatly reduce the search cost from thousands of GPU days to a few.

Despite the remarkable progress, the effectiveness of one-shot NAS is questioned. Current one-shot methods suffer from the ranking disorder issue, that is, the ranking correlation between estimated capabilities and true capabilities of candidates is incorrect. Several recent works [18]–[21] claim that the large weigh-sharing space is the main culprit of inaccurate architecture ranking in one-shot NAS. Works [19] and [20] attempt to reduce the size by modularizing the large weight-sharing space into **blocks** with block-wise knowledge distillation. These methods improved ranking correlation and search efficiency. But the recent work [21] argues that their results are highly correlated with the reference model. Specifically, the candidates that are more similar to the reference model tend to get higher ranks in those methods. Therefore, BossNAS [21] proposed using self-supervised learning to provide block-wise supervision. However, some of the most widely used self-supervised methods require human-designed proxy tasks that must be closely correlated to the target task in some meaningful way, as does the self-supervised NAS.

Moreover, the super-nets are typically optimized by the end-to-end back-propagation. Although empirically proven to be highly successful, the end-to-end back-propagation is considered inefficient. For example, it creates a substantial memory overhead due to the need to store all intermediate feature maps during each forward propagation before the backward propagation. In addition, it hinders the model parallelism for faster parallel training, since the update of lower layers needs to "wait" for error signals back-propagated from upper layers. The aforementioned two issues are much worse in one-shot NAS, because the over-parameterized super-net incorporates much more modules than regular neural networks.

In this paper, to address the aforementioned two problems simultaneously, we propose to modularize the large weight-sharing space of one-shot NAS into **layers** by introducing layer-wisely supervised learning without introducing any reference model or proxy task. To this end, greedy layer-wise learning that learns each layer separately with a local objective is the most direct and simplest approach. But we discover that the naive greedy layer-wise learning hurts the super-net training performance as well as ranking correlation. In particular, it increases representation shift, i.e. different choice modules in the same layer learn very different feature maps, which results in these choice modules not being interchangeable during the training. As shown in recent works [17] and [34], large representation shift will lead to low ranking correlation.

978-1-7281-8671-9/22/$31.00 ©2022 IEEE
Authorized licensed use limited to: Hebei University of Technology. Downloaded on February 02,2023 at 14:34:57 UTC from IEEE Xplore. Restrictions apply.

The recent work [30] hypothesizes that, in greedy layer-wise learning, lower layers are unaware of the existence of upper layers due to gradient isolation, so the full capacity of deep neural networks is not available. Inspired from this insight, we instead learn each layer by using gradients back-propagated from the local objective associated with the adjacent upper layer. As a result, upper layers can recursively send feedbacks to lower layers. Furthermore, to provide every module an equal opportunity to affect modules upon it during the training, we set the input of each layer to be the average of outputs of all choice modules in the adjacent lower layer.

The proposed simple scheme ensures that each choice module can be fairly and sufficiently trained, even the search space is very large. Following recent works [15] and [17], the single-path uniform sampling strategy is chosen as the end-to-end baseline. Extensive experiments are conducted on ImageNet with both supervised and self-supervised objectives. In both cases, the simple scheme reduces representation shift and improves ranking correlation compared with the strong end-to-end baseline. In addition, it reduces 47.4% memory footprint of the super-net training because there is no need to store intermediate feature maps. Moreover, the super-net training can be further accelerated in an asynchronous parallel way. Our contributions are mainly three-fold:

- We propose to modularize the large weight-sharing space of one-shot NAS into **layers** by layer-wisely supervised learning for the first time to improve the effectiveness and efficiency of the current one-shot NAS.
- The proposed non-greedy layer-wise super-net training reduces the representation shift, improves the ranking correlation, reduces the memory footprint, and enables a faster asynchronous super-net training.
- The extensive experiments demonstrate that our method achieves clear improvements over the strong end-to-end baseline. For instance, by only using 9 GPU days, we discover a model that achieves 78.0% top-1 accuracy on ImageNet with 397M FLOPs.

II. Related Work

Neural Architecture Search: The early works on NAS utilize reinforcement learning [4]–[6] or evolutionary algorithms [7]–[9] and the discovered architectures surpass the manually designed counterparts on a variety of tasks. Although these methods [4]–[9] achieve remarkable performances on a variety of tasks, they require thousands of GPU days for each search. Recently, the weight-sharing methods [10]–[24] have shown promising results while reducing search cost to a few GPU days. Generally, the weight-sharing methods all involve training a dense super-net. These method can be roughly classified into two categories: (1) one-stage methods [10]–[13] who couple training and searching within one stage; (2) one-shot methods or two-stage methods [14]–[18] who decouple training and searching into two stages. As shown in [22]–[24], the optimization process in one-stage methods is unstable and easy to fall into bad local minimums. As shown in [14] and [16], one-shot methods are empirically more stable. Nevertheless, as shown in [18]–[21], one-shot methods suffer from the ranking disorder issue due to the large weight-sharing space. In this paper, we first propose to modularize the large search space of one-shot NAS into layers.

Block-Wisely One-Shot NAS: DNA [19] first proposed a scheme of block-wisely super-net training by block-wise knowledge distillation, where a pre-trained teacher model is used to provide block-wise supervision. DONNA [20] optimizes an accuracy predictor by block-wise knowledge distillation, then uses the pre-trained predictor to carrier out rapid NAS for new deployment scenarios. However, distillation-based methods may introduce non-negligible bias from the reference model. To avoid introducing bias from the reference model, BossNAS [21] instead used block-wisely self-supervised contrastive learning to provide block-wise supervision. However, proxy tasks used in self-supervised methods must be carefully designed by human experts to guarantee that they are closely related to the target task. In this paper, we aim to improve the efficiency and effectiveness of one-shot NAS without any teacher model or proxy task.

Layer-wise Learning of DNNs: Greedy layer-wisely unsupervised learning of deep generative models [25] was shown to be effective as initialization for deep supervised models. Previous works [26]–[29] have used layer-wisely supervised learning to train DNNs without fine-tuning and achieve similar accuracy to end-to-end counterparts. The recent work [31] argues that locally supervised learning collapses task-relevant information at early layers and proposed an information propagation loss to preserve useful information. Recently, LoCo [30] argues that greedy layer-wisely contrastive learning cannot deliver the full capacity of DNNs and proposed to recursively send feedbacks from the top to the bottom.

III. Preliminary on One-Shot NAS

In one-shot NAS, the search space $\mathcal{A}$ is represented as an over-parameterized super-net, denotes as $\mathcal{N}(\mathcal{A}, \mathcal{W})$, which covers all architecture candidates $\alpha \in \mathcal{A}$, where $\mathcal{W}$ is the trainable weight of the super-net and is shared across all of architecture candidates. The search of optimal architecture α^* in one-shot NAS is formulated as a two-stage optimization problem. The first-stage optimization problem is given by:

$$\min_{\mathcal{W}} \mathcal{L}_{train}(\mathcal{N}(\mathcal{A}, \mathcal{W})) \tag{1}$$

where $\mathcal{L}_{train}$ denotes loss function on training dataset. To reduce the memory footprint, one-shot methods typically train a super-net by dropping out modules in the super-net with certain probability [16] or uniform sampling among candidates [17] before performing the search. Following recent works [15] and [17], we adopt the single-path uniform sampling strategy as the baseline that only sample one random path from the super-net for each training batch. The second-stage optimization problem is to search neural architectures over the pre-trained super-net and is given by:

$$\min_{\alpha \in \mathcal{A}} Acc_{val}(\mathcal{N}(\alpha, w_\alpha)) \tag{2}$$

Authorized licensed use limited to: Hebei University of Technology. Downloaded on February 02,2023 at 14:34:57 UTC from IEEE Xplore. Restrictions apply.

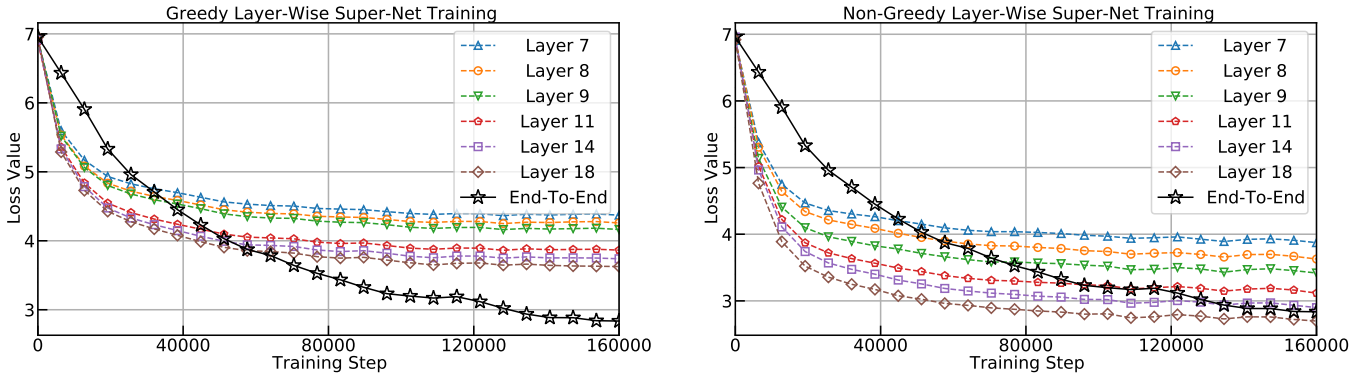

Fig. 1. Following recent works [15] and [17], we adopt the single-path uniform sampling strategy as the end-to-end baseline. **Left:** Although getting a faster convergence at the early phase of super-net training, greedy layer-wise super-net training hurts the super-net training performance compared with the end-to-end baseline. **Right:** By learning each layer with gradients back-propagated from the target associated with the adjacent upper layer, we bridge the training performance gap between greedy layer-wise super-net training and end-to-end baseline.

where $Acc_{val}(\mathcal{N}(\alpha, w_\alpha))$ denotes the accuracy of architecture α on validation dataset. Note the architecture α directly inherits its weights w_α from $\mathcal{W}$. Since the search space usually contains massive architecture candidates (e.g., 3.69×10^{16} in [19]), previous methods resort to heuristics search algorithms to solve Eq. 2, such as random search [14], evolution algorithms [17] or reinforcement learning [33].

IV. Methodology

In this section, the details of our proposed methods are elaborated. The greedy layer-wise super-net training is first introduced and we show that it hurts the super-net training performance as well as ranking correlation. Then, we introduce non-greedy layer-wise super-net training that learns each layer using gradients back-propagated from the local target associated with the adjacent upper layer.

A. Greedy Layer-Wise Super-Net Training

To provide layer-wise supervision without introducing any teacher model or proxy task, greedy layer-wise learning is the simplest approach. Motivated by recent works [14]–[21], we consider an ordinary super-net that consists of n layers and each layer has m choice modules. Let $f_{i,j}(\cdot; w_{i,j})$ denotes the mapping induced by the j-th module at the i-th layer with trainable weight $w_{i,j} \in \mathcal{W}$. For simplicity, the average of outputs of all modules in the i-th layer is used as the input to the $(i+1)$-th layer during the forward propagation:

$$x_{i+1} = \frac{1}{m} \sum_{j=1}^{m} f_{i,j}(x_i; w_{i,j}) \tag{3}$$

where $x_i \in X_i$ denotes the input of the i-th layer. To train the super-net greedily and layer-by-layer, each layer is assigned a local decoder and a local loss. The decoder used in this paper consists of a global average pooling layer and a fully connected layer. During the backward propagation, gradients are not allowed to propagate backward between layers. Instead, each module and each decoder are trained greedily using the associated local loss. Let $g_{i,j}(\cdot; \theta_i)$ denotes the mapping induced by the i-th decoder with the trainable weight θ_i, $\mathcal{L}_i$ denotes the i-th local loss and $y \in Y$ denotes the label. The optimization objective for modules in the i-th layer and the i-th decoder is recursively given by:

$$\min_{\{w_{i,\cdot}\},\theta_i} \sum_{j=1}^{m} \mathcal{L}_i(g_i(f_{i,j}(X_i; w_{i,j}); \theta_i), Y) \tag{4}$$

The recent work [17] proposed to meet the strict fairness during the super-net training. Particularly, strict fairness forces

Algorithm 1 Greedy Layer-Wise Super-Net Training

Input: Training data, the number of iterations T, super-net with n layers (each layer containing m choice modules), n local decoders, n local loss functions.
Output: pre-trained super-net

for iteration $t = 1, \cdots, T$ **do**
  Randomly sample the input data x_1 and label y.
  Clear gradients by $\nabla w_{\cdot,\cdot} \leftarrow 0$ and $\nabla \theta_\cdot \leftarrow 0$.
  for iteration $i = 1, \cdots, n$ **do**
    if $i \leq (n-1)$ **then**
      Initialize x_{i+1} by $x_{i+1} \leftarrow 0$.
    end if
    for iteration $j = 1, \cdots, m$ **do**
      Get feature by $y_{i,j} \leftarrow f_{i,j}(x_i; w_{i,j})$.
      if $i \leq (n-1)$ **then**
        Update x_{i+1} by $x_{i+1} \leftarrow x_{i+1} + \frac{1}{m} y_{i,j}$.
      end if
      Get the gradient for $w_{i,j}$ by:
        $\nabla w_{i,j} \leftarrow \frac{\partial}{\partial w_{i,j}} \mathcal{L}_i(g_i(y_{i,j}; \theta_i), y)$
      Accumulate the gradient for θ_i by:
        $\nabla \theta_i \leftarrow \nabla \theta_i + \frac{\partial}{\partial \theta_i} \mathcal{L}_i(g_i(y_{i,j}; \theta_i), y)$
      Update $w_{i,j}$ by the gradient $\nabla w_{i,j}$.
    end for
    Update θ_i by the gradient $\nabla \theta_i$.
  end for
end for
return super-net parameter $\mathcal{W}$

Authorized licensed use limited to: Hebei University of Technology. Downloaded on February 02,2023 at 14:34:57 UTC from IEEE Xplore. Restrictions apply.

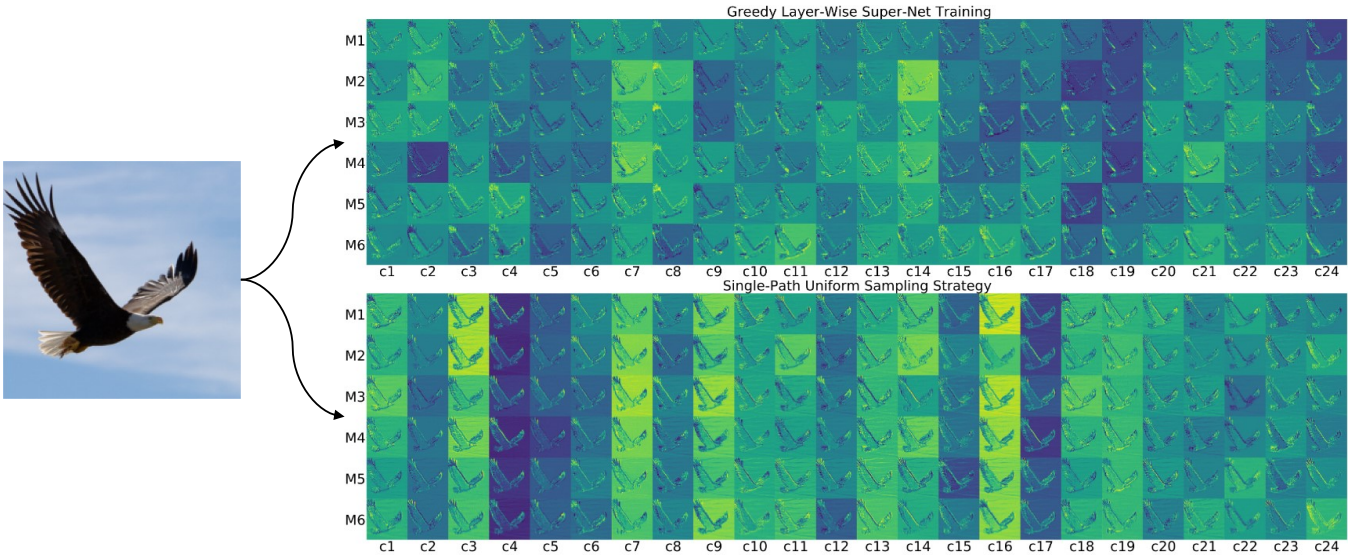

Fig. 2. **Feature maps from all six modules in the first layer of the pre-trained supernet.** As shown in the above figure, different choice modules in the super-net that is trained by greedy layer-wise learning learn much different feature maps and are not interchangeable during the training, compared with the single-path uniform sampling strategy. See quantitative measurement of similarity in Fig. 4.

each module in the super-net to be updated in parallel and only once during each training step. It ensures equal optimization opportunities for all choice blocks throughout the training, and neither overestimates nor underestimates their capacity. Note that both algorithms proposed in this paper meet strict fairness. In particular, after accumulating necessary gradients, each module and local decoder are updated immediately. It is completely equivalent to updating after accumulating gradients of all modules in the super-net. The details of greedy layer-wise super-net training are illustrated in Alg. 1.

Note that the updates of lower layers do not need to "wait' for upper layers, thus it can be sped up in an asynchronous way. Moreover, it greatly reduces the memory footprint since only intermediate activations within a shallow module require to be stored at a time. However, we discover that greedy layer-wise super-net training hurt the super-net training performance compared with the end-toi-end baseline, as shown in Fig. 1. Although getting a faster convergence at the early phase of super-net training, it is soon overtaken by the end-to-end baseline. Furthermore, as shown in Fig. 2, we discover that greedy layer-wise super-net training increased representation shift, that is, different choice modules learn very different feature maps and are not interchangeable during the training. As shown and analyzed in works [17] and [34], large representation shift will lead to low ranking correlation (see more details about ranking correlation in section V-B).

B. Non-Greedy Layer-Wise Super-Net Training

In this paper, we invite the question of whether the large weight-sharing space in one-shot NAS can be modularized into layers by making the super-net training local. As shown in the above subsection, the naive greedy layer-wise learning hurt super-net training performance as well as ranking correlation. Inspired by the recent work [30] which proposed to send feedbacks from the top to the bottom, we propose to learn each layer in super-net by using the gradients propagated from the objective associated with the adjacent upper layer. Specifically, the optimization objective is given by:

$$\min_{\{w_{i,\cdot}\},\theta_i} \sum_{j=1}^{m} \mathbb{E}[\mathcal{L}_{i+1}(g_{i+1}(\widetilde{f}_{i+1}(f_{i,j}(X_i; w_{i,j})); \theta_i), Y)] \quad (5)$$

where $\widetilde{f}_{i+1}(\cdot) \in \{f_{i+1,j}(\cdot; w_{i+1,j})\}$ and it is randomly sampled from modules in the $(i+1)$-th layer. The details of non-greedy layer-wise super-net training is shown in Fig. 3. For instance, at the $(n \cdot t + i)$-th training step, we stack another module randomly sampled from the $(i+1)$-th layer upon each module in the i-th layer, and the i-th decoder used in greedy layer-wise super-net training is replaced with the $(i+1)$-th decoder attached to the $(i+1)$-th layer. The rest of non-greedy layer-wise super-net training is the same as the greedy counterpart (Alg. 1) and is described in Alg. 2. At last, we adopt the evolutionary algorithm [41] to search for the goal architecture under constraints of FLOPs.

Compared with the greedy layer-wise super-net training, the non-greedy algorithm bridges the training performance gap with the end-to-end baseline, as shown in Fig. 1. Moreover, the non-greedy layer-wise super-net training inherits advantages of the greedy counterpart, such as the low memory footprint, fast convergence at the early phase of the training and it also can be sped up in an asynchronous way. Furthermore, it achieves clear improvements over the strong baseline. As shown in Fig. 4, different choice modules in the same layer are interchangeable during the super-net training since they learn similar feature maps, thus it improves ranking correlation (see more details about ranking correlation in section V-B).

Authorized licensed use limited to: Hebei University of Technology. Downloaded on February 02,2023 at 14:34:57 UTC from IEEE Xplore. Restrictions apply.

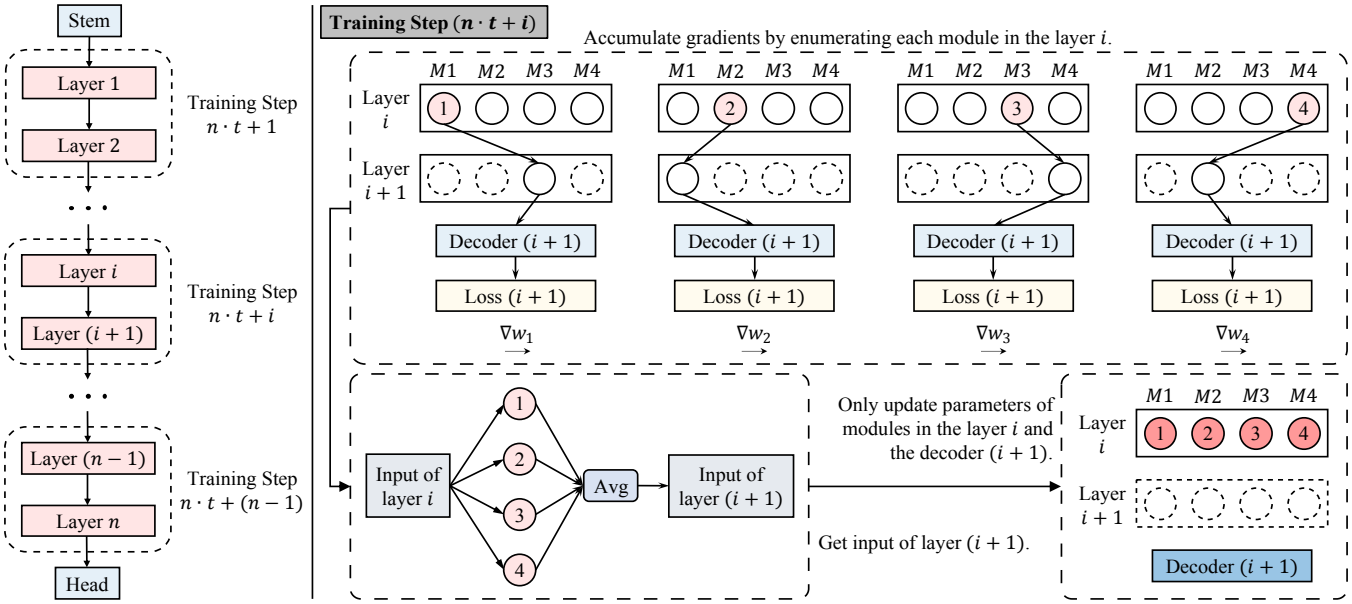

Fig. 3. **Illustration of non-greedy layer-wise super-net training.** Each super-net training step consists of training m modules and a decoder in parallel. The decoder used in this paper consists of a global average pooling layer and a fully-connected layer. The number of training batches or the number of super-net training loops is denoted by integer n. Note that the operations in the second layer of each shallow single-path subnet are randomly sampled from the $(i+1)$-th layer. Then we get the input of layer $(i+1)$ by averaging outputs of all operations in the layer i. Finally, the super-net only gets parameters in the layer i and the decoder $(i+1)$ updated. All operations in the layer i are ensured to be updated once within each training step.

V. Experiments

In this section, implementation details are first introduced. Then, we dissect our method on image classification task with both supervised and self-supervised objectives. Finally, we compare our method with state-of-the-art NAS algorithms.

A. Implementation Details

Search Space. Following recent works [14]–[17], we perform neural architecture search over the search space consisting of mobile inverted bottlenecks [36] and squeeze-excitation modules [37] for fair comparisons. There are six choice modules for each layer, including mobile inverted bottlenecks with kernel sizes of $\{3\times3, 5\times5, 7\times7\}$ and expansion rates of $\{4, 6\}$. The squeeze rate of squeeze-and-excite modules is set to a fixed value 0.25. The search space contains more than 3.32×10^{13} architecture candidates in total. The macro architecture of the search space is shown in Tab. I.

Training Setup. The super-net training algorithm is implemented with Pytorch. For training with supervised objective (classical cross entropy loss), we use the following settings: Adam optimizer momentum 0.9, weight decay 4e-5 and betas $(0.9, 0.999)$; initial learning rate 0.01 with a cosine annealing; standard data augmentation (including random crop and resizing, random horizontal flip with 0.5 probability and color jitter). For training with self-supervised objective, we use the same objective and data augmentation in SimCLR [32].

Retraining Setup. We retrain the discovered architectures for 500 epochs with 4 Nvidia Tesla V100 GPUs on ImageNet using similar settings as EfficientNet [3]: RMSProp optimizer with decay 0.9 and momentum 0.9; dropout rate 0.2; weight decay 4e-5; initial learning rate 0.045 with a warmup [44] in first 4 epochs and cosine annealing; label smoothing [45]; fixed AutoAugment policy [38]; exponential moving average.

TABLE I
The Macro Architecture of the Search Space

Input Shape	Modules	Repeat	Stride
$224^2\times3$	3×3 Conv	1	2
$112^2\times16$	3×3 Depth-Wise Separable Conv	1	2
$56^2\times24$	MBConv / Skip Connect	$1\sim3$	2
$28^2\times40$	MBConv / Skip Connect	$1\sim3$	2
$14^2\times80$	MBConv / Skip Connect	$1\sim3$	1
$14^2\times96$	MBConv / Skip Connect	$1\sim3$	2
$7^2\times192$	MBConv / Skip Connect	$1\sim3$	1
$7^2\times192$	1×1 Conv	1	1
$7^2\times1984$	Global Pool	1	-
1984	FC-1000	1	-

B. Feature Consistency and Ranking Correlation

As shown and analyzed in recent works [17] and [34], large representation shift damages the ranking correlation of one-shot neural architecture search. To demonstrate the effectiveness of our proposed method, we explore and visualize the feature maps from modules in the first layer. As shown in Fig. 2, Fig. 4 and Fig. 5, the greedy layer-wise learning enlarges the representation shift, and the non-greedy layer-wise learning reduces the representation shift compared with the end-to-end baseline (single-path uniform sampling strategy). In order to measure the similarity between feature maps more accurately, we incorporate the cosine similarity to measure the distance among various feature vectors. It ranges from -1 (opposite)

Authorized licensed use limited to: Hebei University of Technology. Downloaded on February 02,2023 at 14:34:57 UTC from IEEE Xplore. Restrictions apply.

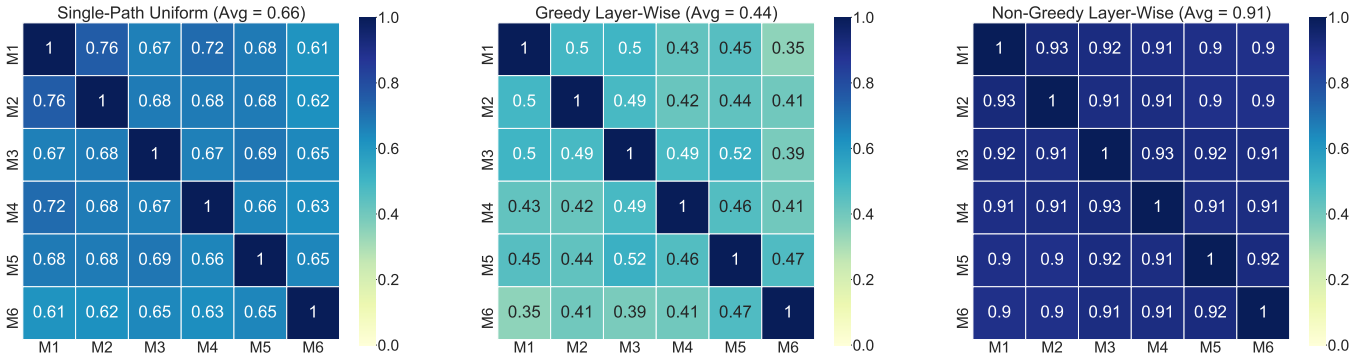

Fig. 4. **Cosine similarity matrices of the outputs from the first layer of six modules in each super-net variant.** Each value in above figure is an average of 1000 samples and the average value within the parentheses is calculated after stripping out the diagonal. Our non-greedy layer-wise super-net training achieves the overall highest feature similarity, intuitively explaining its effectiveness to reduce the representation shift.

TABLE II
RANKING CORRELATION OF DIFFERENT METHODS.

Methods	Objective	Kendall's Tau
Single-Path Uniform Strategy	CE	0.205
Greedy Layer-Wise Learning	CE	0.128
	InfoNCE [39]	0.333
Non-Greedy Layer-Wise Learning	CE	**0.538**
	InfoNCE [39]	0.513

to 1 (identical), where 0 indicates no correlation. In particular, we randomly sampled 1000 images from the validation dataset for the similarity measurement. As shown in Fig. 4, our non-greedy layer-wise learning achieves the highest cosine feature similarity as well as the minimal representation shift.

We further perform an analysis of ranking correlation to evaluate whether the layer-wisely supervised learning of the super-net can improve the ranking of sub-nets. To this end, we randomly sample 32 sub-networks and calculate the ranking correlation between the predicted performance and the true performance of training from scratch. Unfortunately, it is very computational expensive to train such many architecture candidates on large-scale dataset (ImageNet). Therefore, we construct a sub-dataset from ImageNet dataset. The sub-dataset only consists of 100 classes, where each class has 1000 training images and 100 validation images.

The Kendall rank correlation coefficient [40] on this sub-dataset is reported in Tab. II. The greedy layer-wise learning that uses cross entropy loss as layer-wise objective hurts the ranking correlation compared with the strong end-to-end baseline, but the greedy layer-wise learning that uses self-supervised loss (i.e. InfoNCE [39]) as layer-wise objective achieves small improvements over the strong end-to-end baseline. The non-greedy layer-wise learning that uses classical cross entropy loss as layer-wise objective has the highest Kendall rank correlation coefficient.

Algorithm 2 Non-Greedy Layer-Wise Super-Net Training

Input: Training data, the number of iterations T, super-net with n layers (each layer containing m choice modules), n local decoders, n local loss functions.
Output: pre-trained super-net

for iteration $t = 1, \cdots, T$ **do**
  Randomly sample the input data x_1 and label y.
  Clear gradients by $\nabla w_{:,:} \leftarrow 0$ and $\nabla \theta_{:} \leftarrow 0$
  for iteration $i = 1, \cdots, n$ **do**
    if $i \leq (n-1)$ **then**
      Initialize x_{i+1} by $x_{i+1} \leftarrow 0$
    end if
    for iteration $j = 1, \cdots, m$ **do**
      Get feature $y_{i,j}$ by $y_{i,j} \leftarrow f_{i,j}(x_i; w_{i,j})$.
      if $i \leq (n-1)$ **then**
        Update x_{i+1} by $x_{i+1} \leftarrow x_{i+1} + \frac{1}{m} y_{i,j}$
        Sample $\widetilde{f}_{i+1}(\cdot)$ and get feature $\widetilde{y}_{i+1}$ by:
          $\widetilde{y}_{i+1} \leftarrow \widetilde{f}_{i+1}(y_{i,j})$
      else
        Get feature $\widetilde{y}_{i+1}$ by $\widetilde{y}_{i+1} \leftarrow y_{i,j}$.
      end if
      Get the gradient for $w_{i,j}$ by:
        $\nabla w_{i,j} \leftarrow \frac{\partial}{\partial w_{i,j}} \mathcal{L}_{i+1}(g_{i+1}(\widetilde{y}_{i+1}; \theta_{i+1}), y)$
      Accumulate the gradient for θ_{i+1} by:
        $\nabla \theta_{i+1} \leftarrow \frac{\partial}{\partial \theta_{i+1}} \mathcal{L}_{i+1}(g_{i+1}(\widetilde{y}_{i+1}; \theta_{i+1}), y)$
      Update $w_{i,j}$ by the gradient $\nabla w_{i,j}$.
    end for
    Update θ_{i+1} by the gradient $\nabla \theta_{i+1}$.
  end for
end for
return super-net parameter $\mathcal{W}$

C. Compared with State-of-the-Art NAS Methods

Tab. III presents quantitative results under mobile settings on ImageNet. By searching under a FLOPs constraint of 370M, our method discovers an architecture (i.e. LWSL-A) that achieves 77.5% top-1 accuracy on ImageNet with only 370M FLOPs and 6.6M parameters. With a small increase

Authorized licensed use limited to: Hebei University of Technology. Downloaded on February 02,2023 at 14:34:57 UTC from IEEE Xplore. Restrictions apply.

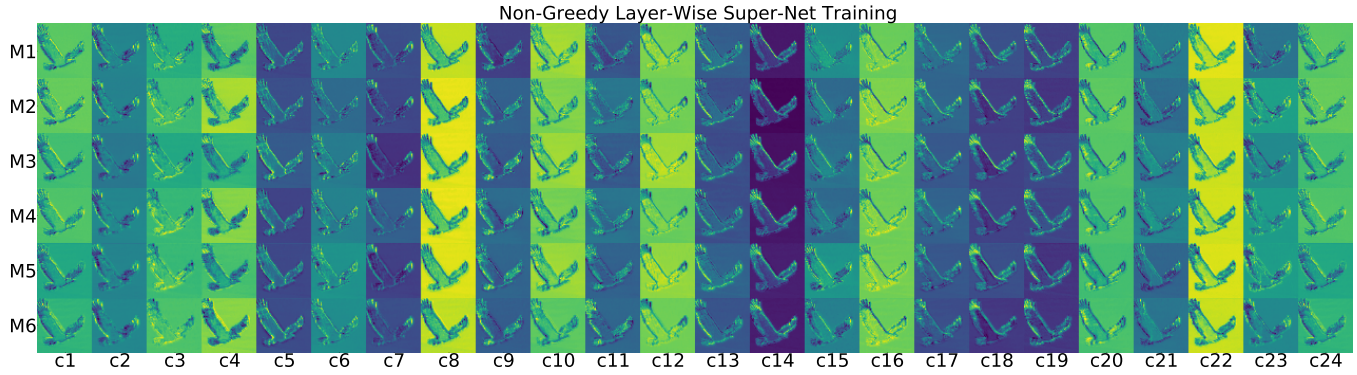

Fig. 5. **Feature maps from all six modules in the first layer of the super-net that is trained by non-greedy layer-wise learning.** Different choice modules in the super-net that is trained by non-greedy layer-wise learning learned very similar feature maps and are interchangeable during the training.

TABLE III
COMPARISON OF MOBILE MODELS ON IMAGENET.

Methods	FLOPs (M)	Param (M)	Top-1 Acc (%)	Cost (GPU days)
MobileNetV2 [36]	300	3.4	72.0	-
MobileNeXt-1.1 [42]	420	4.28	76.7	-
MnasNet-A1 [6]	312	3.9	75.2	≥ 379
Proxyless GPU [11]	465	7.1	75.1	8.1
NASNet-A [5]	564	5.3	76.0	1200
FairNAS-C [17]	321	4.4	76.7	12
Atom-NAS-B+ [13]	329	5.5	77.2	34
FairNAS-B [17]	345	4.5	77.2	12
Atom-NAS-C+ [13]	363	5.9	77.6	34
Single-Path NAS [43]	365	4.3	75.0	1.25
FairNAS-A [17]	388	4.6	77.5	12
EfficientNet-B0 [3]	390	5.3	76.3	$\approx 3k$
LWSL-A (Ours)	370	6.6	77.5	9*
LWSL-B (Ours)	397	6.8	78.0	9*

*: Cost shared among A and B.

TABLE IV
MEMORY USAGE OF SUPER-NET TRAINING

Methods	Memory Usage
Single-Path Uniform Strategy	100%
Greedy Layer-Wise Learning	**43.9%**
Non-Greedy Layer-Wise Learning	52.6%

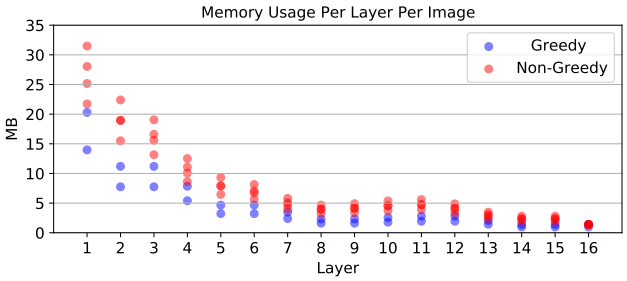

Fig. 6. The super-net training has an imbalanced memory usage distribution.

of allowable FLOPs (i.e. 400M), our method discovers an architecture (i.e. LWSL-B) that achieves 78.0% top-1 accuracy on ImageNet with only 397M FLOPs and 6.8M parameters. . Compared with the manually designed networks, such as MobileNetV2 [36] and MobileNeXt-1.0 [42], the discovered architectures achieves higher accuracy with fewer FLOPs. Compared with FairNAS [17] which adopts a modified single-path uniform sampling strategy, the discovered architecture (i.e. LSWL-A) achieves the similar accuracy with fewer FLOPs. Moreover, as shown in Tab. IV, our method has a much lower memory footprint compared with end-to-end methods. Although layer-wise learning saves memory footprint, as shown in Fig. 6, we discover that the super-net training has a very imbalanced memory usage distribution which prevents us from further reducing memory usage.

VI. CONCLUSION

In this paper, we modularize the large weight-sharing space of one-shot NAS into **layers** and develop the non-greedy layer-wise super-net training strategy which reduces the representation shift, improves the ranking correlation, reduces the memory footprint, and enables a faster asynchronous super-net training. Extensive experiments on ImageNet with both supervised and self-supervised objectives show that our method achieves clear improvements over the strong end-to-end baseline in terms of ranking correlation and memory footprint. By combining evolutionary algorithm to search over the pre-trained super-net, our method discover two models that are comparable to the state-of-the-art.

REFERENCES

[1] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
[2] Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).
[3] Tan, M., and Le, Q. (2019, May). Efficientnet: Rethinking model scaling for convolutional neural networks. In International Conference on Machine Learning (pp. 6105-6114). PMLR.

Authorized licensed use limited to: Hebei University of Technology. Downloaded on February 02,2023 at 14:34:57 UTC from IEEE Xplore. Restrictions apply.

[4] Baker, B., Gupta, O., Naik, N., and Raskar, R. (2016). Designing neural network architectures using reinforcement learning. arXiv preprint arXiv:1611.02167.

[5] Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. (2018). Learning transferable architectures for scalable image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 8697-8710).

[6] Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., and Le, Q. V. (2019). Mnasnet: Platform-aware neural architecture search for mobile. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 2820-2828).

[7] Xie, L., and Yuille, A. (2017). Genetic cnn. In Proceedings of the IEEE international conference on computer vision (pp. 1379-1388).

[8] Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y. L., Tan, J., ... and Kurakin, A. (2017, July). Large-scale evolution of image classifiers. In International Conference on Machine Learning (pp. 2902-2911). PMLR.

[9] Liu, H., Simonyan, K., Vinyals, O., Fernando, C., and Kavukcuoglu, K. (2018, February). Hierarchical Representations for Efficient Architecture Search. In International Conference on Learning Representations.

[10] Liu, H., Simonyan, K., and Yang, Y. (2018, September). DARTS: Differentiable Architecture Search. In International Conference on Learning Representations.

[11] Cai, H., Zhu, L., and Han, S. (2018, September). ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware. In International Conference on Learning Representations.

[12] Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., ... and Keutzer, K. (2019). 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).

[13] Mei, J., Li, Y., Lian, X., Jin, X., Yang, L., Yuille, A., and Yang, J. (2019, September). AtomNAS: Fine-Grained End-to-End Neural Architecture Search. In International Conference on Learning Representations.

[14] Bender, G., Kindermans, P. J., Zoph, B., Vasudevan, V., and Le, Q. (2018, July). Understanding and simplifying one-shot architecture search. In International Conference on Machine Learning (pp. 550-559). PMLR.

[15] Peng, H., Du, H., Yu, H., LI, Q., Liao, J., and Fu, J. (2020). Cream of the Crop: Distilling Prioritized Paths For One-Shot Neural Architecture Search. Advances in Neural Information Processing Systems, 33.

[16] Yu, J., Jin, P., Liu, H., Bender, G., Kindermans, P. J., Tan, M., ... and Le, Q. (2020, August). Bignas: Scaling up neural architecture search with big single-stage models. In European Conference on Computer Vision (pp. 702-717). Springer, Cham.

[17] Chu, X., Zhang, B., and Xu, R. (2021). Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 12239-12248).

[18] Li, X., Lin, C., Li, C., Sun, M., Wu, W., Yan, J., and Ouyang, W. (2020). Improving one-shot nas by suppressing the posterior fading. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 13836-13845).

[19] Li, C., Peng, J., Yuan, L., Wang, G., Liang, X., Lin, L., and Chang, X. (2020). Block-wisely supervised neural architecture search with knowledge distillation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 1989-1998).

[20] Moons, B., Noorzad, P., Skliar, A., Mariani, G., Mehta, D., Lott, C., and Blankevoort, T. (2021). Distilling optimal neural networks: Rapid search in diverse spaces. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 12229-12238).

[21] Li, C., Tang, T., Wang, G., Peng, J., Wang, B., Liang, X., and Chang, X. (2021). Bossnas: Exploring hybrid cnn-transformers with block-wisely self-supervised neural architecture search. arXiv preprint arXiv:2103.12424.

[22] Chen, X., and Hsieh, C. J. (2020, November). Stabilizing differentiable architecture search via perturbation-based regularization. In International Conference on Machine Learning (pp. 1554-1565). PMLR.

[23] Zhang, M., Li, H., Pan, S., Chang, X., and Su, S. (2020). Overcoming multi-model forgetting in one-shot NAS with diversity maximization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 7809-7818).

[24] Chu, X., Wang, X., Zhang, B., Lu, S., Wei, X., and Yan, J. (2020, September). DARTS-: Robustly Stepping out of Performance Collapse Without Indicators. In International Conference on Learning Representations.

[25] Hinton, G. E., Osindero, S., and Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. Neural computation, 18(7), 1527-1554.

[26] Marquez, E. S., Hare, J. S., and Niranjan, M. (2018). Deep cascade learning. IEEE transactions on neural networks and learning systems, 29(11), 5475-5485.

[27] Nøkland, A., and Eidnes, L. H. (2019, May). Training neural networks with local error signals. In International Conference on Machine Learning (pp. 4839-4850). PMLR.

[28] Belilovsky, E., Eickenberg, M., and Oyallon, E. (2019, May). Greedy layerwise learning can scale to imagenet. In International conference on machine learning (pp. 583-593). PMLR.

[29] Löwe, S., O'Connor, P., and Veeling, B. (2019). Putting An End to End-to-End: Gradient-Isolated Learning of Representations. Advances in Neural Information Processing Systems, 32, 3039-3051.

[30] Xiong, Y., Ren, M., and Urtasun, R. (2020). LoCo: Local Contrastive Representation Learning. Advances in Neural Information Processing Systems, 33.

[31] Wang, Y., Ni, Z., Song, S., Yang, L., and Huang, G. (2020, September). Revisiting Locally Supervised Learning: an Alternative to End-to-end Training. In International Conference on Learning Representations.

[32] Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020, November). A simple framework for contrastive learning of visual representations. In International conference on machine learning (pp. 1597-1607). PMLR.

[33] Pham, H., Guan, M., Zoph, B., Le, Q., and Dean, J. (2018, July). Efficient neural architecture search via parameters sharing. In International Conference on Machine Learning (pp. 4095-4104). PMLR.

[34] Peng, J., Zhang, J., Li, C., Wang, G., Liang, X., and Lin, L. (2021). Pi-NAS: Improving neural architecture search by reducing supernet training consistency shift. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 12354-12364).

[35] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2009.

[36] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L. C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4510-4520).

[37] Hu, J., Shen, L., and Sun, G. (2018). Squeeze-and-excitation networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7132-7141).

[38] Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. (2019). Autoaugment: Learning augmentation strategies from data. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 113-123).

[39] Oord, A. V. D., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748.

[40] Kendall, M. G. (1938). A new measure of rank correlation. Biometrika, 30(1/2), 81-93.

[41] Guo, Z., Zhang, X., Mu, H., Heng, W., Liu, Z., Wei, Y., and Sun, J. (2020, August). Single path one-shot neural architecture search with uniform sampling. In European Conference on Computer Vision (pp. 544-560). Springer, Cham.

[42] Zhou, D., Hou, Q., Chen, Y., Feng, J., and Yan, S. (2020). Rethinking bottleneck structure for efficient mobile network design. In Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16 (pp. 680-697). Springer International Publishing.

[43] Stamoulis, D., Ding, R., Wang, D., Lymberopoulos, D., Priyantha, B., Liu, J., and Marculescu, D. (2019). Single-path nas: Designing hardware-efficient convnets in less than 4 hours. arXiv preprint arXiv:1904.02877.

[44] Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., ... and He, K. (2017). Accurate, large minibatch sgd: Training imagenet in 1 hour. arXiv preprint arXiv:1706.02677.

[45] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826).

Authorized licensed use limited to: Hebei University of Technology. Downloaded on February 02,2023 at 14:34:57 UTC from IEEE Xplore. Restrictions apply.