Efficient Zero-Cost Neural Architecture Search for Personalized AI Systems in Cloud-Edge Networks

Kai Huang[a], Yingchi Mao[a], Benteng Zhang[a], Yihan Chen[a], Yuchu Chen[a], and Jie Wu[b]
[a] College of Computer Science and Software Engineering, Hohai University, Nanjing, China
[b] Center for Networked Computing, Temple University, Philadelphia, USA
241307010036@hhu.edu.cn, yingchimao@hhu.edu.cn, 230407040003@hhu.edu.cn,
241307010028@hhu.edu.cn, 2106010213@hhu.edu.cn, jiewu@temple.edu

***Abstract*—Neural Architecture Search (NAS) can discover the optimal neural network architecture within a given SuperNet through automated search, which can improve model performance and reduce computational overhead on resource-constrained devices. Due to the vast SuperNet requiring substantial computational resources for training and evaluation, the search process is costly and difficult to apply directly on End Devices (EDs) with limited computational resources. Moreover, existing methods utilize zero-cost proxies to reduce computational costs in NAS, but overlook limited computational resources on EDs and waste a large amount of computational resources on the cloud server. Deploying NAS on the cloud server can effectively address this issue. The cloud server is used to search for the optimal Subnet, and EDs only need to train the Subnet based on local data. To this end, we propose a nonlinear aggregation-based Neural Architecture Search method based on Feature and Gradient zero-cost proxies (FG-NAS). Specifically, EDs upload local data characteristics to the cloud server, and then the cloud server uses FG-NAS to obtain an optimal SubNet model from the SuperNet based on the uploaded data characteristics. Finally, the cloud server sends the optimal SubNet to EDs, which can reduce the computational burden on EDs. Furthermore, FG-NAS evaluates the accuracy of neural architectures by considering both feature proxies during forward propagation and gradient proxies during backward propagation. Experiments on three datasets demonstrate that compared to current mainstream zero-cost proxy methods, FG-NAS can improve evaluation accuracy by an average of 1.04% and reduce single-network evaluation time by up to 2.45%.**

***Index Terms*—Neural architecture search, zero-cost proxy, cloud-edge collaboration**

I. Introduction

With the increasing prevalence of End Devices (EDs) in Edge Networks (ENs) [1], the deployment and optimization of neural network models have become pivotal issues in the field of cloud-edge collaboration [2]. By utilizing cloud-edge collaboration, compute-intensive NAS tasks can be offloaded to the powerful cloud server to search for the optimal Subnet. EDs only handle lightweight local training tasks. This collaboration can reduce the NAS latency and improve the NAS accuracy. Neural Architecture Search (NAS) [3], a technique for automating the design of high-performance neural networks [4], has emerged as a cornerstone of Artificial Intelligence (AI) research. The objective of NAS is to identify network architectures that excel in specific tasks while being well-suited for efficient inference on EDs. However, ***Challenge 1*** in Fig. 1 has demonstrated that traditional NAS methods often encounter challenges such as excessive computational resource demands and prolonged training durations. For example, Zoph *et al.* proposed the Reinforcement Learning-based NAS method [3], which achieved an 2.50% error rate on the CIFAR-10 dataset but required eight hundred GPUs and weeks of training. Similarly, DARTS necessitated four GPUs and four days of training for searches on the ImageNet dataset. Such methods are impractical for EDs constrained by limited computational and storage capacity. Therefore, developing an efficient and low-cost [5] NAS method is important not only for enhancing the intelligence of EDs but also for advancing the broader adoption of edge computation.

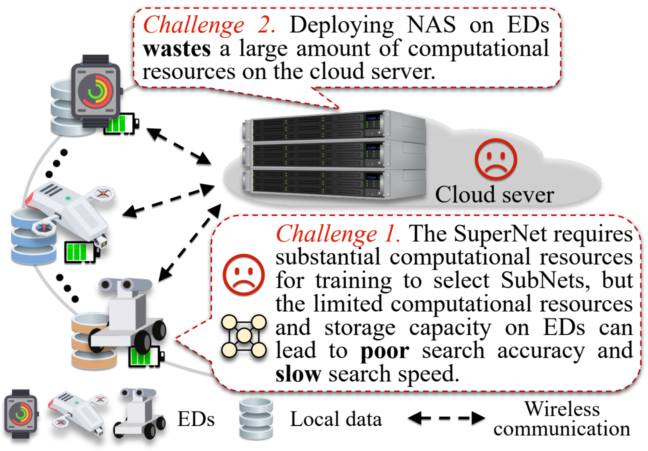
Challenge 2. Deploying NAS on EDs wastes a large amount of computational resources on the cloud server.
Cloud sever
Challenge 1. The SuperNet requires substantial computational resources for training to select SubNets, but the limited computational resources and storage capacity on EDs can lead to poor search accuracy and slow search speed.
EDs
Local data
Wireless communication

Fig. 1. Two challenges in NAS for resource-constrained EDs in cloud-edge networks.

To save computational resources on EDs, *zero-cost proxy*-based methods [6] have garnered attention for the ability to rapidly evaluate network performance without requiring training. For instance, NASWOT leverages statistical characteristics of network activations during forward propagation to predict performance [7], which achieves second-level evaluation time on the CIFAR-10 dataset without requiring GPU support. Similarly, TE-NAS introduced a multi-proxy evaluation framework by integrating a network's expressiveness and trainability [8], which further enhances prediction accuracy. Other methods, such as Zen-NAS [9] and TransNAS [10], utilize network topology or pre-trained model priors to accelerate the search process. However, as shown in ***Challenge 2*** in Fig. 1, existing *zero-cost proxy*-based methods deploy NAS on EDs, which fail to fully utilize the computational resources on the cloud server. In addition, existing *zero-cost proxy*-based methods still face other challenges. *1)* As feature correlation alone may neglect training dynamics, the narrow scope of single-proxy approaches leads to incomplete evaluations, which causes prediction bias. *2)* As linear weighting schemes struggle to scientifically reflect the importance differences among proxies, simplistic combinations of multiple proxies limit assessment precision. *3)* Insufficient generalization resources can lead to poor performance under heterogeneous data distributions on EDs. For example, the prediction accuracy of NASWOT drops by approximately 3.5% on the ImageNet16-120 dataset when data distributions change. These issues indicate that existing *zero-cost proxy*-based methods require further improvements to meet practical needs for resource-constrained EDs.

Motivated by these issues above, we aim to address the single-proxy evaluation instability and multi-proxy computational resource overconsumption inherent in Zero-Cost NAS. To this end, we propose a nonlinear aggregation-based **N**eural **A**rchitecture **S**earch method based on **F**eature and **G**radient zero-cost proxies (FG-NAS). In brief, by synergizing feature proxies from forward propagation and gradient proxies from backward propagation via dynamic nonlinear aggregation, FG-NAS can achieve precise neural architecture search. To reduce the NAS latency and improve the NAS accuracy, we deploy FG-NAS on cloud servers [11], where local EDs upload the data characteristics to the cloud server. The cloud server then uses FG-NAS to search for the optimal model architecture based on the uploaded data characteristics and the SuperNet.

The contributions of this paper are depicted as follows.

- **Multi-Dimensional Feature Proxy.** During forward propagation, FG-NAS quantifies feature information (FI) by computing the entropy of the feature matrix and assesses feature correlation (FC) using Pearson correlation coefficients and nuclear norms, providing a comprehensive measure of the network's expressive capacity.
- **Gradient-based Proxy.** In backward propagation, FG-NAS incorporates gradient change (GC) and important layer gradient information (GI), which utilizes Rademacher random vectors to optimize Jacobian matrix computations. This approach evaluates the network's training capability and convergence speed while reducing computational complexity.
- **Dynamic Nonlinear Aggregation Mechanism.** Depending on whether the input data is labeled, FG-NAS can dynamically adjust the weights of feature and gradient proxies. By employing nonlinear combinations, FG-NAS can amplify the impact of high-ranking proxies and mitigate the interference of low-ranking ones, thereby enhancing evaluation accuracy and robustness.
- **Effectiveness.** Experiments on the CIFAR-$10/100$ and ImageNet16-120 datasets demonstrate that compared to current mainstream zero-cost proxy methods, FG-NAS can improve evaluation accuracy and reduce single-network evaluation time.

This paper is organized as follows. The proposed framework is shown in Section II. The design details of FG-NAS are discussed in Section III. The experiments and analysis are given in Section IV. Finally, we conclude with Section V.

II. Proposed Framework

A. System Model

As shown in Fig. 2, our system model consists of four types of entities as follows.

- **EDs**. Due to limitations in computational power and storage capacity, EDs can typically only deploy and run models that have been distilled and quantized. In addition, they can communicate with a cloud server via an unstable wireless network connection for offline model training and downloading.
- **Cloud server**. A cloud server has sufficient computational power and storage for offline model training. During online inference, the cloud server is not involved.
- **SuperNet**. A supernet is a vast collection of neural network architectures. Training and evaluating them require significant computational resources, so supernets are typically stored on cloud services.
- **SubNet**. Optimal network models are tailored for EDs.

The proposed FG-NAS consists of nine steps as follows.

1) *Upload data characteristics.* The EDs upload local data to the cloud server.
2) *Input SuperNet.* We use local data and the entire SuperNet set as the input for FG-NAS.
3) *Calculate the Feature Information Score.* During forward propagation, the local data from the EDs undergoes PCA dimensionality reduction to obtain the Information Quantity, and the Feature Information Score S_{FI} is calculated using the relevant data characteristics and formulas.
4) *Calculate the Feature Correlation Score.* During forward propagation, the local data from the EDs undergoes PCA dimensionality reduction to obtain the correlation evaluation metrics, and the Feature Correlation Score S_{FC} is calculated using the relevant formula.
5) *Generate Jacobian matrix.* The Jacobian Matrix is calculated using the *Rademacher* random vector.
6) *Calculate Gradient Variation Rate Score.* We calculate the Gradient Variation Rate Score S_{GC} using Spectral Norms σ and the Jacobian Matrix.
7) *Important Layer Selection.* We select the important layer architecture using the Gradient Variation Rate Score S_{GC} and Spectral Norms σ.
8) *Calculate the Important Gradient Score.* We calculate the Important Gradient Score S_{GI} using the mean and variance of gradients during backpropagation.
9) *Output the SubNet ranking and send the optimal SubNet to the EDs.* We obtain the comprehensive score S based on the four individual scores (S_{FI}, S_{FC}, S_{GC}, S_{GI}) and select the network model with the highest comprehensive

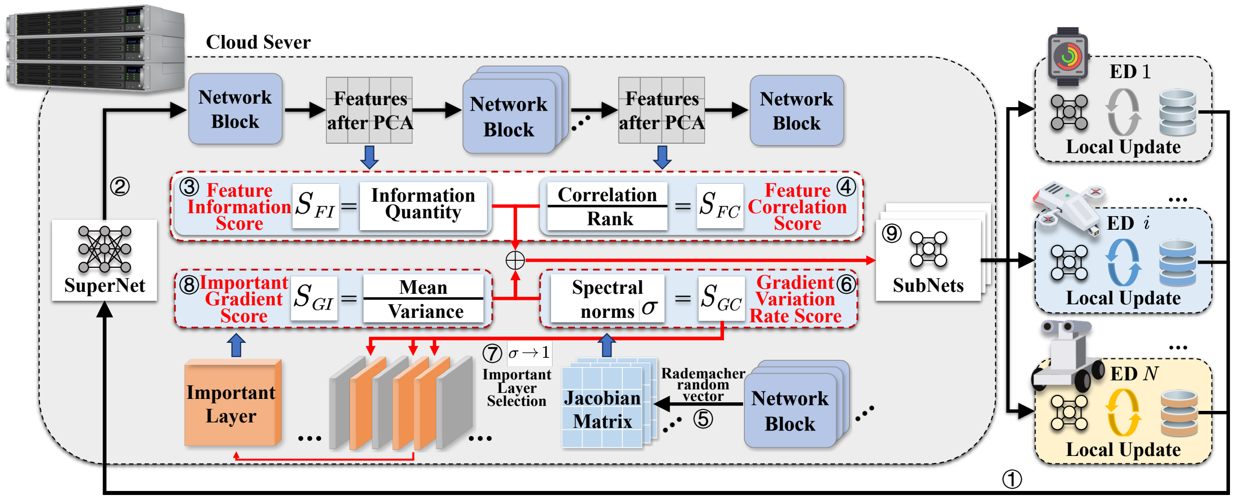
Cloud Sever
Network Block
Features after PCA
Network Block
Features after PCA
Network Block
②
SuperNet
③ Feature Information Score S_{FI} = Information Quantity
Correlation Rank = S_{FC} Feature Correlation Score ④
⑧ Important Gradient Score S_{GI} = Mean / Variance
Spectral norms σ = S_{GC} Gradient Variation Rate Score ⑥
⑨ SubNets
Important Layer
⑦ $\sigma \to 1$ Important Layer Selection
Jacobian Matrix
Rademacher random vector ⑤
Network Block
ED 1
Local Update
ED i
Local Update
ED N
Local Update
①

Fig. 2. The overview of FG-NAS. ① Upload data characteristics. ② Input SuperNet. ⑤ Generate *Jacobian* matrix. ⑨ Output the SubNet ranking and send the optimal SubNet to the ED.

score. Finally, the cloud server sends this network model to the corresponding ED.

B. Problem Formulation

The proposed FG-NAS aims to address the problem of efficient and accurate zero-cost neural architecture search for heterogeneous, resource-constrained EDs in a cloud-edge collaborative system [12]. We have the following definitions.

- $\mathcal{S}$ is the Supernet hosted on the cloud server, representing the SuperNet of candidate neural network architectures.
- $\mathcal{A} = \{\mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_K\} \subseteq \mathcal{S}$ is the set which is K distinct subnet architectures.
- N is the number of participating EDs.
- D_i is the local data characteristics provided by the i-th ED, for $i \in \{1, ..., N\}$. Let $\mathcal{D}_i$ denote the space of characteristics for ED_i, and $D = \{D_1, ..., D_N\}$. These characteristics are uploaded to the cloud server.
- C_i is the set of resource constraints for the i-th ED, such as computational resources and storage capacity.
- $\alpha \in \{0, 1\}$ is a flag that indicates the availability of labels for the data used in proxy calculation.

Other main symbolic parameters are shown in Table I. The core task is to develop an efficient zero-cost evaluation function executed on the cloud server. We denote this function as E, which is defined as

$$E : \mathcal{A} \times \mathcal{D}_i \times \{0, 1\} \to \mathbb{R}, \tag{1}$$

where E takes a candidate architecture $\mathcal{M} \in \mathcal{A}$, the data characteristics $D_i \in \mathcal{D}_i$ relevant to a target ED_i, and the label availability flag α. It outputs a scalar score $s_{\mathcal{M},i} = E(\mathcal{M}, D_i, \alpha)$ that predicts the suitability or potential performance of $\mathcal{M}$ for ED_i without requiring full training. The score $s_{\mathcal{M},i}$ should estimate the true *Performance*$(\mathcal{M}|D_i, C_i)$, which is typically obtained only after costly training and deployment. The underlying goal for NAS in this context is often to find an architecture $\mathcal{M}^*$ that solves

$$\begin{aligned} &\mathcal{M}^* = \arg\max_{\mathcal{M}\in\mathcal{A}} \textit{Performance}(\mathcal{M}|D_i), \\ &\text{subject to} \quad \text{constraints } C_i. \end{aligned} \tag{2}$$

Since solving (2) directly is infeasible due to the cost of evaluating *Performance*$(\cdot)$, we aim to leverage the zero-cost evaluation function E to generate a ranked list $R = (\mathcal{M}_{(1)}, \mathcal{M}_{(2)}, ..., \mathcal{M}_{(K)})$ of all architectures in $\mathcal{A}$, which is ordered by the scores $s_{\mathcal{M},i}$. This ranking R should effectively approximate the ideal ranking R^* derived from the true *Performance*$(\cdot)$. The quality of the ranking R is assessed by its correlation with R^*, aiming to maximize a rank correlation coefficient (i.e., *Spearman* coefficient ρ). Therefore, the optimization problem of this paper can be defined as

$$\text{Maximize} \quad \rho(R, R^*), \tag{3}$$

where R^* is the ground-truth ranking based on actual performance under constraints C_i.

As can be seen from the above optimization problem, the main challenge of this paper is to design a zero-cost evaluation function based on (1). It should be computationally efficient and capture architecture potential with limited and heterogeneous data. The function robustly combines multiple proxy signals and adapts to label availability α. It produces a reliable ranking based on (3) to guide the selection of optimal SubNets $\mathcal{M}^*$, which are suitable for deployment on diverse EDs under specific constraints C_i.

III. The Design Details of FG-NAS

A. Feature Proxy Based on Forward Propagation

The original feature matrix is denoted as $F_i \in \mathbb{R}^{c \times n}$ to represent the c-dimensional output characteristics of the i-th layer, where n indicates the number of characteristics. The characteristics are centralized by performing a mean

TABLE I
LIST OF MAIN SYMBOLIC PARAMETERS

Symbol	Descriptions
d	Number of eigenvalues
n	Total number of networks
v	Amount of feature information retained
ρ	Pearson correlation coefficient between features
$\mathcal{S}$	A Supernet encompassing a multitude of network architectures
Q	Information content
D	The set of input data characteristics from all EDs
M	Total number of training samples
D_i	The data feature of the current ED
$\overline{f_i^i}$	The mean of the i-th row
$\mathcal{M}^*$	A best-suited network architecture for the ED
$f_l^{i,m}$	The m-th feature point in the i-th row
S_{FI}	Network feature information content score
S_{GI}	Important layer gradient information score
S_{FC}	Total feature correlation score
S_{GC}	Neural network gradient change rate score
$F_i(k)$	The k-th column eigenvector in the feature matrix
$Block_l$	Operation of the neural network module at l -th layer

normalization on each column of the feature matrix (i.e., each feature), ensuring that the mean of each feature is zero, which is defined as

$$\overline{F}_i(k) = F_i(k) - \frac{1}{n}\sum\nolimits_{k=1}^{n} F_i(k), \tag{4}$$

where $F_i(k)$ is the k-th column feature vector in the feature matrix, with the size of $c \times 1$. The covariance matrix C_i can measure the correlation between characteristics, which is given by

$$C_{oi} = \frac{1}{n-1}\overline{F}_i^{\intercal}\overline{F}_i, \tag{5}$$

where $C_{oi} \in \mathbb{R}^{n\times n}$ can be used to determine the direction of maximum variation in the neural network characteristics. The eigenvalues are sorted, and the Top-k largest eigenvalues λ are selected by

$$V = \sum\nolimits_{i=1}^{k} \frac{\lambda_i}{\sum_{i=1}^{d}\lambda_i} \geq v, \tag{6}$$

where d is the number of eigenvalues and v is the amount of retained feature information, which is typically set to 0.9. The eigenvectors p that satisfy this condition are used to form a new matrix $V \in \mathbb{R}^{n\times k}$. Finally, the feature matrix F_i is mapped to the new vector space, resulting in a reduced-dimensionality feature matrix $f_i \in \mathbb{R}^{c\times k}$, which is given by

$$f_i = f_i V. \tag{7}$$

The amount of information Q is calculated by

$$Q = -k \log p, \tag{8}$$

where k is a constant. For a dataset of size n, denoted by the probability p_i for each occurrence, the entropy of this dataset is calculated by

$$S = -k\sum\nolimits_{i=1}^{n} p_i \log p_i. \tag{9}$$

The feature matrix distribution of each layer of the neural network output can be defined as

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{x^2}{2\sigma^2}\right), \tag{10}$$

where x is the element in the feature matrix and σ is the variance of the feature matrix. Based on (9) and (10), the calculation formula is given by

$$S[f_i] = \frac{\ln(2\pi\sigma^2)+1}{2}, \tag{11}$$

where $S[f_i]$ is the information quantity of the i-th layer feature and f_i is the output feature matrix of the i-th layer in the neural network. Finally, the total feature information of the network is the sum of all layer-wise information quantities

$$S_{FI} = \sum\nolimits_{l=1}^{N} S\left[f_l\right], \tag{12}$$

where S_{FI} is the feature information score of the network and N is the number of primary blocks in the neural network.

For a feature matrix $f_i \in \mathbb{R}^{c\times k}$, and l is the output of the l-th layer of a neural network, the Pearson correlation coefficient between characteristics is calculated by

$$\rho(f_l^i, f_l^j) = \frac{\sum_{m=1}^{k}\left(f_l^{i,m} - \overline{f}_l^i\right)\left(f_l^{j,m} - \overline{f}_l^j\right)}{\sqrt{\sum_{m=1}^{k}\left(f_l^{i,m} - \overline{f}_l^i\right)^2\left(f_l^{j,m} - \overline{f}_l^j\right)^2}}, \tag{13}$$

where f_l^i is the i-th row of the feature matrix, $f_l^{i,m}$ is the m-th feature point in the i-th row, and $\overline{f}_l^i$ is the mean of the i-th row. Let $sum(P_{f_l})$ denote the sum of all elements in the Pearson correlation matrix $P_{f_l} \in \mathbb{R}^{c\times c}$. For the feature matrix $f_l \in \mathbb{R}^{c\times k}$, its nuclear norm is defined as the trace of the square root of the product of the matrix and its transpose. The nuclear norm is calculated by

$$\parallel f_l \parallel_{nuc} = tr\left(\sqrt{f_l^T f_l}\right). \tag{14}$$

Finally, by calculating the ratio of the nuclear norm to the feature correlation coefficient, the total feature correlation score S_{FC} can be defined as

$$S_{FC} = \sum\nolimits_{l=1}^{N} \frac{\|f_l\|_{nuc}}{sum\left(P_{f_l}\right)}. \tag{15}$$

B. Gradient Proxy Based on Backpropagation

For the l-th layer, the Jacobian matrix is calculated by

$$J_l = \frac{\partial f_l}{\partial f_{l-1}}, \tag{16}$$

where, the partial derivative of the current output f_l with respect to the input f_{l-1}. During the forward propagation of a neural network, for the input $f_{l-1} \in \mathbb{R}^{c\times n}$ of any main module, it undergoes relatively complex transformations to produce the output $f_l \in \mathbb{R}^{c'\times n}$, which can be approximated as a linear transformation

$$f_l = Block_l f_{l-1} \approx A_l f_{l-1}, \tag{17}$$

where $Block_l$ is the neural network module operation at the l-th layer and A_l is the approximate linear transformation matrix, the gradient propagation for the backpropagation is calculated by

$$g_{l-1} \approx A_l^T g_l = J_l g_l, \tag{18}$$

where g_l is the gradient of the l-th layer, thus the linear transformation matrix A_l^T is the Jacobian matrix of the l-th layer $J_l \in \mathbb{R}^{c' \times c}$. Inspired by the Hutchinson method, FG-NAS introduces a *Rademacher* random vector $v \in \mathbb{R}^c \times 1$ to approximate the Jacobian matrix J_ι, where the elements of the vector are randomly drawn from $\{1, -1\}$ with equal probabilities. When calculating expectations, v follows a *Rademacher* distribution, which has three important properties

$$E[v_i] = 0; E[v_i^2] = 1; E[v_i v_j] = 0, \tag{19}$$

where $i \neq j$. FG-NAS uses vector v to replace gradient g during backpropagation, obtains vector v' through the Jacobian matrix, and multiplies v' on both sides of the equation, and the expectation of both sides is calculated by

$$E[v'v^T] = E[J_l v v^T] = J_l E[vv^T] = J_l. \tag{20}$$

According to the expected property of the Rademacher random vector, $E[vv^T]$ is the unit matrix, so we only need to find the mean of $v'v^T$ to get the Jacobian matrix. In the actual calculation process, multiple v' are obtained by calculating multiple random samples v, and finally their mean is calculated by

$$v_k' = Block_l(v_k^T), J_l = \frac{1}{n} \sum\nolimits_{k=1}^{n} v_k' v_k^T, \tag{21}$$

where $Block_l$ is the operation of the back propagation of the l-th layer of the neural network. Finally, the Jacobian matrix is decomposed into singular values, and then the spectral norm is calculated by

$$J_l = UMV^T, \sigma_l = \|J_l\|_2 = \sigma_{\max}(M), \tag{22}$$

where M is a diagonal matrix, and the spectral norm of the Jacobian matrix J_t is the maximum value in the diagonal matrix. The neural network gradient change rate score is calculated by

$$S_{GC} = \sum\nolimits_{l=1}^{N} 2 - \sigma_l - \frac{1}{\sigma_l}. \tag{23}$$

During each training iteration, the neural network calculates the gradient of the loss function concerning each parameter through backpropagation. This gradient determines the direction and magnitude of the parameter updates. The mean gradient is calculated by

$$E[g(x)] = \frac{1}{M} \sum\nolimits_{i=1}^{M} \left| \frac{\partial L(x_i, y_i; \theta)}{\partial \theta} \right|, \tag{24}$$

where $L(x_i, y_i; \theta)$ is the loss function of the network for the i-th sample (x_i, y_i), θ is the parameter of the current layer, and M is the total number of training samples. Then, the gradient variance can be given by

$$Var[g(x)] = \frac{1}{M} \sum\nolimits_{i=1}^{M} \left(\frac{\partial L(x_i, y_i; \theta)}{\partial \theta} - E[g(x_i)] \right)^2. \tag{25}$$

To accelerate network evaluation and reduce computational overhead, FG-NAS uses the gradient change rate σ_ι as an importance evaluation metric to select layers where gradients can propagate stably for computation. Given a range of gradient changes $\{\sigma_l \in [k, m] \mid 0 \leqslant k \leqslant 1, m \geqslant 1\}$, FG-NAS selects important layers within the specified range to calculate the mean and variance of the gradients and finally computes the average. The gradient information score S_{GI} of the important layers is defined as

$$S_{GI} = \frac{1}{\sum_{l=1}^{N} (1 \mid \sigma_l)} \sum\nolimits_{l=1}^{N} \left(\frac{E[g(x)]}{Var[g(x)]} \mid \sigma_l \right). \tag{26}$$

After setting the extraction ratio γ of important layers, k and m can be calculated by

$$\sum\nolimits_{l=1}^{N} (1 \mid \sigma_l) = \gamma N, \tag{27}$$

where $km = 1$ and the important layer proportion γ is generally set within $[0.9, 0.5]$.

C. Dynamic Nonlinear Aggregation

FG-NAS adopts a nonlinear combination method that integrates logarithmic and exponential functions to combine gradient-based metrics with feature map-based metrics for calculating network rankings. Since the gradient information computation of important layers relies on data labels [23], to better adapt to different data distributions, FG-NAS introduces a coefficient $\alpha \in \{0, 1\}$, which indicates whether the input data has labels. When input data lacks labels, we set $\alpha = 0$. The gradient-based proxy ranking R_G is calculated by

$$R_G = \begin{cases} \ln(R_{GI} R_{GC}) + R_{GI}^2 + R_{GC}^2, & \alpha = 1 \\ R_{GC}, & \alpha = 0 \end{cases} \tag{28}$$

where R_{GI} and R_{GC} are the important layer gradient information and gradient change rate proxy ranking, respectively. The feature-based proxy ranking R_F is calculated by

$$R_F = \ln(R_{FI} R_{FC}) + R_{FI}^2 + R_{FC}^2, \tag{29}$$

where R_{FI} and R_{FC} are the feature information quantity and correlation proxy ranking, respectively. Based on R_G and R_F, the overall ranking R of the network is defined as

$$R = \ln(R_F R_G) + R_F^2 + R_G^2. \tag{30}$$

D. Complexity Analysis of FG-NAS

We analyze the computational complexity of the proposed FG-NAS method for evaluating a single candidate network architecture $\mathcal{M} \in \mathcal{A}$. The complexity depends on several factors, such as L, the number of principal layers/blocks analyzed, M, the mini-batch size, $c \times n$, the feature map dimensions before PCA, $c \times k$, the dimensions after PCA, N_{rad}, the

Algorithm 1: FG-NAS

```
Input: SuperNet, Input data, Labeled flag
Output: Ranked list $R$ of SubNets
Initial Rankings = Empty list
for each SubNet ∈ SuperNet do
    $features$ = ForwardPropagation(SubNet, input data)
    $F$ = PCA($features$)
    Feature Information Score:
    $S\left[f_l\right] = \frac{\ln\left(2\pi\sigma^2\right)+1}{2}$, $f_l \in F$
    $S_{FI} = \sum_{l=1}^{N} S\left[f_l\right]$
    Feature Correlation Score:
    $\| f_l \|_{nuc} = tr\left(\sqrt{f_l^T f_l}\right)$
    $S_{FC} = \sum_{l=1}^{N} \frac{\|f_l\|_{nuc}}{sum\left(P_{f_l}\right)}$
    Gradient Change Rate Score:
    $J_l = UMV^T$
    $\sigma_l = \|J_l\|_2 = \sigma_{\max}(M)$
    $S_{GC} = \sum_{l=1}^{N} 2 - \sigma_l - \frac{1}{\sigma_l}$
    Important Layer Gradient Information Score:
    $E[g(x)] = ComputeGradientMean($Input data$)$
    $Var[g[x]] = ComputeGradientVariance($Input data$)$
    $S_{GI} = \frac{1}{\sum_{l=1}^{N}(1 \mid \sigma_l)} \sum_{l=1}^{N} \left(\frac{E[g(x)]}{Var[g(x)]} \mid \sigma_l\right)$
    $R_F = \ln\left(R_{FI}R_{FC}\right) + R_{FI}^2 + R_{FC}^2$
    if Labeled flag $= 1$ then
        $R_G = \ln\left(R_{GI}R_{GC}\right) + R_{GI}^2 + R_{GC}^2$
    else
        $R_G = R_{GC}$
    end
    return $R = \ln\left(R_F R_G\right) + R_F^2 + R_G^2$
end
function $ComputeGradientMean(Inputdata)$
begin
    for each sample $i = 1, ..., M$ do
        $g_i = \frac{\partial L(x_i, y_i; \theta)}{\partial \theta}$
    end
    return $E[g(x)] = \frac{1}{M} \sum_{i=1}^{M} g_i$
end
function $ComputeGradientVariance(Inputdata)$
begin
    for each sample $i = 1, ..., M$ do
        $v_i = \left(\frac{\partial L(x_i, y_i; \theta)}{\partial \theta} - E\left[g\left(x\right)\right]\right)^2$
    end
    return $Var[g(x)] = \frac{1}{M} \sum_{i=1}^{M} v_i$
end
```

number of Rademacher samples for Jacobian approximation, and the single-sample forward/backward propagation costs (F_{ops} and B_{ops}). The n is the feature dimension per channel, which is consistent with its use in Section III-A.

1) Feature Proxies (S_{FI} and S_{FC}). Calculation requires one initial forward propagation for the mini-batch ($\mathcal{O}(M \cdot F_{ops})$). Subsequent per-layer operations include PCA dimensionality reduction and computations for entropy, correlation, and nuclear norm, with matrix operation costs ($C_{matrix,F}$) primarily dependent on dimensions L, c, n, k.

2) Gradient Proxies (S_{GC} and S_{GI}). These proxies require backpropagation. Computing S_{GC} involves Jacobian approximation using N_{rad} Rademacher samples (requiring computation roughly equivalent to $\mathcal{O}(L \cdot N_{rad})$ partial backward propagation, cost B'_{ops} each) and SVD computation ($\mathcal{O}(L \cdot C_{SVD}(c, c'))$). Computing S_{GI} requires standard backpropagation for the mini-batch ($\mathcal{O}(M \cdot B_{ops})$). The cost is dominated by the backward propagation.

3) Aggregation.

Combining the four proxy scores involves a constant number of arithmetic operations per network, resulting in negligible complexity $\mathcal{O}(1)$.

4) The total complexity per architecture is the sum of the costs for feature and gradient proxies, which is $\mathcal{O}(MF_{ops} + (LN_{rad}B'_{ops} + MB_{ops}) + L \cdot C_{matrix\_ops})$. The C_{matrix_ops} is the significant per-layer cost of matrix operations (including PCA, Correlation, Nuclear Norm, SVD). As M and N_{rad} are typically small constants, the complexity is primarily determined by a limited number of forward/backward propagation and matrix computations. Therefore, the proposed FG-NAS method maintains a low computational complexity per architecture evaluation compared to training-based approaches.

IV. Performance Evaluation

A. Experimental Settings

Experimental Environment. All experiments in this section are conducted on the cloud server equipped with two NVIDIA A100 GPUs with 80GB memory and 256GB RAM. We choose several Jetson Xavier NX development boards equipped with 8GB memory and 6-core ARM CPUs as EDs.

SuperNets. We have three SuperNets as follows.

- *NAS-Bench-201* [14] consists of 15,625 network architectures, each utilizing a unique cell structure. It provides various evaluation metrics, such as test accuracy and training loss on the CIFAR-10/100 datasets.
- *MobileNetV2* [15] includes candidate architectures built using inverted residual blocks, which vary in block depth, width, and expansion ratio.
- *AutoFormer* [16] is specifically designed to evaluate NAS methods for Vision Transformers, which can be divided into tiny, small, and base subsets based on model size.

Datasets. We have three datasets as follows.

- *CIFAR-10* [17] dataset contains $60,000$ 32×32 pixel color images, divided into 10 classes, with $6,000$ images per class. It is used for image-classification tasks and is commonly employed to test model performance.
- *CIFAR-100* [17] dataset contains 100 classes with 600 32×32 pixel images per class. It is often used to evaluate model performance on multi-class classification tasks.
- *ImageNet16-120* [18] dataset is a subset of the ImageNet dataset used for image classification tasks. It includes 120 classes, with $1,280$ images per class, totaling $153,600$ images. The images in IN16-120 are 16×16 pixels, making them smaller than the original ImageNet dataset, which is used for rapid training and experimentation.

Baselines. We have eight baseline methods as follows.

TABLE II
CORRELATION BETWEEN DIFFERENT ZERO-COST PROXIES AND MODEL ACCURACY IN THE NAS-BENCH-201 SUPERNET

Method	*Spearman* ρ			Time(ms)
	CIFAR-10	CIFAR-100	ImageNet16-120	
#Params/FLOPS	0.753	0.727	0.691	-
SNIP	0.615	0.619	0.539	304.8
NASWOT	0.743	0.769	0.760	**33.1**
TE-NAS	0.731	0.728	0.680	1209.1
ZiCo	0.809	0.785	0.778	342.6
FG-NAS	**0.853**	**0.849**	**0.843**	316.3

TABLE III
EXPERIMENTAL RESULTS IN THE MOBILENETV2 SUPERNET

Constraint (Params, FLOPs)	Method	Type	*Top*-1	Search Cost (GPU Days)
(5M, 600M)	OFA	TB	78.7	50
	ZiCo	TF	79.1	0.4
	FG-NAS	TF	**79.5**	**0.35**
(3M, 450M)	OFA	TB	77.7	50
	ZiCo	TF	78.1	0.4
	FG-NAS	TF	**78.4**	**0.33**

- *#Params/FLOPS* [19]. The number of parameters or floating-point operations (FLOPS) is directly used as an indicator of network complexity and potential performance. Generally, networks with more parameters possess stronger learning capabilities but incur higher computational costs.
- *SNIP(B)* [20] evaluates network trainability by measuring the sensitivity of individual weights to the loss function. A higher overall sensitivity indicates greater trainability and a stronger potential for effective training.
- *NASWOT(F)* [21] analyzes the correlation of activation values during the forward propagation. Lower correlation among activations suggests a better ability to partition the input space and extract diverse characteristics, thus implying stronger representational power.
- *TE-NAS(G+L)* [8] evaluates network representational capacity and trainability. It measures the number of linear regions and the Neural Tangent Kernel (NTK) condition number. More linear regions and smaller condition numbers indicate better performance and easier training.
- *ZiCo(B)* [22] assesses trainability by analyzing the distribution of network gradients. Networks with higher mean gradients and lower standard deviations exhibit more stable training behavior and superior trainability.
- *TF-TAS(G+L)* [24] evaluates network performance by combining its expressive capacity and task adaptability.
- *OFA* [25] achieves adaptation to different network configurations with a single training by training a supernet that supports multiple sub-networks.
- *AutoFormer* [16] employs a weight-sharing supernet and progressive shrinking to efficiently search Transformer architectures. It directly evaluates subnet performance using inherited weights without retraining, significantly reducing search time and resource consumption.

Metrics. We have three metrics as follows.

- *Spearman* coefficient ρ is used to assess the relationship between the predicted ranking of networks and their actual performance, which is given by

$$\rho = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2-1)}, \tag{31}$$

  where d_i is the difference between the predicted ranking and the actual ranking for the i-th network, and n is the total number of networks.
- *Single network evaluation time*, i.e., the time required to obtain the comprehensive ranking of the network using the current method.
- *Top-1 accuracy* refers to the classification accuracy of the searched neural network on the test set, i.e., the proportion of instances where the predicted label matches the true label, which is given by

$$Top\text{-}1 = \frac{\text{Number of correctly predicted samples}}{\text{Total number of samples}}. \tag{32}$$

B. Correlation Comparison Analysis

To validate the accuracy of FG-NAS in predicting network accuracy, we conduct experiments on the NAS-Bench-201 SuperNet, calculating *Spearman* coefficient ρ and evaluation time. In each run, $3,000$ networks are randomly sampled for evaluation, and the experiment is repeated 5 times to compute the average results. The experimental results are shown in Table II. This experiment is conducted on a public server.

FG-NAS performs best in terms of correlation, achieving improvements of 0.044, 0.064, and 0.065 over the best baseline ZiCo across three datasets, while reducing the single network evaluation time by approximately 74% compared to the multi-proxy [28] method TE-NAS. Compared to TE-NAS, FG-NAS utilizes feature information and feature correlation during forward propagation to measure the network's feature extraction capability, making it applicable to all types of networks. To assess gradient propagation stability, FG-NAS replaces gradients with *Rademacher* random vectors to optimize matrix computations, thereby reducing computational load. Compared to ZiCo, FG-NAS incorporates feature-based metrics [26] to enhance search accuracy and reduce the dependency on data labels. To further decrease computational complexity, FG-NAS uses the gradient change rate as an importance selection metric, selecting layers with stable gradient propagation for gradient computation. Additionally, FG-NAS employs dynamic nonlinear aggregation to emphasize high-ranking metrics and penalize low-ranking ones, which improves the accuracy of network performance prediction.

C. Comparison of NAS under Resource Constraints.

We conduct comparative experiments in the MobileNetV2 SuperNet and the AutoFormer SuperNet, respectively. Our experiments broadly categorize NAS methods [27] into two primary categories, such as Training-Based (TB) and Training-Free (TF). The experiments in this section are conducted on the embedded Jetson Xavier NX edge device.

TABLE IV
EXPERIMENTAL RESULTS IN THE AUTOFORMER SUPERNET

Constraint (Params, FLOPs)	Method	Type	*Top*-1	Search Cost (GPU Days)
(25M, 5G)	AutoFormer	TB	74.7	24
	TF-TAS	TF	75.3	0.5
	FG-NAS	TF	**75.9**	**0.25**
(5M, 1.5G)	AutoFormer	TB	81.7	24
	TF-TAS	TF	81.9	0.6
	FG-NAS	TF	**82.0**	**0.31**

TABLE V
EXPERIMENTAL RESULTS OF FG-NAS ABLATION

Method	*Spearman ρ*			Time(ms)
	CIFAR-10	CIFAR-100	ImageNet16-120	
-FP	0.803	0.782	0.779	282.6
-GP	0.787	0.771	0.762	**36.7**
-NL	0.839	0.815	0.798	311.4
FG-NAS	**0.853**	**0.849**	**0.843**	316.3

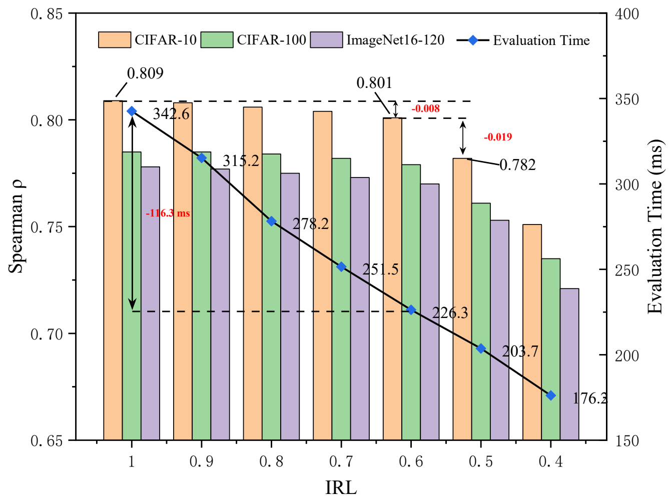

Fig. 3. Ablation Experiment with Different Importance Layer Ratios

The experiment firstly compares FG-NAS with the training-based method OFA and the best zero-cost proxy method ZiCo on MobileNetV2. During the search process, an evolutionary search strategy is employed, with the evolutionary algorithm set to iterate $100,000$ times to ensure thorough exploration of the entire SuperNet. The candidate set size is 1024, and each iteration selects the highest-ranked network for evolutionary mutation. The experimental results are shown in Table III. Under the given constraints, FG-NAS achieves the best experimental results, outperforming the training-based OFA method. FG-NAS completes the search process in just 0.35 and 0.33 days under the two constraint conditions, respectively, and improves accuracy by 0.4% and 0.3% compared to ZiCo. This demonstrates the efficiency of FG-NAS. FG-NAS reduces the computational load of the evaluation process by retaining calculations from important layers and improves the accuracy of evaluation results through a combination of zero-cost proxies from multiple perspectives, thereby enhancing the precision of the search outcomes.

The experiment on AutoFormer compares FG-NAS with the training-based method AutoFormer and the best Transformer architecture search method TF-TAS. The experiment sets constraints on the Tiny and Small subsets, respectively, and randomly samples $10,000$ architectures for evaluation. The experimental results are shown in Table IV. Under the given constraints, compared to the TF-TAS method, FG-NAS reduces the search time by approximately 50%, while improving the accuracy of the search results by 0.6% and 0.1%, respectively. The experimental results demonstrate that FG-NAS is not limited to specific types of network architectures, validating the generalization capability of the feature-based proxy and gradient-based proxy approach.

D. FG-NAS Ablation Experiments

FG-NAS is primarily divided into three components such as Feature Proxy (FP), Gradient Proxy (GP), and Dynamic Nonlinear Aggregation (NL). To determine the impact of three components on FG-NAS, we conduct ablation experiments and compare the results with the original method in the NAS-Bench-201. The experimental results are shown in Table V. To validate the effectiveness of the important layer selection strategy, we compare different Important Layer Ratios (ILR). The experimental results are presented in Fig. 3.

The experimental results in Table V firstly indicate that the gradient proxy has the highest correlation with model accuracy. After removing the gradient proxy, the *Spearman* coefficient ρ on the CIFAR-10, CIFAR-100, and ImageNet16-120 datasets decrease by 0.066, 0.078, and 0.081, respectively. This is because gradient-related information can reflect the trainability of the network. Secondly, after removing the feature proxy, the *Spearman* coefficient ρ on the three datasets decreases by 0.050, 0.067, and 0.064, respectively, demonstrating that the amount of feature information and feature correlation effectively reflect the network's feature extraction capability. Finally, when replacing the Dynamic Nonlinear Aggregation (NL) with Linear Aggregation (L), the experimental results on the three datasets decrease by 0.014, 0.034, and 0.045, respectively. This suggests that linear aggregation cannot effectively handle the differences between the scores of various proxies, leading to network selection bias. In contrast, nonlinear aggregation can emphasize high-ranking proxies and penalize low-ranking ones, enabling neural architecture search to focus more on network architectures that consistently rank high across proxies.

As shown in Fig. 3, when the Important Ratio of Layers (IRL) is set to 0.6, the *Spearman* coefficient ρ on CIFAR-10 drops by only 0.008, while the evaluation time drops by 116.3 ms. This indicates that the important layer selection strategy can minimize evaluation time without significantly compromising assessment accuracy. However, when the IRL is further reduced to 0.5, the *Spearman* coefficient ρ drops substantially 0.019 because excessive layer removal leads to critical information loss.

V. Conclusion

To tackle the problems of weak correlation between zero-cost proxies and model accuracy, heavy reliance on labeled data, high computational demand, and wasteful consumption of cloud server computational resources, we propose a non-linear aggregation-based Neural Architecture Search method based on Feature and Gradient zero-cost proxies (FG-NAS). Specifically, FG-NAS evaluates neural architectures globally by combining feature proxies from forward propagation and gradient proxies from backward propagation. It quantifies feature information and correlation to enhance forward proxy reliability and assesses training capability through gradient change rate and mean-variance. An importance selection strategy reduces gradient computation, while dynamic nonlinear aggregation adjusts proxy weights based on data labels, which improves evaluation accuracy and search efficiency. Experiments on the CIFAR-10, CIFAR-100, and ImageNet16-120 datasets demonstrate that compared to current mainstream zero-cost proxy methods, FG-NAS can improve evaluation accuracy by an average of 1.04% and reduce single-network evaluation time by up to 2.45%. These improvements demonstrate that FG-NAS is a promising method to significantly improve the accuracy of neural architecture evaluation and save computational resources on EDs.

Acknowledgment

This work was supported in part by the Key Research and Development Program of China under Grant 2022YFC3005401; in part by the Technology Talent and Platform Program of Yunnan Province under Grant 202405AK340002; in part by the Technology Project of Huaneng Group under Grant HNKJ20-H46 and Grant HNKJ24-H167; and in part by the High Performance Computing Platform, Hohai University. *(Corresponding authors: Yingchi Mao and Benteng Zhang.)*

References

[1] B. Zhang *et al.*, "Overcoming Forgetting Using Adaptive Federated Learning for IIoT Devices With Non-IID Data," in *IEEE Internet of Things Journal,* vol. 12, no. 12, pp. 21025-21037, 15 June 15, 2025.

[2] D. Yao and B. Li, "PerFedRLNAS: One-for-All Personalized Federated Neural Architecture Search," *AAAI*, vol. 38, no. 15, pp. 16398-16406, Mar. 2024.

[3] B. Zoph and Q. V. Le, "Neural Architecture Search with Reinforcement Learning," *arXiv* arXiv:1611.01578, 2016.

[4] H. Zhou, J. Peng, C. Liao, and J. Li, "Application of Deep Learning Model Based on Image Definition in Real-Time Digital Image Fusion," *J Real-Time Image Proc*, vol. 17, no. 3, pp. 643–654, Jun. 2020.

[5] N. H. Luong, Q. M. Phan, *et al.*, "Lightweight Multi-Objective Evolutionary Neural Architecture Search with Low-Cost Proxy Metrics," *Information Sciences,* vol. 655, p. 119856, 2024.

[6] J. Lukasik, M. Moeller, and M. Keuper, "An Evaluation of Zero-Cost Proxies-From Neural Architecture Performance Prediction to Model Robustness," *International Journal of Computer Vision,* pp. 1–18, 2024.

[7] J. Mellor, J. Turner, A. Storkey, and E. J. Crowley, "Neural Architecture Search without Training," in *Proceedings of the International Conference on Machine Learning (ICML)*, PMLR, 2021, pp. 7588–7598.

[8] W. Chen, X. Gong, and Z. Wang, "Neural Architecture Search on ImageNet in Four GPU Hours: A Theoretically Inspired Perspective," *arXiv* arXiv:2102.11535, 2021.

[9] M. Lin *et al.*, "Zen-NAS: A Zero-Shot NAS for High-Performance Image Recognition," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 347–356.

[10] I. U. Haq, B. S. Lee, and D. M. Rizzo, "TransNAS-TSAD: Harnessing Transformers for Multi-Objective Neural Architecture Search in Time Series Anomaly Detection," *Neural Comput & Applic*, vol. 37, no. 4, pp. 2455–2477, Feb. 2025.

[11] Y. Jiang, Z. Li, B. Zhao, X. Zhang, and X. Dong, "Foreign Object Detection in the Inspection of Cloud Server Center Using Separable Self-Attention," *Intelligent Service Robotics*, vol. 18, no. 2, pp. 279–292, Mar. 2025.

[12] B. Zhang, Y. Mao, X. He, P. Ping, H. Huang and J. Wu, "Exploring the Privacy-Accuracy Trade-Off Using Adaptive Gradient Clipping in Federated Learning," in *IEEE Transactions on Network Science and Engineering,* vol. 12, no. 3, pp. 2254-2265, May-June 2025.

[13] J. Lee, D. Kim, and B. Ham, "Network Quantization with Element-Wise Gradient Scaling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 6448–6457.

[14] X. Dong and Y. Yang, "NAS-Bench-201: Extending the Scope of Reproducible Neural Architecture Search," *arXiv* arXiv:2001.00326, 2020.

[15] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted Residuals and Linear Bottlenecks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4510–4520.

[16] M. Chen, H. Peng, J. Fu, and H. Ling, "Autoformer: Searching Transformers for Visual Recognition," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 12270–12280.

[17] A. Krizhevsky and G. Hinton, "Learning Multiple Layers of Features from Tiny Images," 2009.

[18] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A Large-Scale Hierarchical Image Database," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255, 2009.

[19] C. White, M. Khodak, R. Tu, S. Shah, S. Bubeck, and D. Dey, "A Deeper Look at Zero-Cost Proxies for Lightweight Nas," *ICLR Blog Track*, 2022.

[20] Lee, Namhoon, Thalaiyasingam Ajanthan, and Philip HS Torr. "Snip: Single-Shot Network Pruning Based on Connection Sensitivity." *arXiv* arXiv:1810.02340, 2018.

[21] J. Mellor, J. Turner, A. Storkey, and E. J. Crowley, "Neural Architecture Search Without Training," in *Proceedings of the International Conference on Machine Learning (ICML)*, PMLR, 2021, pp. 7588–7598.

[22] G. Li, Y. Yang, K. Bhardwaj, and R. Marculescu, "Zico: Zero-shot NAS via Inverse Coefficient of Variation on Gradients," *arXiv* arXiv:2301.11300, 2023.

[23] B. Zhang, Y. Mao, X. He, H. Huang and J. Wu, "Balancing Privacy and Accuracy Using Significant Gradient Protection in Federated Learning," in *IEEE Transactions on Computers,* vol. 74, no. 1, pp. 278-292, Jan. 2025.

[24] Q. Zhou *et al.*, "Training-Free Transformer Architecture Search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 10894–10903.

[25] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-All: Train One Network and Specialize it for Efficient Deployment," *arXiv* arXiv:1908.09791, 2019.

[26] N. Sinha, A. El Rahman Shabayek, A. Kacem, P. Rostami, C. Shneider, and D. Aouada, "Hardware Aware Evolutionary Neural Architecture Search Using Representation Similarity Metric," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision,* 2024, pp. 2628–2637.

[27] B. Wu, X Dai, P Zhang, Y Wang, F Sun, Y Wu, Y Tian, P Vajda, Y Jia, K Keutzer, "Fbnet: Hardware-Aware Efficient Convnet Design via Differentiable Neural Architecture Search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 10734–10742.

[28] Y. Xue, C. Chen, and A. Słowik, "Neural Architecture Search Based on a Multi-Objective Evolutionary Algorithm with Probability Stack," *IEEE Transactions on Evolutionary Computation,* vol. 27, no. 4, pp. 778–786, 2023.