MNGNAS: Distilling Adaptive Combination of Multiple Searched Networks for One-Shot Neural Architecture Search

Zhihua Chen, Guhao Qiu, Ping Li, *Member, IEEE*, Lei Zhu, Xiaokang Yang, *Fellow, IEEE*, and Bin Sheng, *Member, IEEE*

***Abstract*—Recently neural architecture (NAS) search has attracted great interest in academia and industry. It remains a challenging problem due to the huge search space and computational costs. Recent studies in NAS mainly focused on the usage of weight sharing to train a SuperNet once. However, the corresponding branch of each subnetwork is not guaranteed to be fully trained. It may not only incur huge computation costs but also affect the architecture ranking in the retraining procedure. We propose a multi-teacher-guided NAS, which proposes to use the adaptive ensemble and perturbation-aware knowledge distillation algorithm in the one-shot-based NAS algorithm. The optimization method aiming to find the optimal descent directions is used to obtain adaptive coefficients for the feature maps of the combined teacher model. Besides, we propose a specific knowledge distillation process for optimal architectures and perturbed ones in each searching process to learn better feature maps for later distillation procedures. Comprehensive experiments verify our approach is flexible and effective. We show improvement in precision and search efficiency in the standard recognition dataset. We also show improvement in correlation between the accuracy of the search algorithm and true accuracy by NAS benchmark datasets.**

***Index Terms*—Neural architecture search, knowledge distillation, multiple searched networks, image recognition.**

Manuscript received 7 December 2021; revised 10 January 2023; accepted 26 June 2023. This work was supported in part by the National Natural Science Foundation of China under Grants 62272164, 62272298, and 62077037, in part by the National Key Research and Development Program of China under Grant 2022YFC2407000, in part by the Interdisciplinary Program of Shanghai Jiao Tong University under Grants YG2023LC11 and YG2022ZD007, in part by the College-level Project Fund of Shanghai Jiao Tong University Affiliated Sixth People's Hospital under Grant ynlc201909, in part by the Medical-industrial Cross-fund of Shanghai Jiao Tong University under Grant YG2022QN089, in part by the Science and Technology on Space Intelligent Control Laboratory under Grant HTKJ2022KL502010, and in part by The Hong Kong Polytechnic University under Grants P0042740, P0030419, P0043906, and P0044520. *(Corresponding authors: Zhihua Chen; Bin Sheng.)*

Zhihua Chen and Guhao Qiu are with the Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai 200237, China (e-mail: 257866189@qq.com; czh@ecust.edu.cn).

Ping Li is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, and also with the School of Design, The Hong Kong Polytechnic University, Hong Kong (e-mail: p.li@polyu.edu.hk).

Lei Zhu is with the ROAS Thrust, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou 511400, China, and also with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong (e-mail: leizhu@ust.hk).

Xiaokang Yang and Bin Sheng are with the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: xkyang@sjtu.edu.cn; shengbin@sjtu.edu.cn).

I. Introduction

THE convolution neural network has achieved huge success in various computer vision tasks such as image recognition, semantic segmentation, and object detection [1]–[4]. As the network architectures designed in the recognition task are always used as pretrained backbone and finetune in the other tasks, image recognition has always been seen as a fundamental task for the neural network design. Many deep learning models need to stack many convolutional layers, which is very computationally intensive. Due to the need to deploy deep learning models on low-end hardware, designing lightweight neural network architectures has become a hot research topic. Commonly used methods are the method of manually designed network architecture and the neural network search method. Manual-designed neural networks generally design special modules first, then stack them and add skip connections. However, designing lightweight networks manually is inefficient and it is difficult to find the architecture that has both high performance and low latency for rapid response due to impossibility to fully explore the search space.

In recent years, neural network search (NAS) has attracted a growing interest which aims to automatically find the architecture combinations and corresponding hyperparameters. Following studies have shown architectures obtained by NAS algorithm are able to achieve higher performance compared to human designed architectures. The core idea of NAS is to use a search strategy to find the optimal network architecture for multiple goals at the same time in a pre-designed discrete search space at a limited computational cost. A straightforward method for NAS choosing the modules is to traverse all the possible candidate blocks, train them to convergence and evaluate. However, the whole process is too time-consuming. Taking into account the discontinuous search process and the search space, the early methods regarded the search process as a black box optimization problem and used reinforcement learning and evolutionary algorithms to find new architectures [5]–[7]. However, these early methods have a huge amount of calculation costs, which limits the wide usage.

Due to the huge search space and long training time of a single architecture, it is necessary to design specific search strategy and performance estimation algorithms to find high-precision architectures with few parameter sizes efficiently. The efficient search strategy weight sharing based algorithm is proposed [8]–[10]. Simplifying one-shot algorithm introduces

a specific network that forces the candidate network to share parameters in a complex network, called SuperNet in most of later studies. In later studies, the whole process can be divided into two parts, SuperNet training part and candidate network searching part. In weight sharing based method, the weight learned by SuperNet is directly used to estimate the performance of focused candidate subnetwork. Note that there is no strong correlation between the accuracy obtained by directly using SuperNet weight and obtained by standalone training. The standalone ranking is the focusing one and direct obtained ranking contains error. A common solution to the weak correlation is to retrain the architecture with few training epochs. However it is difficult to design the number of training epochs. Therefore it would be significant to design a more flexible and efficient retraining method to get the accurate performance ranking.

In this paper, we propose a specific knowledge distillation algorithm for retraining part in the searching process. The main idea is inspired from the fact that the convergence speed of using soft label training in distillation is faster than the general classification of hard labels [11]. In addition, multi-teacher distillation is more robust than single-teacher distillation. The teacher models in MNGNAS algorithm includes three parts. The first part is the model with higher accuracy obtained from the previous training, the second part is the architecture of the same search iteration and the third part is the candidate subnetwork with a little perturbation. The perturbation is the subnetwork with a few different candidate blocks. The first and second part of teacher models is used to increase the diversity of extracted features. The third part is used to help the teacher model to learn better knowledge useful for later knowledge distillation. Considering the computation burden, the third part of teacher models is only used in the optimal models. In most multi teacher knowledge distillation algorithms, the competition and compromise among different teacher mode features is common. In order to alleviate this problem, this paper proposes a method of dynamic coefficients, and solves the final weights through quadratic programming.

Our proposed algorithm can be used as a retraining auxiliary module in searching process for various one-shot-based NAS algorithms. Experiments show that SuperNet training methods like SPOS, FairNAS and MixPath combining the MNGNAS algorithm can find the high-precision architecture. The proposed MNGNAS algorithm has improved performance compared with corresponding primary algorithm. We prove that the proposed method can generalize well on three different search spaces and four image recognition datasets. The results have shown the effectiveness of all of the components which improve the final searched network accuracy.

Our main contributions can be summarized as follows:

- We introduce a NAS retraining method applied in the searching process, termed multiple network guided distillation neural architecture search (MNGNAS), which is flexible within arbitrary search spaces. We use a high-precision model group obtained by searching and searched architecture in the same iteration as the ensemble teacher models. Our proposal is easily incorporated into most existing one-shot-based NAS algorithms to improve search efficiency.
- The proposed adaptive ensemble distillation uses an optimization algorithm to determine the most suitable combination coefficient of feature maps. It makes better use of all features extracted from the searched architecture and avoids the domination of a single architectural feature.
- We introduce a specific knowledge distillation algorithm named perturbation-aware knowledge distillation for the optimal model in every searching epoch. Considering the feature difference caused by capacity difference, we add another distillation loss to help the models learn features more suitable for the further knowledge distillation process. The updated distillation process is to use the current model as the teacher model to guide the candidate models with few block perturbations.
- The adaptive knowledge distillation in the proposed MNGNAS algorithm helps to achieve competitive performance for the one-shot NAS algorithm on several image recognition datasets, including CIFAR10, CIFAR100, ImageNet16-120, and ImageNet. Our method can achieve higher accuracy with less search cost compared with most neural network search algorithms. We also conducted the experiment on the NAS benchmark dataset NAS-Bench-101, NAS-Bench-201, and NAS-Macro-Bench to verify that our proposed MNGNAS algorithm can obtain searching results that are closer to the standalone-training-accuracy compared to the recent proposed one-shot-based method.

The remainder of this paper is organized as follows: Section II introduces the related work of knowledge distillation and neural architecture search algorithm. In Section III, the adaptive ensemble knowledge distillation and perturbation-aware knowledge distillation algorithm in the proposed MNGNAS algorithm are described in details. Section IV gives the experimental settings and results. Section V gives some ablation experiments about similar distillation algorithm and different hyperparameter settings. In section VI, we conclude this paper.

II. Related Work

Here, we briefly review two research fields related to our work. We will firstly review the related work about knowledge distillation, followed by four streams of neural architecture search methods, i.e., reinforcement based NAS, evolution based NAS, gradient based NAS and one-shot-based NAS.

A. Knowledge Distillation

General knowledge distillation The vanilla knowledge distillation algorithm is the process which trains small student model with the supervision of a high capacity teacher model and encourages the similarity between lightweight student model and heavy teacher model. Considering that the prediction results of the teacher model contain more similarity information between categories, the knowledge distillation method helps the lightweight model obtain better results by adding the loss function of the difference between the prediction results [11], [12]. Subsequent algorithms introduce different

loss functions to improve the performance. Neural selectivity transfer [13] algorithm uses the Maximum mean Discrepancy as the distillation loss function. Relational knowledge distillation [14] proposes to transfer output relations instead of individual outputs. The transferred knowledge includes the distance loss function between two samples and the angular loss function between the three samples. FitNet [15] algorithm proposes to calculate the differences of feature map in each stage as the additional distillation loss.

Distillation from multi-teacher models Ensembles of neural networks are known to be much more robust and accurate than a single model. Each neural network is trained independently. The main problem is that using too many models introduces too much inference time. The subsequent knowledge distillation algorithm points out that the multiple teacher models can help the student model obtain more accurate outputs. Knowledge flow [16] algorithm proposes to add transformed and scaled intermediate representations from multi teacher models to the student model. MEAL [17] algorithm proposes using multiple teacher models to transfer knowledge to students, and the transferring feature map is determined by selection modules. In addition, the discriminator network is used to judge the input feature map from teachers or students to enhance the performance of distillation algorithm. MEALv2 algorithm is an improved algorithm and proposed using the average feature map as the transferring feature map. Hydra [18] algorithm proposes a distillation algorithm that uses a lightweight student network with multiple auxiliary branches to learn the features of the ensemble teacher model separately. There always exists compromise and competition between different teacher models in previous distillation methods. It is difficult to alleviate it by simple averaging operation or selecting the best model. We propose the adaptive knowledge distillation method which uses the optimization algorithm to obtain the aggregation coefficient which can partially alleviate the compromise and competition problem.

Online knowledge distillation Since the general knowledge distillation needs to train one or more teacher and student models separately, the computational cost in training process is relatively high. Online knowledge distillation proposes a method of using multiple simple models to transfer knowledge to each other to improve the performance of small models. Deep Mutual Learning [19] is an algorithm proposed earlier, which proposes to transfer knowledge among multiple students instead of one-way transfer in traditional knowledge distillation algorithms from only teacher to student model. ONE ensemble [20] proposes a multi-model online distillation algorithm using main module sharing and multiple auxiliary branches. In addition, it is proposed to use the gating function to predict the importance coefficient of each branch. OKDDIP [21] algorithm proposes a two level online knowledge distillation algorithm. The First level distillation is to train peer models by their own training targets. The second level is to transfer the knowledge from peer models to the group leader models. Self-distillation [22] algorithm proposes to use deep feature maps to transfer knowledge to shallow feature maps to avoid individual teacher model training.

B. Neural Architecture Search

The goal of NAS is to automatically design the network topology without human intervention and find the optimal architecture with highest standalone accuracy as well as satisfying the hardware constraint. The most accurate solution to NAS is to train each of the architectures within the search space from scratch to convergence and compare their performance. However it is impractical due to the huge number of search architectures and training costs. The commonly used solution is to train only the architecture in the designed search subspace using various heuristic search strategies. According to the searching strategy, NAS algorithms can be briefly categorized into Reinforcement learning (RL) based method, evolution algorithm (EA) based method, gradient based method and one-shot-based method.

NAS-designed neural networks MobileNetV3 [23] is a block-wise search algorithm and uses the inverted residual bottleneck as the main component in the search space. FBNet [24] algorithm is a cell-wise search algorithm and the search strategy is based on the gradient descent method. EfficientNet [25] algorithm uses search algorithms to find the best combination of network depth, network width and input resolution. MnasNet [26] algorithm takes accuracy and delay time as optimization targets at the same time and it is designed on the MobileNet search space. These architectures all require tremendous searching cost.

Reinforcement-learning-based methods Earliest RL-based method [5] uses the RNN-based controller to generate and search architecture, and use accuracy as the reward of the policy gradient strategy to update the controller parameters. Later proposed method [6] proposes the improved usage of Q-learning and uses the greedy exploration and experimental replay to choose the CNN layers.

Evolution-based methods The general EA-based method is to use the validation accuracy as fitness and the mutation and crossover operators in the evolutionary algorithm to search for new architectures. The method based on evolutionary algorithm generally uses the block as the search unit, and the combination of modules searched at each level is represented by a fixed-length string. Selection, crossover and mutation operations are used to generate new searching architectures. Evolution classifiers [7] propose an evolutionary algorithm in search space of variable depth. Regularized Evolution [27] algorithm proposes the aging evolution algorithm which discards the earliest training model.

Although reinforcement-based NAS and evolution-based NAS have achieved impressive results, they are extremely computationally expensive and generally require several hundred GPU hours, which severely restricts its application prospect. The main disadvantages of the RL-based method and the EA-based method is that each individual network is trained separately during the search process. Weight-sharing-based neural architecture search has recently become the mainstream in NAS studies and it significantly improves the computational efficiency. Weight-sharing method can be mainly divided into gradient-based method and one-shot method.

Gradient-based methods DARTS [28] is the earliest

gradient-based method. As the search space is reduced to a continuous structure space, the gradient descent method can be used to optimize the structure and weight alternately. The search algorithm based on gradient descent needs to expand the depth after the architecture search to complete the training of the architecture parameters. The PDARTS [29] algorithm states that there exists a depth gap between the searched architecture and the final architecture, that is, the searched shallow architecture may not be the optimal combination of operations in the deep architecture. PDARTS proposes a progressive expanding procedure. StablizingDARTS [30] and RobustDARTS [31] algorithm propose a regularization method to improve search performance and the norm and maximum eigenvalue of hessian matrix are are used as the basis for stopping the search respectively. Subsequent improvements to the DARTS algorithm noticed that the searched architecture is prone to generate excessive skip-connection layers. DARTS+ [32] algorithm proposes to use the early stop strategy that directly stops when there are too many skip-connections to avoid overfitting problem. FairDARTS [33] algorithm proposes to replace the softmax operation in the original DARTS algorithm with sigmoid to transform the competitive relationship of different modules into a cooperative relationship. NoisyDARTS [34] algorithm proposes to add noise to the skip connection branch to alleviate the problem of unfair competition between skip-connection and other modules.

One-shot-based methods One-shot method encodes the search space into a over-parameterized network SuperNet. It decouples SuperNet training from architecture searching. The main process is: training the SuperNet containing all search modules; using the designed search strategy to iterate the searching architecture and fine-tuning the corresponding architecture based on the weight of the SuperNet and selecting the best architecture to evaluate. The weight sharing in the one-shot method is to preserve the weights of all paths during the training of the subnetwork. The previously proposed multi-branch SuperNet network has the problem of excessive co-adaption between modules. Simplify one-shot algorithm [8] proposes to use path dropout strategy to alleviate the problem of excessive dependence between different modules, but this method requires artificial setting of dropout hyperparameters. Single Path One shot (SPOS) algorithm [9] proposes a training method that only one path is trained in each training step and uses uniform sampling of each module branch to make the accuracy of the search process closer to the standalone training accuracy. FairNAS algorithm [35] pointed out that the training accuracy is the closet to the true accuracy when the SuperNet training method is consistent with the general single network training method. A method for updating SuperNet weights after training multiple architectures is proposed. MixPath [10] algorithm proposes a special regularization layer shadow batch normalization to train multiple paths corresponding to multiple modules at the same time in the SuperNet. CARS algorithm [36] uses a SuperNet training method similar to the SPOS algorithm. The difference is that the search process is regarded as a multi-objective optimization problem, and the optimization target is selected as accuracy and FLOPs metrics. AttentiveNAS [37] algorithm also uses the similar SuperNet training process as SPOS and proposes to retrain only the pareto best and worst architecture to reduce the computational cost in the searching process. NSAS algorithm [38] proposes that it is easy to forget the previously trained parameters when training the SuperNet, and proposes the usage of continuous learning to improve the SuperNet training process.

There exists some studies that use the knowledge distillation process to guide the training of SuperNet. DNA [39] algorithm factorizes the search space into block-wise search. The distillation algorithm uses the output of the previous block as the teacher of the next block layer by layer. BossNAS [40] algorithm is an improved algorithm of DNA and proposes a online collaborative learning method that uses a self-supervised method to optimize each layer in candidate subnetworks. Cream of the Crop algorithm [41] uses the same search space and searching strategy as SPOS and proposes using the search prioritized path to guide the training of following searching architectures by knowledge distillation. AlphaNet [42] algorithm proposes a specific α-divergence regularization that focuses on avoiding over-estimation and under-estimation problem of the teacher network guidance.

In our method, we use the same SuperNet training strategy as one-shot-based method. Unlike previous distillation based method, our proposed algorithm MNGNAS is that we propose using the adaptive ensemble distillation method based on multiple models with better performance obtained by search and the architectures in the same searching epoch instead of the single-teacher distillation method in the previous method.

NAS Bench datasets Since NAS experiments generally require a lot of computing resources, it is difficult to reproduce the experiments and compare the algorithms fairly. Some recent studies provide records of the accuracy, FLOPs and other information corresponding to each architecture in the pre-designed search space. NAS-Bench-101 [43] only focuses on the CIFAR10 dataset. The search space includes 3×3 convolution, 1×1 convolution and 3×3 average pooling. NAS-Bench-201 [44] focuses on CIFAR10, CIFAR100 and ImageNet16-120 dataset and uses the MobileNet-like search space. In addition, the dataset gives the accuracy of the validation set during the training process. NATS-Bench [45] provides more detailed information of a topology search space and loss change in training process which can be used in all cell-based NAS algorithms. NAS-Macro-Bench [46] focuses on the topology structure and corresponding training, test accuracy information on a MobileNet-like search space.

III. Multi-Network-Guided Neural Architecture Search

In this section, we start by briefly reviewing the technical background related to our work and then introduce the details of the proposed MNGNAS algorithm.

A. Preliminaries

1) Neural Architecture Search Problem: In general, the NAS problem can be expressed as: Given a search space $\mathcal{A}$, a group of resource constraints b for architecture δ and

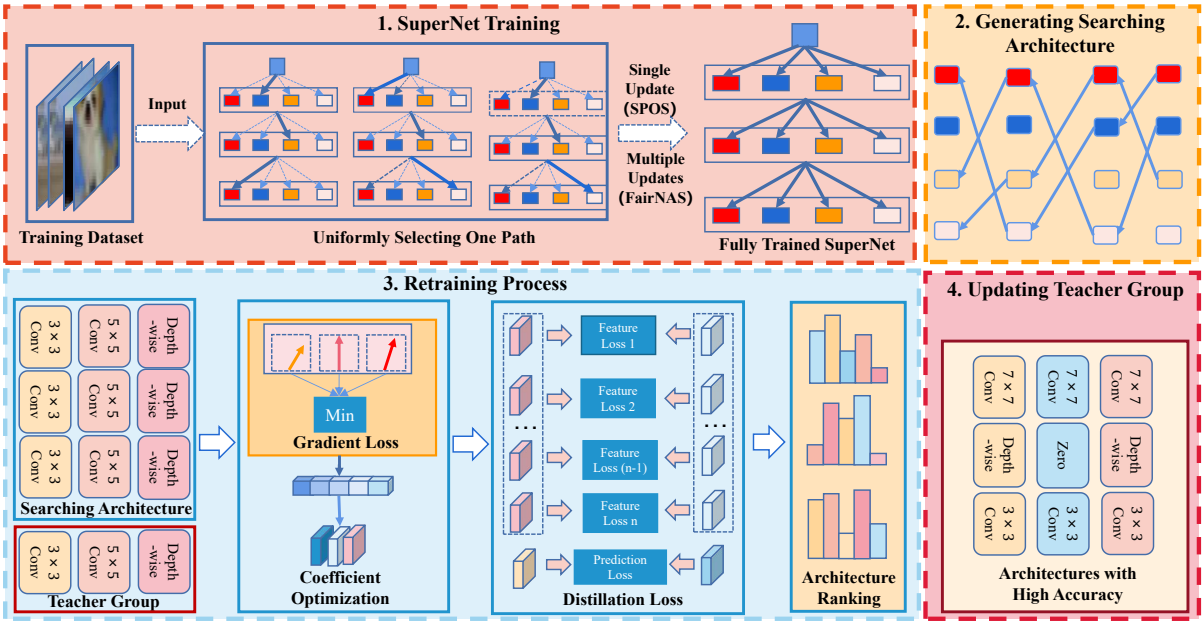


Fig. 1: The framework of whole process of our proposed algorithm. The algorithm includes four parts: **1.** Training of SuperNet. According to the SuperNet training strategy like SPOS and MixPath, subnetwork is selected for training and updated the parameters of the corresponding SuperNet path. **2.** Generating new searching architecture. We use evolution algorithm to generate new searching architectures. **3.** Retraining. The training of each architecture is based on the adaptive integrated distillation optimization algorithm. Teacher models include models with higher accuracy in the previous search process and other models in the current search iteration. **4.** Updating teacher group. We select the architecture with highest accuracy to update the large teacher group and small teacher group. Part 2, 3, and 4 correspond to the search process, which requires multiple iterations until the predefined number of search epochs is reached.

a performance estimator function E, find the optimal architecture and corresponding weights (δ^*, θ^*) simultaneously in the search space by maximizing the performance evaluation function. Formally, we want to enforce as

$$\begin{aligned} &E(\delta_i, \Theta_i; D_{val}) \leq E(\delta_j, \Theta_j; D_{val}) \\ &\Rightarrow Acc(\delta_i, \Theta_i; D_{test}) \leq Acc(\delta_j, \Theta_j; D_{test}), \end{aligned} \tag{1}$$

for all pairs of architectures in the search space. Here D_{val} and D_{test} refer to the validation and test dataset respectively.

In general, the searched network in the search space can be represented as the string $[C_1, C_2, \ldots, C_l]$, then, the corresponding deep learning architecture is defined as:

$$F^o_{w,\delta}(x) = (C_{w_1} \circ \cdots \circ C_{w_i})(x), \tag{2}$$

where C_{w_i} denotes as a searched block using the network weight, $F^o_{w,\delta}(x)$ denotes as the output feature map, and $\circ$ refers to the operations to connect several layers in sequential. The feature map F^m_i obtained in the i-th layer is defined as:

$$F^m_i(x) = (C_1 \circ \ldots \circ C_i)(x). \tag{3}$$

2) SuperNet Training Procedure and Challenges: The search space $\mathcal{A}$ is represented by a directed graph named SuperNet α containing all combinations of operations and each branch represents a combination. The goal of NAS task is to find the optimal architecture $\alpha_c \in \mathcal{A}$ in the validation dataset while meeting the specific constraint. In general, the process of one-shot neural architecture can be defined as:

$$\begin{aligned} W_c &= \arg\min_{W_c} L_{val}(\delta_c, W_c; D_{val}) + \lambda ||W_c||^2 \\ \mathbf{s.t.}\ \ \delta_c &= \arg\min_{\forall \delta \in \mathcal{A}} L_{tra}(\delta, W; D_{tra}) + \lambda ||W_c||^2 \ , \\ B(\delta_c) &\leq b \end{aligned} \tag{4}$$

where L_{tra} and L_{val}, respectively, refers to the loss in training dataset and in validation dataset, $B(\delta)$ function refers to the resource constraint function including FLOPs and latency for the architecture δ, b refers to the corresponding constraint value, $\lambda||W_c||^2$ refers to the weight decay term that prevents the searched model from overfitting, D_{train} and D_{val} refer to the training and validation dataset, respectively.

The main idea of one-shot method is to use the weight sharing method to train the SuperNet network once, and directly use the trained SuperNet weight to obtain the accuracy during the search process. Evolutionary algorithm is always used as the search strategy. Different from training the single network separately, there exists parameter overlap between different subnetworks and it is impossible to guarantee that the results of SuperNet training are the same as those of individual training. Even we cannot guarantee that the accuracy ranking obtained by SuperNet training is the same as the real ranking. A simple solution is to fine-tune the weights corresponding to the SuperNet during the searching process. However, due to huge training cost, the number of retraining epochs in the searching process is not easy to determine. We propose to

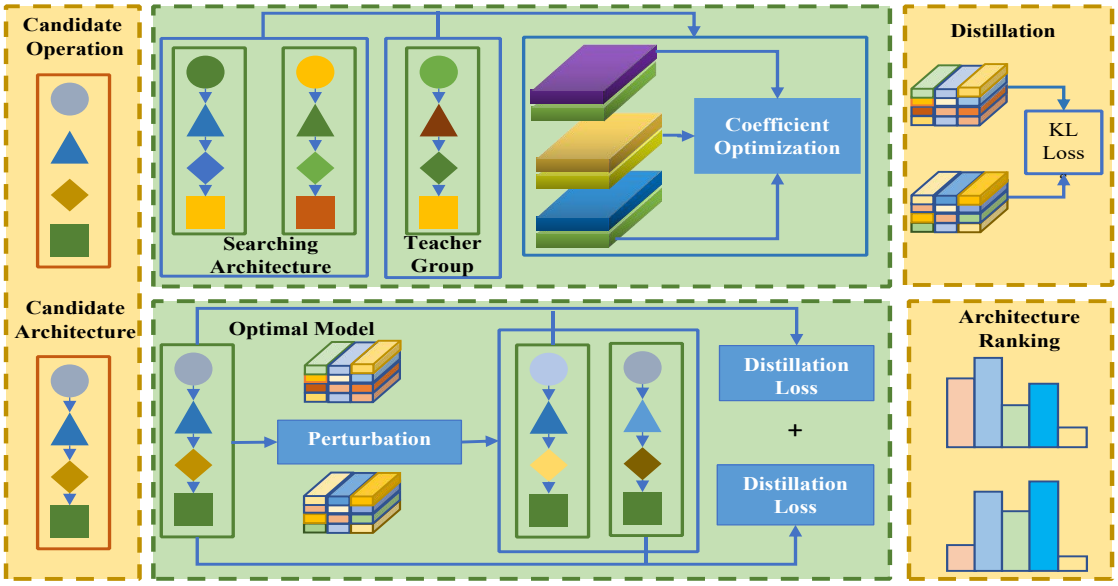


Fig. 2: The overall framework of the distillation procedure in the proposed MNGNAS algorithm. The proposed algorithm takes the generated architectures and current ensemble teacher group as input and output the updated ensemble group and searched architecture ranking. We use the optimization algorithm to obtain the adaptive coefficient. The weight of optimal architecture is further fine-tuned by the perturbation-aware knowledge distillation algorithm which adds the loss term to calculate the feature loss between optimal architecture and perturbed architecture.

use adaptive ensemble knowledge distillation to get a more accurate performance ranking with fewer training epochs.

B. Overall Framework

As shown in Fig. 1, our proposed algorithm MNGNAS is composed of four parts: (1) Training the SuperNet by uniform sampling strategy like SPOS algorithm. (2) Using the searched teacher group and the searched architecture in the same iteration to train the current training network according to the proposed adaptive ensemble knowledge distillation algorithm for every searching architectures and the perturbation-aware distillation algorithm for the optimal architectures. The detailed procedure is shown in Fig. 2. (3) Selecting the models with higher accuracy on the validation set and updating the searched teacher model. (4) Generating new candidate architectures according to the evolution algorithm. In the next section, we will introduce the proposed knowledge distillation algorithm for general model retraining in detail.

C. Adaptive Ensemble Knowledge Distillation

In searching procedure, the aim is to design an appropriate strategy to get the accurate performance ranking with only a few training epochs. We propose to use the idea of knowledge distillation to speed up deep learning model convergence and use the method of multiple teacher distillation to maintain the diversity of features in the distillation algorithm. In addition, considering the need to maintain model diversity as well as avoiding the problem of the majority of lightweight model after multiple iterations of evolutionary algorithm, we divide the whole teacher model group into the large model group T_l and the small model group T_s .

The distillation loss defined in our proposed algorithm is composed of two parts: the intermediate layer difference and the prediction layer difference. The prediction layer difference is defined as:

$$L_{dis}^{final}(T,S) = \mathbb{E}_{x\sim\mathcal{D}_{tra}}(KL(f_s^T(x)||f_s^S(x))). \quad (5)$$

The intermediate layer difference is defined as:

$$L_{dis}^{inter}(T,S) = \mathbb{E}_{x\sim\mathcal{D}_{tra}}||f_j^T(x) - f_j^S(x)||^2, \quad (6)$$

where f_j^T and f_j^S refer to the jth feature map obtained by the teacher model T and the student model S. The distillation loss is the addition of prediction layer difference loss L_{dis}^{final} and intermediate layer difference loss L_{dis}^{inter}. The distillation loss is abbreviated as L_{dis} in later content.

In our proposed MNGNAS algorithm, the teacher model includes the teacher model group T_l or T_s and the model group $\{S_j\}_{j\neq i}^{n_{TC}}$ for cooperative learning. n_C refers to the number of currently searched model. The loss in cooperative learning is defined as:

$$L_{dis}(S_i,\{S_j\}|\Theta_{S_i}) = \frac{1}{n_C-1}\sum_{j=1,j\neq i}^{n_C-1} L_{dis}(S_i,S_j|\Theta_{S_i}). \quad (7)$$

In the first search process, we only use the label as the supervision to retrain the searching model. In the subsequent search process, the searching teacher model group and the model in the same searching epoch are used to guide the structure training. We use KL divergence to calculate the difference between feature maps. The distillation loss of the ith model in the first training searching epoch is calculated as $L_{dis}(T,S_i)$, where $f_j^{S_i}$ refers to the jth feature map obtained

Algorithm 1: Adaptive Ensemble Knowledge Distillation Process

Data: Searched teacher model group T_l and T_s.
Result: The architecture of the current epoch to complete the training.

Calculate the predictions p_S generated by the searched architecture in the current epoch;
for *i=0 to* $Epoch_{ft}$ **do**
  Calculate the predictions $\{p_{TS}\}^{n_{TS}}$ and $\{p_{TL}\}^{n_{TL}}$ generated by the searched small and large teacher group;
  Calculate the predictions $\{p_{S_i}\}^{n_C}$ generated by the current searching architectures;
  Calculate the optimal coefficient based on the optimization algorithm for the ith student described in Eq. (20);
  Calculate the distillation loss L_{dis} based on the adaptive coefficient and corresponding teacher models;
  Compute the gradient ∇_{W_j} by $\partial L_{dis}/\partial W_j$;
  Update the network weight W_j based on Eq. (10);
end

by the ith model. The distillation loss of the ith model in the training searching epoch is defined as:

$$L_{dis}^{tra} = \alpha L_{dis}(T, S_i|\Theta_{s_i}) + \beta L_{dis}(\{S_j\}, S_i|\Theta_{S_i}), \quad (8)$$

where $L_{TI}(\cdot)$ is the same as the loss in the first training epoch. The feature map used for distillation in the search teacher model group is selected as the average of the feature map. α and β are the adaptive ensemble coefficients. The calculation process is described in next section. The distillation loss between searched teacher models and training models is defined as:

$$L_{TS}(f^{TS}, f^{S_i}) = \frac{1}{N}\sum_{j=1}^{N} KL(f_j^{S_i}, \sum_{k=1}^{n_T} f_j^{S_k}), \quad (9)$$

where $f_j^{S_k}$ is the feature map predicted by kth model in the searched teacher group. N is the stage number of the S_i architecture.

The training loss in retraining architecture procedure includes classification loss L_{cls} and distillation loss L_{dis} and is calculated as $L = L_{cls} + L_{dis}$. Then the parameter update of the model S_i is defined as:

$$\begin{aligned}\Theta_{S_i}^{t+1} = \Theta_{S_i}^{t} - \eta\nabla_{\Theta_{S_i}}(&L_{cls}(y, p(S_i; x, \Theta_{S_i})) \\ &+ \alpha L_{dis}(p(S_i; x, \Theta_{S_i}), p(T; x, \Theta_T))) \\ &+ \beta L_{dis}(p(S_i; x, \Theta_{S_i}), p(S_j; x, \Theta_{S_j})),\end{aligned} \quad (10)$$

where $\Theta_{S_i}^{t+1}$ is the updated parameters of the S_i model in the $t+1$ period. y is the groundtruth label and $p(S_i; x, w_{S_i})$ is the predicted feature map from the network S_i. $p(TS; x, w_{TI})$ is the average feature map predicted by the searched teacher group. Algorithm 1 provides a meta-algorithm of the distillation process of the proposed MNGNAS algorithm.

D. Adaptive Weighting Coefficient

In general ensemble knowledge distillation, the feature maps generated from all teacher models are considered equally. The feature map is usually generated from the average value from all teacher models. There always exists competitions and conflicts among the teacher models and the final feature maps may be determined by few teachers. In our proposed algorithm, we want to get the guidance from all teachers.

The optimization algorithm is used to find an optimal direction of the current model by learning the weight coefficients of each teacher model. The detailed process is defined as:

$$\begin{aligned}&\min_{d, v_e, v_c, \xi_e, \xi_c} v_e + v_c + C_1\sum_{e=1}^{n_T}\xi_e + C_2\sum_{c=1}^{n_C-1}\xi_c + \frac{1}{2}\|\boldsymbol{d}\|^2 \\ &\mathbf{s.t.}\left\langle\nabla_{\boldsymbol{\Theta}}\ell_{m_i}^e\left(\boldsymbol{\Theta}^{(\tau)}\right), \boldsymbol{d}\right\rangle \le v_e + \xi_e \\ &\quad\left\langle\nabla_{\boldsymbol{\Theta}}\ell_{m_i}^c\left(\boldsymbol{\Theta}^{(\tau)}\right), \boldsymbol{d}\right\rangle \le v_c + \xi_c \\ &\quad\xi_e \ge 0, \quad \xi_c \ge 0\end{aligned}, \quad (11)$$

where $\boldsymbol{\Theta}^{(\tau)}$ refers to the parameter of the student network. $\ell_m^e\left(\boldsymbol{\Theta}^{(\tau)}\right)$ refers to the distillation loss regarding the student model and the mth teacher model. $\langle\cdot,\cdot\rangle$ refers to the inner product of two vectors. $\ell_m^c\left(\boldsymbol{\Theta}^{(\tau)}\right)$ refers to the collaborative learning loss regarding the current model m_i .$\boldsymbol{d}$ refers to the optimizing gradient descent direction we want the student network want to learn. C_1 and C_2 refer to the regularization parameter. ξ_e and ξ_c refer to the slack variable that allow the violation of $\left\langle\nabla_{\boldsymbol{\Theta}}\ell_m^e\left(\boldsymbol{\Theta}^{(\tau)}\right), \boldsymbol{d}\right\rangle \le v_e$ and $\left\langle\nabla_{\boldsymbol{\Theta}}\ell_m^c\left(\boldsymbol{\Theta}^{(\tau)}\right), \boldsymbol{d}\right\rangle \le v_c$.

To effectively solve the optimization problem, we get the dual problem which is defined as:

$$\begin{aligned}&\min_{\alpha,\beta}\frac{1}{2}\left\|\sum_{e=1}^{n_T}\alpha_e\nabla_{\boldsymbol{\Theta}}\ell_m^e\left(\boldsymbol{\Theta}^{(\tau)}\right) + \sum_{c=1}^{n_C-1}\beta_c\nabla_{\boldsymbol{\Theta}}\ell_m^c\left(\boldsymbol{\Theta}^{(\tau)}\right)\right\|^2 \\ &\mathbf{s.t.}\sum_{e=1}^{n_T}\alpha_e = 1, \quad \sum_{c=1}^{n_C-1}\beta_c = 1 \\ &\quad 0 \le \alpha_e \le 1, \quad 0 \le \beta_c \le 1\end{aligned}. \quad (12)$$

Moreover, by examining the KKT condition, we can get the optimal descent direction $\boldsymbol{d}^*$ as:

$$\boldsymbol{d}^* = -\sum_{e=1}^{n_T}\alpha_e\nabla_{\boldsymbol{\theta}}\ell_m^e\left(\boldsymbol{\Theta}^{(\tau)}\right) - \sum_{c=1}^{n_C-1}\beta_c\nabla_{\boldsymbol{\Theta}}\ell_m^c\left(\boldsymbol{\Theta}^{(\tau)}\right). \quad (13)$$

In addition, we can get the optimal coefficient parameter α_e^* and β_c^* as:

$$\begin{aligned}&\alpha_e^*\left(\left\langle\nabla_{\boldsymbol{\Theta}}\ell_m^e\left(\boldsymbol{\Theta}^{(\tau)}\right), \boldsymbol{d}\right\rangle - v_e^* - \xi_e\right) = 0 \\ &\beta_c^*\left(\left\langle\nabla_{\boldsymbol{\Theta}}\ell_m^c\left(\boldsymbol{\Theta}^{(\tau)}\right), \boldsymbol{d}\right\rangle - v_c^* - \xi_c\right) = 0 \cdot \\ &\alpha_e^*\xi_e^* = C_1\xi_e^*, \quad \alpha_c^*\xi_c^* = C_2\xi_c^*\end{aligned} \quad (14)$$

In this way, we have:

$$\begin{aligned} -\|\boldsymbol{d}\|^2 - v_e^* - C_1 \sum_{e=1}^{n_T} \xi_e = 0 \\ -\|\boldsymbol{d}\|^2 - v_c^* - C_2 \sum_{c=1}^{n_C-1} \xi_c = 0 \end{aligned}, \tag{15}$$

and

$$\begin{aligned} \left\langle \nabla_{\boldsymbol{\Theta}} \ell_{m_i}^e \left(\boldsymbol{\Theta}_e^{(\tau)} \right), \boldsymbol{d}_c^{(\tau)} \right\rangle &\leq v_e + \xi_e^* \\ &= -\|d^{(\tau)}\|^2 + \xi_e^* - C_1 \sum_{e=1}^{n_T} \xi_e^*, \end{aligned} \tag{16}$$

$$\begin{aligned} \left\langle \nabla_{\boldsymbol{\Theta}} \ell_{m_i}^c \left(\boldsymbol{\Theta}^{(\tau)} \right), \boldsymbol{d}^{(\tau)} \right\rangle &\leq v_c + \xi_c^* \\ &= -\|d^{(\tau)}\|^2 + \xi_c^* - C_2 \sum_{c=1}^{n_C-1} \xi_c^*. \end{aligned} \tag{17}$$

We want the $d^{(\tau)}$ to be a decent direction then the dynamic weight parameter should satisfy:

$$\xi_e^* - C_1 \sum_{e=1}^{n_T} \xi_e^* \leq 0, \quad \xi_c^* - C_2 \sum_{c=1}^{n_C-1} \xi_c^* \leq 0. \tag{18}$$

Considering the general feature difference loss, the gradient can be expressed as:

$$\begin{aligned} \nabla_{\boldsymbol{\Theta}} \ell_m^e(\boldsymbol{\Theta}^{(\tau)}) &= \nabla_{\boldsymbol{\Theta}} \ell_m^e(\frac{1}{2}||p(S;\Theta) - p(S_c;\Theta_c)||^2) \\ &= ||p(S;\Theta) - p(S_c;\Theta_c)||, \end{aligned} \tag{19}$$

then, the adaptive ensemble coefficient α and β is solved by the optimization that can be described as:

$$\begin{aligned} &\min_{\alpha,\beta} \frac{1}{2}|| \ p(S;\Theta) - \sum_{e=1}^{n_T} \alpha_e p(S_e;\Theta_e) - \sum_{c=1}^{n_C-1} \beta_c p(S_c;\Theta_c)||^2 \\ &\mathbf{s.t.} \sum_{e=1}^{n_T} \alpha_e = 1, \quad \sum_{c=1}^{n_C-1} \beta_c = 1 \\ &0 \leq \alpha_e \leq 1, \quad 0 \leq \beta_c \leq 1 \end{aligned} \tag{20}$$

In this sense, the optimization process can be seen as a dynamic weighting strategy. Through optimization algorithm, the distillation algorithm can be more tolerant to noise feature maps collected from candidate architectures.

E. Perturbation-Aware Knowledge Distillation

Considering the capacity difference in teacher and students may leading to the degradation in knowledge distillation algorithm, we design a specific knowledge distillation process for optimal architectures in every searching epochs.

The core idea is to let the optimal architectures learn more suitable features for later knowledge distillation algorithms. As the crossover and mutation operation only affects few operations, we propose to use the optimal architectures to guide a few architectures with few block perturbations as a regularization term to reduce the differences among optimal architecture and architectures in following searching process in feature level. The detailed process is shown in Algorithm 2. Here we give the formal expression of architectures with one perturbation. The architecture set is defined as:

$$S_j^P = \{S_i | \exists k \quad C_j^k \neq C_i^k, \quad \forall m \neq k \quad C_j^m = C_i^m\}, \tag{21}$$

where S_j is the focused optimal architecture. C_k is the perturbed block.

The loss of the optimal architecture training is defined as:

$$L_{dis}(S_j) = \mathbf{E}_{x \sim \mathcal{D}_{tra}, S_i \in S_j^P} KL(f_{S_j}(x), f_{S_i}(x)), \tag{22}$$

where f_{s_j} and f_{s_i} refers to the output of architecture S_j and S_i.

The parameter Θ_{S_j} update of the model S_j is defined as:

$$\begin{aligned} \Theta_{S_j}^{t+1} =& \Theta_{S_j}^t - \eta \nabla_{\Theta_{S_j}} (L_{cls}(y, p(S_i; x, \Theta_{S_i})) - \\ & \nabla_{\Theta_{S_j}} (L_{dis}(p(S_j; x, \Theta_{S_j}), p(S_j^P; x, \Theta_{S_j^P})))), \end{aligned} \tag{23}$$

where $p(S_i; x, \Theta_{S_i})$ refers to the predicted feature map from the focused optimal architecture S_j. $p(S_j^P; x, \Theta_{S_j^P})$ refers to the predicted feature map obtained from the perturbed architectures.

Algorithm 2: Perturbation-Aware Knowledge Distillation Process

Data: Optimal Architecture S_j and its initial weight W_{S_j}.
Result: The weight of architecture S_j after fine-tuning.
Obtain the perturbed architecture set S_j^P based on the rule shown in Eq. (22);
for S_i *in* S_j^P **do**
  Obtain the initial weight of S_i from optimal architecture S_j and SuperNet weight;
  Calculate the predictions on training dataset generated by perturbed architecture S_i;
  Fine-tuning the weight of S_j based on Eq. (23);
end

F. Candidate Architecture Updating Process

After the retraining process in searching process described in previous two sections, we obtain the retrained architectures in the current searching period and new searching architecture set is needed to be updated. The core updating step set is based on the evolution algorithm. Firstly, a population of networks $\{\alpha_k\}_{k=1}^{n_P}$ is randomly initialized. In addition, there need to be $2n_s$ architectures divided into the large model teacher group and the small model teacher group. The large teacher model group selects the highest accuracy of the architecture that satisfies the parameter constraints of the large teacher model. The small teacher model group is similar, and the one that satisfies the constraints of the small model is selected.

After each search epochs, all candidate architectures need to be sorted according to the accuracy value on the validation set. The n_T architectures with the highest accuracy is used to update the search teacher group T_s and T_l according to FLOPS. In the multiple teacher group update process, we

Algorithm 3: Searching algorithm

Data: Search Space $\mathcal{A}$, Pretrained SuperNet S, FLOPS threshold $flops_{thre}$.
Result: Optimal architecture

$T_s = \varnothing, T_l = \varnothing$;
for *i=0 to Epoch* **do**
  Sample n candidate architectures $[\delta_1, \delta_2, \ldots, \delta_n]$ following FLOPs constraint;
  if T_s == $\varnothing$ *and* T_l == $\varnothing$ **then**
    Retrain all the searched architecture $\{\delta_i\}_{i=1}^{n_C}$;
  else
    if *flops*$(\alpha_i) \leq flops_{thre}$ **then**
      Use T_s and $\{S_j\}_{j=1,j\neq i}^{n_C}$ as the ensemble teacher model to train all the searched model S_j;
    else
      Use T_l and $\{S_j\}_{j=1,j\neq i}^{n_C}$ as the ensemble teacher model to train all the searched model S_j;
    end
  end
  Select the high-precision architecture, and replace low-precision architectures in T_s and T_l set;
end

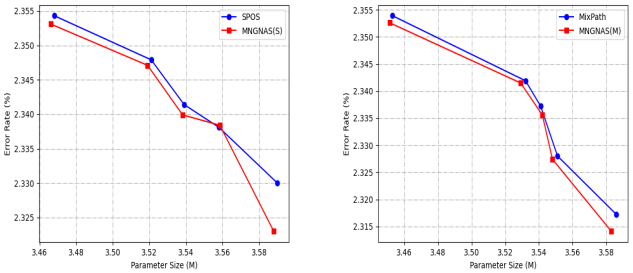
consider both the FLOPs and accuracy metrics. The updated architecture in small teacher group T_s is formally defined as:

$$T_s^{up} = \{\alpha | Acc(\delta, \Theta; D_{val}) \geq Acc(\delta_k, \Theta_k; D_{val}), \\ Flops(\alpha) \leq Flops(\delta_k) \quad \forall \delta_k \in T_s\}. \tag{24}$$

In the process of generating new architectures, crossover and mutation operators need to be used maintain the diversity of the search architecture. Crossover operation means that the same modules in the candidate architecture are retained, and different modules are randomly selected. The mutation operation is to randomly select a module to become another candidate one. Main process is to sort the structure with the highest fitness obtained by the search with the fitness in the search teacher group and keep the first n_{TS} architectures. The $n_{removed}$ architecture with the worst accuracy is removed. New candidate architectures for the next iteration are generated by the mutation and crossover operations on the remaining architectures. The procedure of our proposed MNGNAS algorithm is described in Algorithm 3.

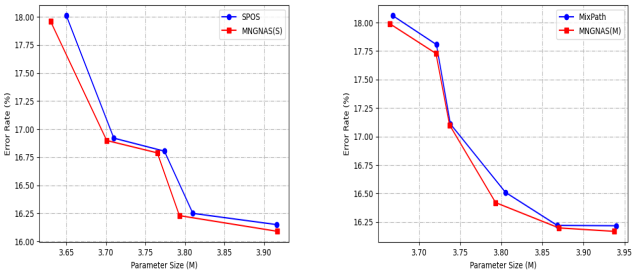
IV. Experimental Results

In this section, we evaluate the proposed MNGNAS algorithm on the image recognition task and conduct a series of experiments on CIFAR, ImageNet as well as recently proposed NAS Benchmark dataset. Furthermore we apply our proposed method into the chained and cell based search space to verify the generalization of our algorithm. The input image size for CIFAR10 and CIFAR100 is 32×32. ImageNet is a challenging dataset as the input resolution is 224×224 and contains 1000 classes. ImageNet16-120 contains the same image as ImageNet but with a small resolution as 16×16.



(a) SPOS (b) MixPath

Fig. 3: Test errors of some models searched by SPOS with our proposed MNGNAS algorithm on CIFAR10 dataset. NAS algorithms are based on the MobileNet search space.



(a) SPOS (b) MixPath

Fig. 4: Test errors of some models searched by SPOS with our proposed MNGNAS algorithm on CIFAR100 dataset. NAS algorithms are based on the MobileNet search space.

A. Implementation Details

On the image classification task, our proposed MNGNAS is evaluated in three search spaces: ShuffleNet, DARTS and MobileNet. The searched block includes 3×3 convolution, 5×5 convolution, 7×7 convolution and separable convolution block. In the SuperNet training process, we train all the subnetworks using momentum SGD with a batch size of 96 and train the whole SuperNet for 600 epochs with a simple data augmentation strategy. In the searching process, we set the number of architectures n_{TS} stored in the search teacher group T_l and T_s as 5. We use the evolution algorithm to generate new candidate architectures. Both crossover and mutation ratios are set to 0.5. The number of models to search n_S in each epoch is set to 30. The number of models removed due to low accuracy in each search epoch is set to 10.

As competitors, We use the representative one-shot NAS methods including SPOS, FairNAS and MixPath. These two methods are the current state-of-the-arts one-shot-based methods. We also compare our proposed algorithm with gradient based method including DARTS [28], PDARTS [29], GDAS [52] and DARTS- [53]. To validate the generalization of our retraining strategy, we consider combining with different SuperNet training strategies.

TABLE I: Test error rates, parameter size and search cost for our proposed MNGNAS algorithm, human designed networks and other NAS architectures on the CIFAR10 and CIFAR100 dataset. The NAS algorithms involved in the comparison include both gradient based method and one-shot-based method. Some of the algorithms give the results on the ShuffleNet, DARTS and MobileNet search spaces respectively. MNGNAS (S) and MNGNAS (M) respectively refer to the models proposed in this article that correspond to the combination of SPOS and MixPath training strategies. The main comparison is the performance comparison between the algorithm proposed in this article and the combination of SPOS and MixPath MNGNAS (S), MNGNAS (M) and the original algorithm.

Search Space	Algorithm	CIFAR10				CIFAR100			
		#Param	#MAdds	Search Cost (GPU Day)	Error	#Param(M)	#MAdds	Search Cost (GPU Day)	Error
-	ResNet101 [3]	44.5	1619	-	4.38	44.5	1619	-	21.62
	ResNeXt [47]	68.2	8893		3.58	68.2	8893		20.29
	ShuffleNetV2 [48]	3.2	404		5.83	3.2	404		18.18
	MobileNetV2 [49]	8.8	706		7.36	8.8	706		22.31
DARTS	ENAS [50]	4.6	804	0.8	2.89	5.1	933	0.9	17.13
	SNAS [51]	2.9	551	1.5	2.92	3.1	573	1.8	17.53
	DARTS [28]	3.3	574	1.5	2.76	3.8	651	5	17.54
	PDARTS [29]	3.4	557	0.3	2.50	3.6	585	0.3	16.55
	GDAS [52]	3.4	519	0.21	2.93	3.4	-	0.2	18.38
	DARTS- [53]	3.5	568	0.4	2.50	3.3	-	0.4	17.51
	NSAS [38]	3.54	-	0.4	2.73	3.54	-	0.4	18.02
	SPOS [9]	3.24	564	0.5	2.41	3.62	597	0.7	16.31
	MNGNAS (S)	3.24	560	0.6	2.38	3.49	579	0.7	16.21
	FairNAS [35]	3.32	572	0.6	2.46	3.70	633	0.8	16.42
	MNGNAS (F)	3.22	548	0.7	2.44	3.60	593	0.8	16.37
	MixPath [10]	3.34	572	0.6	2.43	3.64	615	0.7	16.30
	MNGNAS (M)	3.32	570	0.6	2.39	3.58	590	0.8	16.27
ShuffleNet	SPOS	2.37	389	0.3	2.72	2.32	326	0.7	16.31
	MNGNAS (S)	2.24	387	0.4	**2.68**	2.32	325	0.7	**16.13**
MobileNet	SPOS [9]	3.58	735	0.5	2.31	3.91	722	0.8	16.24
	MNGNAS (S)	3.61	726	0.6	**2.30**	3.92	748	0.9	**16.23**
	FairNAS [35]	3.54	723	0.5	2.30	3.89	741	0.9	16.22
	MNGNAS (F)	3.52	720	0.6	2.32	3.92	751	0.9	16.17
	MixPath [10]	3.58	710	0.6	2.26	3.91	746	0.8	16.13
	MNGNAS (M)	3.52	713	0.6	**2.22**	3.91	744	0.9	**16.12**

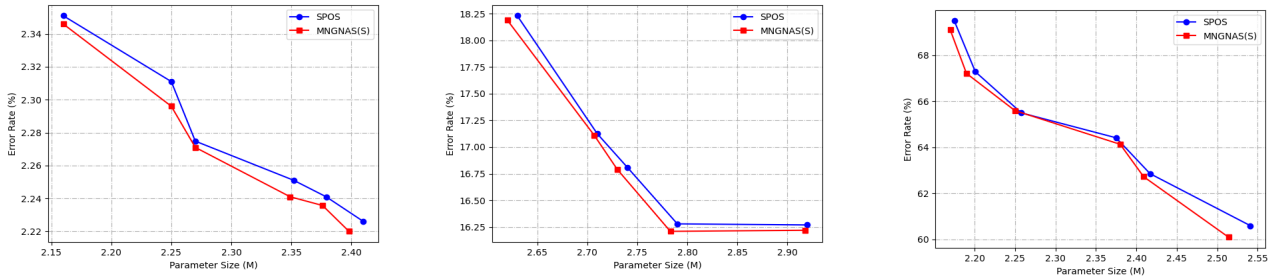
B. Evaluation Metrics

The evaluation process is to train the final obtained architecture on the training dataset and validate on the test dataset like the training methods of most image recognition models. The strategy to obtain the final architecture is a little different in block-based algorithm and cell-based algorithm. In block-based algorithm, the final architecture is to select one operation in corresponding path of the SuperNet. In cell-based algorithm, we need to stack more searched cell blocks to form the final architecture. In addition to analyzing the top architectures, we also conduct the correlation analysis. We calculate the correlation between the search accuracy and the true accuracy in the NAS-Bench-101 search space. The Kendall Tau metrics is used to evaluate the correlation.

C. Results on CIFAR10 and CIFAR100 Datasets

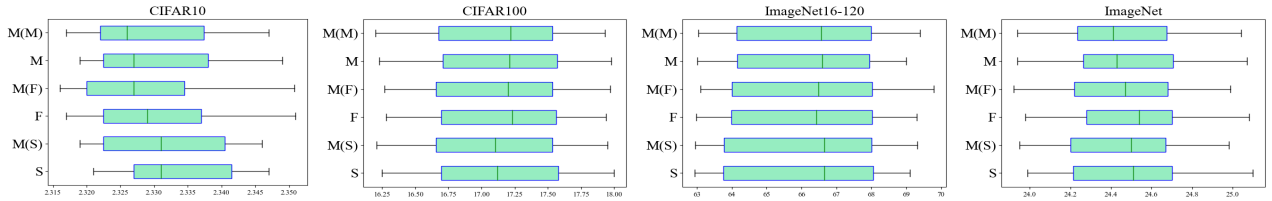
Fig. 3, Fig. 4 and Fig. 5(a)(b) compare the accuracy and FLOPS value of the architecture obtained from the original one-shot algorithms and the proposed MNGNAS algorithm. Fig. 5 shows the architecture obtained by using the SPOS training strategy on the ShuffleNet search space. Fig. 3 and Fig. 4 show the architecture obtained by using different training strategies on the MobileNet search space. The two-row line chart corresponds to SPOS and MixPath respectively. It is observed that under the close FLOPS conditions, our proposed algorithm can achieve a lower error rate most of the time. To compare the performance of MNGNAS with state-of-the-art NAS methods fairly, we follow the experimental setting in SPOS, FairNAS and MixPath algorithm. For this experiment, we consider accuracy in test dataset, MAdds, FLOPs and search cost (GPU Days) as the two objective of interest. The comparison results in CIFAR10 and CIFAR100 dataset are provided in Table I and can be summarized as:

1) Compared to primary one-shot-based NAS algorithm SPOS, FairNAS and MixPath, MNGNAS greatly improve the search results. Under the condition of similar search time, the MNGNAS algorithm can search for architectures with fewer parameters and higher accuracy in both ShuffleNet, MobileNet and DARTS search space. MNGNAS algorithm can obtain 0.03% and 0.1% decrease obtained by SPOS 0.02% and 0.5% decrease obtained by FairNAS and 0.04% and 0.03% decrease SuperNet training strategy in CIFAR10, CIFAR100 dataset and DARTS respectively. In MobileNet search space, MNG-



(a) CIFAR10 (b) CIFAR100 (c) ImageNet16-120

Fig. 5: Test errors of some models searched by SPOS with our proposed MNGNAS algorithm on different image recognition datasets. NAS algorithms are based on the ShuffleNet search space. SuperNet is trained on the training dataset and evaluated on the validation dataset.



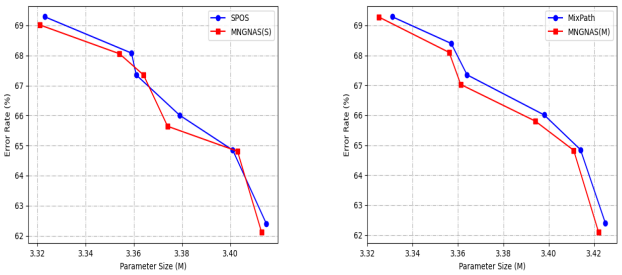
(a) CIFAR10 (b) CIFAR100 (c) ImageNet16-120 (d) ImageNet

Fig. 6: Box plot of error rates on different image recognition datasets. The figure shows the distribution of the error rate corresponding to the architecture obtained by different search algorithms. S, F, and M on the y-axis in the figure are the abbreviations for SPOS, FairNAS and MNGNAS algorithms respectively. M(.) is the abbreviation of MNGNAS and SuperNet training strategy algorithm respectively.

NAS algorithm can obtain 0.01%, 0.02% decrease obtained by SPOS and 0.05% and 0.01% decrease obtained by MixPath SuperNet training strategy.

2) Our proposed MNGNAS algorithm outperforms most of the gradient based NAS algorithms in accuracy with much lower computational consumption in the DARTS search space. In CIFAR10 dataset, the proposed algorithm can obtain 0.09% decrease compared with the PCDARTS algorithm with the similar parameters. In CIFAR100 dataset, the proposed algorithm can obtain 0.45% decrease compared with PDARTS algorithm with the similar parameters.
3) Our proposed MNGNAS algorithm can also outperform manual designed models in both accuracy and computational consumption. In both CIFAR10 and CIFAR100 dataset. The architectures obtained on the ShuffleNet search space outperform manually designed ShuffleNetV1 and ShuffleNetV2. Similarly, the architectures obtained on the MobileNet search space outperform manually designed MobilenetV1 and MobilenetV2.

The statistical values of error rate obtained by one-shot-based algorithm are represented in Fig. 6(a) and Fig. 6(b). As it shows, compared with the SPOS, FairNAS and MixPath algorithms, the proposed algorithm has a lower average error rate, and the minimum value of error rate obtained by the search is also lower.

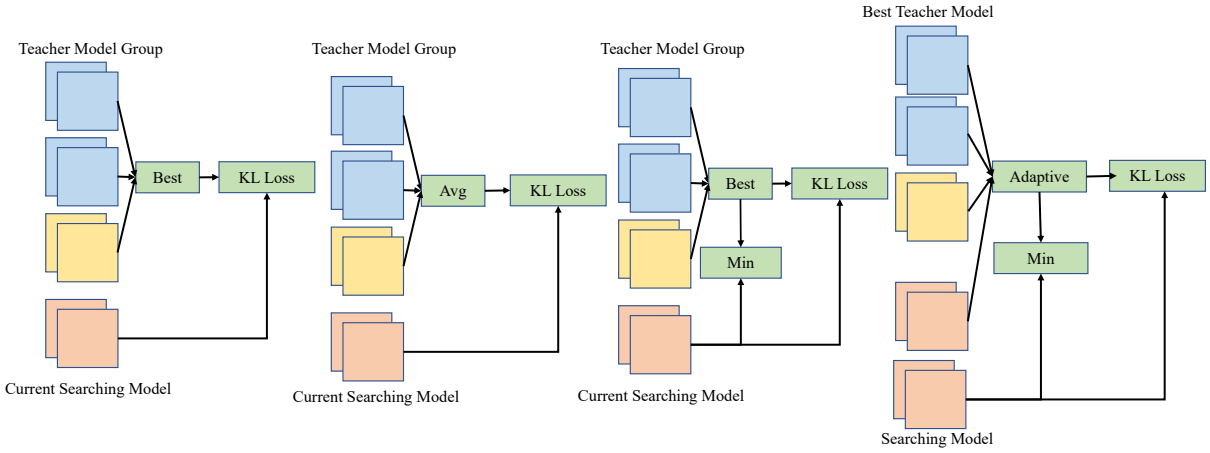


(a) SPOS (b) MixPath

Fig. 7: Test errors of some models searched by SPOS with our proposed MNGNAS algorithm on ImageNet16-120 dataset. NAS algorithms are based on the MobileNet search space.

D. Results on ImageNet16-120 Dataset

Different from the architecture on the CIFAR10 and CIFAR100 datasets, considering the resolution of the image, only a three-stage architecture is used on the ImageNet16-120 dataset. We make comparison with gradient based method and one-shot-based method where the quantitative results are summarized in Table II. Our proposed MNGNAS algorithm obtains slightly accuracy compared to corresponding one-shot method with fewer FLOPs. Compared with SPOS algorithm,



(a) B Algorithm (b) C Algorithm (c) D Algorithm (d) MNGNAS

Fig. 8: The graph of optimal architecture obtained by the proposed MNGNAS algorithm.

TABLE II: TOP-1 metrics for one-shot-based NAS architectures on the ImageNet16-120 dataset. The algorithms give the results on the DARTS, ShuffleNet and MobileNet search spaces respectively.

Search Space	Algorithm	#Params	Search Cost (GPU Day)	Top-1
DARTS	ENAS [50]	3.32	0.3	16.32
	DARTS [28]	3.24	0.3	16.32
	SETN [54]	3.27	0.4	32.52
	SPOS [9]	3.28	0.5	35.17
	MNGNAS (S)	3.24	0.6	35.53
	MixPath [10]	3.21	0.4	36.05
	MNGNAS (M)	3.22	0.5	36.05
ShuffleNet	SPOS [9]	2.54	0.4	39.31
	MNGNAS (S)	2.52	0.4	**39.53**
MobileNet	SPOS [9]	3.43	0.5	39.11
	FairNAS [35]	3.36	0.5	39.15
	MixPath [10]	3.44	0.5	39.44
	MNGNAS (S)	3.41	0.6	**39.52**
	MNGNAS (F)	3.43	0.7	**39.72**
	MNGNAS (M)	3.42	0.6	**39.74**

the proposed algorithm can obtain 0.22% and 0.41% increase in TOP-1 metrics in ShuffleNet and MobileNet search space. Compared with MixPath algorithm, the proposed algorithm can obtain 0.57% increase in TOP-1 metrics.

Fig. 5(c) and Fig. 7 compare the accuracy and FLOPS value of the architecture obtained from the original one-shot algorithms and the proposed MNGNAS algorithm. It shows that under the approximate FLOPS conditions, our proposed algorithm can achieve a lower error rate most of the time. Fig. 8 and Fig. 9 demonstrate the detailed optimal architecture obtained in MobileNet and DARTS search space on different image recognition datasets respectively.

E. Results on ImageNet dataset

In this section, we apply the proposed MNGNAS algorithm to both MobileNet and DARTS search space. We verify the performance of transfer learning as well as the searching on a large model in ImageNet dataset.

TABLE III: TOP-1 metrics for NAS obtained architectures on the ImageNet dataset. The algorithms give the results on the DARTS search spaces respectively. † represents directly searching on the ImageNet dataset.

Algorithm	Search Cost (GPU Day)	Params	TOP-1	TOP-5
SPOS [9]	1.25	4.6	75.5	92.5
FairNAS [35]	1.5	4.6	75.3	92.4
FairDARTS [33]	3.0	4.3	75.6	92.6
DARTS [28]	4.0	5.3	76.0	92.7
GDAS [52]	-	5.3	74.9	92.2
CyDAS [55]	-	5.4	75.9	92.6
PCDARTS [56]	3.8	5.3	75.8	92.7
RLNAS [57]	-	5.5	75.9	92.0
MNGNAS (S)	-	5.3	**76.2**	92.9
MNGNAS (S) †	1.41	5.4	75.8	92.8
MNGNAS (F)	-	5.3	75.9	92.8
MNGNAS (F) †	1.67	5.3	76.1	92.7
MNGNAS (M)	-	5.3	**76.4**	**92.9**
MNGNAS (M) †	1.56	5.5	76.1	92.7

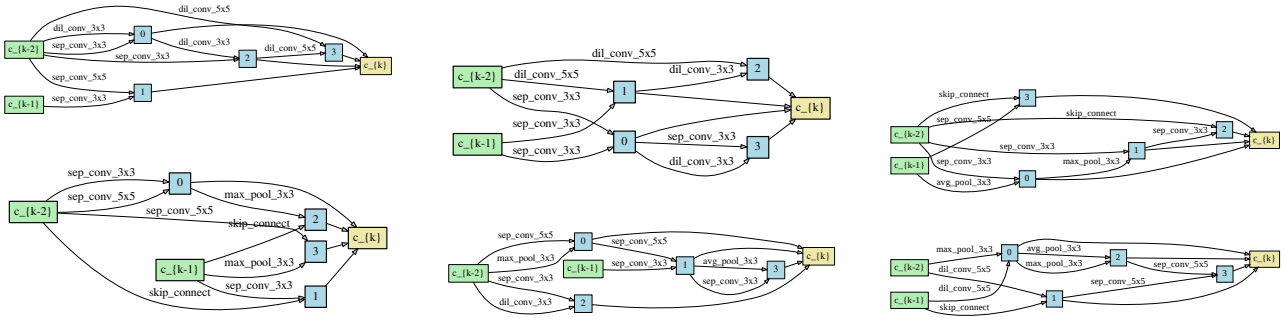
(a) DARTS search space: The experimental results are shown in Table III. Our proposed MNGNAS algorithm can obtain higher TOP-1 value in both directly searching and transferring settings. MNGNAS with FairNAS SuperNet training strategy can obtain highest TOP-1 among all compared training strategy, with 0.5% increase compared with CyDAS [55] and RLNAS [57].

(b) MobileNet search space: The experimental results are shown in Table IV. As it shows, the proposed MNGNAS algorithm obtains highest TOP-1 value. MNGNAS algorithm with FairNAS training strategy obtains 0.4% increase compared with primary FairNAS algorithm.

F. Results on NAS Benchmark Datasets

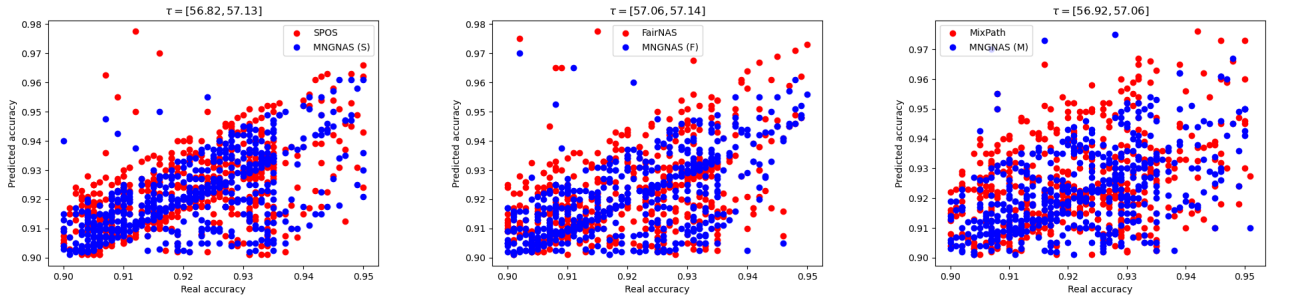
In this section, we apply MNGNAS to the NAS-Bench-101, NAS-Bench-201 and NAS-Macro-Bench search space.

(a) NAS-Bench-101: We apply the proposed method in one of the cell-based search space NAS-Bench-101. We mainly



(a) CIFAR10 (b) CIFAR100 (c) ImageNet

Fig. 9: The cell topology found in different image recognition datasets and DARTS search space. The cell topology is searched on CIFAR10, CIFAR100 and ImageNet datasets respectively. The two lines in the figure represent the topology of the normal and reduction modules respectively.



(a) SPOS (b) FairNAS (c) MixPath

Fig. 10: The Scatter plot of the real accuracy of the architecture extracted by NAS Bench101 and the accuracy obtained by the NAS algorithm. The algorithm to be compared is the original one-shot search algorithm SPOS, FairNAS, MixPath and the MNGNAS algorithm proposed in this paper for retraining the search architecture.

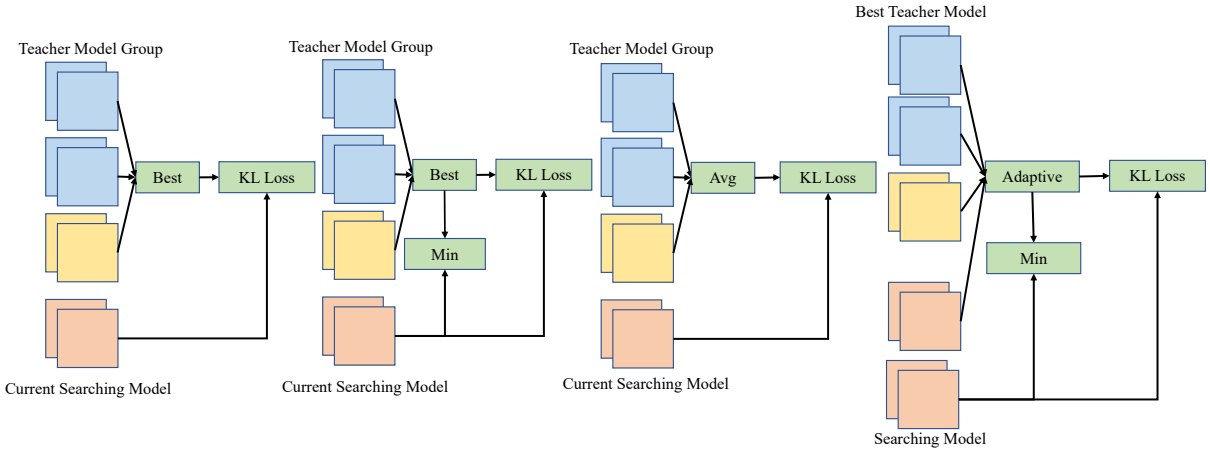
TABLE IV: TOP-1 and TOP-5 metrics for NAS obtained architectures on the ImageNet dataset. The algorithms give the results on the MobileNet search spaces respectively.

Method	Params	Flops	TOP-1	TOP-5
MobileNetV3 [23]	-	217	75.2	-
SPOS [9]	5.4	472	76.3	-
FairNAS [35]	4.6	488	77.5	93.2
FBNet [24]	4.4	375	74.9	92.1
ProxylessNAS [58]	7.0	457	75.1	92.7
MNGNAS (S)	4.7	392	77.4	93.6
MNGNAS (F)	4.5	375	77.9	93.9
MNGNAS (M)	5.0	455	77.5	93.6

give the correlation coefficient between the accuracy obtained by the algorithm and the actual accuracy. Fig 10 gives the scatter plot of actual accuracy and accuracy predicted by the MNGNAS algorithm and shows that the correlation between true accuracy and predicted accuracy obtained by MNGNAS algorithm and primary one-shot-based method. The blue and red point represents the searching results obtained by MNGNAS algorithm and primary one-shot-based NAS algorithms respectively. Obviously, there exists more points obtained by MNGNAS algorithm near the center line, which shows the superior performance compared with primary algorithm. We give the correlation coefficient above the scatter plot and it also shows the superior performance of our MNGNAS algorithm.

(b) NAS-Bench-201: In this part, we apply the proposed method in another cell-based search space NAS-Bench-201. To fairly compare with recently proposed method, we both give the average and optimal architecture obtained accuracy value obtained by the proposed MNGNAS algorithm. We conduct the experiments for four times and use 4 different random seeds. The experimental results are given in Table V. As it shows, the proposed MNGNAS algorithm can obtain relatively higher accuracy in most image recognition datasets and SuperNet training strategies. In CIFAR10 and ImageNet16-120 dataset, MNGNAS algorithm with SPOS training strategy obtains highest accuracy. In CIFAR100, MNGNAS algorithm with SPOS or MixPath training strategy obtains highest accuracy. The optimal architecture obtained by MNGNAS also achieves higher accuracy value compared with corresponding one-shot-based NAS algorithm.

(c) NAS-Macro-Bench: We also apply the one-shot-based NAS algorithm in the block-based search space. We consider three state-of-the-art one-shot-based architectures namely SPOS [9], FairNAS [35] and MixPath [10]. Moreover, we



(a) B Algorithm (b) C Algorithm (c) D Algorithm (d) MNGNAS

Fig. 11: The overall framework of one-shot-based NAS algorithm with similar distillation algorithms.

TABLE V: Test error rates with standard deviation for our proposed algorithm based on the NAS-Bench-201 search space.

	CIFAR10		CIFAR100		ImageNet16-120	
	Valid	Test	Valid	Test	Valid	Test
Optimal	91.61	94.37	73.49	73.51	46.77	47.31
REA [27]	91.25±0.31	94.02±0.31	72.28±0.95	72.23±0.84	45.71±0.77	45.77±0.80
REINFORCE [5]	91.12±0.12	93.90±0.26	71.80±0.94	71.86±0.89	45.37±0.74	45.64±0.78
DARTS [28]	39.77±0.00	54.30±0.00	38.57±0.00	38.97±0.00	18.87±0.00	18.41±0.00
GDAS [52]	90.01±0.46	93.23±0.23	24.05±8.12	24.20±8.08	42.84±1.79	43.16±2.64
SNAS [51]	90.10±1.04	92.77±0.88	69.69±0.39	69.34±1.89	42.84±1.79	43.16±2.64
DARTS- [53]	91.03±0.44	93.80±0.40	71.36±1.51	71.53±1.51	44.87±1.46	45.12±0.81
SPOS [9]	91.23±0.16	93.41±0.14	71.83±0.46	71.82±0.38	44.87±1.43	45.72±2.15
	(91.44)	(93.62)	(72.32)	(72.36)	(45.45)	(45.92)
MNGNAS (S)	**91.27±0.12**	93.41±0.13	**71.91±0.44**	**71.99±0.45**	44.81±1.34	45.73±2.19
	(91.49)	(93.77)	**(72.53)**	**(72.60)**	(45.69)	(45.97)
FairNAS [35]	91.33±0.09	93.27±0.08	71.77±0.38	71.94±0.43	44.79±1.63	45.17±1.83
	(91.45)	(93.70)	(72.29)	(72.39)	(45.53)	(45.89)
MNGNAS (F)	**91.35±0.11**	93.16±0.15	71.80±0.47	71.96±0.41	44.81±1.82	45.24±1.92
	(91.79)	(93.87)	(72.41)	(72.67)	(45.62)	(45.94)
MixPath [10]	91.28±0.14	92.94±0.31	71.75±0.41	71.71±0.38	44.73±1.79	45.41±2.06
	(91.48)	(93.57)	(72.33)	(72.43)	(45.39)	(45.84)
MNGNAS (M)	91.27±0.12	93.01±0.29	71.77±0.33	**71.99±0.43**	**44.85±1.53**	45.42±2.01
	(91.55)	(93.68)	(72.44)	**(72.58)**	**(45.61)**	(45.92)

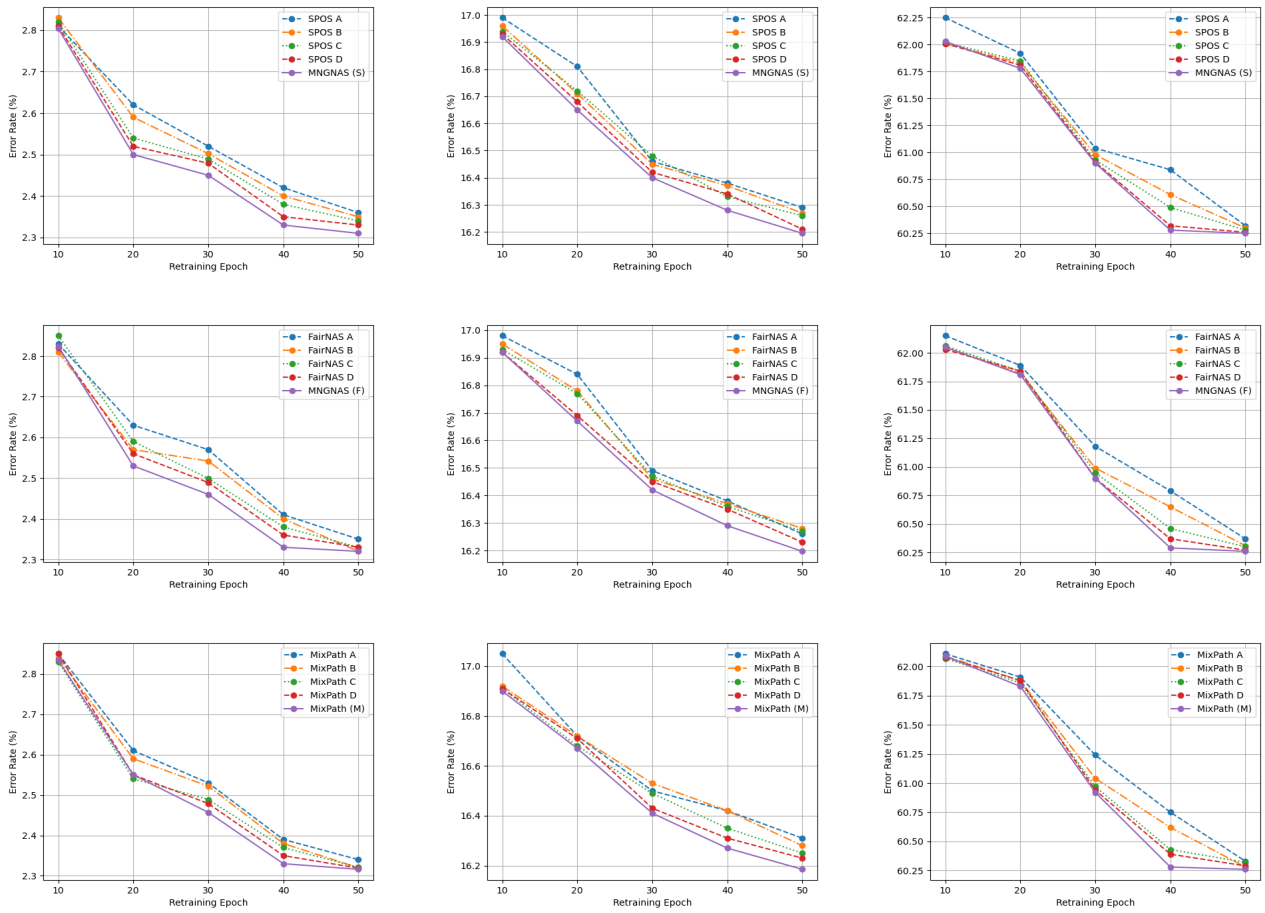
TABLE VI: Accuracy, parameter size, Flops and search cost for our proposed method and recently proposed NAS algorithm on the NAS-Macro-Bench search space.

Methods	FLOPS	Params	Top-1	Top-5	Search Cost
EfficientNet [25]	390	5.3	76.3	93.2	-
SCARLET [59]	280	6.0	75.6	92.6	12
MnasNet [26]	312	3.9	75.2	92.5	-
GreedyNAS [60]	284	4.7	76.2	92.5	<1
MCT-NAS [46]	280	4.9	76.3	92.6	<1
ProxylessNAS [58]	320	4.0	74.6	92.2	-
ST-NAS [61]	326	5.2	76.4	93.1	-
SPOS [9]	322	3.8	76.2	92.5	1.1
MNGNAS (S)	323	3.8	76.5	92.6	1.3
FairNAS [35]	326	3.9	76.7	92.6	1.2
MNGNAS (F)	324	3.9	**76.8**	92.9	1.4
MixPath [10]	327	3.9	76.4	92.4	1.1
MNGNAS (M)	330	4.2	76.7	92.5	1.2

TABLE VII: Object detection results on MS-COCO dataset. The detection model is based on RetinaNet combined with different backbone networks.

Backbone	FLOPs	Params	mAP	AP_{50}	AP_{75}
MobileNetV2 [49]	6.1	3.4	28.3	46.7	29.3
MobileNetV3 [23]	4.5	-	29.9	49.3	30.8
SPOS [9]	7.4	4.3	30.7	49.8	32.2
FairNAS [35]	8.0	5.9	32.4	52.4	33.9
MNGNAS (S)	7.37	4.2	**32.3**	53.5	36.5
MNGNAS (F)	7.88	5.7	**33.8**	55.2	37.2

compare our method with reinforcement and Monte Carlo based method including EfficientNet [25] and MCT-NAS [46]. The comparison results are given in Table VI. As it shows, one-shot-based method and our method can be seen as time-saving methods and obtain approximate TOP-1 value. Our proposed method MNGNAS can obtain higher TOP-1 compared with corresponding method. MNGNAS algorithm with FairNAS algorithm obtains highest TOP-1 value.



(a) CIFAR10 (b) CIFAR100 (c) ImageNet16-120

Fig. 12: Effect of different retraining epochs and different distillation methods on error rate of the searched architecture. We give the training loss in the 40 searching epochs by different distillation algorithms. The SuperNet training strategies are SPOS, FairNAS and MixPath in three lines respectively.

TABLE VIII: Comparison results on different distillation algorithms based on the ShuffleNet search space. The compared algorithms are similar to the proposed algorithm. We give the results on different image recognition datasets.

Distillation Algorithm	CIFAR10			CIFAR100			ImageNet16			ImageNet		
	#Params	#MAdds	Error	#Params	#MAdds	Error	#Params	#MAdds	Top-1	#Params	#MAdds	TOP-1
w/o KD	2.57	392	2.31	2.31	362	16.19	2.43	376	39.50	3.41	328	73.0
Single KD	2.53	385	2.29	2.31	361	16.34	2.45	378	39.49	3.41	331	73.2
Average Ensemble KD	2.52	393	2.26	2.32	371	16.20	2.41	375	39.45	3.41	335	73.1
Searched adaptive Ensemble KD	2.53	394	2.28	2.33	350	16.18	2.43	377	39.46	3.42	338	73.3
MNGNAS (S)	2.51	387	**2.24**	2.30	341	**16.10**	2.41	376	**39.51**	3.43	344	**73.5**

G. Results on COCO Object Detection Datasets

We validate the searched models on the downstream task in computer vision. We give the experimental results of object detection as a fundamental computer task which needs to detect object of different classes. We select the lightweight object detection model RetinaNet and retain most of the training hyperparameters. Table VII shows the related results of Flops, parameter size and mAP value. It shows that the backbone network obtained by the proposed MNGNAS algorithm obtains a superior tradeoff: higher mAP value with less Flops compared with primary one-shot-based method SPOS, FairNAS and manually designed backbones MobileNet.

V. Ablation Study

A. Effects of Knowledge Distillation Algorithm

In this section, we design the comparison groups for MNGNAS algorithm with general distillation and general ensemble distillation and keep the same SuperNet training algorithm. We compare the accuracy and parameters with similar distillation algorithms. The algorithms can be described as: (A) Searching algorithms that uses the general training strategy without the knowledge distillation process, (B) searching algorithm

TABLE IX: Comparison results on different distillation algorithms based on the MobileNet search space. The compared algorithms are similar to the proposed algorithm. We give the results on different image recognition datasets and both ShuffleNet and MobileNet search space.

SuperNet Training Algorithm	Distillation Algorithm	CIFAR10			CIFAR100			ImageNet16-120			ImageNet		
		#Params	#MAdds	Error	#Params	#MAdds	Error	#Params	#MAdds	TOP-1	#Params	#MAdds	TOP-1
SPOS	w/o KD	3.53	743	2.36	3.99	757	16.26	3.58	731	39.15	3.49	743	73.5
	Single KD	3.51	744	2.32	3.97	758	16.28	3.44	728	39.31	3.50	745	73.4
	Adaptive Ensemble KD	3.48	733	2.32	3.92	753	16.19	3.42	723	39.43	3.48	743	73.5
	Searched adaptive Ensemble KD	3.49	739	2.31	3.91	750	16.17	3.42	721	39.31	3.46	735	73.6
	MNGNAS	3.47	726	**2.31**	3.91	748	**16.23**	3.41	719	**39.54**	3.41	719	**73.8**
FairNAS	w/o KD	3.51	725	2.33	3.76	731	16.28	3.40	718	39.46	3.43	720	73.6
	Single KD	3.54	731	2.29	3.72	722	16.25	3.44	722	39.38	3.44	722	73.5
	Adaptive Ensemble KD	3.53	729	2.31	3.74	730	16.24	3.43	721	39.42	3.41	718	73.7
	Searched adaptive Ensemble KD	3.49	723	2.33	3.73	732	16.21	3.47	725	39.46	3.44	723	73.6
	MNGNAS	3.51	725	2.31	3.74	732	16.21	3.44	722	39.52	3.43	721	73.7
MixPath	w/o KD	3.51	757	2.38	3.97	748	16.26	3.44	722	39.46	3.42	720	73.4
	Single KD	3.51	724	2.36	3.93	752	16.25	3.46	727	39.28	3.40	716	73.5
	Adaptive Ensemble KD	3.49	748	2.34	3.95	746	16.23	3.45	726	39.61	3.45	726	39.61
	Searched adaptive Ensemble KD	3.49	749	2.33	3.94	745	16.23	3.43	723	39.54	3.42	720	73.7
	MNGNAS	3.48	746	**2.32**	3.91	744	**16.19**	3.42	720	**39.73**	3.43	722	73.7

TABLE X: Test error rates with standard deviation for our proposed algorithm based on the NAS-Bench-201 search space.

	CIFAR10		CIFAR100		ImageNet16-120	
	Valid	Test	Valid	Test	Valid	Test
SPOS A	91.25±0.24	93.09±0.13	71.78±0.41	71.83±0.51	44.79±1.07	45.82±1.93
SPOS B	91.23±0.14	93.07±0.17	71.82±0.38	71.85±0.41	44.76±1.01	45.84±1.51
SPOS C	91.24±0.21	93.16±0.27	71.91±0.28	71.88±0.71	44.64±1.23	45.79±1.24
SPOS D	91.26±0.26	93.29±0.24	71.89±0.17	71.94±0.81	44.72±1.43	45.81±1.06
	(91.42)	(93.65)	(72.48)	(72.57)	(45.54)	(45.94)
MNGNAS (S)	91.27±0.12	93.80±0.40	71.91±0.44	71.99±0.45	44.81±1.34	45.73±2.19
	(91.49)	(93.76)	(72.53)	(72.59)	(45.69)	(45.97)
FairNAS A	91.17±0.23	93.17±0.11	71.63±0.28	71.88±0.23	44.68±1.42	45.19±1.21
FairNAS B	91.19±0.27	93.16±0.14	71.64±0.19	71.88±0.27	44.67±1.01	45.18±1.33
FairNAS C	91.19±0.22	93.18±0.24	71.66±0.11	71.90±0.35	44.69±1.41	45.19±1.20
FairNAS D	91.20±0.24	93.09±0.24	71.74±0.11	71.94±0.35	44.74±1.41	45.21±1.20
	(91.63)	(93.82)	(72.32)	(72.65)	(45.61)	(45.95)
MNGNAS (F)	91.35±0.18	93.16±0.21	71.80±0.14	71.99±0.46	44.79±1.24	45.24±1.92
	(91.79)	(93.87)	(72.41)	(72.69)	(45.69)	(45.97)
MixPath A	91.30±0.13	93.12±0.19	71.72±0.16	71.96±0.42	44.78±1.14	45.20±1.13
MixPath B	91.31±0.11	93.14±0.17	71.71±0.12	71.98±0.32	44.81±1.21	45.21±1.34
MixPath C	91.33±0.17	93.15±0.18	71.73±0.13	71.97±0.33	44.77±1.31	45.22±1.82
MixPath D	91.34±0.23	93.15±0.24	71.76±0.19	71.97±0.36	44.79±1.41	45.24±1.71
	(91.50)	(93.60)	(72.33)	(72.51)	(45.52)	(45.89)
MNGNAS (M)	91.36±0.12	93.18±0.29	71.76±0.33	71.98±0.43	44.79±1.24	45.29±1.92
	(91.55)	(93.68)	(72.44)	(72.58)	(45.61)	(45.92)

TABLE XI: Comparison of Kendall Tau using different distillation algorithms and SuperNet training algorithms. We select 1000 architectures from NAS-Bench-101 dataset.

	SPOS	FairNAS	MixPath
w/o KD	56.46	57.15	57.13
Single KD	56.82	57.29	57.31
Adaptive Ensemble KD	57.13	57.51	57.81
Searched Adaptive Ensemble KD	57.19	57.45	58.01
MNGNAS	57.32	57.48	**58.04**

that only uses the model with a highest accuracy in the previous searching epoch as the teacher model, (C) searching algorithm that only uses the model with a higher accuracy in previous searching epoch as the general ensemble distillation algorithm, (D) searching algorithm that only uses the model with a higher accuracy in previous searching epoch as the adaptive ensemble teacher model, and (E) searching algorithm (MNGNAS) that uses the model with a higher accuracy in the previous searching epoch and models on the same search iteration as teacher model and uses the adaptive ensemble knowledge distillation. The detailed procedure of B, C, D and MNGNAS algorithm is shown in Fig. 11.

In Table VIII and Table IX, the error rate and TOP-1 metric in the compared datasets are shown. The search space is ShuffleNet and MobileNet respectively. We give the detailed results with different SuperNet training algorithms and knowledge distillation algorithms. Compared with other knowledge distillation algorithms in ShuffleNet search space, the proposed method can obtain the lowest error rate as well as highest TOP-1 accuracy metrics in both CIFAR and ImageNet dataset. In MobileNet search space, MNGNAS with SPOS obtains lowest error rate in CIFAR10 dataset. MNGNAS with FairNAS obtains lowest error rate in CIFAR100 dataset. MNGNAS with MixPath obtains highest accuracy value in ImageNet16-120

TABLE XII: Comparison results on MNGNAS algorithm with different number of teacher models. The results are obtained on ShuffleNet search space and different SuperNet training strategy.

Teacher Number	CIFAR10			CIFAR100			ImageNet16-120			ImageNet		
	S	F	M	S	F	M	S	F	M	S	F	M
4	2.41	2.36	2.29	17.16	16.88	16.89	60.98	60.73	60.41	27.82	27.62	27.92
6	2.37	2.31	2.27	16.76	16.41	16.37	60.63	60.63	60.34	27.31	27.41	27.54
8	2.29	2.25	2.23	16.42	16.28	16.23	60.56	60.51	60.28	26.91	26.83	26.99
10	2.27	2.21	2.21	16.23	16.23	16.19	60.56	60.42	60.27	26.89	26.84	26.88
12	2.26	2.21	2.22	16.22	16.21	16.19	60.56	60.41	60.28	26.89	26.83	26.88
14	2.28	2.23	2.26	16.22	16.21	16.22	60.59	60.41	60.28	26.90	26.83	26.89

TABLE XIII: Comparison results on MNGNAS algorithm with different number of teacher models. The results are obtained on MobileNet search space and different SuperNet training strategy.

Teacher Number	CIFAR10			CIFAR100			ImageNet16-120			ImageNet		
	S	F	M	S	F	M	S	F	M	S	F	M
4	2.37	2.38	2.27	17.19	16.91	16.95	60.97	60.89	60.54	27.89	27.74	27.77
6	2.35	2.33	2.25	16.80	16.57	16.65	60.63	60.68	60.33	27.52	27.54	27.56
8	2.27	2.29	2.22	16.41	16.26	16.19	60.54	60.54	60.26	26.86	26.95	26.93
10	2.24	2.25	2.21	16.27	16.25	16.18	60.52	60.41	60.25	26.85	26.82	26.86
12	2.25	2.24	2.22	16.27	16.26	16.19	60.52	60.41	60.26	26.83	26.81	26.86
14	2.25	2.25	2.24	16.28	16.25	16.19	60.53	60.42	60.25	26.85	26.82	26.88

TABLE XIV: Comparison results on MNGNAS algorithm with different number of teacher models. We give the results on the DARTS search space. We give the results on different image recognition datasets and different SuperNet training strategy.

Teacher Number	CIFAR10			CIFAR100			ImageNet16			ImageNet		
	S	F	M	S	F	M	S	F	M	S	F	M
4	2.43	2.52	2.65	17.32	17.13	17.12	60.72	60.81	60.84	27.53	27.56	27.58
6	2.32	2.38	2.41	16.85	16.81	16.78	60.58	60.51	60.53	27.44	27.47	27.49
8	2.28	2.29	2.38	16.67	16.56	16.53	60.47	60.44	60.41	27.45	27.43	27.44
10	2.27	2.26	2.29	16.36	16.41	16.35	60.46	60.42	60.41	27.44	27.42	27.41
12	2.27	2.25	2.29	16.33	16.40	16.36	60.45	60.43	60.42	27.45	27.43	27.41
14	2.28	2.25	2.27	16.32	16.38	16.39	60.46	60.42	60.42	27.44	27.42	27.42

dataset. MNGNAS with SPOS obtains highest accuracy value in ImageNet. In addition, our method can obtain lowest error rate as well as highest accuracy in most of the combinations of SuperNet training strategy and knowledge distillation algorithms. In Table X, the experimental results with different knowledge distillation algorithm in NAS-Bench-201 search space are given. We both give the mean accuracy value and optimal model obtained accuracy value. As the table shows, our proposed algorithm outperforms both mean accuracy and optimal model accuracy in most datasets. We also make a comparison between MNGNAS and similar distillation algorithms on the NAS-Bench-101 dataset. As reported in Table XI, our proposed algorithm can obtain much higher correlation coefficient compared with other distillation algorithms with several SuperNet training strategy. MNGNAS with MixPath training strategy can obtain the highest correlation coefficient.

B. Effects of Retraining Epochs

In this section, we provide an analysis on the impact of the retraining epochs n_{retra} in searching process. When we increase the retraining epochs, more accurate predicted accuracy can be obtained and the search cost will increase significantly. The error rates obtained by one-shot-based method based on different distillation algorithms with different retraining epochs are shown in Fig. 12. It shows that increasing the number of retraining epochs significantly decrease the error rate in the beginning but increase fewer after 40 epochs. Our MNGNAS algorithm can get the architecture with lower error rate within relative less retraining epochs. In practice, we set n_{retra} as 40. We also make comparisons with different retraining epochs on the NAS-Bench-101 search space. We give the ablation experiments of the combinations of different retraining epochs and distillation algorithm in Fig. 13.

C. Effects of the Number of Teacher Models

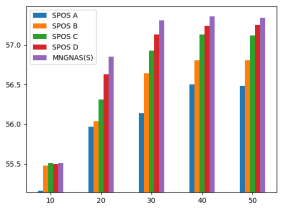
In this section, we investigate the effect of the number of teacher models in searching process. The number of teacher models here refers to the optimal models from previous searching epochs. Before the number of search epochs reaches the preset number of teacher models, we select all optimal models in the previous search period as the teacher model group. We train MNGNAS (S) and MNGNAS (M) with different teacher model number $n_{TS} \in \{2, 4, 6, 8, 10, 12, 14\}$ and report the error rates. We give the experimental results on ShuffleNet, MobileNet and DARTS search space and choose the SPOS as the SuperNet training strategy. The error rates obtained by the proposed MNGNAS algorithm are shown in Table XII, Table XIII and Table XIV. In most combinations, when we increase n_T from 4 to 10, the error rates become much lower. When we continue to increase n_T, the error rate becomes nearly unchanged and it will bring much computation costs. In practice, we set n_T 10. In Fig. 14, We make comparisons with different teacher model number on the NAS-Bench-101 search space. As it shows, We can also obtain the highest correlation coefficient with nearly 10 teacher models. We give the experimental results of MNGNAS algorithm with different

TABLE XV: Comparison results on MNGNAS algorithm with different number of teacher models. The results are obtained on NAS-Bench-201 search space and different SuperNet training strategy.

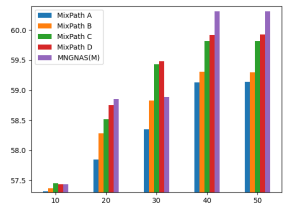
Teacher number	CIFAR10		CIFAR100		ImageNet16-120	
	valid	test	valid	test	valid	test
4	91.30±0.10	92.89±0.21	71.63±0.23	71.80±0.23	44.65±1.44	45.19±1.52
6	91.29±0.20	92.91±0.24	71.67±0.36	71.83±0.27	44.72±1.63	45.21±1.43
8	91.29±0.18	92.97±0.28	71.68±0.34	71.86±0.34	44.76±1.18	45.26±1.23
10	91.27±0.12	93.01±0.29	71.69±0.33	71.90±0.43	44.85±1.24	45.29±1.92
12	91.26±0.19	92.93±0.21	71.66±0.28	71.83±0.24	44.85±1.17	45.28±1.87
14	91.25±0.14	92.93±0.26	71.65±0.21	71.82±0.23	44.83±1.05	45.27±1.71

TABLE XVI: Comparison results on MNGNAS with different perturbed architecture number. The results are obtained on DARTS search space.

Perturbed Architecture Number	CIFAR10			CIFAR100			ImageNet		
	S	F	M	S	F	M	S	F	M
0	2.29	2.25	2.27	16.68	16.57	16.74	26.92	26.94	26.90
2	2.28	2.25	2.26	16.54	16.42	16.61	26.88	26.86	26.88
4	2.27	2.24	2.26	16.38	16.39	16.58	26.86	26.82	26.83
6	2.28	2.26	2.26	16.38	16.37	16.57	26.85	26.83	26.82

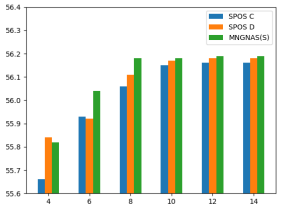


(a) SPOS

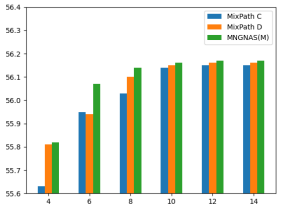


(b) MixPath

Fig. 13: The Kendall Tau metric (τ) during searching for different distillation algorithms and retraining epochs with primary algorithm and our MNGNAS algorithm. The ground truth value is obtained from the NAS-Bench-101 dataset.



(a) SPOS



(b) MixPath

Fig. 14: The Kendall Tau metric (τ) during searching for different distillation algorithms and teacher models with primary algorithm and our proposed MNGNAS algorithm. The ground truth value is obtained from the NAS-Bench-101 dataset.

teacher models in the NAS-Bench-201 search space in Table XV. As it shows, we can obtain high accuracy value when set the teacher number as 10 similar to the parameter settings in other search spaces. And when increasing the value, it will almost only bring more computation costs.

D. Effects of the Number of Perturbed Architecture

In this section, we investigate the effect of the number of perturbed architectures in searching process. We train MNGNAS with different SuperNet strategy and different perturbed architecture number in $\{2, 4, 6\}$. We also give the comparison results without perturbed architecture fine-tuning process and its corresponding number is 0. We give the experimental results on DARTS search space in Table XVI. As it shows, the perturbed architecture strategy can obtain lower error rate and 4 perturbed architectures are the optimal hyperparameter.

VI. Conclusion and Future Work

In this paper, we introduced the MNGNAS algorithm to improve the searching efficiency in one-shot-based NAS algorithm by the ensemble knowledge distillation algorithm. The core idea is to use multiple high-accuracy architectures as the teacher to distill the following searched architecture. The teacher model in our algorithm includes the model with higher accuracy in the previous search process and the model with the same search iteration. Considering the competition and compromise problem in general ensemble knowledge distillation algorithm, we introduce the adaptive coefficient based on optimization algorithm. In addition, we design a specific knowledge distillation strategy for optimal architectures. The loss calculating the feature differences between focused architectures and perturbed architectures is added to learn more suitable feature maps for later process. Extensive experiments demonstrated that we improved the accuracy and compactness of one-shot-based NAS methods on several benchmark datasets. Ablation studies showed that our method could obtain architectures with higher accuracy under fewer search epochs than similar distillation algorithms. Our proposed method can also be applied to improve performance in related computer vision tasks, which will be investigated as our future work.

References

[1] S.-H. Gao, M.-M. Cheng, K. Zhao, X.-Y. Zhang, M.-H. Yang, and P. Torr, "Res2Net: A new multi-scale backbone architecture," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 2, pp. 652–662, 2021.

[2] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4700–4708.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[4] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, "Squeeze-and-excitation networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 42, no. 8, 2020, pp. 2011–2023.

[5] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *International Conference on Learning Representations*, 2017, pp. 1–16.

[6] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," in *International Conference on Learning Representations*, 2017, pp. 1–18.

[7] R. Esteban, M. Sherry, S. Andrew, S. Saurabh, L. S. Yutaka, T. Jie, V. L. Quoc, and K. Alexey, "Large-scale evolution of image classifiers," in *International Conference on Machine Learning*, vol. 70, 2017, pp. 2902–2911.

[8] G. Bender, P.-J. Kindermans, B. Zoph, V. Vasudevan, and Q. Le, "Understanding and simplifying one-shot architecture search," in *International Conference on Machine Learning*, 2018, pp. 550–559.

[9] Z. Guo, X. Zhang, H. Mu, W. Heng, Z. Liu, Y. Wei, and J. Sun, "Single path one-shot neural architecture search with uniform sampling," in *European Conference on Computer Vision*, 2020, pp. 544–560.

[10] X. Chu, X. Li, S. Lu, B. Zhang, and J. Li, "MixPath: A unified approach for one-shot neural architecture search," *arXiv:2001.05887*, pp. 1–22, 2020.

[11] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv:1503.02531*, pp. 1–9, 2015.

[12] L. Wang and K.-J. Yoon, "Knowledge Distillation and Student-Teacher Learning for visual intelligence: A review and new outlooks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 6, pp. 3048–3068, 2022.

[13] Z. Huang and N. Wang, "Like what you like: Knowledge distill via neuron selectivity transfer," *arXiv:1707.01219*, pp. 1–9, 2017.

[14] W. Park, D. Kim, Y. Lu, and M. Cho, "Relational knowledge distillation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3967–3976.

[15] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "FitNets: Hints for thin deep nets," in *International Conference on Learning Representations*, 2015, pp. 1–13.

[16] I.-J. Liu, J. Peng, and A. G. Schwing, "Knowledge Flow: Improve upon your teachers," in *International Conference on Learning Representations*, 2019, pp. 1–17.

[17] Z. Shen, Z. He, and X. Xue, "MEAL: Multi-model ensemble via adversarial learning," in *AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 4886–4893.

[18] L. Tran, B. S. Veeling, K. Roth, J. Świątkowski, J. V. Dillon, S. Mandt, J. Snoek, T. Salimans, S. Nowozin, and R. Jenatton, "Hydra: Preserving ensemble diversity for model distillation," in *ICML Workshop on Uncertainty and Robustness in Deep Learning*, 2020, pp. 1–5.

[19] Y. Zhang, T. Xiang, T. M. Hospedales, and H. Lu, "Deep mutual learning," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4320–4328.

[20] X. Lan, X. Zhu, and S. Gong, "Knowledge distillation by on-the-fly native ensemble," in *Advances in Neural Information Processing Systems*, 2018, pp. 7528–7538.

[21] D. Chen, J.-P. Mei, C. Wang, Y. Feng, and C. Chen, "Online knowledge distillation with diverse peers," in *AAAI Conference on Artificial Intelligence*, 2020, pp. 3430–3437.

[22] L. Zhang, C. Bao, and K. Ma, "Self-Distillation: Towards efficient and compact neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 8, pp. 4388–4403, 2022.

[23] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan *et al.*, "Searching for mobilenetv3," in *IEEE International Conference on Computer Vision*, 2019, pp. 1314–1324.

[24] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, "FBNet: Hardware-aware efficient convnet design via differentiable neural architecture search," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 734–10 742.

[25] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*, 2019, pp. 6105–6114.

[26] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2820–2828.

[27] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 4780–4789.

[28] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *International Conference on Learning Representations*, 2018, pp. 1–13.

[29] X. Chen, L. Xie, J. Wu, and T. Qi, "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation," in *IEEE International Conference on Computer Vision*, 2019, pp. 1294–1303.

[30] X. Chen and C. Hsieh, "Stabilizing differentiable architecture search via perturbation-based regularization," in *International Conference on Machine Learning*, vol. 119, 2020, pp. 1554–1565.

[31] A. Zela, T. Elsken, T. Saikia, Y. Marrakchi, T. Brox, and F. Hutter, "Understanding and robustifying differentiable architecture search," in *International Conference on Learning Representations*, 2019, pp. 1–28.

[32] H. Liang, S. Zhang, J. Sun, X. He, W. Huang, K. Zhuang, and Z. Li, "DARTS+: Improved differentiable architecture search with early stopping," *arXiv:1909.06035*, pp. 1–15, 2020.

[33] X. Chu, T. Zhou, B. Zhang, and J. Li, "Fair Darts: Eliminating unfair advantages in differentiable architecture search," in *European Conference on Computer Vision*, 2020, pp. 465–480.

[34] X. Chu and B. Zhang, "Noisy differentiable architecture search," in *British Machine Vision Conference*, 2021, pp. 1–14.

[35] X. Chu, B. Zhang, and R. Xu, "FairNAS: Rethinking evaluation fairness of weight sharing neural architecture search," in *IEEE International Conference on Computer Vision*, 2021, pp. 12 239–12 248.

[36] Z. Yang, Y. Wang, X. Chen, B. Shi, C. Xu, C. Xu, Q. Tian, and C. Xu, "CARS: Continuous evolution for efficient neural architecture search," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1829–1838.

[37] D. Wang, M. Li, C. Gong, and V. Chandra, "AttentiveNAS: Improving neural architecture search via attentive sampling," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6418–6427.

[38] M. Zhang, H. Li, S. Pan, X. Chang, C. Zhou, Z. Ge, and S. W. Su, "One-shot neural architecture search: Maximising diversity to overcome catastrophic forgetting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 9, pp. 2921–2935, 2021.

[39] C. Li, J. Peng, L. Yuan, G. Wang, X. Liang, L. Lin, and X. Chang, "Block-wisely supervised neural architecture search with knowledge distillation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1989–1998.

[40] C. Li, T. Tang, G. Wang, J. Peng, B. Wang, X. Liang, and X. Chang, "BossNAS: Exploring hybrid cnn-transformers with block-wisely self-supervised neural architecture search," in *IEEE International Conference on Computer Vision*, 2021, pp. 12 261–12 271.

[41] H. Peng, H. Du, H. Yu, Q. LI, J. Liao, and J. Fu, "Cream of the Crop: Distilling prioritized paths for one-shot neural architecture search," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1–10, 2020.

[42] D. Wang, C. Gong, M. Li, Q. Liu, and V. Chandra, "AlphaNet: Improved training of supernets with alpha-divergence," in *International Conference on Machine Learning*, vol. 139, 2021, pp. 10 760–10 771.

[43] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter, "NAS-Bench-101: Towards reproducible neural architecture search," in *International Conference on Machine Learning*, vol. 97, 2019, pp. 7105–7114.

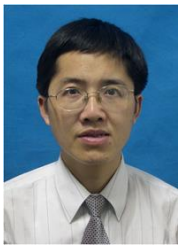
[44] X. Dong and Y. Yang, "NAS-Bench-201: Extending the scope of reproducible neural architecture search," in *International Conference on Learning Representations*, 2020, pp. 1–16.

[45] X. Dong, L. Liu, K. Musial, and B. Gabrys, "NATS-Bench: Benchmarking nas algorithms for architecture topology and size," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–14, 2021.

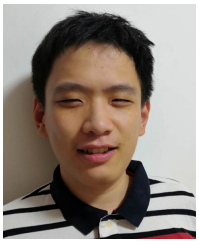
[46] X. Su, T. Huang, Y. Li, S. You, F. Wang, C. Qian, C. Zhang, and C. Xu, "Prioritized architecture sampling with monto-carlo tree search," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 968–10 977.

[47] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5987–5995.

[48] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet v2: Practical guidelines for efficient cnn architecture design," in *European Conference on Computer Vision*, 2018, pp. 116–131.
[49] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetv2: Inverted residuals and linear bottlenecks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.
[50] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," in *International Conference on Machine Learning*, vol. 80, 2018, pp. 4092–4101.
[51] S. Xie, H. Zheng, C. Liu, and L. Lin, "SNAS: stochastic neural architecture search," in *International Conference on Learning Representations*, 2018, pp. 1–17.
[52] H. Cai, L. Zhu, and S. Han, "Searching for a robust neural architecture in four gpu hours," in *International Conference on Learning Representations*, 2019, pp. 1761–1770.
[53] X. Chu, X. Wang, B. Zhang, S. Lu, X. Wei, and J. Yan, "DARTS-: robustly stepping out of performance collapse without indicators," in *International Conference on Learning Representations*, 2021, pp. 1–22.
[54] X. Dong and Y. Yang, "One-shot neural architecture search via self-evaluated template network," in *IEEE International Conference on Computer Vision*, 2019, pp. 3680–3689.
[55] H. Yu, H. Peng, Y. Huang, J. Fu, H. Du, L. Wang, and H. Ling, "Cyclic differentiable architecture search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 1, pp. 211–228, 2023.
[56] Y. Xu, L. Xie, W. Dai, X. Zhang, X. Chen, G.-J. Qi, H. Xiong, and Q. Tian, "Partially-connected neural architecture search for reduced computational redundancy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 9, pp. 2953–2970, 2021.
[57] X. Zhang, P. Hou, X. Zhang, and J. Sun, "Neural architecture search with random labels," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 907–10 916.
[58] X. Dong and Y. Yang, "ProxylessNAS: Direct neural architecture search on target task and hardware," in *International Conference on Learning Representations*, 2019, pp. 1–13.
[59] X. Chu, B. Zhang, Q. Li, R. Xu, and X. Li, "SCARLET-NAS: Bridging the gap between stability and scalability in weight-sharing neural architecture search," in *IEEE International Conference on Computer Vision Workshops*, 2021, pp. 317–325.
[60] S. You, T. Huang, M. Yang, F. Wang, C. Qian, and C. Zhang, "GreedyNAS: Towards fast one-shot NAS with greedy supernet," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1996–2005.
[61] R. Guo, C. Lin, C. Li, K. Tian, M. Sun, L. Sheng, and J. Yan, "Powering one-shot topological NAS with stabilized share-parameter proxy," in *European Conference on Computer Vision*, vol. 12359, 2020, pp. 625–641.



Zhihua Chen received the Ph.D. degree in computer science from the Shanghai Jiao Tong University, Shanghai, China, in 2006. He is currently a Full Professor with the Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai, China. His current research interests include neural network design, deep learning, and computer vision.

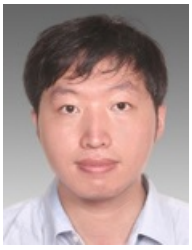


Guhao Qiu received the M.Eng. degree in computer science from the East China University of Science and Technology, Shanghai, China, in 2021. He is currently pursuing the Ph.D. degree in computer science with the Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai, China. His current research interests include deep learning, lightweight neural network design, and neural architecture search.

Ping Li (Member, IEEE) received the Ph.D. degree in computer science and engineering from The Chinese University of Hong Kong, Hong Kong, in 2013. He is currently an Assistant Professor with the Department of Computing and an Assistant Professor with the School of Design, The Hong Kong Polytechnic University, Hong Kong. He has published over 200 top-tier scholarly research articles, pioneered several new research directions, and made a series of landmark contributions in his areas. He has an excellent research project reported by the *ACM TechNews*, which only reports the top breakthrough news in computer science worldwide. More importantly, however, many of his research outcomes have strong impacts to research fields, addressing societal needs and contributed tremendously to the people concerned. His current research interests include image/video stylization, colorization, artistic rendering and synthesis, computational art, and creative media.

Lei Zhu received the Ph.D. degree in computer science and engineering from The Chinese University of Hong Kong, Hong Kong, in 2017. He is currently an Assistant Professor with The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China, and also an Affiliate Assistant Professor with The Hong Kong University of Science and Technology, Hong Kong. He was a Postdoctoral Research Associate with the University of Cambridge, U.K. His current research interests include computer graphics and computer vision.

Xiaokang Yang (Fellow, IEEE) received the B.S. degree from Xiamen University, Xiamen, China, in 1994, the M.S. degree from the Chinese Academy of Sciences, Shanghai, China, in 1997, and the Ph.D. degree from Shanghai Jiao Tong University, Shanghai, in 2000. He is currently a Distinguished Professor with the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University. From 2000 to 2002, he was a Research Fellow with the Centre for Signal Processing, Nanyang Technological University, Singapore. From 2002 to 2004, he was a Research Scientist with the Institute for Infocomm Research, Singapore. From 2007 to 2008, he visited the Institute for Computer Science, University of Freiburg, Freiburg im Breisgau, Germany, as an Alexander von Humboldt Research Fellow. He has published over 200 refereed articles. He has filed 60 patents. His current research interests include image processing and communication, computer vision, and machine learning. He is a member of Asia Pacific Signal and Information Processing Association, the VSPC Technical Committee, the IEEE Circuits and Systems Society, and the MMSP Technical Committee, and the IEEE Signal Processing Society. He is an Associate Editor of the IEEE Transactions on Multimedia and a Senior Associate Editor of the IEEE Signal Processing Letters. He was a Series Editor of Springer CCIS and an Editorial Board Member of Digital Signal Processing. He is also the Chair of the Multimedia Big Data Interest Group, MMTC Technical Committee, and the IEEE Communication Society.



Bin Sheng (Member, IEEE) received the Ph.D. degree in computer science and engineering from The Chinese University of Hong Kong, Hong Kong, in 2011. He is currently a Full Professor with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China. He is an Associate Editor of the IEEE Transactions on Circuits and Systems for Video Technology. His current research interests include virtual reality and computer graphics.