Effective, Efficient and Robust Neural Architecture Search

Zhixiong Yue[1,2,3], Baijiong Lin[2], Yu Zhang[2,4], and Christy Liang[3]
[1]*Research Institute of Trustworthy Autonomous Systems, Southern University of Science and Technology*
[2]*Department of Computer Science and Engineering, Southern University of Science and Technology*
[3]*School of Computer Science, University of Technology Sydney*
[4]*Peng Cheng Laboratory*
yuezx@mail.sustech.edu.cn, {bj.lin.email,yu.zhang.ust}@gmail.com, jie.liang@uts.edu.au

2022 International Joint Conference on Neural Networks (IJCNN) | 978-1-7281-8671-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/IJCNN55064.2022.9892654

***Abstract*—Designing neural network architecture for embedded devices is practical but challenging because the models are expected to be not only accurate but also enough lightweight and robust. However, it is challenging to balance those trade-offs manually because of the large search space. To solve this problem, we propose an Effective, Efficient, and Robust Neural Architecture Search (E2RNAS) method to automatically search a neural network architecture that balances the performance, robustness, and resource consumption. Unlike previous studies, the objective function of the proposed E2RNAS method is formulated as a multi-objective bi-level optimization problem with the upper-level subproblem as a multi-objective optimization problem that considers the performance, robustness, and resource consumption. To solve the proposed objective function, we integrate the multiple-gradient descent algorithm, a widely studied gradient-based multi-objective optimization algorithm, with the bi-level optimization. Experiments on benchmark datasets show that the proposed E2RNAS method can find robust architecture with low resource consumption and comparable classification accuracy.**

***Index Terms*—neural architecture search, adversarial robustness, out-of-distribution**

I. Introduction

Deep learning has achieved great successes in many areas, such as computer vision, natural language processing, speech, gaming. The design of the neural network architecture is essential to such success. However, such design relies heavily on experts' knowledge and experience, and even experienced experts cannot design the optimal architecture. Therefore, Neural Architecture Search (NAS), which aims to design the architecture of neural networks in an automated way, has attracted great attention in recent years.

Although NAS has demonstrated the capability to find neural network architecture with state-of-the-art performance in various tasks [1]–[3], conventional NAS methods are typically only designed to optimize the accuracy during the architecture searching process while neglecting other significant objectives, resulting in very limited application scenarios.

Actually, performance is not the only factor to be considered in real-world applications. On the contrary, resource consumption and robustness may be more critical. For example, a deep neural network with high computational burden and storage demands is difficult to be deployed to embedded devices (*e.g.* mobile phone and IoT device). Besides, it is well known that the trained neural networks are easily misled by adversarial examples and can not exactly distinguish in- and out-of-distribution samples, making them hard to deploy in safety-sensitive applications such as autonomous driving. Hence, there exists an open problem: *Can we find an automatic way to design a robust architecture with competitive performance to be deployed in resource-aware platforms?*

Corresponding author: Yu Zhang.

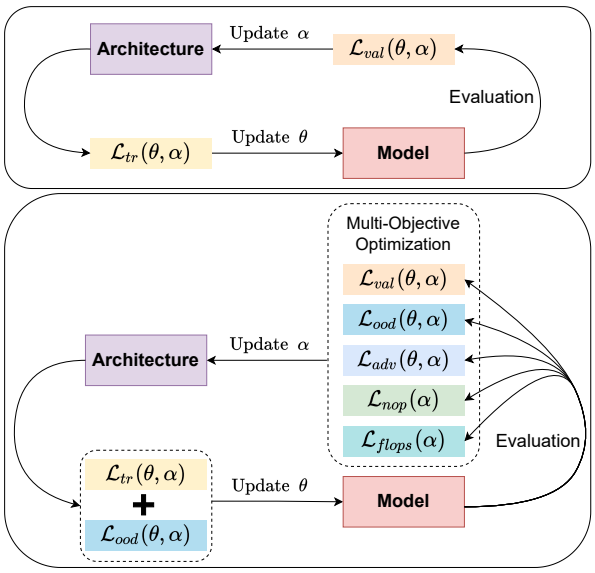

Fig. 1. Comparison of the architecture searching procedure between DARTS [1] (top) and the proposed E2RNAS (bottom). We formulate E2RNAS as a multi-objective bi-level optimization problem with two key differences with DARTS: 1) we train the model with both in- and out-of-distribution data samples to improve the robustness. 2) we evaluate the E2RNAS model with five objectives, including the validation loss $\mathcal{L}_{val}(\boldsymbol{\theta},\boldsymbol{\alpha})$ for effectiveness and the number of parameters $\mathcal{L}_{nop}(\boldsymbol{\alpha})$, the number of operations $\mathcal{L}_{flops}(\boldsymbol{\alpha})$ for efficiency, and the out-of-distribution robustness loss $\mathcal{L}_{ood}(\boldsymbol{\theta},\boldsymbol{\alpha})$ and the adversarial robustness loss $\mathcal{L}_{adv}(\boldsymbol{\theta},\boldsymbol{\alpha})$ for robustness.

This open problem implicitly indicates a trade-off among multiple objectives in NAS. Recently, some studies have considered multiple objectives during the architecture searching process. However, most of those works (*e.g.* [3]–[5]) only focus on the hardware-aware NAS problem, *i.e.* designing an architecture that can be deployed in resource-limited devices. There exist few studies [6], [7] that statistically investigate the influence of architecture on the robustness, such as the

978-1-7281-8671-9/22/$31.00 ©2022 IEEE

adversarial robustness and out-of-distribution robustness from a NAS perspective. However, they do not consider the trade-off between the performance and robustness.

To answer the open problem, in this paper, we propose an **E**ffective, **E**fficient, and **R**obust **N**eural **A**rchitecture **S**earch method (E2RNAS) to design an architecture by explicitly balancing the trade-off among the performance, resource consumption and robustness. Specifically, we consider the validation accuracy for the effectiveness, the number of parameters and FLOPs for the efficiency, and the out-of-distribution robustness and adversarial robustness for the robustness. Built on Differentiable Architecture Search (DARTS) [1], the proposed E2RNAS formulates the entire objective function as a *multi-objective bi-level* optimization problem where the upper-level subproblem is a multi-objective optimization problem by considering the effectiveness, efficiency, and robustness. To solve the resultant problem, we propose a gradient-based optimization algorithm by combining the Multiple Gradient Descent Algorithm (MGDA) [8] and the bi-level optimization algorithm. In summary, the contributions of this paper are three-fold.

- We propose the E2RNAS method for searching effective, efficient, and robust network architecture, a practical DARTS-based framework for multi-objective NAS and can be seamlessly combined with DARTS and its variants.
- We formulate the objective function of the E2RNAS method as a multi-objective bi-level optimization problem and propose an efficient gradient-based algorithm to solve it.
- Experiments on benchmark datasets show that the proposed E2RNAS method can find robust architecture with less resource consumption and comparable classification accuracy.

II. Related Works

A. Multi-Objective Optimization

Multi-objective optimization aims to optimize more than one objective function simultaneously. Among different techniques to solve multi-objective problems, we are interested in gradient-based multi-objective optimization algorithms [8], which leverage the Karush-Kuhn-Tucker (KKT) conditions [9] to find a common descent direction for all objectives. In this paper, we utilize one such method, *i.e.* MGDA [8]. With n objective functions $\{\mathcal{L}_i(\boldsymbol{\theta})\}_{i=1}^n$ to be minimized, MGDA is an iterative method by first solving the following quadratic programming problem as

$$\min_{\gamma_1,\cdots,\gamma_n} \left\| \sum_{i=1}^{n} \gamma_i \nabla_{\boldsymbol{\theta}} \mathcal{L}_i(\boldsymbol{\theta}) \right\|_2^2 \text{ s.t. } \gamma_i \geq 0,\ \sum_{i=1}^{n} \gamma_i = 1, \quad (1)$$

where $\|\cdot\|_2$ denotes the ℓ_2 norm of a vector and γ_i can be viewed as a weight for the ith objective, and then minimizing $\sum_{i=1}^n \gamma_i \mathcal{L}_i(\boldsymbol{\theta})$ with respect to parameters $\boldsymbol{\theta}$. When convergent, the MGDA can find a Pareto-stationary solution.

B. Multi-Objective NAS

Due to the complex application scenarios in the real world, recent works on NAS take multiple objectives instead of only accuracy one objective into consideration, which implicitly indicates a multi-objective optimization problem. Specifically, to search an efficient architecture to be deployed in resource-limited platforms, some workers take the resource-constraint objectives such as the number of parameters, FLOPs, latency, and energy consumption into consideration [4]. Among those works, different techniques are applied to solve this multi-objective optimization problem. For example, some work [2] apply evolutionary algorithms to approximate the entire Pareto front, but the search cost is quite high. Some work [3] regard the combination of multiple targets like accuracy and latency as rewards to optimize the controller sampling an architecture from the search space using reinforcement learning algorithms. The most relevant to our work is the gradient-based method [5], [10], [11], especially the DARTS-based method [5], [10]. GOLD-NAS [10] regards the resource constraint like FLOPs as the regularization terms, whose coefficients gradually increase to prune the architecture during the search procedure. Built on DARTS [1], RC-DARTS [5] considers to search architecture with high accuracy while constraining the model size and FLOPs of the searched architecture within user-defined intervals. Therefore, the proposed objective function is formulated as a constrained optimization problem, and a projected gradient descent method is applied to solve it.

C. Robust Architecture Search

To search a robust architecture, some works [7], [12] investigate the influence of architecture on adversarial robustness from a NAS perspective and then discover a family of adversarially robust architecture based on their observations. Different with [7], [12], [13] consider adversarial robustness as an optimized objective to search architecture that can defend multiple types of adversarial attacks. This problem is formulated as a multi-objective optimization and solved by an evolutionary algorithm. Besides, similar with [7], [12], [6] study what topology of neural network architecture is best for out-of-distribution robustness.

III. The E2RNAS Method

In this section, we present the proposed E2RNAS method. We first give an overview of the DARTS method and then introduce how to achieve two kinds of robustness and formulate the objectives to constrain the resource cost, including the number of parameters and FLOPs in the searched architecture. Finally, we present the multi-objective bi-level formulation of the proposed E2RNAS method and its optimization.

A. Preliminary

DARTS [1] aims to learn a Directed Acyclic Graph (DAG) called cell, which can be stacked to form a neural network architecture. Each cell consists of N nodes $\{d_i\}_{i=0}^{N-1}$, each of which denotes a hidden representation. $\mathcal{O}$ denotes a discrete operation space. The edge (d_i, d_j) of the DAG represents

an operation function $o(\cdot)$ (*e.g.* skip connection or 3×3 pooling) from $\mathcal{O}$ with a probability $\alpha_o^{(i,j)}$ to perform at the node d_i. Therefore, we can formulate each edge (d_i, d_j) as a weighted sum function to combine all the operations in $\mathcal{O}$ as $f_{i,j}(d_i) = \sum_{o\in\mathcal{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o'\in\mathcal{O}}\exp(\alpha_{o'}^{(i,j)})} o(d_i)$. An intermediate node d_j is the sum of its predecessors, *i.e.* $d_j = \sum_{i<j} f_{i,j}(d_i)$. The output of the cell, *i.e.* node d_{N-1}, is the concatenation of all the output of nodes excluding the two input nodes d_0 and d_1. Therefore, $\boldsymbol{\alpha} = \{\alpha_o^{(i,j)}\}_{(i,j)\in\mathbf{E},\ o\in\mathcal{O}}$ can parameterize the searched architecture, where $\mathbf{E}$ denotes the set of all the edges from all the cells. Let $\mathcal{D}_{tr}$ and $\mathcal{D}_{val}$ denote the training dataset and validation dataset, respectively. DARTS is to solve a bi-level optimization problem as

$$\begin{aligned} &\min_{\boldsymbol{\alpha}}\ \mathcal{L}_{val}(\boldsymbol{\theta}^*(\boldsymbol{\alpha}), \boldsymbol{\alpha}) \\ &\text{s.t.}\ \ \boldsymbol{\theta}^*(\boldsymbol{\alpha}) = \arg\min_{\boldsymbol{\theta}}\ \mathcal{L}_{tr}(\boldsymbol{\theta}, \boldsymbol{\alpha}), \end{aligned} \tag{2}$$

where $\boldsymbol{\theta}$ denotes all the weights of the neural network, the average training loss of a neural network is represented by $\mathcal{L}_{tr}(\boldsymbol{\theta}, \boldsymbol{\alpha}) = \frac{1}{|\mathcal{D}_{tr}|}\sum_{(\mathbf{x},y)\in\mathcal{D}_{tr}}\ \ell(\boldsymbol{\theta}, \mathbf{x}, y)$ with parameter weight $\boldsymbol{\theta}$ and an architecture $\boldsymbol{\alpha}$ and $\ell(\boldsymbol{\theta}, \mathbf{x}, y)$ denotes the loss function for each sample. $\mathcal{L}_{val}(\boldsymbol{\theta}^*(\boldsymbol{\alpha}), \boldsymbol{\alpha})$ is defined similarly. Here $\min_{\boldsymbol{\alpha}} \mathcal{L}_{val}(\boldsymbol{\theta}^*(\boldsymbol{\alpha}), \boldsymbol{\alpha})$ is called the **Upper-Level (UL)** subproblem and $\min_{\boldsymbol{\theta}} \mathcal{L}_{tr}(\boldsymbol{\theta}, \boldsymbol{\alpha})$ is called the **Lower-Level (LL)** subproblem. When the search procedure finishes, the final architecture can be determined by the operation with the largest probability in each edge from each cell, *i.e.* , $o^{(i,j)} = \arg\max_{o\in\mathcal{O}}\ \ \alpha_o^{(i,j)}$.

B. Objective Functions for Robustness

In E2RNAS, we expect the searched architecture to be robust, which means that the trained model with the searched architecture can distinguish test samples whether from in- or out-of-distribution, and its performance is stable when adding some perturbations to the in-distribution samples. To improve the robustness of the searched architecture, we consider both Out-of-Distribution (OoD) robustness and adversarial robustness as objective functions in the UL subproblem.

1) Out-of-Distribution Robustness: Let $\mathcal{C}_{in}$ denote the label distribution of the training dataset $\mathcal{D}_{tr}$ and the validation dataset $\mathcal{D}_{val}$. We use an OoD dataset $\mathcal{D}_{ood}$ with its corresponding label distribution $\mathcal{C}_{out}$ and $\mathcal{C}_{in}\cap\mathcal{C}_{out} = \emptyset$ to measure the OoD robustness of the neural network. Following [14], we formulate the OoD robustness objective function as

$$\mathcal{L}_{ood}(\boldsymbol{\theta}, \boldsymbol{\alpha}) = \frac{1}{|\mathcal{D}_{ood}|}\sum_{\mathbf{x}\in\mathcal{D}_{ood}} \mathrm{KL}(\mathcal{U}\|\mathcal{S}(\mathbf{x}, \boldsymbol{\theta})), \tag{3}$$

where $\mathcal{U}$ represents a discrete uniform distribution, *i.e.* $\mathcal{U} = (\frac{1}{k}, \cdots, \frac{1}{k})$ if the neural network is a k-class classification model, $\mathcal{S}(\mathbf{x}, \boldsymbol{\theta})$ is the probability distribution of $\mathbf{x}$ predicted by the neural network with weights in $\boldsymbol{\theta}$, and $\mathrm{KL}(\cdot\|\cdot)$ denotes the Kullback-Leibler (KL) divergence to measure the distance between $\mathcal{U}$ and $\mathcal{S}(\mathbf{x}, \boldsymbol{\theta})$. Namely, the predictive distributions of OoD samples are forced to be uniform distributions. In this way, the maximum predictive probability of OoD samples is lower than in-distribution samples so that the neural network can distinguish in- and out-of-distribution to avoid the over-confidence in its predictions.

2) Adversarial Robustness: To evaluate the adversarial robustness of the neural network with the searched architecture $\boldsymbol{\alpha}$, we first perturb each data sample in the validation dataset $\mathcal{D}_{val}$ by PGD adversarial attack [15] to generate a perturbed validation dataset denoted by $\mathcal{D}_{val}^{adv}$. Then we compute the average loss on this perturbed dataset as

$$\mathcal{L}_{adv}(\boldsymbol{\theta}, \boldsymbol{\alpha}) = \frac{1}{|\mathcal{D}_{val}^{adv}|}\sum_{(\mathbf{x},y)\in\mathcal{D}_{val}^{adv}} \ell(\theta, \mathbf{x}, y). \tag{4}$$

C. Objective Functions of Resource Constraints

architecture with less resource consumption have more application scenarios even in resource-constrained mobile devices. Therefore, we regard resource constraints as the desired objectives and mainly focus on the number of parameters and multiply-add operations (*i.e.* FLOPs).

1) Number of Parameters: By following DARTS [1], we determine the operation on each edge of each cell in the final architecture as the one with the largest probabilities. So the number of parameters in an architecture can be computed as $\mathcal{N}(\boldsymbol{\alpha}) = \sum_{(i,j)\in\mathbf{E}}\ \ n_{\arg\max_{o\in\mathcal{O}}\ \alpha_o^{(i,j)}}$, where n_o denotes the number of parameters corresponding to the operation $o(\cdot)$. Note that $\arg\max$ is a non-differentiable operation, making the computation of the gradient of $\mathcal{N}(\boldsymbol{\alpha})$ with respect to $\boldsymbol{\alpha}$ infeasible. To make such operation differentiable, we use the softmax function to approximate the $\arg\max$ operation and then formulate it as $\hat{\mathcal{N}}(\boldsymbol{\alpha}) = \sum_{(i,j)\in\mathbf{E}}\sum_{o\in\mathcal{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o'\in\mathcal{O}}\exp(\alpha_{o'}^{(i,j)})} n_o$. Furthermore, to prevent the model to search over-simplified architecture (*i.e.* the one containing too many parameter-free operations) that leads to unsatisfactory performance, we add a lower bound L to the parameter size $\hat{\mathcal{N}}(\boldsymbol{\alpha})$. Therefore, the objective function of the number of parameters can be formulated as

$$\mathcal{L}_{nop}(\boldsymbol{\alpha}) = |\hat{\mathcal{N}}(\boldsymbol{\alpha}) - L|. \tag{5}$$

2) FLOPs: Similar to the number of parameters, FLOPs also only depend on the architecture $\boldsymbol{\alpha}$. Thus we formulate the objective function of FLOPs similarly to the number of parameters (*i.e.* Eq. (5)). First, we carefully compute the FLOPs f_o of each operation $o(\cdot)$ in $\mathcal{O}$. Then we use the softmax function to approximate the $\arg\max$ operation and calculate the total FLOPs $\hat{\mathcal{F}}(\boldsymbol{\alpha}) = \sum_{(i,j)\in\mathbf{E}}\sum_{o\in\mathcal{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o'\in\mathcal{O}}\exp(\alpha_{o'}^{(i,j)})} f_o$. Finally we constrain $\hat{\mathcal{F}}(\boldsymbol{\alpha})$ with a lower bound F and formulate the objective function of FLOPs as

$$\mathcal{L}_{flops}(\boldsymbol{\alpha}) = |\hat{\mathcal{F}}(\boldsymbol{\alpha}) - F|. \tag{6}$$

In practice, we can specify the minimum constraints L in Eq. (5) and F in Eq. (6) to search an architecture with an expected model size. On the other hand, we can also get a set of Pareto-optimal architecture by adjusting the minimum constraints as shown in Table I.

D. Multi-Objective Bi-Level Formulation

E2RNAS aims to search the architecture $\boldsymbol{\alpha}$ to minimize the validation loss for the effectiveness, the number of parameters and FLOPs for the efficiency, and the OoD robustness and adversarial robustness for the robustness. Thus, we combine Eqs. (3), (4), (5) and (6) to formulate the entire objective function as

$$\begin{aligned}\min_{\boldsymbol{\alpha}}\ & U(\boldsymbol{\theta}^*(\boldsymbol{\alpha}),\boldsymbol{\alpha}) = (\mathcal{L}_{val}(\boldsymbol{\theta}^*(\boldsymbol{\alpha}),\boldsymbol{\alpha}), \mathcal{L}_{ood}(\boldsymbol{\theta}^*(\boldsymbol{\alpha}),\boldsymbol{\alpha}),\\ & \qquad \mathcal{L}_{adv}(\boldsymbol{\theta}^*(\boldsymbol{\alpha}),\boldsymbol{\alpha}), \mathcal{L}_{nop}(\boldsymbol{\alpha}), \mathcal{L}_{flops}(\boldsymbol{\alpha}))\\ \text{s.t.}\ & \boldsymbol{\theta}^*(\boldsymbol{\alpha}) = \arg\min_{\boldsymbol{\theta}} (\mathcal{L}_{tr}(\boldsymbol{\theta},\boldsymbol{\alpha}) + \mathcal{L}_{ood}(\boldsymbol{\theta},\boldsymbol{\alpha})). \end{aligned} \tag{7}$$

where $\boldsymbol{\theta}^*(\boldsymbol{\alpha})$ indicates that the network weights $\boldsymbol{\theta}$ depends on the network architecture $\boldsymbol{\alpha}$. Objective loss functions $\mathcal{L}_{val}, \mathcal{L}_{ood}, \mathcal{L}_{adv}$ depend on the network weights and architecture. Objective loss functions $\mathcal{L}_{nop}, \mathcal{L}_{flops}$ only depend on the network architecture.

Problem (7) is similar to the bi-level optimization problem (2) in the original DARTS, where the LL subproblem is similar, but there exists significant differences in that the UL subproblem contains five objectives, as shown in Figure 1. On the other hand, different from RC-DARTS [5] that directly adds the resource constraint into the original DARTS objective function (2) as a constraint and formulates the objective function as a constrained optimization problem or GOLD-NAS [10] that regards resource constraint as the regularization term and then optimizes as a single-objective optimization problem, we evaluate all the objectives in the UL subproblem and cast it as a multi-objective optimization problem solving by a gradient-based multi-objective method, *i.e.* MGDA.

Therefore, problem (7) is a multi-objective bi-level optimization problem which is also a generalization of problem (2) in the DARTS and we use the MOML method [16] to solve it. It can be understood as a two-stage optimization. Firstly, when given an architecture parameter $\boldsymbol{\alpha}$, we can learn a model with optimal model weights $\boldsymbol{\theta}^*$ via the empirical risk minimization on both training dataset and OoD dataset. Secondly, given $\boldsymbol{\theta}^*$, the architecture parameter $\boldsymbol{\alpha}$ is updated on the validation dataset by making a trade-off among its performance, robustness, and resources consumption. Therefore, we can solve the problem (7) in two stages, which are described as follows.

a) Updating $\boldsymbol{\theta}$: Given the architecture parameter $\boldsymbol{\alpha}_t$, $\boldsymbol{\theta}$ can be simply updated as

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\theta}} (\mathcal{L}_{tr}(\boldsymbol{\theta}_t,\boldsymbol{\alpha}_t) + \mathcal{L}_{ood}(\boldsymbol{\theta}_t,\boldsymbol{\alpha}_t)), \tag{8}$$

where t denotes the index of the iteration and $\eta_{\boldsymbol{\theta}}$ denotes the learning rate.

b) Updating $\boldsymbol{\alpha}$: After obtaining $\boldsymbol{\theta}_{t+1}$, we can optimize the UL subproblem to update the architecture parameter $\boldsymbol{\alpha}$. As the UL subproblem is a multi-objective optimization problem, we adopt the MGDA to solve it. In MGDA, we first need to solve problem (1), which requires the computation of the gradients of the five objectives with respect to $\boldsymbol{\alpha}$. The gradient of $\mathcal{L}_{nop}(\boldsymbol{\alpha}_t)$ and $\mathcal{L}_{flops}(\boldsymbol{\alpha}_t)$ with respect to $\boldsymbol{\alpha}$ is easy to compute, while the gradient of the remaining three objectives with respect to $\boldsymbol{\alpha}$ is a bit complicate as $\boldsymbol{\theta}^*(\boldsymbol{\alpha}_t)$ is also a function of $\boldsymbol{\alpha}$ and it is too expensive to obtain $\boldsymbol{\theta}^*(\boldsymbol{\alpha}_t)$. Therefore, we use a second-order approximation as

$$\begin{aligned}&\nabla_{\boldsymbol{\alpha}} \mathcal{L}_{val}(\boldsymbol{\theta}^*(\boldsymbol{\alpha}_t),\boldsymbol{\alpha}_t)\\ \approx&\nabla_{\boldsymbol{\alpha}} \mathcal{L}_{val}(\boldsymbol{\theta}_{t+1} - \eta_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\theta}} \mathcal{L}_{tr}(\boldsymbol{\theta}_{t+1},\boldsymbol{\alpha}_t),\boldsymbol{\alpha}_t).\end{aligned} \tag{9}$$

where ξ denotes the learning rate for inner optimization. Obviously when $\eta_{\boldsymbol{\theta}} = 0$, $\boldsymbol{\theta}_{t+1}$ becomes an approximation of $\boldsymbol{\theta}^*(\boldsymbol{\alpha}_t)$ and Eq. (9) degenerates to the first-order approximation, which can speed up the gradient computation and reduce the memory cost but lead to worse performance [1]. So we use the second-order approximation in Eq. (9). Similarly, the gradient of $\mathcal{L}_{adv}(\boldsymbol{\theta}^*(\boldsymbol{\alpha}_t),\boldsymbol{\alpha}_t)$ and $\mathcal{L}_{ood}(\boldsymbol{\theta}^*(\boldsymbol{\alpha}_t),\boldsymbol{\alpha}_t)$ can be computed approximately. Then we solve the problem (1) to get the weight $\Gamma = (\gamma_1, \cdots, \gamma_5)$ for five objectives, respectively. While the problem (1) has no analytical solution, we apply the Frank-Wolfe algorithm [17] to solve it.

After that, we can update $\boldsymbol{\alpha}_t$ as

$$\boldsymbol{\alpha}_{t+1} = \boldsymbol{\alpha}_t - \eta_{\boldsymbol{\alpha}} \nabla_{\boldsymbol{\alpha}_t} \big(U(\boldsymbol{\theta}^*(\boldsymbol{\alpha}_t),\boldsymbol{\alpha}_t)\Gamma^T\big), \tag{10}$$

where $\eta_{\boldsymbol{\alpha}}$ denotes the learning rate for $\boldsymbol{\alpha}$. The whole algorithm is summarized in Algorithm 1.

Algorithm 1 E2RNAS

Input: Dataset $\mathcal{D}_{tr}$ and $\mathcal{D}_{val}$, OoD dataset $\mathcal{D}_{ood}$, batch size B, perturbation size ϵ, minimum constraint L and F, learning rates $\eta_{\boldsymbol{\alpha}}$ and $\eta_{\boldsymbol{\theta}}$

Output: Learned architecture parameter $\boldsymbol{\alpha}$

```
Randomly initialized α_0 and θ_0, t := 0;
while not converged do
    Sample a mini-batch of size B;
    Update θ_{t+1} according to Eq. (8);
    Compute five objective functions and the corresponding gradients;
    Compute weights Γ by solving problem (1);
    Update α_t according to Eq. (10);
    t := t + 1;
end while
```

IV. Experiments

In this section, we empirically evaluate the E2RNAS method on CIFAR-10 [18], CIFAR-100 [18] and ImageNet-1K [19] datasets.

A. Experimental Datasets

The CIFAR-10 dataset contains 50,000 training images and 10,000 testing images from 10 classes, each of which has 6,000 images with a 32×32 resolution in total. The CIFAR-100 dataset contains 100 classes grouped into 20 super-classes, with 500 training images and 100 testing images for each class. For the ImageNet dataset, we use the ImageNet-1K benchmark, which contains 1K high-level categories from the original 22K categories. For OoD test, we use the Street View House Numbers (SVHN) dataset [20], which consists of the images of house numbers captured from the Google street view

and contains 10 classes with 73,257 images used for training and 26,032 images for testing.

B. Implementation Details

1) Search Space: The search space adopts the same setting as DARTS [1]. There are two types of cells, *i.e*. the reduction cell and the normal cell. The reduction cells are located at the $1/3$ and $2/3$ of the total depth of the network. Other cells in the network belong to the normal cell. There are 7 nodes in each cell for both reduction and normal cells, including four intermediate nodes, two input nodes, and one output node. In both normal and reduction cell, the set of operations $\mathcal{O}$ contains eight operations, including 3×3 separable convolutions, 5×5 separable convolutions, 3×3 dilated separable convolutions, 5×5 dilated separable convolutions, 3×3 max pooling, 3×3 average pooling, identity, zero. For the convolution operator, the ReLU-Conv-BN order is used. Each separable convolution is applied twice.

2) Search Settings: In the search process, by following DARTS [1], half of the whole training set is used for training a model and the other half for validation. A small network of 8 cells is trained for 50 epochs with the batch size as 64 and initial channels as 16. For the **adversarial robustness** objective, the adversarial examples are generated by the FGSM attack follow the setting in [21] with the perturbation size $\epsilon = 2$. For the **OoD robustness** objective, we use the SVHN dataset [20] as the OoD dataset, which is introduced in Section IV-A. For the **FLOPs** objective, we use a function f_o to approximate the FLOPs of the corresponding operation $o(\cdot)$. Let C denote the number of channels of input and output channels for the operation. W and H denote the width and height of the feature map. Ignoring padding and bias, the FLOPs of an $\omega \times \omega$ separable convolution is $2(CHW+C^2HW+\omega^2CHW+2CHW)$ and the FLOPs of an $\omega \times \omega$ dilated separable convolutions is $CHW + C^2HW + \omega^2CHW + 2CHW$ with $stride = 1$. The FLOPs of a 3×3 average pooling is CHW. The FLOPs of a 3×3 max pooling, identity or none operation is zero. The FLOPs of a skip connection is 0 if $stride = 0$ and is $C^2HW+2\times CHW$ if $stride = 1$. The ADAM optimizer with the learning rate 3×10^{-4}, the momentum $\beta = (0.5, 0.999)$, and the weight decay 1×10^{-3} is used to update $\boldsymbol{\alpha}$ in the UL subproblem. The SGD optimizer with the momentum 0.9 and the weight decay 3×10^{-4} is used to update $\boldsymbol{\theta}$ in the LL subproblem. The proposed method is implemented in PyTorch 0.3.1, and all the experiments are conducted on one single NVIDIA Tesla V100S GPU.

3) Retrain Settings: Following DARTS [1], a large network of 20 cells is retrained on the full training set for 600 epochs, with the batch size as 96, the initial number of channels 36, a cutout of length 16, the dropout probability 0.2, and auxiliary towers of weight 0.4.

4) Evaluation Metrics: For the **performance** objective, the accuracy is tested on the full testing set. For the **adversarial robustness** objective, adversarial examples are generated using the PGD attack [15] with the perturbation size $\epsilon = 2/255$ on the full testing set. The PGD attack takes 10 iterative steps with the step size of 2.5ϵ as suggested in [22]. For the **OoD robustness** objective, we evaluate the effectiveness of distinguishing between in- and out-of-distribution samples by measuring the Area Under the Precision-Recall (AUPR) curve, which is a threshold-independent metric [14], [23] and the PR curve describes the relationship between precision and recall, where precision is computed by TP/(TP+FP), recall is computed by TP/(TP+FN), and TP, FP, TN, and FN denote true positive, true negative, false positive, and false negative. We specify the OoD images as positives to compute the AUPR in this paper.

C. Analysis on Experimental Results

1) Searched Architecture on CIFAR-10: The normal and reduction cells searched by the E2RNAS on the CIFAR-10 dataset are presented in Figure 2. The found reduction cell does not contain any operation with parameters, which reduces the parameter size of the architecture. Moreover, it is notable that both normal and reduction cells do not include the max pooling operation and begin with the average pooling operation, which indicates the found architecture is potentially OoD robust based on the observations in [6]. This also coincides with experimental results in Table I.

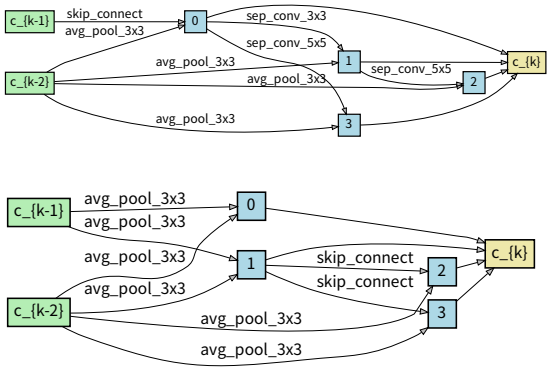

Fig. 2. The found normal cell (top) and reduction cell (bottom) on the CIFAR-10 dataset by the proposed E2RNAS method. This architecture corresponds to "E2RNAS-S1" in Table I.

2) Architecture Evaluation on CIFAR-10: The comparison of the proposed E2RNAS method with state-of-the-art NAS methods on the CIFAR-10 dataset is shown in Tables I and II. Two variants of the E2RNAS method denoted by E2RNAS-S1 and E2RNAS-S2 can obtain a set of Pareto-optimal architecture by adjusting L in Eq. (5), Specifically, we set $L = 2.5$ in Eq. (5) for E2RNAS-S1 and $L = 3.5$ for E2RNAS-S2. Notably, E2RNAS outperforms NAS methods [27], [28] by searching for a more lightweight architecture with lower search costs of three or four orders of magnitude and a comparable test error rate. Moreover, although ENAS [29] slightly outperforms E2RNAS in terms of the search time, it finds a much larger architecture with a higher test error. Compared with the original DARTS in [1], E2RNAS achieves a better trade-off among accuracy, efficiency, and robustness. Specifically, although the test error of E2RNAS is slightly higher than DARTS, E2RNAS can find more efficient and

TABLE I

COMPARISON WITH GRADIENT-BASED NAS METHODS ON THE **CIFAR-10** DATASET. † REPRESENTS TRAINING WITHOUT THE CUTOUT AUGMENTATION. ‡ INDICATES THE USE OF THE PROVIDED GENOTYPE IN THE ORIGINAL PAPER. ↑ (↓) INDICATES A LARGER (LOWER) VALUE IS BETTER. THE SEARCH COST IS RECORDED ON ONE SINGLE NVIDIA TESLA V100S GPU AND INCLUDES VALIDATION TIME WHILE SEARCHING. WE SET $L = 2.5$ IN EQ. (5) FOR E2RNAS-S1 AND $L = 3.5$ FOR E2RNAS-S2.

Architecture	Multiple-Objective	Test Err. (%) ↓	Params (M) ↓	FLOPs (M) ↓	PGD Acc. (%) ↑	OoD AUPR (%) ↑	Search Cost (GPU days) ↓
DARTS‡ [1]		2.59	3.3	539	25.68	31.44	0.6
P-DARTS‡ [24]		**2.37**	3.4	543	39.39	21.46	0.25
PC-DARTS‡ [25]		3.78	2.72	442	35.19	30.93	**0.24**
E2RNAS-S1	√	2.87	**2.55**	**425**	36.39	31.38	0.98
E2RNAS-S2	√	2.75	3.53	586	**42.81**	**34.64**	0.98

TABLE II

COMPARISON WITH VARIOUS ARCHITECTURE ON THE **CIFAR-10** DATASET. "RL", "EVO." AND "SMBO" STAND FOR REINFORCEMENT LEARNING-BASED, EVOLUTION-BASED AND SEQUENTIAL MODEL-BASED OPTIMIZATION NAS METHOD, RESPECTIVELY. "MO-G", "MO-RL" AND "MO-EVO." STAND FOR MULTI-OBJECTIVE GRADIENT-BASED, MULTI-OBJECTIVE REINFORCEMENT LEARNING-BASED AND MULTI-OBJECTIVE EVOLUTION-BASED NAS METHOD, RESPECTIVELY. "-" INDICATES THAT THE CORRESPONDING RESULT IS NOT REPORTED. FOR E2RNAS-S2, WE SET $L = 3.5$ IN EQ. (5).

Architecture	Test Err. (%) ↓	Params (M) ↓	Search Cost (GPU days) ↓	Search Method
DenseNet-BC [26]	3.46	25.6	-	manual
AmoebaNet-B [27]	2.55	2.8	3150	Evo.
PNAS [28]	3.41	3.2	225	SMBO
ENAS [29]	2.89	4.6	0.5	RL
LEMONADE [2]	3.05	4.7	80	MO-Evo.
Proxyless-R [11]	2.30	5.8	-	MO-RL
RAPDARTS [30]	2.83	2.8	12	MO-G
GOLD-NAS-K [10]	2.57	3.3	1.1	MO-G
RC-DARTS-C42 [5]	2.81	3.3	1	MO-G
FPNASNet [31]	3.01	5.76	-	MO-G
E2RNAS-S2	2.75	3.53	0.98	MO-G

robust architecture. For example, compared with DARTS, the E2RNAS-S1 architecture with only 2.55 MB model size and 425 MB FLOPs significantly improves the PGD accuracy by 11.11%, while the corresponding AUPR slightly decreases by 0.06%. Besides, although E2RNAS considers five objectives to trade-off, its search process is slightly slower than DARTS with only one objective. On the inference latency, DARTS is with 28.64 ms latency on a single NVIDIA Tesla V100S GPU, while E2RNAS-S1 is only with 19.42 ms and E2RNAS-S2 is with 27.49 ms on the same device. Therefore, those results indicate the efficiency of E2RNAS.

Even E2RNAS can achieve comparable performance with variants of DARTS such as P-DARTS [24], and PC-DARTS [25]. For example, compared E2RNAS-S2 with P-DARTS and PC-DARTS, E2RNAS can find an architecture with a higher AUPR and comparable performance on other objectives.

On the other hand, compared with multi-objective NAS methods, E2RNAS outperforms LEMONADE [2], RC-DARTS-C42 [5], RAPDARTS [30], and FPNASNet [31] by finding more lightweight and effective architecture in a shorter search time. Moreover, different from Proxyless-R [11] that searches for architecture with good performance but a large model size, E2RNAS can make a better trade-off between the accuracy and parameter size. Besides, E2RNAS achieves competitive performance with a faster search process when compared with GOLD-NAS-K [10].

In summary, experimental results in Table I show that E2RNAS can efficiently search a significantly robust architecture with a lower model size and comparable classification accuracy, compared with state-of-the-art NAS methods.

3) Experimental Results on CIFAR-100: We also evaluate the proposed E2RNAS method on the CIFAR-100 dataset. The comparison of E2RNAS with DARTS [1], P-DARTS [24] and PC-DARTS [25] is presented in Table III. The experimental results on the CIFAR-100 dataset are similar to that on the CIFAR-10 dataset in that E2RNAS can find a lightweight and robust architecture with a slightly decreased test accuracy. For example, compared with DARTS, E2RNAS-S1 with only 2.3 MB model size and 375 MB FLOPs can significantly improve the PGD accuracy and AUPR. Moreover, compared E2RNAS-S2 with P-DARTS and PC-DARTS, E2RNAS also outperforms them in the robustness with a slight drop in the test accuracy. These quantitative experiments indicate that E2RNAS can search robust architecture with lower resource consumption and comparable performance.

TABLE III

COMPARISON WITH STATE-OF-THE-ART NAS METHODS ON THE **CIFAR-100** DATASET. ‡ INDICATES THE USE OF THE PROVIDED GENOTYPE IN THE ORIGINAL PAPER. WE SET $L = 2.5$ IN EQ. (5) FOR E2RNAS-S1 AND $L = 3.5$ FOR E2RNAS-S2.

Architecture	Test Acc. (%) ↑	Params (M) ↓	FLOPs (M) ↓	PGD Acc. (%) ↑	OoD AUPR (%) ↑
DARTS‡ [1]	**83.94**	3.4	539	13.66	32.83
P-DARTS‡ [24]	83.54	3.5	543	17.34	32.43
PC-DARTS‡ [25]	83.06	3.7	568	18.44	32.75
E2RNAS-S1	79.39	**2.3**	**375**	**22.26**	32.88
E2RNAS-S2	81.70	3.9	643	21.21	**33.30**

D. The Generalization of E2RNAS

1) Combine with other gradient based NAS methods: A notable benefit of E2RNAS is its generalization ability, which means the proposed E2RNAS method can be seamlessly combined with other NAS methods, especially DARTS-based methods, to make a better trade-off among multiple objectives. To reveal it, we combine E2RNAS with two variants of DARTS, *i.e.* P-DARTS [24] and PC-DARTS [25], denoted

TABLE IV

EVALUATION OF THE GENERALIZATION ABILITY OF THE PROPOSED E2RNAS METHOD ON THE CIFAR-10 AND CIFAR-100 DATASETS. "{E2RNAS *on* #METHOD}" MEANS THE ARCHITECTURE SEARCHED BY COMBINING "METHOD" WITH E2RNAS. ‡ INDICATES THE PROVIDED GENOTYPE IN THE ORIGINAL PAPER. THE SEARCH COST IS RECORDED ON ONE SINGLE NVIDIA TESLA V100S GPU AND INCLUDES VALIDATION TIME WHILE SEARCHING.

Dataset	Architecture	Test Err. (%) ↓	Params (M) ↓	FLOPs (M) ↓	PGD Acc. (%) ↑	OoD AUPR (%) ↑
CIFAR-10	P-DARTS‡ [24]	**2.37**	3.4	543	39.39	21.46
	E2RNAS *on P-DARTS*	3.37	**3.2**	**509**	**47.85**	**29.82**
	PC-DARTS‡ [25]	**2.72**	3.6	568	35.19	26.07
	E2RNAS *on PC-DARTS*	3.78	**2.7**	**443**	**42.38**	**38.93**
CIFAR-100	P-DARTS‡ [24]	**16.46**	3.5	543	17.34	32.43
	E2RNAS *on P-DARTS*	17.51	**3.0**	**480**	**24.45**	**33.38**
	PC-DARTS‡ [25]	**16.94**	3.7	568	18.44	32.75
	E2RNAS *on PC-DARTS*	17.13	**3.1**	**492**	**24.37**	**34.64**

by "E2RNAS *on P-DARTS*" and "E2RNAS *on PC-DARTS*", respectively.

Although these two methods improve the search process of DARTS from different perspectives, their objective functions are still similar to the original DARTS, *i.e*. problem (2). Therefore, built on their method and following their experimental settings, we can adapt the proposed E2RNAS method to combine with them for further improvements, *i.e*. taking resource constraint and robustness into consideration and then reformulating their objective function as a multi-objective bi-level optimization problem similar to the problem (7) to further balance a trade-off among performance, resource constraint and robustness.

We evaluate those generalized methods and compare them with their corresponding original method on both the CIFAR-10 and CIFAR-100 datasets. The corresponding experimental results and comparison are presented in Table IV. The results show that E2RNAS can make a better trade-off among multiple objectives, which is similar to the results on DARTS, *i.e*. finding a more lightweight architecture with significantly increased robustness and slightly decreased test accuracy. Besides, we find that the search time only slightly increases when combining E2RNAS with P-DARTS and PC-DARTS, even by only 0.04 GPU days with P-DARTS on the CIFAR-10 dataset, which indicates the efficiency of E2RNAS.

2) Transfer Architecture to ImageNet-1K: We further transfer the architecture searched by E2RNAS on the CIFAR-10 dataset to the larger ImageNet-1K dataset. Table V shows that the E2RNAS-S2 method performs better than DARTS in terms of the test error and PGD accuracy. The trade-off between the accuracy and robustness is consistent with previous experiments. This experiment shows that the architecture searched by E2RNAS on a smaller dataset is transferable to a larger dataset while keeping the trade-off between different objectives.

E. Ablation Study and Discussion

This section studies how each design in E2RNAS influences its performance on different objectives. We first discuss the design of the LL subproblem of the problem (7) and then investigate the effectiveness of MGDA in the UL subproblem. The corresponding results are presented in Table VI.

TABLE V

TRANSFER SEARCHED ARCHITECTURE TO THE **IMAGENET-1K** DATASET. ‡ INDICATES THE USE OF THE PROVIDED GENOTYPE IN THE ORIGINAL PAPER.

Architecture	Test Err. (%) ↓	Params (M) ↓	FLOPs (G) ↓	PGD Acc. (%) ↑	OoD AUPR (%) ↑
DARTS‡	26.76	4.72	69.5	0.82	14.07
E2RNAS-S1	30.83	**3.76**	**55.4**	0.70	14.46
E2RNAS-S2	**26.07**	4.93	73.3	**4.72**	**14.54**

TABLE VI

ABLATION STUDY ON THE **CIFAR-10** DATASET UNDER THE SAME MINIMUM CONSTRAINTS L AND F IN EQ. (5) AND EQ. (6), RESPECTIVELY. MGDA IS APPLIED TO MAKE A TRADE-OFF AMONG MULTIPLE OBJECTIVES IN UL SUBPROBLEM AND IF WITHOUT MGDA (*i.e*. "E2RNAS *w/o MGDA*"), IT MEANS EQUAL WEIGHTS OF THE FIVE OBJECTIVES IN PROBLEM (7) ARE USED. "E2RNAS *w/ AT in LL*" DENOTES USING ADVERSARIAL TRAINING IN LL SUBPROBLEM AND "E2RNAS *w/o OoD in LL*" REPRESENTS THAT WE ONLY MINIMIZE $\mathcal{L}_{tr}(\boldsymbol{\theta}, \boldsymbol{\alpha})$ IN LL SUBPROBLEM.

Architecture	Test Err. (%) ↓	Params (M) ↓	FLOPs (M) ↓	PGD Acc. (%) ↑	OoD AUPR (%) ↑
E2RNAS	2.76	3.53	586	40.32	31.46
E2RNAS *w/o MGDA*	2.72	4.0	667	39.76	31.17
E2RNAS *w/ AT in LL*	3.34	3.0	476	52.40	32.53
E2RNAS *w/o OoD in LL*	3.26	2.7	439	39.22	31.37

1) Design of LL subproblem: E2RNAS aims to solve a multi-objective bi-level problem (7), where the UL subproblem optimizes the architecture $\boldsymbol{\alpha}$ by evaluating multiple objectives using the model $\boldsymbol{\theta}$ learned in the LL subproblem. Hence, a well-designed LL subproblem can improve the performance of our proposed E2RNAS method.

It is well known that adversarial training can significantly improve the adversarial robustness of the model but maybe result in a bad clean performance. Moreover, we find that the trade-off between performance and adversarial robustness exists in the NAS domain. The corresponding results are shown in Table VI. Comparing "E2RNAS *w/ AT in LL*" with E2RNAS, despite the PGD accuracy significantly increasing, the test accuracy also decreases, which fits the observation in [7]. Therefore, we do not apply adversarial training in the E2RNAS method to achieve comparable performance on classification accuracy.

Different from adversarial robustness, we find that training the model in the LL subproblem with OoD data samples (*i.e*. E2RNAS *vs*. "E2RNAS *w/o OoD in LL*" in Table VI) can not only improve the OoD metric but also make a better trade-off among other objectives, especially test error *vs*. model size. Hence, we add the OoD loss in the LL subproblem to achieve better performance.

2) Effectiveness of MGDA: The MGDA method is applied to solve the UL subproblem of the problem (7), which is a multi-objective problem to minimize five objectives. If without the MGDA method, it means that we solve the UL subproblem by minimizing an equally weighted sum of five objectives (*i.e*. $\Gamma = (0.2, \cdots, 0.2)$ in Eq. (10)). We quantitatively compare the performance of E2RNAS with and without MGDA (*i.e*. "E2RNAS" *vs*. "E2RNAS *w/o MGDA*" in Table VI) and find that solving with MGDA achieves much better results on the parameter size, FLOPs, OoD metric (*i.e*. AUPR) and PGD accuracy. So instead of using equal weights, applying MGDA can find a good solution of weights and make a trade-off among multiple objectives.

V. Conclusion

This paper proposes the E2RNAS method that optimizes multiple objectives simultaneously to search an effective, efficient, and robust architecture. The proposed objective function is formulated as a multi-objective bi-level problem, and we design an algorithm to integrate the MGDA with the bi-level optimization. Experiments demonstrate that E2RNAS can find robust architecture with optimized model size and comparable classification accuracy on various datasets. In our future study, we are interested in extending the proposed E2RNAS method to search for multiple Pareto-optimal architecture at one time.

Acknowledgements. This work is supported by NSFC key grant 62136005 and NSFC general grant 62076118.

References

[1] H. Liu, K. Simonyan, and Y. Yang, "DARTS: differentiable architecture search," in *Proceedings of the 7th International Conference on Learning Representations*, 2019.

[2] T. Elsken, J. H. Metzen, and F. Hutter, "Efficient multi-objective neural architecture search via lamarckian evolution," in *Proceedings of the 7th International Conference on Learning Representations*, 2019.

[3] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2820–2828.

[4] H. Benmeziane, K. E. Maghraoui, H. Ouarnoughi, S. Niar, M. Wistuba, and N. Wang, "A comprehensive survey on hardware-aware neural architecture search," *arXiv preprint arXiv:2101.09336*, 2021.

[5] X. Jin, J. Wang, J. Slocum, M.-H. Yang, S. Dai, S. Yan, and J. Feng, "Rc-darts: Resource constrained differentiable architecture search," *arXiv preprint arXiv:1912.12814*, 2019.

[6] R. Ardywibowo, S. Boluki, X. Gong, Z. Wang, and X. Qian, "NADS: neural architecture distribution search for uncertainty awareness," in *Proceedings of the 37th International Conference on Machine Learning*, vol. 119, 2020, pp. 356–366.

[7] C. Devaguptapu, D. Agarwal, G. Mittal, P. Gopalani, and V. N. Balasubramanian, "On adversarial robustness: A neural architecture search perspective," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2021, pp. 152–161.

[8] J.-A. Désidéri, "Multiple-gradient descent algorithm (MGDA) for multi-objective optimization," *Comptes Rendus Mathematique*, vol. 350, no. 5, pp. 313–318, 2012.

[9] H. W. Kuhn and A. W. Tucker, "Nonlinear programming," in *Traces and Emergence of Nonlinear Programming*. Springer, 2014, pp. 247–258.

[10] K. Bi, L. Xie, X. Chen, L. Wei, and Q. Tian, "Gold-nas: Gradual, one-level, differentiable," *arXiv preprint arXiv:2007.03331*, 2020.

[11] H. Cai, L. Zhu, and S. Han, "Proxylessnas: Direct neural architecture search on target task and hardware," in *Proceedings of the 7th International Conference on Learning Representations*, 2019.

[12] M. Guo, Y. Yang, R. Xu, Z. Liu, and D. Lin, "When NAS meets robustness: In search of robust architectures against adversarial attacks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2020, pp. 628–637.

[13] J. Liu and Y. Jin, "Multi-objective search of robust neural architectures against multiple types of adversarial attacks," *Neurocomputing*, vol. 453, pp. 73–84, 2021.

[14] K. Lee, H. Lee, K. Lee, and J. Shin, "Training confidence-calibrated classifiers for detecting out-of-distribution samples," in *Proceedings of the 6th International Conference on Learning Representations*, 2018.

[15] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Proceedings of the 5th International Conference on Learning Representations*, 2017.

[16] F. Ye, B. Lin, Z. Yue, P. Guo, Q. Xiao, and Y. Zhang, "Multi-objective meta learning," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[17] M. Frank and P. Wolfe, "An algorithm for quadratic programming," *Naval research logistics quarterly*, vol. 3, no. 1-2, pp. 95–110, 1956.

[18] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097–1105, 2012.

[20] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," 2011.

[21] E. Wong, L. Rice, and J. Z. Kolter, "Fast is better than free: Revisiting adversarial training," in *Proceedings of the 8th International Conference on Learning Representations*, 2020.

[22] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proceedings of the 6th International Conference on Learning Representations*, 2018.

[23] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," in *Proceedings of the 5th International Conference on Learning Representations*, 2017.

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

[25] Y. Xu, L. Xie, X. Zhang, X. Chen, G.-J. Qi, Q. Tian, and H. Xiong, "Pc-darts: Partial channel connections for memory-efficient architecture search," in *Proceedings of the 7th International Conference on Learning Representations*, 2019.

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

[27] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 4780–4789.

[28] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 19–34.

[29] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *Proceedings of the International Conference on Machine Learning*, 2018, pp. 4095–4104.

[30] S. Green, C. M. Vineyard, R. Helinski, and Ç. K. Koç, "RAPDARTS: resource-aware progressive differentiable architecture search," in *Proceedings of the International Joint Conference on Neural Networks*, 2020, pp. 1–7.

[31] J. Cui, P. Chen, R. Li, S. Liu, X. Shen, and J. Jia, "Fast and practical neural architecture search," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 6509–6518.