IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 2, 2021, accepted September 16, 2021, date of publication September 27, 2021, date of current version October 6, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3115911

A Feature Fusion Based Indicator for Training-Free Neural Architecture Search

LINH-TAM TRAN, MUHAMMAD SALMAN ALI, AND SUNG-HO BAE, (Member, IEEE)
Department of Computer Science and Engineering, Kyung Hee University, Yongin 17104, Republic of Korea

Corresponding author: Sung-Ho Bae (shbae@khu.ac.kr)

This work was supported by the Technology Innovation Program or the Industrial Strategic Technology Development Program funded by the Ministry of Trade, Industry and Energy (MOTIE), South Korea (Memristor Fault-Aware Neuromorphic System for 3D Memristor Array) under Grant 10085646.

ABSTRACT Neural Architecture Search Without Training (NASWOT) has been proposed recently to replace the conventional Neural Architecture Search (NAS). Pioneer works only deploy one or two indicator(s) to search. Nevertheless, the quantitative assessment for indicators is not fully studied and evaluated. In this paper, we first review several indicators, which are used to evaluate the network in a training-free manner, including the correlation of Jacobian, the output sensitivity, the number of linear regions, and the condition number of the neural tangent kernel. Our observation is that each indicator is responsible for characterizing a network in a specific aspect and there is no single indicator that achieves good performance in all cases, e.g. highly correlated with the test accuracy. This motivated us to develop a novel indicator where all properties of a network are taken into account. To obtain better indicator that can consider various characteristics of networks in a harmonized form, we propose a Fusion Indicator (FI). Specifically, the proposed FI is formed by combining multiple indicators in a weighted sum manner. We minimize the mean squared error loss between the predicted and actual accuracy of networks to acquire the weights. Moreover, as the conventional training-free NAS researches used limited metrics to evaluate the quality of indicators, we introduce more desirable metrics that can evaluate the quality of training-free NAS indicator in terms of fidelity, correlation and rank-order similarity between the predicted quality value and actual accuracy of networks. That is, we introduce the Pearson Linear Coefficient Correlation (PLCC), the Root Mean Square Error (RMSE), the Spearman Rank-Order Correlation Coefficient (SROCC), and Kendall Rank-Order Correlation Coefficient (KROCC). Extensive experiments on NAS-Bench-101 and NAS-Bench-201 demonstrate the effectiveness of our FI, outperforming existing methods by a large margin.

INDEX TERMS Neural architecture search, training-free neural architecture search, fusion indicator, evaluation metrics.

I. INTRODUCTION

Deep neural networks (DNNs) have shown remarkable performance on various computer vision tasks. Since the success of AlexNet on ImageNet [1] classification task in 2012 [2], many high-performance networks have been introduced [3]–[5] where these networks have been designed by experts. However, manual design is not an optimal choice especially when the network goes deeper. Moreover, the process for designing these networks requires immense time and effort. To reduce the cost of designing the network, researchers have studied to automate the process, leading to Neural Architecture Search (NAS). Instead of designing the architecture, experts design the search algorithms that find good candidates (e.g., the number of layers, filters and types of activation, etc.) on a given search space.

NAS is able to discover superior architecture on various computer vision tasks such as image classification [6]–[8], object detection [9], [10]. However, it suffers from several limitations. Firstly, the search cost is extremely high. It takes years of reinforcement learning (RL) [6] to search for networks which achieve state-of-the-art accuracy on large-scale dataset such as ImageNet. The most expensive part in RL approach is the training from scratch of child networks. To alleviate this limitation, subsequent works have suggested to search on smaller configuration (e.g., cell) [11], performed with shared weights [12], search on a continuous search space for NAS [7], or incorporate Bi-layer parallel training [13].

The associate editor coordinating the review of this manuscript and approving it for publication was Danilo Pelusi.

Secondly, there has a barrier to apply NAS to practical applications. That is, we need to search for a new network when the environment (e.g target task or hardware) is changed. For example, the architecture found on this dataset may not work well on others [14], or the latency for a same network is different when being executed on different hardware devices [15]. Moreover, the budget for searching is still high (e.g hours of GPU).

In order to search for best architecture in a short period time (e.g., a few minutes), NAS without training (NASWOT) [16] is introduced. Specifically, NASWOT proposed a training-free indicator that predicts the score which is highly correlated with the actual accuracy of a network, guiding the search process. Because the step to calculate this indicator does not require any training of networks, the total cost for NAS reduces significantly. With its simplicity, they have first demonstrated the possibility of performing NAS without involving any training.

Meanwhile, there have been growing works on deep learning theory that enables us to understand the behavior of DNNs [17]–[23]. TE-NAS [24] has made the first attempt to apply the indicators derived from theoretical works for revealing network characteristics, namely trainability and the expressivity, for training-free NAS. Different from [16], the search algorithm of TE-NAS is inspired by pruning-from-scratch.

Training-free NAS can replace the conventional NAS algorithms. However, prior works suffer from several limitations. [16] only uses one indicator, namely correlation Jacobian of a batch of images augmented with Cutout [25], by taking one characteristic (robustness to the input perturbation) of the network into account. As [16] uses only one indicator, it cannot represent other aspects of the network such as trainability or expressivity. So, it turns out to have limited performance in correlation between predicted score and actual accuracy. Reference [24] deals with this problem by using two indicators. However, the score (criterion for selecting the best network) is calculated with the sum of two ranks from each indicator, meaning that the two indicators have the same importance. However, this does not guarantee the optimal solution because different indicators may have different behaviors and contributions to the final scores in different importance.

To solve this problem, this paper investigate several training-free indicators and harmonizes them in a fusion framework. The FI is capable of measuring the performance of networks under various aspects. Instead of treating all indicators equally, we propose a simple training approach to find the appropriate weights for each indicator. The main contributions of the paper can be summarized as follows:

- We first collect and analyze several indicators that can be used to estimate the network's performance without training. The collected four indicators are the correlation of Jacobian (CJ), output sensitivity (OS), condition number of Neural Tangent Kernel (CNNTK), and the number of Linear Regions (NLR).
- We propose a FI which is a combination of output from multiple indicators with learned weights. The proposed indicator benefits from various properties of a network such as trainability, expressivity, generalibility and robustness against perturbation.
- We introduce new quantitative metrics to measure goodness of indicators for training-free NAS.

The rest of the paper is organized as follows. Section II introduces several related works. Section III presents the FI. Section IV demonstrates the experimental results. Section V is the conclusion of the paper.

II. RELATED WORKS

A. NEURAL ARCHITECTURE SEARCH

Neural architecture search (NAS) has attracted much attentions nowadays because of the ability to discover superior architectures automatically. However, most NAS algorithms require a huge amount of resources.

The earliest works on NAS were based on reinforcement learning [6], [11], [26], [27]. In [6], a controller was trained to generate the network's configuration which were used to construct the network. The exhaustive training and evaluation of child-network and the macro search (e.g searching the entire network) made this method unaffordable for practical applications. Particularly, the method in [6] used 800 GPUs and finished the searching phase in 28 days. To deal with this limitation, [11] searched for cells (i.e., normal cell and reduction cell). These were stacked to build the complete network. The method achieved 11.2x less search cost than [6]. Another type of NAS algorithm was based on evolutionary algorithms [28]–[30], reduced the search cost to weeks of GPUs.

Many attempts have been made to perform NAS just in a few hours [7], [12], [14], [31]. Especially ENAS [12] allowed sharing the weights among candidates. Thus, the most expensive part in the searching phase was eliminated. DARTS [7] proposed to search in a differentiable manner. During searching, a supernet was trained and the network was obtained by removing operators which has a low weight.

B. TRAINING-FREE NEURAL ARCHITECTURE SEARCH

Conventional NAS algorithms required a heavy search cost which is unaffordable to most applications especially for training the candidate networks. Therefore, if the performance of architectures is predicted without any training, the budget for NAS can be reduced significantly. Reference [16] was the first to demonstrate the feasibility of performing NAS without any training. The authors in [16] empirically found the positive correlation between the correlation of Jacobian matrix among augmented input images and the network's performance. Thus, they suggested using this indicator to score the quality of the networks. Finally, NAS in [16] was performed with a simple search strategy based on Random Search where the indicator is used to replace the training process with simple prediction of the quality in a network. This work opened a new direction

for NAS where one can utilize some indicators to estimate the network's performance. However, the major drawback of this method is that the whole framework relies on a single indicator which only captures one characteristic of a network.

TE-NAS [24] leveraged two training-free indicators, that is, NLR and CNNTK. The NLR is used to measure the expressivity of DNNs [21], [22]. The CNNTK measures the trainability of DNNs [23]. Based on these two indicators, they proposed a pruning-based algorithm to perform the search. Particularly, the criterion for pruning was based on the sum of two ranks measured by NLR and of CNNTK. The process was repeated until a certain stopping criteria was met.

In summary, the above works [16], [24] focus on performing NAS without involving any training by leveraging several training-free indicators. The search algorithm is based on random or pruning approach. Although training-free NAS has demonstrated a potential alternative for conventional NAS algorithms, naively summing the ranks in [24] does not reflect the importance of each indicator. In order to solve the aforementioned limitation, first, we consider multiple indicators where each indicator can express different characteristic of a network. Overall, there are four indicators in our Fusion method. Second, instead of assigning an equal weight for each indicator in a weighted sum manner, we adopt a training approach to find appropriate weights. This way we can highlight the important of each indicator. Furthermore, we perform extensive evaluation for the indicators in a systematic manner.

C. NAS BENCHMARKS

Reproducible and benchmarking NAS datasets are the most important factors for comparing the algorithms. There have been a lot of efforts to develop a benchmark dataset [32], [33]. Particularly, NAS-Bench-101 [32] has made the first effort to build a dataset for benchmarking NAS. There are 423k architectures in this dataset. Each architecture is trained on CIFAR-10 under the same hyper-parameters. NAS-Bench-101 provides the test accuracy of all architectures on this search space.

NAS-Bench-201 [33] extends NAS-Bench-101 [32] by adding more operators (i.e., None and skip-connect), supporting more NAS algorithms (i.e., differentiable NAS), and evaluating on more datasets (i.e., CIFAR-100 and ImageNet-16-120 [34]). The network in NAS-Bench-201 is a cell-based structure where the cell is defined as a densely-connected directed acyclic graph with 4 nodes. With this configuration, the total architecture in NAS-Bench-201 is 15625. All networks are trained under the same settings. Specifically, they are trained from scratch with Nesterov momentum SGD for 200 epochs. The initial learning rate is 0.1 and is decayed with cosine annealing. The weight decay is set to 0.0005 and the batch size is 256.

In our work, we utilise NAS-Bench-101 and NAS-Bench-201 to demonstrate the effectiveness of the proposed FI on all classification tasks in the dataset.

III. METHOD

The motivation of training-free NAS is to select high-performance potential networks from a search space without any training. In order to design a better indicator, we study several methods that can evaluate some characteristics of the network before training, i.e., CJ, OS, NLR, and CNNTK. In this section, we will first summarize these methods and then explain our proposed FI.

A. TRAINING-FREE INDICATORS

To understand the neural network (NN) behavior, we use CJ, OS, NLR, and CNNTK. These methods provide essential knowledge for characterizing NNs.

1) ROBUSTNESS AGAINST PERTURBATION VIA CJ

In order to score a network at an initial state, [16] designs CJ that computes the correlation of activations of a network with a mini-batch of n augmented images. Specifically, to compute the score, CJ first calculates the derivative of output y with respect to input x of this batch:

$$\mathrm{J} = \left(\frac{\partial y(x_1)}{\partial x_1}, \frac{\partial y(x_2)}{\partial x_2}, \ldots, \frac{\partial y(x_n)}{\partial x_n} \right). \tag{1}$$

Then we calculate the correlation for this Jacobian matrix $\sum_{\mathrm{J}}$ and count the number of entries that are smaller than a predefined threshold (β). Thus, the score for Jacobian is defined as:

$$S_{\mathrm{CJ}} = \sum_{i,j} \mathbb{1} \left(0 < (\sum_{\mathrm{J}})_{i,j} < \beta \right), \tag{2}$$

where $\mathbb{1}$ is the indicator function.

2) GENERALIZATION VIA OS

For improving the network's generalization ability, authors in [35] proposed an ensemble approach called OS that can estimate the degree of generalization power of a network. Forouzesh *et al.* [17] further extended the results of [35] where the goal is to study the relation between the sensitivity and generalization in NNs. For determining the sensitivity of the network, external noise is added to the input. Let x be the input vector, θ be the parameters of a NN f_θ, and ε be the noise which is sampled from a uniform distribution, then the error err is defined as:

$$err_y = f_\theta(x + \varepsilon) - f_\theta(x). \tag{3}$$

The averaged error is calculated as below:

$$\overline{err_y} = \frac{1}{N} \sum_{n=1}^{N} err_y^n, \tag{4}$$

where N is the number of classes and err_y^n indicates the n-th value of err_y. The sensitivity of an NN can be measured by computing the variance of the output error. To this end, the score of sensitivity is formulated as:

$$S_{\mathrm{OS}} = Var[\mathbf{X}], \tag{5}$$

where $\mathbf{X}$ is the vector of averaged error for M samples and *Var* is the variance.

3) EXPRESSIVITY OF NEURAL NETWORK VIA NLR

To answer why deep networks outperform a shallow network, [36] analyzes the deep ReLU networks with respect to their complexity. They have shown that, given the same computational resources, a deep network can divide the input space into many regions than a shallow one. Motivated by [21], [36] provided an in-depth analysis on NLR for convolution neural networks (CNNs). Following [24], NLR can be used to measure the expressivity of NNs. Let $\mathcal{N}$ be a ReLU CNN and θ be the parameters of $\mathcal{N}$ sampled from some distributions. The score for NLR can then be calculated as:

$$S_{\text{NLR}} = \#\{R(P;\theta) : R(P;\theta) \neq \emptyset \text{ for some } P\}, \tag{6}$$

where $R(\cdot)$ is the region corresponding to P and θ, and is defined as:

$$R(P;\theta) := \{x^0 \in \mathbb{R}^{C\times H\times W} : z(x^0;\theta)\cdot P(z) > 0, \forall z \in \mathcal{N}\}, \tag{7}$$

where $z(x^0;\theta)$ is the pre-activation of a neuron z and P is an activation pattern such that $P(z) \in \{-1, 1\}$ for each neuron z in $\mathcal{N}$.

4) TRAINABILITY OF NEURAL NETWORKS VIA CNNTK

In [20] a new tool was proposed to help understand the behavior of DNNs during training, which is called NTK. It has been proved that using NTK, we can obtain the time evolution of linearized NNs at time t without running gradient descent:

$$\mu_t(X_{\text{train}}) = (\mathbf{Id} - e^{-\eta\hat{\Theta}_{\text{train,train}}t})Y_{\text{train}}, \tag{8}$$

where $\mu_t(x) = \mathbb{E}[z_i^L(x)]$ is the expected outputs of infinitely wide network, z_i^L is the output of i-th neuron in the last layer L, η is the learning rate, and $\hat{\Theta}_{\text{train,train}}$ is the NTK between two training inputs. X_{train} and Y_{train} are the input and target which are drawn from the training set. $\mathbf{Id}$ is a constant. The trainability of NNs is studied in [23]. Let λ_i be the i-th eigenvalue in the D diagonal matrix and U be the unitary matrix of $\hat{\Theta}_{\text{train,train}}$, i.e., $\hat{\Theta}_{\text{train,train}} = UDU^{-1}$. Then Eq. (8) can be rewritten as:

$$\tilde{\mu}_t(X_{\text{train}})_i = (\mathbf{Id} - e^{-\eta\lambda_i t})\tilde{Y}_{\text{train},i}, \tag{9}$$

where $\tilde{\mu}_t(X_{\text{train}})_i = U\mu_t(X_{\text{train}})$ and $\tilde{Y}_{\text{train},i} = UY_{\text{train}}$.

Let λ_0 and λ_m be the minimum and maximum eigenvalues of $\hat{\Theta}_{\text{train,train}}$. In Eq. (9), the maximum learning rate scales as $\eta \sim 2/\lambda_0$ in [37]. Thus, the smallest eigenvalue will converge exponentially at a rate given by $1/k$, where $k = \lambda_0/\lambda_m$ and is the condition number. If the condition number of the NTK diverges, the network is untrainable. In our work, we inverse the condition number of NTK such that with a higher value, we can get better trainability. Thus, the score for trainability can be written as:

$$S_{\text{CNNTK}} = \frac{1}{k}. \tag{10}$$

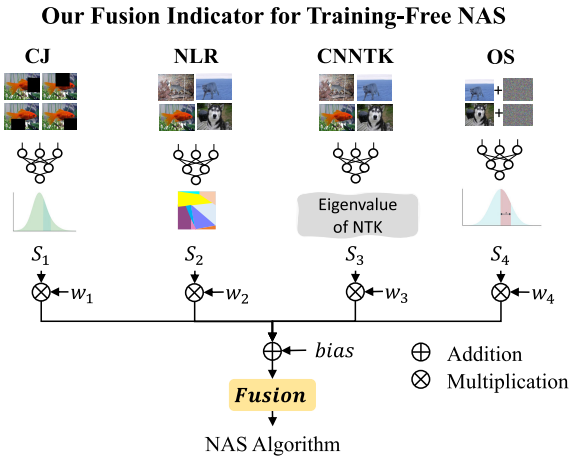

FIGURE 1. The illustration of the proposed fusion indicator. CJ, NLR, CNNTK, OS are the four training-free indicators. Each indicator represents a single characteristic of a network. In the fusion indicator, each standalone indicator contributes a certain property for ranking the neural network.

B. FUSION INDICATOR

In order to evaluate NNs without any training, the process requires a powerful indicator for correctly characterizing the network. Each indicator has its purpose and significance in specifying our model. Using these indicators separately makes them ineffective and weak since they only characterize the network with a specific factor hence inculcating a certain bias in the network. Thus, this motivates us to design the FI where all indicators are combined. Let $\mathcal{S}$ be the set of training free indicators (e.g., $S_{\text{CJ}}, S_{\text{NLR}}, S_{\text{CNNTK}}, S_{\text{OS}}$), our score for FI is defined as:

$$S_{\text{FI}} = \sum_{i=1}^{n} w_i S_i + b, \tag{11}$$

where n is the total indicators, b is the bias, w_i and S_i are the weight and the score using i-th indicator in $\mathcal{S}$. The overview of the proposed Fusion Indicator is demonstrated in Figure 1. We hypothesize that our proposed FI can take all indicators' advantage and make our training-free framework more reliable. We want our FI to reflect our network's accuracy closely. We are interested in minimizing the difference between the score and the accuracy. Specifically, we minimize the following loss function:

$$\mathcal{L} = \text{MSE}(S_{\text{FI}}, Acc), \tag{12}$$

where MSE is the mean squared error. S_{FI} and Acc in Eq. (12) denote the score and the accuracy of a network. Thus, our S_{FI} is the predicted accuracy of a network. The model parameters in Eq. (11) are trained with the stochastic gradient descent (SGD) method. Note that it is prohibitive to use support vector regression or linear regression in this case

since there are a huge numbers of samples in NAS-Bench-101/202 datasets.

IV. EXPERIMENTAL RESULTS

A. TRAINING-FREE INDICATOR ASSESSMENT

We use four metrics to evaluate the performance of the indicator. The first two metrics are the Spearman Rank-Order Correlation Coefficient (SROCC) and the Kendall Rank-Order Correlation Coefficient (KROCC). The third metric and the fourth metric are the Pearson Linear Correlation Coefficient (PLCC) and the Root Mean Square Error (RMSE) between the score and the accuracy after nonlinear regression. The first two metrics measure the prediction monotonicity of the indicator while the third one measures the linear correlation between the actual accuracy and the predicted one. These quality assessment metrics are made for evaluating such characteristics (fidelity and correlation) and are widely used in practice [38]–[40]. To compute PLCC and RMSE, we apply the following logistic function as suggested in [41]:

$$f(x) = \beta_1 \left(\frac{1}{2} - \frac{1}{1 + e^{\beta_2 (x - \beta_3)}} \right) + \beta_4 x + \beta_5, \qquad (13)$$

where x is the score (predicted accuracy) and $\beta_i (i = 1, 2, 3, 4, 5)$ are the set of parameters that minimize the least squares error between the output from indicator and the network's accuracy.

Since our FI is a learning-based method, the optimal w may not generalize well on other test data. To prevent overfitting, we use n-fold cross-validation. Specifically, we use 5-fold cross-validation and average the results obtained from 5 folds to get the overall performance. In all experiments, we run the 5-fold cross-validation 10 times and average the results to obtain the final performance.

B. RESULTS ON NAS-BENCH-101

We compare the performance of our FI and standalone indicators, i.e. CJ, NLR, CNNTK, OS on CIFAR-10 of NAS-Bench-101 search space. Because we have four indicators, there are 11 combinations for our FI. FI_2 means two indicators are used and so on. We use {CJ, OS} for FI_2, {CJ, CNNTK, OS} for FI_3, and {CJ, NLR, CNNTK, OS} for FI_4. We refer the reader to the Ablation section for the performance of other combinations. The results are shown in Table 1.

As shown in Table 1, the proposed FI outperforms other indicators significantly in all performance assessment methods. Notably, our FI achieves 5% higher KROCC and SROCC than the CJ-based single indicator [16]. Compared to NLR, CNNTK and OS, the proposed FI has around 24% and 30% higher KROCC and SROCC, respectively. Additionally, it is worth noting that increasing the number of indicator does not improve the performance on NAS-Bench-101. One possible reason for this is that there are only three operations (i.e., 3×3 convolution, 1×1 convolution, 3×3 max pool) in the search space. This reduces the diversity of the network architectures even though NAS-Bench-101 contains more architectures than NAS-Bench-201. As a result, several indicators do not contribute to the overall performance.

TABLE 1. Performance of several training-free indicators measured on CIFAR-10 of NAS-Bench-101. The first, second, and third ranked performances are highlighted in blue, red, and black bold, respectively.

Indicator	PLCC	RMSE	SROCC	KROCC
CJ	0.4882	0.0355	0.6423	0.4658
NLR	0.2371	0.0395	0.3887	0.2699
CNNTK	0.2506	0.0394	0.4045	0.2788
OS	0.2238	0.0396	0.3889	0.2664
FI_2	**0.4935**	**0.0353**	**0.6961**	**0.5140**
FI_3	**0.4931**	**0.0353**	**0.6956**	**0.5136**
FI_4	**0.4922**	**0.0353**	**0.6947**	**0.5128**

C. RESULTS ON NAS-BENCH-201

We further evaluate the performance of training-free indicators on NAS-Bench-201. We use {CNNTK, OS} for FI_2, {NLR, CNNTK, OS} for FI_3, and all indicators for FI_4. The performance measured by PLCC, RMSE, SROCC, and KROCC for all competitors on this dataset are shown in Table 2.

From Table 2, it can be seen that the proposed FI outperforms standalone indicators in all performance assessment methods for all datasets on NAS-Bench-201. It is worth noting that OS performs the best compared to CJ, NLR, and CNNTK. On CIFAR-10, FI_2 has 7.29% and 8.58% higher SROCC and KROCC than OS. Adding more indicators to our FI further increases the performance. Specifically, FI_3 has 7.43% and 8.85% SROCC and KROCC improvement over OS. When using all indicators, FI_4 reaches its peak with 0.88 and 0.7017 SROCC and KROCC, respectively.

On CIFAR-100, our FI performs consistently well. Standalone indicator achieves less than 0.8 SROCC and 0.6 KROCC. By contrast, the proposed FI obtains more than 0.86 SROCC and 0.68 KROCC. On ImageNet-16-120, we can see that FI_4 has the best performance for all assessment metrics. Compared to OS, FI_4 has 9% and 11% higher SROCC and KROCC.

Additionally, we illustrate the scatter plots of the predicted scores versus the accuracy measured by CJ, NLR, CNNTK, and OS in Figure 2. We show the test accuracy with respect to the score of CJ, NLR, CNNTK, and OS in the first, second, third, and fourth column of Figure 2, respectively. We also show the scatter plots of FI_4 for 5-fold cross-validation (CV) in Figure 3. Each column in Figure 3 represents the test accuracy with respect to the score of each fold, namely Fold-1, Fold-2, Fold-3, Fold-4, and Fold-5 in CV. In both figures, the green line is the best fitting curve. Figure 2 demonstrates that the OS indicator has a higher correlation than others. From Figure 3, we can see that there is a very strong correlation between the predicted score and the accuracy when using the proposed FI.

In summary, the performance of our FI improves consistently when adding more indicators. This may come from the diversity of the search space. Besides the three operations

TABLE 2. Performance of several training-free indicators measured on CIFAR-10, CIFAR-100, and ImageNet-16-120 of NAS-Bench-201 search space. The first, second, and third ranked performances are highlighted in blue, red, and black bold, respectively.

Indicator	CIFAR-10				CIFAR-100				ImageNet-16-120			
	PLCC	RMSE	SROCC	KROCC	PLCC	RMSE	SROCC	KROCC	PLCC	RMSE	SROCC	KROCC
CJ	0.5146	0.1109	0.7400	0.5280	0.5918	0.0981	0.7420	0.5308	0.7839	0.0591	0.7161	0.5288
NLR	0.7997	0.0777	0.7107	0.5245	0.8002	0.0730	0.7196	0.5295	0.8180	0.0547	0.7246	0.5378
CNNTK	0.9534	0.0390	0.7308	0.5389	0.8963	0.0540	0.7447	0.5542	0.7452	0.0634	0.6026	0.4279
OS	0.9466	0.0417	0.8027	0.6109	0.8729	0.0594	0.7856	0.5976	0.8362	0.0522	0.7705	0.5834
FI_2	**0.9587**	**0.0363**	**0.8756**	**0.6965**	**0.9195**	**0.0477**	**0.8687**	**0.6867**	**0.9039**	**0.0406**	**0.8494**	**0.6753**
FI_3	**0.9613**	**0.0354**	**0.8770**	**0.6992**	**0.9220**	**0.0470**	**0.8731**	**0.6940**	**0.9133**	**0.0386**	**0.8481**	**0.6744**
FI_4	**0.9596**	**0.0362**	**0.8800**	**0.7017**	**0.9209**	**0.0473**	**0.8775**	**0.6987**	**0.9185**	**0.0375**	**0.8607**	**0.6935**

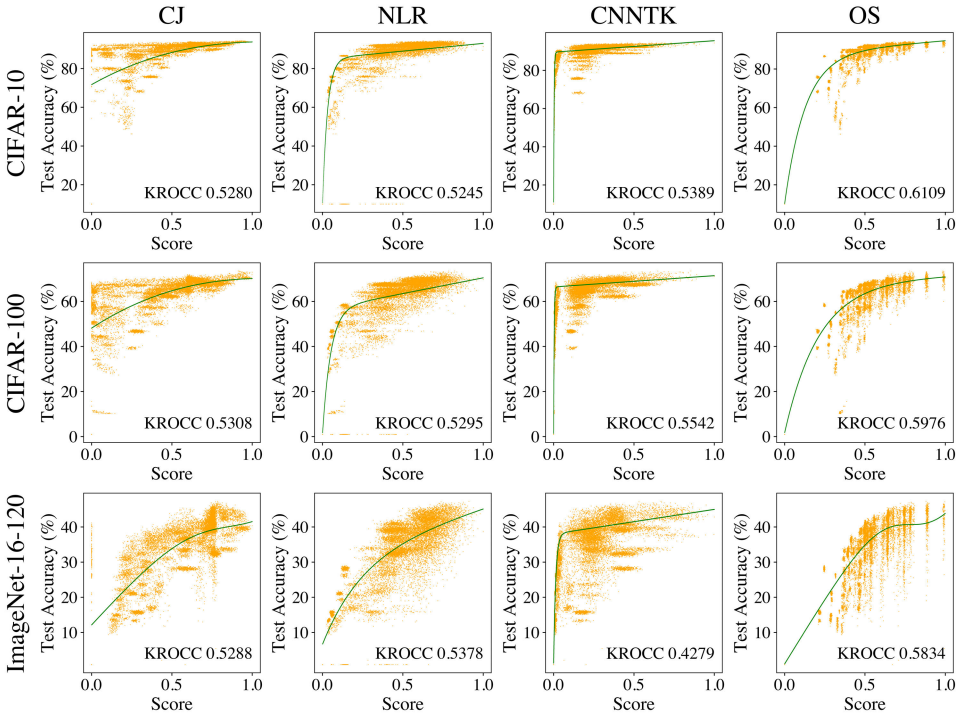

FIGURE 2. The plots of the score for all architectures in NAS-Bench-201 against the test accuracies on CIFAR-10, CIFAR-100, and ImageNet-16-120. For better visualization, the score is scaled to the range of 0 and 1. The best fitting curve is shown in green line. The Kendall Tau values show a strong correlation for all indicators.

used in [32], NAS-Bench-201 uses two more operators, zero and skip-connect, which makes the search space richer. Thus, the proposed FI can fully utilize all characteristics of a network to evaluate the network correctly.

D. CROSS DATASETS TEST FOR FI

We perform cross dataset tests to verify the generalizability of our FI for different datasets. To do this, we obtain the weights for the FI trained on one dataset (e.g., CIFAR-10) and evaluate it on other datasets. For example, we use the score evaluated on CIFAR-10 to train the weight for the FI and use the same weights to perform testing on CIFAR-100 and ImageNet-16-120. We denote the weight obtained from CIFAR-10, CIFAR-100, and ImageNet-16-120 as $\mathbf{w}_{\text{CIFAR-10}}$, $\mathbf{w}_{\text{CIFAR-100}}$, and $\mathbf{w}_{\text{ImageNet-16-120}}$, respectively. The performance results are listed in Table 3.

TABLE 3. Performance of FI_4 on NAS-Bench-201, measured by KROCC.

Weight	CIFAR-10	CIFAR-100	ImageNet-16-120
$\mathbf{w}_{\text{CIFAR-10}}$	-	0.7099	0.6904
$\mathbf{w}_{\text{CIFAR-100}}$	0.6855	-	0.6789
$\mathbf{w}_{\text{ImageNet-16-120}}$	0.6712	0.6871	-

As shown in Table 3, we can see that the proposed FI achieves a high KROCC. For all datasets, the value of KROCC is greater than 0.67. The weights obtained on one dataset are highly compatible with other datasets. This shows the generality and robustness of our approach.

E. IMPROVING NAS WITHOUT TRAINING USING FUSION INDICATOR

The ultimate goal of NAS without Training is to replace the heavy cost of training candidate networks with inexpensive

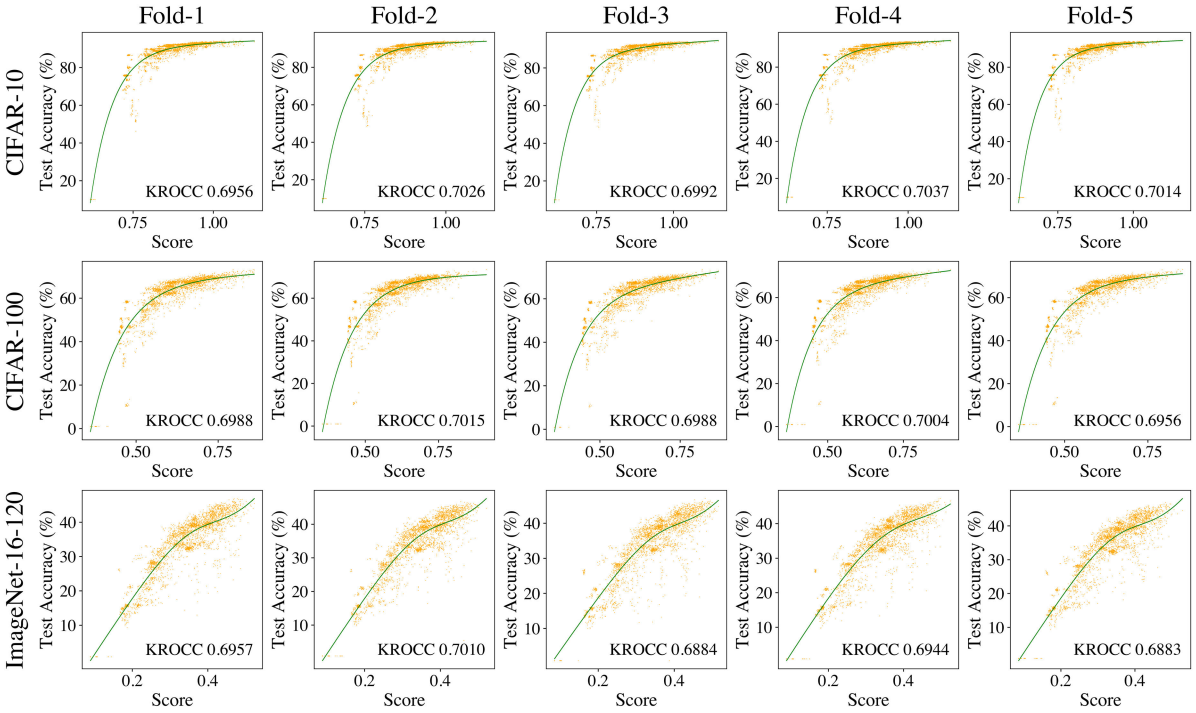

FIGURE 3. Scatter plots of predicted score, FI_4, against the accuracy for each fold on CIFAR-10, CIFAR-100, and ImageNet-16-120 of NAS-Bench-201 search space. The best fitting curve is shown in green line.

ones (e.g., CJ, NLR…). In order to demonstrate the effectiveness of training-free indicators, we incorporate the score from training-free indicator into Aging Evolution (AE), an evolutionary algorithm for NAS [42]. We replace the network accuracy obtained from the network training step (**train and evaluation** in [42]) with the **predicted score** from the indicator. For the mutation phase, the network which has the highest predicted score is selected as a parent. After mutation, the child network is scored using the indicator. Finally, the output is the network which has the highest predicted score. We choose the AE algorithm due to its simplicity.

We compare the performance of AE using CJ, NLR, CNNTK, OS, and the proposed FI_4 on CIFAR-100 with NAS-Bench-201 search space. We evaluate around 300 networks. We run the experiment 100 times and show the average results in Figure 4. As demonstrated in Figure 4, it is clear that the proposed FI achieves higher test accuracy for the network with the highest predicted score than others. Specifically, NLR has the lowest test accuracy for the network with the highest predicted score. The three indicators CJ, CNNTK, and OS achieve comparable test accuracy. It is noticeable that the proposed FI achieves much higher accuracy than others, where we perform experiments 100 times for each case and take an average of the 100 test accuracies.

F. ABLATION STUDIES

In this section, we study the behavior of the training-free indicator. We investigate how weight initialization affects performance. We also study the behavior of the proposed FI when more training-free indicators are added.

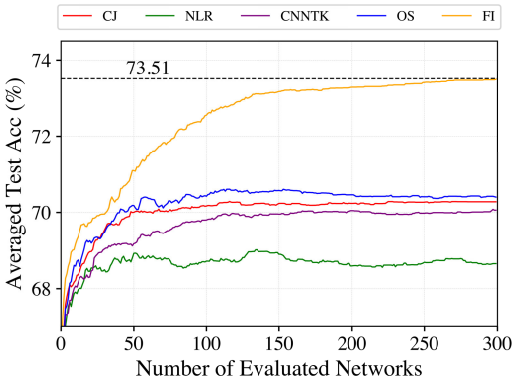

FIGURE 4. Performance of Aging Evolution using different training-free indicators on CIFAR-100 with NAS-Bench-201 search space. The black dot horizontal line is the best test accuracy on this benchmark.

1) HOW WEIGHT INITIALIZATION METHODS EFFECT PERFORMANCE?

When we develop an indicator for training-free NAS framework, the most critical factor is how robust our indicator is. At the initial state, our network is unstable. There are several initialization methodologies for a network. However, in this study, we assess the performance of training-free indicators using the following initialization methods:

- **Uniform distribution:** For uniform distribution, the weights are initialized from $\mathcal{U}(-\sqrt{k}, \sqrt{k})$, where $k = groups/(C_{in} * \prod_{i=0}^{1} k[i])$.
- **Normal distribution:** The weights are sampled from $\mathcal{N}(0, std^2)$. For OS indicator, we only use the standard

TABLE 4. KROCC of indicators with different weight sampling methods on CIFAR-10 of NAS-Bench-201.

Indicator	Uniform	Kaiming	Normal
CJ	**0.5208**	0.4080	0.3263
NLR	**0.5245**	0.5220	0.5190
CNNTK	**0.5389**	0.5040	0.4373
OS	-	-	**0.6109**

normal distribution without any batch normalization as mentioned in [17].

- **Kaiming He:** Following [43], the weights are sampled from $\mathcal{N}(0, std^2)$, where $std = gain/fanmode^{1/2}$.

We compute KROCC for each indicator with different weight sampling methods. Table 4 demonstrates the quantitative performance of each indicator for different weight sampling methods. As shown in Table 4, the performance of training-free indicators depends on how the weights are sampled, especially for CJ. One possible explanation for this is that the hypothesis of the Jacobian indicator requires the same local linear operators, where the linear maps are the Jacobian of augmented input x, should have low correlation. This means that if any weight initialization method violates the hypothesis, the Jacobian indicator will not work (e.g., strong correlation with different augmented input x).

To avoid exhaustively computing all possible weight sampling combinations for FI, we use a simple technique that selects the best sampling method for each indicator. Thus, uniform distribution is used for computing CJ, NLR, CNNTK and normal distribution is used for OS.

2) NUMBER OF INDICATORS FOR FI

We also investigate the performance of our FI by combining different indicators on CIFAR-10. There are 11 combinations in total. The weights are sampled following the above analysis. KROCC is used to measure performance. We list the results in Table 5.

From Table 5, it can be seen that on NAS-Bench-101 combining indicators further improves the performance. For example, NLR, CNNTK, OS achieves around 0.27 KROCC but combining them enhances the performance to 0.30 $\sim$ 0.34 KROCC. The best combination is {CJ, OS}.

On NAS-Bench-201, we observe that when two indicators are used, the performance measured by KROCC has a high standard deviation (e.g., the value of KROCC is ranging from 0.5610 to 0.6965). On a search space that has higher diversity (e.g., more operators), fusing more indicators helps improve the performance of the network. Specifically, our best KROCC increases from 0.6965 to 0.6993 and 0.7016 when three, and four indicators are used respectively. We find that when using two indicators, the combination of CNNTK and OS outperforms other combinations. It is natural since CNNTK and OS are the top-two standalone indicator. The overall performance increases when more indicators are used. The behaviour is consistent for all quality assessment methods. We can see that the best correlation is obtained when all indicators are used. Figure 5 shows the trend of

TABLE 5. Comparison with different combination of indicators for FI on CIFAR-10, measured by KROCC.

Indicator	NAS-Bench-201	NAS-Bench-101
CJ, NLR	0.6212	0.4685
CJ, CNNTK	0.5610	0.4651
CJ, OS	0.6583	**0.5140**
NLR, CNNTK	0.6312	0.3157
NLR, OS	0.6161	0.3025
CNNTK, OS	**0.6965**	0.3477
CJ, NLR, CNNTK	0.6492	0.4678
CJ, NLR, OS	0.6629	0.5136
CJ, CNNTK, OS	0.6990	**0.5137**
NLR, CNNTK, OS	**0.6993**	0.3463
CJ, NLR, CNNTK, OS	**0.7016**	**0.5128**

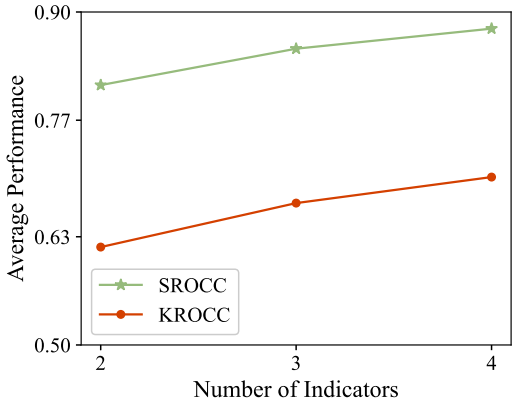

FIGURE 5. Overall comparison for different combination of indicators measured on CIFAR-10, NAS-Bench-201.

increasing the number of indicators for FI on NAS-Bench-201 improves the performance.

For comparing the difference in performance, we conduct the statistical tests between the proposed Fusion Indicator and the standalone one. We first define the absolute difference values between the predicted score $\hat{y}$ (after nonlinear regression) and the actual accuracy y as $\Delta = |y - \hat{y}|$. The first test verifies the normality of Δ and the second test determines whether the Δ from one indicator are statistically indistinguishable from another indicator. The significance level is 5% for both tests. We use Shapiro-Wilk test for normality and the results (p-value) are shown in Table 6. For the second test, Wilcoxon rank-sum is performed because there is no normal distribution case as in Table 6. The results for the second test are shown in Table 7. In most cases, the indicators are statistically different from each other, except for CNNTK versus NLR on NAS-Bench-101, which is not different. In general, the proposed FI statistically improves the performance.

G. THREAT TO VALIDITY

There can be a few factors that threaten the validity of this research. We briefly list several potential threats:

- **Search space:** There are several search spaces which are used in other NAS algorithms such as AmoebaNet [42],

TABLE 6. The results from Shapiro-Wilk tests on CIFAR-10.

Indicator	p-value NAS-Bench-201	p-value NAS-Bench-101
CJ	0	0
NLR	0	0
CNNTK	0	0
OS	0	0
FI_2	0	0
FI_3	0	0
FI_4	0	0

If p-value>0.05, normal distribution

TABLE 7. The statistical significance matrix on CIFAR-10 with 95% confidence. Each element in the table is a codeword for 2 symbols. The first and second position in the symbol indicate the result of the hypothesis test on NAS-Bench-201 and NAS-Bench-101.

Indicator	CJ	NLR	CNNTK	OS
CJ	- -	11	11	11
NLR	11	- -	10	11
CNNTK	11	10	- -	11
OS	11	11	11	- -
FI_2	11	11	11	11
FI_3	11	11	11	11
FI_4	11	11	11	11

0 means the indicator on the row is not significantly different than the indicator on the column.
1 means the indicator on the row is significantly different than the indicator on the column.

FBNet [44]. The search space of NAS-Bench-101/201 is much smaller than the aforementioned search space.

- **Dataset:** Current NAS benchmark only uses small dataset such as CIFAR-10, CIFAR-100, and reduced ImageNet-16-120. Thus, it is common to ask whether the training-free indicator works well on large-scale datasets (e.g., ImageNet [1]).
- **Bias in the experiment:** Since the training-free indicator evaluates the network before training, the score is affected by the value of the network's parameters.

Regarding the first and the second threats, there is no benchmark dataset for these search spaces at the current state. It is because training on a large-scale dataset requires a lot of computational costs, as mentioned in [33]. Due to this reason, most NAS benchmark datasets use smaller search spaces and train the networks on a small dataset such as CIFAR-10. However, we will validate the performance of training-free indicators as well as the Fusion indicator once they are available. In order to counter the last threat, we conduct the experiments multiple times (i.e., 10 times) with different random seeds and take the average as the final results. This mitigates the influence of random weights on the results. Extensive experiments on two recently released NAS-Bench-101/201 confirm that the Fusion Indicator brings many benefits in ranking the networks.

V. CONCLUSION

In this paper, we proposed a simple yet effective method for training-free NAS, called FI. The core motivation of our work is to harmonize several characteristics, including correlation of Jacobian, the number of linear regions, the condition number of the neural tangent kernel, and the output sensitivity of a network in a weighted sum manner. Through extensive experiments, our work demonstrated that the proposed FI achieves a higher correlation between the predicted score and the network's accuracy than standalone indicator in all quality assessment methods. The best combination on NAS-Bench-101 and NAS-Bench-201 is {CJ, OS} and {CJ, NLR, CNNTK, OS}, respectively. The large search space such as NAS-Bench-201 can fully use the characteristics effectively because of the diversity in their search space. Interestingly, we find that fusing only two indicators (e.g., {CNNTK, OS} on NAS-Bench-201) achieved comparable performance to fuse four indicators.

We encourage the researchers in this field to discover more characteristics of a network so that we can develop a powerful training-free indicator. The proposed FI can be applied to query-based NAS algorithms such as random search or evolutionary search easily. We plan to develop an efficient search method that uses the proposed FI in our future work.

REFERENCES

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Red Hook, NY, USA: Curran Associates, 2012, pp. 1097–1105.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015, *arXiv:1512.03385*. [Online]. Available: https://arxiv.org/abs/1512.03385

[4] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4700–4708.

[5] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*. [Online]. Available: https://arxiv.org/abs/1704.04861

[6] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," 2016, *arXiv:1611.01578*. [Online]. Available: https://arxiv.org/abs/1611.01578

[7] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–12.

[8] 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 *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2820–2828.

[9] B. Chen, G. Ghiasi, H. Liu, T.-Y. Lin, D. Kalenichenko, H. Adam, and Q. V. Le, "MnasFPN: Learning latency-aware pyramid architecture for object detection on mobile devices," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 2820–2828.

[10] N. Wang, Y. Gao, H. Chen, P. Wang, Z. Tian, C. Shen, and Y. Zhang, "NAS-FCOS: Fast neural architecture search for object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11943–11951.

[11] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," 2017, *arXiv:1707.07012*. [Online]. Available: https://arxiv.org/abs/1707.07012

[12] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," 2018, *arXiv:1802.03268*. [Online]. Available: https://arxiv.org/abs/1802.03268

[13] J. Chen, K. Li, K. Bilal, X. Zhou, K. Li, and P. S. Yu, "A bi-layered parallel training architecture for large-scale convolutional neural networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 5, pp. 965–976, May 2019.

[14] Y. Xu, L. Xie, X. Zhang, X. Chen, G. Qi, Q. Tian, and H. Xiong, "PC-DARTS: Partial channel connections for memory-efficient differentiable architecture search," 2019, *arXiv:1907.05737*. [Online]. Available: https://arxiv.org/abs/1907.05737

[15] C. Li, Z. Yu, Y. Fu, Y. Zhang, Y. Zhao, H. You, Q. Yu, Y. Wang, C. Hao, and Y. Lin, "HW-NAS-Bench: Hardware-aware neural architecture search benchmark," in *Proc. Int. Conf. Learn. Represent.*, 2021. [Online]. Available: https://openreview.net/forum?id=_0kaDkv3dVf

[16] J. Mellor, J. Turner, A. Storkey, and E. J. Crowley, "Neural architecture search without training," 2020, *arXiv:2006.04647*. [Online]. Available: https://arxiv.org/abs/2006.04647

[17] M. Forouzesh, F. Salehi, and P. Thiran, "Generalization comparison of deep neural networks via output sensitivity," in *Proc. 25th Int. Conf. Pattern Recognit. (ICPR)*, Jan. 2021, pp. 7411–7418.

[18] R. Novak, Y. Bahri, D. A. Abolafia, J. Pennington, and J. Sohl-Dickstein, "Sensitivity and generalization in neural networks: An empirical study," 2018, *arXiv:1802.08760*. [Online]. Available: https://arxiv.org/abs/1802.08760

[19] K. Kawaguchi, L. P. Kaelbling, and Y. Bengio, "Generalization in deep learning," 2017, *arXiv:1710.05468*. [Online]. Available: https://arxiv.org/abs/1710.05468

[20] A. Jacot, F. Gabriel, and C. Hongler, "Neural tangent kernel: Convergence and generalization in neural networks," 2018, *arXiv:1806.07572*. [Online]. Available: https://arxiv.org/abs/1806.07572

[21] H. Xiong, L. Huang, M. Yu, L. Liu, F. Zhu, and L. Shao, "On the number of linear regions of convolutional neural networks," in *Proc. 37th Int. Conf. Mach. Learn.*, vol. 119, H. D. III and A. Singh, Eds. Jul. 2020, pp. 10514–10523.

[22] M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. Sohl-Dickstein, "On the expressive power of deep neural networks," 2017, *arXiv:1606.05336*. [Online]. Available: https://arxiv.org/abs/1606.05336

[23] L. Xiao, J. Pennington, and S. S. Schoenholz, "Disentangling trainability and generalization in deep learning," 2019, *arXiv:1912.13053*. [Online]. Available: https://arxiv.org/abs/1912.13053

[24] W. Chen, X. Gong, and Z. Wang, "Neural architecture search on ImageNet in four GPU hours: A theoretically inspired perspective," in *Proc. Int. Conf. Learn. Represent.*, 2021. [Online]. Available: https://openreview.net/forum?id=Cnon5ezMHtu

[25] T. Devries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," 2017, *arXiv:1708.04552*. [Online]. Available: https://arxiv.org/abs/1708.04552

[26] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," 2016, *arXiv:1611.02167*. [Online]. Available: https://arxiv.org/abs/1611.02167

[27] Z. Zhong, J. Yan, W. Wu, J. Shao, and C.-L. Liu, "Practical block-wise neural network architecture generation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2423–2432.

[28] L. Xie and A. Yuille, "Genetic CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1379–1388.

[29] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin, "Large-scale evolution of image classifiers," in *Proc. 34th Int. Conf. Mach. Learn. (ICML)*, Sydney, NSW, Australia, vol. 70, Aug. 2017, pp. 2902–2911.

[30] T. Elsken, J. H. Metzen, and F. Hutter, "Efficient multi-objective neural architecture search via Lamarckian evolution," in *Proc. Int. Conf. Learn. Represent.*, Sep. 2019. [Online]. Available: https://openreview.net/forum?id=ByME42AqK7

[31] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation," 2019, *arXiv:1904.12760*. [Online]. Available: https://arxiv.org/abs/1904.12760

[32] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter, "Nas-bench-101: Towards reproducible neural architecture search," in *Proc. ICML*, 2019, pp. 7105–7114.

[33] X. Dong and Y. Yang, "NAS-Bench-201: Extending the scope of reproducible neural architecture search," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 2–17.

[34] P. Chrabaszcz, I. Loshchilov, and F. Hutter, "A downsampled variant of ImageNet as an alternative to the CIFAR datasets," 2017, *arXiv:1707.08819*. [Online]. Available: https://arxiv.org/abs/1707.08819

[35] J. Yang, X. Zeng, S. Zhong, and S. Wu, "Effective neural network ensemble approach for improving generalization performance," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 6, pp. 878–887, Jun. 2013.

[36] R. Pascanu, G. Montufar, and Y. Bengio, "On the number of response regions of deep feed forward networks with piece-wise linear activations," 2014, *arXiv:1312.6098*. [Online]. Available: https://arxiv.org/abs/1312.6098

[37] J. Lee, L. Xiao, S. S. Schoenholz, Y. Bahri, R. Novak, J. Sohl-Dickstein, and J. Pennington, "Wide neural networks of any depth evolve as linear models under gradient descent," *J. Stat. Mech. Theory Exp.*, vol. 2020, no. 12, Dec. 2020, Art. no. 124002.

[38] L. Zhang, Y. Shen, and H. Li, "VSI: A visual saliency-induced index for perceptual image quality assessment," *IEEE Trans. Image Process.*, vol. 23, no. 10, pp. 4270–4281, Aug. 2014.

[39] S.-H. Bae and M. Kim, "DCT-QM: A DCT-based quality degradation metric for image quality optimization problems," *IEEE Trans. Image Process.*, vol. 25, no. 10, pp. 4916–4930, Oct. 2016.

[40] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[41] H. R. Sheikh, M. F. Sabir, and A. C. Bovik, "A statistical evaluation of recent full reference image quality assessment algorithms," *IEEE Trans. Image Process.*, vol. 15, no. 11, pp. 3440–3451, Nov. 2006.

[42] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, Jul. 2019, pp. 4780–4789.

[43] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," 2015, *arXiv:1502.01852*. [Online]. Available: https://arxiv.org/abs/1502.01852

[44] B. Wu, K. Keutzer, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, and Y. Jia, "FBNet: Hardware-aware efficient ConvNet design via differentiable neural architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 10734–10742.

LINH-TAM TRAN received the bachelor's degree from the Department of Computer Science and Engineering, Ho Chi Minh City University of Technology, Vietnam, in 2014, and the M.S. degree from Hongik University, Seoul, South Korea, in 2018. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, Kyung Hee University, Suwon, South Korea. His research interest includes neural architecture search for practical applications.

MUHAMMAD SALMAN ALI received the bachelor's degree in computer science from the National University of Sciences and Technology (NUST), Islamabad, Pakistan, in 2018. He is currently pursuing the M.S. degree leading to the Ph.D. degree with Kyung Hee University, South Korea. His research interests include deep learning interpretation and the effect of soft errors on deep neural networks. He was a recipient of the gold medal for being a high-achiever during his UG studies.

SUNG-HO BAE (Member, IEEE) received the B.S. degree from Kyung Hee University, South Korea, in 2011, and the M.S. and Ph.D. degrees from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2012 and 2016, respectively. From 2016 to 2017, he was a Postdoctoral Associate with the Computer Science and Artificial Intelligence Laboratory (CSAIL), Massachusetts Institute of Technology (MIT), MA, USA. Since 2017, he has been an Assistant Professor with the Department of Computer Science and Engineering, Kyung Hee University. He has been involved in model compression/interpretation for deep neural networks and inverse problems in image processing and computer vision.