REP: An Interpretable Robustness Enhanced Plugin for Differentiable Neural Architecture Search

Yuqi Feng, *Graduate Student Member, IEEE*, Yanan Sun, *Senior Member, IEEE*, Gary G. Yen, *Fellow, IEEE*, and Kay Chen Tan, *Fellow, IEEE*

***Abstract*—Neural architecture search (NAS) is widely used to automate the design of high-accuracy deep architectures, which are often vulnerable to adversarial attacks in practice due to the lack of adversarial robustness. Existing methods focus on the direct utilization of regularized optimization process to address this critical issue, which causes the lack of interpretability for the end users to learn how the robust architecture is constructed. In this paper, we introduce a robust enhanced plugin (REP) method for differentiable NAS to search for robust neural architectures. Different from existing peer methods, REP focuses on the robust search primitives in the search space of NAS methods, and naturally has the merit of contributing to understanding how the robust architectures are progressively constructed. Specifically, we first propose an effective sampling strategy to sample robust search primitives in the search space. In addition, we also propose a probabilistic enhancement method to guarantee natural accuracy and adversarial robustness simultaneously during the search process. We conduct experiments on both convolutional neural networks and graph neural networks with widely used benchmarks against state of the arts. The results reveal that REP can achieve superiority in terms of both the adversarial robustness to popular adversarial attacks and the natural accuracy of original data. REP is flexible and can be easily used by any existing differentiable NAS methods to enhance their robustness without much additional effort.**

***Index Terms*—Neural architecture search, adversarial attacks, adversarial robustness, search space, robust search primitives.**

I. Introduction

DEEP neural networks (DNNs) have played a crucial role in addressing many real-world applications [1]. However, most DNN architectures are manually designed by domain experts, which is often time-consuming, labor-intensive and error-prone. Neural architecture search (NAS) [2] which aims at automatically designing DNN architectures without vast expertise have received extensive attention. Specifically, NAS automates the design of deep architectures by formulating it as an optimization problem, which is often discrete, bi-level, computationally expensive and with complicated constraints.

A traditional NAS method is composed of three parts: the search space, the search strategy, and the performance evaluation. Specifically, the search space defines all the possible DNN architectures, where the optimal one is found through the corresponding search strategy. The search strategy refers to the algorithm which is used to search the optimal architecture in the search space. Currently, the reinforcement learning (RL) [3], [4], [5], the evolutionary algorithm (EA) [6] and the differentiable algorithm [7] are the three popular search strategies to realize NAS. Moreover, the performance evaluation decides how efficient the NAS methods would be through providing a performance reward to the corresponding search strategy. In practice, the computational budget of the differentiable NAS is much smaller than that of RL-based and EA-based ones [7]. This is because a supernet is adopted in the performance evaluation phase of the differentiable NAS, and the performance of architectures is directly evaluated with the weights inherited from the trained supernet instead of training each of them like the RL-based and EA-based NAS do.

No matter which search strategy is utilized, the performance evaluation of existing NAS methods mainly targets on the natural accuracy of the resulted neural architectures, i.e., the performance of the neural architectures inputted with original data. When it comes to practical applications, such as deploying at a vulnerable environment, the adversarial robustness is also non-negligible and needs to be greatly concerned for the sake of safety. Specifically, the adversarial robustness refers to the accuracy of a neural network inputted with adversarial examples, which are often the original data perturbed by specific algorithms. As evidenced by a very recent literature [8], the neural architectures searched by NAS methods are vulnerable against most adversarial examples.

The adversarial robust NAS is an emerging topic, and the current research perspective is relatively simple-minded. Existing studies mainly focus on presenting different adversarial robustness metrics as regularization into the search strategy of NAS, and then the adversarial robustness of the architecture is directly achieved through the black-box optimization process. For example, Mok et al. [9] take the perspective of the input loss landscape. They indicate that a smooth input loss landscape can make the model more resistant to perturbations and results in a robust model. Consequently, the input loss landscape is regularized as an additional search target into the search process

Received 17 June 2024; revised 9 December 2024; accepted 14 February 2025. Date of publication 18 February 2025; date of current version 25 March 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 62276175, and in part by the Research and Develop Program of West China Hospital of Stomatology Sichuan University under Grant RD-03-202403. Recommended for acceptance by L. Nie. *(Corresponding author: Yanan Sun.)*

Yuqi Feng, Yanan Sun, and Gary G. Yen are with the College of Computer Science, Sichuan University, Chengdu 610065, China (e-mail: feng770623@gmail.com; ysun@scu.edu.cn; gyen@okstate.edu).

Kay Chen Tan is with the Department of Computing, The Hong Kong Polytechnic University,, Hong Kong (e-mail: kctan@polyu.edu.hk).

The source code is available at https://github.com/fyqsama/REP.

Digital Object Identifier 10.1109/TKDE.2025.3543503

for the robust NAS. Dong et al. [10] start from the Lipschitz constant and the adversarial robustness of architectures, and then explore the relationship between both. After that, they conclude that the smaller the Lipschitz constant is, the better adversarial robustness the models have. Finally, they improve the adversarial robustness by reducing the Lipschitz constant during the optimization process with an explicit regularization term. Moreover, Hosseini et al. [11] propose two adversarial robustness metrics based on the certificated lower bound and Jacobian regularization, and both metrics are enumerated and regularized to the objective functions of differentiable NAS to perform joint maximal optimization. In addition, Cheng et al. [12] design an adversarial noise estimator to generate adversarial examples under different attack strengths, and the adversarial losses of these examples are optimized together with the natural loss. As a result, the neural architectures with wide spectrum adversarial robustness can be determined.

Although these methods have demonstrated the effectiveness in their respective experiments, it is not clear why the resulting architectures are robust when they are used. Specifically, these methods mainly committed to search for robust neural architectures through optimization with additional regularization components measuring adversarial robustness. However, the optimization process is black-box with poor interpretability, and it cannot explain which elements lead to the improvement of adversarial robustness. This would inevitably harm the further development of robust NAS and the end users still have sufficient confidence to safely explore the resulting architectures in real-world practice. In this paper, we propose a plugin approach for robust NAS called Robust Enhanced Plugin (REP), which takes the perspective of improving interpretability regarding the construction of robust neural architectures. Different from previous researches that are often achieved via regularization terms, what we particularly concentrated in this work is the robust search primitives in the search space, which are beneficial for the adversarial robustness of the architecture. By constructing the final architecture upon the robust search primitives, it is easy to understand why the resulting architecture is robust and what cause the improvement of the adversarial robustness of the resulting neural architectures. As its name states, REP has the merit of good flexibility owing to its plugin nature and can be generally applied to various differentiable NAS methods which are very popular among the existing NAS methods. In summary, this paper makes the following contributions:

- We develop REP to reveal a new research perspective on developing robust NAS method focusing on robust search primitives, which tends to improve the interpretability on the construction of robust architectures. Precisely, we propose a sampling strategy for finding the robust search primitives in the search space. Because the primitives are the basic building blocks to construct the whole architecture, the details of generating a robust architecture become publicly available.
- We propose a probabilistic enhancement search strategy to additionally combat for the natural accuracy to original data. This is with the practical fact that existing work can only guarantee the adversarial robustness. In the proposed search strategy, the probability of the robust search primitives being selected is enhanced in the form of a distance metric, while other search primitives can still be selected to the architecture to guarantee the natural accuracy. Consequently, the adversarial robustness and the natural accuracy can be simultaneously achieved.
- We show REP with the characteristics for both euclidean data and non-euclidean data. In particular, we evaluated REP for convolutional neural network (CNN) on CIFAR-10, CIFAR-100, and ImageNet datasets, and graph neural network (GNN) on Cora, CiteSeer, and PubMed datasets against various kinds of popular adversarial attacks. In addition, the ablation study also proves that the designed components can positively improve the performance of REP. Furthermore, REP also demonstrates promising transferability in terms of both adversarial robustness and natural accuracy.

The remainder of this article is organized as follows. The background is introduced in Section II. Then, the preliminary and the proposed REP method are detailed in Section III. After that, the design and the results of experiments are demonstrated in Sections IV and V, respectively. Finally, the conclusion and discussion are presented in Section VI.

II. Related Work

To help readers better understand REP, the related work of NAS and adversarial attacks and defenses are detailed in Sections II-A and II-B, respectively.

A. Neural Architecture Search (NAS)

Generally, the NAS methods are mainly divided into three categories according to the different search strategies adopted: RL-based NAS [3], [13], EA-based NAS [6], [14], and differentiable NAS [7]. The earliest proposed NAS method [3] is based on RL, which first constructs a controller to generate the architecture representation, and then the corresponding architecture is trained from scratch and evaluated for obtaining reward. This process is repeated until the termination condition is met. After that, the final architecture is selected from the candidates. The EA-based NAS initially encodes a set of different architectures into population individuals, and then the fitness of each individual is evaluated to iteratively guide the heuristic search through genetic operators, until the architecture with the promising fitness is found conditioned with some criterion. When evaluating the fitness of individuals, the EA-based NAS needs to train each individual on the training set and get the validation accuracy of them on the validation set. For both kinds of NAS, the training process of each neural architecture leads to much computational budget of the RL-based NAS and EA-based NAS.

As for the differentiable NAS, they often relax the discrete NAS problem into a continuous one through a supernet, where the connection possibility of nodes is modeled as classification problems. After that, the gradient-based algorithm is adopted for solving. Owing to the efficiency and efficacy, the differentiable NAS methods currently dominate the development of NAS and have diverse variants [15], [16], and this is also the reason why

the work in this paper is dedicated to the differentiable NAS. Among the variants of the differentiable NAS, DARTS [7] is the most popular one. Specifically, there are nodes and edges organized as a directed acyclic graph (DAG) in the supernet of DARTS, where each node $x(i)$ is a latent representation (such as a feature map), and each edge (i, j) between nodes i and j is associated with an operation $o^{(i,j)}$ adopted on node i. DARTS constructs a mixed operation $\overline{o}^{(i,j)}$ on each edge formulated by (1):

$$\overline{o}^{(i,j)}(x) = \sum_{o \in \mathcal{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{(i,j)})} o(x), \tag{1}$$

where $\mathcal{O}$ represents all the candidate operations and $\alpha_o^{(i,j)}$ denotes the weight of operation o on edge (i, j). Therefore, the choice of operations on each edge is parameterized as a vector $\alpha^{(i,j)}$ with dimension $|\mathcal{O}|$. After that, the differentiable algorithm, such as the gradient-based algorithm, is used for learning a set of weights $\alpha = \{\alpha^{(i,j)}\}$. With the α learned, the operation on edge (i, j) can be obtained through (2):

$$o^{(i,j)} = \arg\max_{o \in \mathcal{O}} \alpha_o^{(i,j)}. \tag{2}$$

Finally, the resulting architecture is determined. Based on the idea of supernet in DARTS, some recent work has further explored the topology search in the supernet. For example, Gu et al. [17] decouple the operation and topology search in DARTS. Yang et al. [18] search for the topology based on the designed cells. Although these methods have achieved decent performance, they still suffer from the limitation of poor interpretability caused by the black-box properties of the gradient-based algorithms. Specifically, they directly used the optimized weights of search primitives to determine their merits. However, because the optimization process is a black-box process, they cannot know why the selected search primitives are beneficial, leading to the poor interpretability of the search process. This is also the motivation of REP in this paper.

B. Adversarial Attacks and Defenses

Adversarial attacks add imperceptible perturbations to the input data to check if the attacked neural network models are robust [11]. This is based on the observation that neural networks are easily fooled by adversarial examples [19].

In practice, the fast gradient sign method (FGSM) [20], projected gradient descent (PGD) [21], automatic projected gradient descent (AutoPGD) [22], and Carlini&Wagner (C&W) [23] attacks are often used to generate adversarial examples for CNNs, which is achieved by modifying a small number of pixels in the input image of CNN. Specifically, the FGSM attack first computes the gradient of the model on the input data, and then the direction of the gradient is determined. After that, the perturbation is generated by multiplying the direction of the gradient by the preset step size. Finally, the adversarial example is determined by adding the perturbation to the input data. Furthermore, the PGD attack employs a multi-step perturbation adding method. In each step, the direction of the gradient will be redetermined to add perturbations more precisely. Therefore, PGD attack is much stronger than FGSM attack. To further improve the PGD attack, AutoPGD adds the automatic adjustment of attack parameters to PGD. Specifically, AutoPGD can adjust the step size for each step in PGD. The experiments [22] have shown that AutoPGD attack is slightly stronger than PGD attack. Different from three methods introduced above, C&W attack takes the perspective of optimization to generate adversarial examples. The C&W attack minimizes the distance between the original data and the adversarial examples generated. Meanwhile, the probability of adversarial examples being classified into a wrong class is maximized. By solving this optimization problem, C&W can generate stronger adversarial examples than those generated by FGSM, PGD, and AutoPGD.

For GNNs, the DICE [24] and the node embedding [25] attacks are representative attacks, which aim to generate adversarial examples through adding or deleting edges in the original graph. Specifically, DICE generates adversarial examples by randomly deleting edges between the nodes of the same class and adding edges between the nodes of the different classes in the original graph. The percentage of add or delete operations performed on edges cannot exceed the predetermined perturbation rate. Moreover, the node embedding attack adds or deletes a certain number of edges in the original graph with the limit of the predetermined perturbation rate, in order to damage the embedding of the GNN learned from the input data. Then the performance of GNN suffers in the case of the poorly learned embedding. Although the attack schemes are different for CNNs and GNNs, the effectiveness of REP would not be affected when applied to both CNNs and GNNs. This is because REP only needs the values of adversarial robustness, which is independent to the particular attack schemes. Specifically, the robust search primitives are selected by comparing the adversarial robustness of architectures sampled in adjacent epochs. In the comparison process, only the values of the adversarial robustness are compared, no matter which scheme is adopted to obtain the values.

On the other hand, the adversarial defenses aim to improve the adversarial robustness of neural networks to resist adversarial attacks. The popular defense methods mainly include the adversarial example detection [26], the random smooth [27], the robust loss functions [28], [29], and the adversarial training [30], [31]. Among those, the adversarial training is widely used due to its simplicity and effectiveness [9]. Specifically, adversarial training first generates adversarial examples from the training data according to the trainable parameters of the model in the ith training epoch. After that, the adversarial examples are used to train the model, and the trainable parameters of the model in the $(i + 1)$th epoch will be obtained. This process is repeated until the end of the training. However, all these methods assume the neural network whose architecture has been established in advance, and then the adversarial robustness is obtained through the learning process driven by these methods. In this paper, the proposed work aims to improve the robustness of neural networks via NAS.

III. Methodology

In this section, we first present the preliminary in Section III-A. Then, the overall framework of the proposed REP method is shown in Section III-B. After that, the two main

components of REP are detailed in Sections III-C and III-D, respectively. Finally, the interpretability and effectiveness of REP are discussed in Sections III-E and III-F, respectively.

A. Preliminary

The proposed REP method is a plugin method, which relies on the specific search spaces of the differentiable NAS method to be plugged by REP. However, existing NAS methods have different search spaces with different definitions. In order to better use REP, the search spaces in REP are redefined, and the details are provided below.

In differentiable NAS methods, the search space is indicated by a supernet where nodes and edges are formed as a DAG. In REP, we define a search primitive as one edge e affiliated with an operation o and represented as (e, o). With this, the search space $\mathcal{S}$ can be redefined as nodes with a set of search primitives $\mathcal{S} = \{(e, o) \mid e \in \mathcal{E}, o \in \mathcal{O}\}$ where $\mathcal{E}$ denotes the set of all possible edges between nodes and $\mathcal{O}$ denotes the set of all the candidate operations on edges. Moreover, an architecture with n search primitives can be denoted as $\{b_1, b_2, \ldots, b_n \mid b_i \in \mathcal{S}, i = 1, 2, \ldots, n\}$. Based on these, we further give the definition of robust search primitives. Given two different pairs of architectures $\mathcal{A}_i$, $\mathcal{A}_j$ and $\mathcal{A}_k$, $\mathcal{A}_l$, the adversarial robustness of them satisfies $\mathcal{R}_i > \mathcal{R}_j$ and $\mathcal{R}_k < \mathcal{R}_l$. In this condition, if a search primitive b satisfies $b \in \mathcal{A}_i, \mathcal{A}_l$ and $b \notin \mathcal{A}_j, \mathcal{A}_k$, this search primitive is considered as a robust search primitive.

B. Algorithm Overview

Based on the definitions above, the overall framework of the proposed REP method is shown in Algorithm 1, where the contributions of this paper are in bold and italic. To begin with, the search space $\mathcal{S}$ and the search strategy $\mathcal{P}$ of the NAS method which is plugged by REP are given, and through a series of main components of REP, the architecture with both high natural accuracy and adversarial robustness is discovered. The main components of REP consist of three consequential parts: adversarial robustness evaluation (lines 3-13), robust search primitives sampling strategy (line 14), and probability enhancement search strategy (line 15).

In the first part, two empty sets $\mathcal{A}$ and $\mathcal{R}$ which will record the searched architectures and their respective adversarial robustness are created. Then, a counter i indicating the current search epoch is initialized (line 3). After that, the given NAS method is iteratively performed for N search epochs. In each search epoch i, the architecture $\mathcal{A}_i$ searched in this epoch is recorded and stored (lines 5-6). After that, the adversarial robustness of the architecture is evaluated and recorded (lines 7-11). In particular, to accelerate the evaluation process, if the architecture $\mathcal{A}_i$ is the same as an architecture $\mathcal{A}_j$ where $j < i$, its adversarial robustness $\mathcal{R}_i$ will be directly referred to the same value as $\mathcal{R}_j$ (line 8). Otherwise, the weights of the architecture $\mathcal{A}_i$ are directly inherited from the supernet, and then its adversarial robustness will be evaluated by the adversarial attack (line 10). In the second part, the proposed robust search primitives sampling strategy is performed upon the stored architectures and their adversarial robustness values to sample robust search primitives (line 14).

Algorithm 1: Overall Framework of REP.

Input: The search space $\mathcal{S}$ and search strategy $\mathcal{P}$ of the given NAS method, the target dataset, the search epoch N of $\mathcal{P}$.
Output: The searched $arch$.

1 $\mathcal{A} \leftarrow \emptyset$, $\mathcal{R} \leftarrow \emptyset$, $i \leftarrow 0$;
2 **while** $i < N$ **do**
3 $\mathcal{A}_i \leftarrow$ record the architecture $\mathcal{A}_i$ in the i-th epoch;
4 $\mathcal{A} \leftarrow \mathcal{A} \cup \mathcal{A}_i$;
5 **if** $\mathcal{A}_i = \mathcal{A}_j$ *recorded in the previous* j*-th epoch* **then**
6 $\mathcal{R}_i \leftarrow$ directly refer to the adversarial robustness $\mathcal{R}_j$;
7 **else**
8 Inherit the weights of $\mathcal{A}_i$ in the supernet and then evaluated the adversarial robustness $\mathcal{R}_i$ through adversarial attack;
9 **end**
10 $\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{R}_i$;
11 $i \leftarrow i + 1$;
12 **end**
13 Sample the robust search primitives $\mathcal{B}^*$ from $\mathcal{A}$ through the proposed ***sampling strategy for robust search primitives*** with $\mathcal{R}$;
14 $arch \leftarrow$ Search for the architecture in $\mathcal{S}$ via the proposed ***probability enhancement search strategy***;
15 **return** $arch$.

Finally, the final architecture is obtained by the proposed search strategy (line 15). For a better understanding of how REP works with differentiable NAS methods, an example is shown in Fig. 1 where the DARTS [7] algorithm is chosen as the target NAS method to be plugged by REP.

C. Sampling Strategy for Robust Search Primitives

In order to sample the robust search primitives from the pre-acquired architectures $\mathcal{A}$ with adversarial robustness values $\mathcal{R}$ efficiently, we propose a novel sampling strategy focusing on the relationship between the adversarial robustness of the architectures and the search primitives in them. The pseudo-code of the proposed sampling strategy is shown in Algorithm 2. In particular, given the set of architectures $\mathcal{A}$ with their adversarial robustness $\mathcal{R}$, through the robust search primitive sampling (lines 3-14), the robust search primitives are finally determined (line 15).

In the robust search primitives sampling process, the total number N of sampling iterations is first initialized to the number of architectures in $\mathcal{A}$ (line 3). Then, two empty sets $\mathcal{B}_1$ and $\mathcal{B}_2$ are initialized to store the sampled search primitives (line 4), and the counter of the iterations is set to one (line 5). Next, the sets of search primitives are obtained through a single traversal of $\mathcal{A}$ and $\mathcal{R}$ (lines 6-14). Specifically, given two adjacent architectures $\mathcal{A}_i$ and $\mathcal{A}_{i+1}$ with respective adversarial robustness $\mathcal{R}_i$ and $\mathcal{R}_{i+1}$, if the adversarial robustness $\mathcal{R}_i < \mathcal{R}_{i+1}$, the search primitives in architecture $\mathcal{A}_{i+1}$ but not appearing in $\mathcal{A}_i$, which are denoted

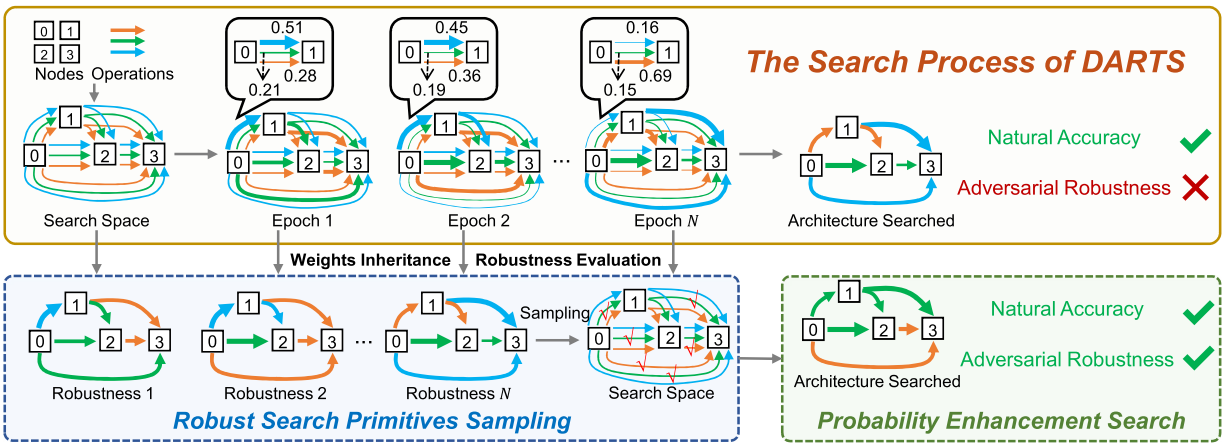


Fig. 1. An example of REP when DARTS is plugged by REP. The illustration contains two parts: the flow of DARTS algorithm (upper part) and the process of REP (lower part). In the upper part, given the pre-determined nodes and operations, through two main steps, i.e., the search space initialization and the search process, the architecture with high natural accuracy but poor adversarial robustness is determined. To be more visible, the search primitives with larger weights are denoted as thicker lines while the smaller weights are denoted as thinner lines. In the lower part, the proposed REP method first takes effects of the search process of DARTS, and the architecture with both high natural accuracy and adversarial robustness is searched through the two sequential parts highlighted in blue and green.

Algorithm 2: Sampling Robust Search Primitives.

Input: Set of architectures $\mathcal{A}$ with adversarial robustness $\mathcal{R}$ of them
Output: Set of robust search primitives $\mathcal{B}^*$

1 $N \leftarrow$ number of architectures in $\mathcal{A}$;
2 $\mathcal{B}_1, \mathcal{B}_2 \leftarrow \emptyset$;
3 $i \leftarrow 1$;
4 **while** $i < N$ **do**
5 **if** $\mathcal{R}_i < \mathcal{R}_{i+1}$ **then**
6 Add search primitives to $\mathcal{B}_1$ by $\mathcal{B}_1 = \mathcal{B}_1 \cup (\mathcal{A}_{i+1} - \mathcal{A}_i)$
7 **end**
8 **if** $\mathcal{R}_i > \mathcal{R}_{i+1}$ **then**
9 Add search primitives to $\mathcal{B}_2$ by $\mathcal{B}_2 = \mathcal{B}_2 \cup (\mathcal{A}_i - \mathcal{A}_{i+1})$
10 **end**
11 $i \leftarrow i + 1$;
12 **end**
13 Obtain the sampled robust search primitives $\mathcal{B}^*$ by $\mathcal{B}^* = \mathcal{B}_1 \cap \mathcal{B}_2$;
14 **return** $\mathcal{B}^*$

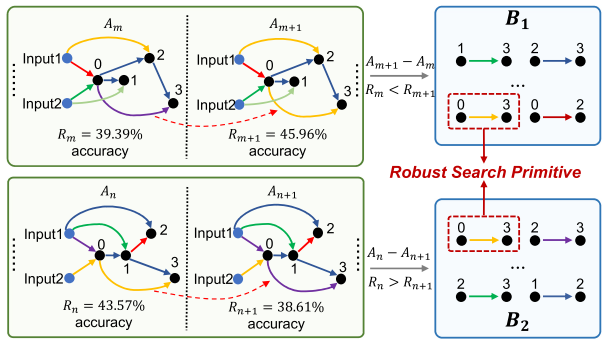
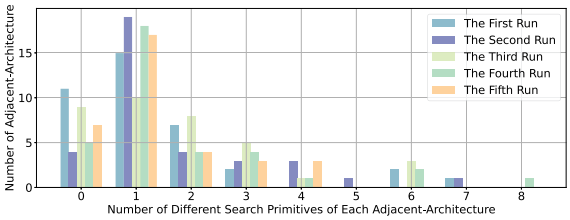


Fig. 2. An example of the proposed sampling strategy for robust search primitives. The illustration begins with two groups of adjacent architectures $\mathcal{A}_m, \mathcal{A}_{m+1}$ and $\mathcal{A}_n, \mathcal{A}_{n+1}$, through the proposed sampling strategy, the robust search primitives from these architectures are sampled in the end. The sampling strategy consists of two parts: the acquisition of sets $\mathcal{B}_1$ (the upper part) and $\mathcal{B}_2$ (the lower part). In the first part, taking the architectures $\mathcal{A}_m$ and $\mathcal{A}_{m+1}$ as an example, the adversarial robustness of them exists in the relationship $\mathcal{R}_m < \mathcal{R}_{m+1}$ and there are two different search primitives between these two architectures which are marked with a red dashed line. Following the sampling strategy, the search primitive between nodes 0 and 3 is added to $\mathcal{B}_1$. In the second part, the acquisition of $\mathcal{B}_2$ follows the same process. At the end, the robust search primitive in these architectures is selected by taking the intersection of $\mathcal{B}_1$ and $\mathcal{B}_2$.

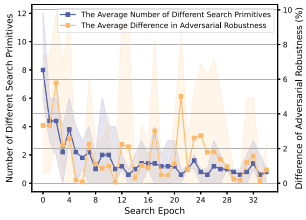
as $\mathcal{A}_{i+1} - \mathcal{A}_i$, are added to $\mathcal{B}_1$ (line 8). Otherwise, the search primitives in $\mathcal{A}_i - \mathcal{A}_{i+1}$ are added to $\mathcal{B}_2$ (line 11). Once the two sets $\mathcal{B}_1$ and $\mathcal{B}_2$ are obtained, at the end, the set of robust search primitives $\mathcal{B}^*$ is taken from the intersection of both $\mathcal{B}_1$ and $\mathcal{B}_2$ (line 15). For a clearer understanding of the proposed sampling strategy, we demonstrate an example of four neural architectures to show the process of robust search primitives sampling in Fig. 2.

The sampling process is the core step of the proposed REP method and the design of such a sampling strategy is based on our observations. Specifically, our observations focus on sets $\mathcal{A}$ and $\mathcal{R}$. For architectures in $\mathcal{A}$, two architectures in adjacent epochs have small changes between their search primitives. Meanwhile, we find that the adversarial robustness of architectures with adjacent indexes in $\mathcal{R}$ are usually different. In addition, we also observe that architectures in non-adjacent epochs are often quite different from each other.

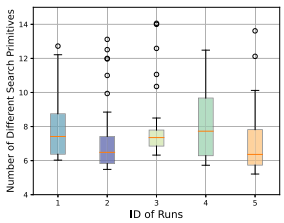
To support the above claims, we independently perform REP five times to sample robust search primitives during the search process of DARTS, and then visualize them in Fig. 3 for better demonstration. Specifically, in Fig. 3(a), the horizontal axis denotes the distribution regarding the number of different primitives of each adjacent-architecture, and the vertical axis refers to the corresponding number of adjacent-architectures. In Fig. 3(b), the horizontal axis denotes the search epoch, the left vertical axis (for the purple line) denotes the number of different primitives



(a) Distribution Regarding the Number of Different Primitives of Each Adjacent-Architecture and the Corresponding Number of Adjacent-Architectures



(b) Search Epoch and the Corresponding 1) Number of Different Primitives 2) Difference of Adversarial Robustness of Each Adjacent-Architecture



(c) The ID of Runs and the Corresponding Distribution Regarding the Number of Different Primitives of Non-Adjacent-Architecture

Fig. 3. Visualizations for different variables related to the sampling process of robust search primitives.

of each adjacent-architecture, the right vertical axis (for the orange line) denotes the difference of adversarial robustness of each adjacent-architecture. In Fig. 3(c), the horizontal axis denotes the ID of runs, the vertical axis denotes the distribution regarding the number of different primitives of each non-adjacent-architecture.

As shown in Fig. 3(a), there is the largest number of adjacent-architectures with only one different primitive. Meanwhile, there is also a large number of adjacent-architectures with zero or two different primitives. These observations demonstrate the adjacent-architectures are often similar. Furthermore, as shown by the orange line in Fig. 3(b), the difference of adversarial robustness shows relatively large changes among adjacent-architectures. Consequently, it is demonstrated that adjacent-architectures are similar but their adversarial robustness is not. Based on above observations, we analyze that, because the training settings for architectures are the same, the small changes of search primitives in adjacent architectures can be the reason for the change of adversarial robustness. Meanwhile, the small changes of search primitives are beneficial to narrowing the candidates when sampling robust search primitives, which can make the sampling strategy more effective. Therefore, the proposed sampling strategy spotlights on the adjacent architectures in $\mathcal{A}$.

In addition, we also explain the reason for not sampling robust search primitives in non-adjacent-architectures. As shown in Fig. 3(c), in each run, the minimum number of different primitives in non-adjacent-architectures is constantly above five, concentrating between six and ten. Obviously, non-adjacent-architectures have much more different primitives comparing with those of adjacent ones, and this is harmful to narrow the candidates of robust search primitives. Therefore, we do not sample robust search primitives in non-adjacent-architectures.

D. Probability Enhancement Search Strategy

To search the architecture with high natural accuracy and adversarial robustness based on the robust search primitives sampled, we propose a probability enhancement search strategy. Following the conventions of differentiable NAS [7], [32], we denote the architecture with a set of weights α and learn the weights in α to determine the resulting architecture.

Specifically, based on the pre-defined sets $\mathcal{E}$ and $\mathcal{O}$ detailed in our definitions in Section III-A, we can easily know that the α is a matrix with the dimension of $|\mathcal{E}| \times |\mathcal{O}|$. Because the number of edges in search space $\mathcal{S}$ is $|\mathcal{E}|$, and the number of candidate operations on each edge is $|\mathcal{O}|$. Moreover, each position (m, n) in α denotes the weight of an edge $e_m \in \mathcal{E}$ affiliated with a certain operation $o_n \in \mathcal{O}$, i.e., the weight of a search primitive (e_m, o_n).

Motivated by the above characteristics of α, we introduce a matrix α_R with the same dimension as α:

$$\alpha_R^{m,n} = \begin{cases} 1, & \text{if } (e_m, o_n) \text{ is a robust search primitive,} \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

As shown in (3), the positions of robust search primitives in α_R are set with one, while other positions are set with zero. Obviously, when the distance between α and α_R becomes smaller, the weight of robust search primitives in α will be larger, and more robust search primitives will be contained in the architecture. Therefore, we introduce a distance metric $\psi(\alpha)$ in (4) to denote the distance between α and α_R:

$$\psi(\alpha) = \|\alpha - \alpha_R\|_2^2. \tag{4}$$

With the defined distance metric $\psi(\alpha)$, we only need to minimize it in the search strategy and the final architecture will have more robust search primitives. moreover, we also need to ensure the natural accuracy of the architecture. The search process is formulated as (5) and (6):

$$\min_{\alpha} \quad \mathcal{L}_{val}(\omega^*(\alpha), \alpha) + \lambda\psi(\alpha), \tag{5}$$

$$\text{s.t.} \quad \omega^*(\alpha) = \arg\min_{\omega} \mathcal{L}_{train}(\omega, \alpha), \tag{6}$$

where $\mathcal{L}_{train}(\cdot)$ and $\mathcal{L}_{val}(\cdot)$ denote the loss on the training set and the validation set, λ is a tradeoff parameter which balances the validation loss of the architecture and the distance metric $\psi(\alpha)$ when optimizing α, and and ω denotes the set of trainable parameters of the network. By solving the above optimization problem, the architecture parameter α is determined once the search process is ended. Finally, the resulting architecture is determined based on α according to (2).

The addition of the distance metric $\psi(\alpha)$ in (5) can benefit two aspects. First, the natural accuracy of the derived architecture can be improved because $\psi(\alpha)$ can prevent the architecture parameter α from overfitting on the validation set. Specifically, as evidenced by the literature [33], purely minimizing the validation loss $\mathcal{L}_{val}(\omega^*(\alpha), \alpha)$ on the validation set can lead to the drop of natural accuracy in the later stage of the search process. The empirical results in this literature have also shown that such drop is caused by the overfitting of α on the validation set. Consequently, the distance metric $\psi(\alpha)$ can play the role

of regularization to prevent α from overfitting on the validation set. Therefore, the natural accuracy can be improved. Second, the adversarial robustness of the searched architecture is achieved at the same time when $\psi(\alpha)$ is introduced. This is because the inclusion of $\psi(\alpha)$ leads to more robust search primitives in the derived architectures, thus the adversarial robustness can be improved. The further experimental verification of this point is conducted in Section V-D.

Furthermore, it is possible that all operations for one node are all robust search primitives in an architecture. In this case, the proposed REP method selects the one which is most beneficial to natural accuracy. Specifically, these robust search primitives are viewed of the same importance in α_R (the corresponding values are all set to one) in the probability enhanced search process. Through the optimization shown in (5), given the probability of selecting each of these primitives is equally enhanced, the one which is most beneficial for minimizing the validation loss will be determined. This is because there are only two objectives in this optimization, i.e., the validation loss and the distance metric $\psi(\alpha)$ corresponding to α_R. In addition, there may be some other methods to handle the above case, such as giving some weights to those robust search primitives. These weights need to be determined by additional complex algorithms, and we will invest more effort to this point in future work.

E. Discussion on Interpretability of REP

To help the readers better understand the interpretability of REP, the corresponding discussions are presented in this section. The interpretability of REP can be divided into two aspects: the interpretability of the process and the results.

First, the interpretability of the process is mainly brought by the proposed probability enhancement search strategy. In the search strategy, we focus on the distance metric $\psi(\alpha)$ in (4) which is added to the objective function (5) with a weight λ. Meanwhile, $\psi(\alpha)$ is minimized together with the validation loss $\mathcal{L}_{val}$. With the minimized $\psi(\alpha)$, the weights of the robust search primitives in α will be greater and the resulting architecture will have more robust search primitives. In this process, though the optimization is still black-box, we can plainly know that how the metric $\psi(\alpha)$ added to the objective function leads the search process. In other words, it explains that, after adding this metric, which search primitives can be selected by the algorithm and used to construct the final architecture because of the $\psi(\alpha)$ added. This leads to the higher interpretability of the process than the existing robust NAS. For example, after searching the architecture based on the input loss landscape in AdvRush [9], we can only know the architecture searched has smooth input loss, but we cannot know how the input loss landscape leads the search process and which search primitives are selected by the algorithm due to the metric of input loss landscape added.

Second, the interpretability of the results mainly comes from the robust search primitives sampled by the proposed sampling strategy. After the search process, we can clearly know why the architecture searched is robust with the robust search primitives sampled. Because one can clearly observe the robust search primitives contained by the architecture searched by REP, which make the architecture robust. Furthermore, to show the robust search primitives do lead to the robustness improvement and then demonstrate the interpretability of the results, we conduct ablation studies in Section V-D. In contrast, when the robust architectures are searched by black-box optimization process in AdvRush [9], we cannot know that why this kind of design can lead to smoother input loss landscape and better adversarial robustness, and we can only know the architecture is robust. This leads to the poor interpretability of the results of the existing robust NAS methods.

F. Analysis for the Effectiveness of REP

The effectiveness of REP can be attributed to the adversarial robustness evaluation in the sampling process for robust search primitives. Specifically, the adversarial robustness evaluation is performed based on a certain (or more) adversarial attack. According to the results of the adversarial attack, the robust search primitives are sampled from architectures which demonstrate superior adversarial robustness under this attack. Consequently, the adversarial robustness of the derived architecture, which is composed of these robust search primitives sampled, will be improved under this attack. Furthermore, owing to the transferability and generalizability of adversarial attacks [34], an architecture which performs robustly under one kind of attack often also performs robustly under other kinds of attacks. As a result, though the robust search primitives are sampled from robust architectures under the certain attack, they are still beneficial to the adversarial robustness under other kinds of attacks. Therefore, the architectures derived by REP can achieve better results under different adversarial attacks. For example, the robust search primitives sampled based on FGSM or PGD attacks are still effective under the C&W attack, as evidence by the results shown in Table VII.

IV. Experimental Design

In this section, the settings of the experiments are introduced in detail. To begin with, the benchmark datasets are presented in Section IV-A. After that, the peer competitors chosen are introduced in Section IV-B. Finally, the parameter settings of the experiments are detailed in Section IV-C.

A. Benchmark Datasets

To demonstrate the proposed REP method is efficient for both euclidean data and non-euclidean data, the experiments are performed on popular benchmark datasets for both CNN and GNN. Following the conventions [12], [35], [36], the benchmark datasets CIFAR-10, CIFAR-100 [37], and ImageNet [38] are selected for CNN, and Cora, CiteSeer, and PubMed [39] are selected for GNN.

B. Peer Competitors

In order to show the superiority of the proposed REP method in terms of both natural accuracy and adversarial robustness, we choose the state-of-the-art models for the comparison. Specifically, following the convention of robust NAS community [9],

[10], the competitors selected by us fall into three categories: human-designed models, the models searched by representative differentiable NAS baselines and the models searched by state-of-the-art robust NAS methods.

For CNN, we choose the well-known human-designed models, i.e., ResNet [40] and DenseNet [41]. The differentiable NAS baselines are DARTS [7], PDARTS [15], and PCDARTS [42], which are representative differentiable NAS methods. The robust NAS methods are SDARTS-ADV [43], RobNet [44], RACL [10], and AdvRush [9], which can achieve the state-of-the-art performance on both natural accuracy and adversarial robustness.

For GNN, we also compare REP with human-designed models, the differentiable NAS baseline, and the robust graph NAS method. The human-designed models are GCN [45], GAT [46], and GraphSAGE [47], which are all well-known and commonly used GNN models. Specifically, all of these models are widely used for a variety of GNN-related tasks represented by the node classification task. Besides, we choose the popular SANE [32] method as the differentiable NAS baseline. In addition, the state-of-the-art robust graph NAS method G-RNA [35] is also chosen as the peer competitor. In the experiments, we referred to the official implementation of G-RNA in AutoGL [48] library for comparisons.

C. Parameter Settings

In our experiments, most of the parameter settings are determined according to the conventions in communities of differentiable NAS [7], [32] and robust NAS [9], [11], [49]. Specifically, the detailed settings for search and measurement and other implementation details are introduced as follows.

Parameter Settings for Search: In the probability enhancement search process, the parameter λ which balances the natural accuracy of the architecture and the distance metric $\psi(\alpha)$ is set to 0.01, by following the conventions in [9], [11]. Furthermore, it was also shown by AdvRush that the setting of 0.01 is effective in achieving both natural accuracy and adversarial robustness. Moreover, SGD [50] is used to optimize the network with the learning rate of 0.025, the momentum of 0.9 and the weight decay of $3e^{-4}$. Adam [51] is used to optimize the architecture parameters with the learning rate of $3e^{-4}$ and the weight decay of $1e^{-3}$. For CNN, the epoch is set to 50 and 10 for sampling and probability enhanced search, and the batch size is set to 64in order to fit in a single GPU. For GNN, the epoch of search process is set to 150 and 50 for the above two steps, respectively.

Parameter Settings for Measurement: After the search process, we choose the architecture with the highest natural accuracy on the validation set as the final architecture as the competitors do. To evaluate the performance of models in the presence and absence of adversarial defensive measures, the adversarial training and standard training are performed to train the models. For the adversarial training, CNN is trained for 200 epochs and GNN is trained for 600 epochs. For the standard training, CNN and GNN are both trained for 600 epochs. Following the conventions [7], [9], [32], [52], 20 cells are stacked to form the CNN architectures on CIFAR-10 and CIFAR-100, and the number of initial channels is set to 36. In terms of ImageNet, the number of cells is set to 14, and the number of initial channels is set to 48. In addition, during the training of CNNs, the learning rate is decayed by the factor of 0.1 at the 100th and 150th epochs. During the training of GNNs, the learning rate is decayed by cosine annealing. Moreover, the batch size of the adversarial training and the standard training for CNNs is set to 64 and 96, respectively. By following the suggestion in [33], the cutout operation [53] is used to prevent overfitting in the training processes. In addition, following the most common training method of the peer competitors reported in their seminal papers, SGD is used to optimize the weights in the network with learning rate of 0.025, momentum of 0.9 and weight decay of $3e^{-4}$.

After training, the natural accuracy and adversarial robustness are chosen to measure the performance of the models. In particular, the adversarial attacks for measurement are set with general settings of the peer competitors to be representative [9], [11], [24]. For CNN, FGSM is set with the total perturbation scale $\epsilon = 8/255$. C&W is set with attack iteration 100, the parameter $c = 0.5$ and learning rate 0.01. PGD attack is set with total attack iterations of 20, total perturbation scale $\epsilon = 8/255$ and single step size of $\epsilon = 2/255$. AutoPGD attack is set with total attack iterations of 20, total perturbation scale $\epsilon = 8/255$ with l_∞ bound and cross-entropy loss. For GNN, the random attack, DICE, and node embedding attack are used with 5% perturbation rates in the original graph. In addition, the harmonic robustness score (HRS) [8] is also adopted for evaluation. This is because HRS is calculated based on natural accuracy and adversarial robustness and can evaluate models in the combination of both aspects.

Other Implementation Details: The experimental codes were implemented on Python 3.7 and with the deep learning library PyTorch 1.6.0 [54]. For the implementation for GNN, we used the library named PyTorch Geometric [55]. When generating adversarial examples, we used the open-source adversarial attack toolboxes TorchAttacks [56] and DeepRobust [57] for CNN and GNN, respectively.

V. Experimental Results and Analysis

In this section, first of all, the overall results are presented in Section V-A. After that, the experimental results of REP under different adversarial attack strengths are shown in Section V-B. Then, the analysis on the proposed sampling strategy for robust search primitives is conducted in Section V-C. Finally, the results of extensive ablation studies are carried out in Section V-D.

A. Overall Results

To show the effectiveness of REP comprehensively, the set of architectures $\mathcal{A}$ are obtained on CIFAR-10 and Cora during 50 and 150 search epochs of DARTS [7] and SANE [32] for CNN and GNN, respectively. Then, the robust search primitives are sampled from $\mathcal{A}$ by REP. Finally, the derived architecture (denoted as REP for simplicity) is determined through REP based on the robust search primitives sampled. Notably, the experimental results are composed of four parts. In the first two

TABLE I
OVERALL RESULTS FOR CNN ON CIFAR-10, CIFAR-100, AND IMAGENET

Dataset & Method	Models	Params	Natural	Adversarial Attack				HRS(↑)	Search Cost (GPU days)
				FGSM	PGD^{20}	$APGD_{CE}$	C&W		
CIFAR-10, Adv	ResNet-18 [40]	11.2M	84.09	54.64	45.86	44.54	42.48	59.35	-
	DenseNet-121 [41]	7.0M	85.95	58.46	50.49	49.11	35.68	63.61	-
	NASNet [3]	4.3M	85.84	58.70	49.70	49.10	42.97	62.95	2,000
	AmoebaNet [6]	3.2M	85.97	59.85	51.71	51.20	39.21	64.58	3,150
	DARTS [7]	3.3M	85.17	58.74	50.45	48.32	41.25	63.37	0.5
	PDARTS [15]	3.4M	85.37	59.12	51.32	49.96	41.03	64.10	0.3
	SDARTS-ADV [43]	3.3M	85.64	59.18	49.93	49.22	28.09	63.08	1.3
	RobNet [44]	5.5M	82.79*	58.38*	52.74*	50.41	46.07	64.43	22.6
	RACL [10]	3.6M	84.63	58.57	50.62	49.42	44.13	63.35	0.5
	AdvRush [9]	4.2M	87.30*	60.87*	53.07*	51.83*	45.13	66.01	0.7
	WsrNet [12]	3.0M	83.94	56.12	47.17	46.46	23.34	60.40	4.0
	REP	3.6M	**87.57**	**62.45**	**54.37**	**53.70**	**48.02**	**67.09**	0.7
CIFAR-10, Std	DARTS [7]	3.3M	97.24*	50.56	0.06	0.01	0.04	0.12	0.5
	PDARTS [15]	3.4M	97.50*	54.51	0.27	0.13	0.10	0.54	0.3
	SDARTS-ADV [43]	3.3M	97.39*	52.00	0.19	0.06	0.07	0.38	1.3
	RACL [10]	3.6M	96.76	52.38	0.22	0.07	0.19	0.44	0.5
	AdvRush [9]	4.2M	**97.58***	55.91*	0.17	0.06	0.08	0.34	0.7
	REP	3.6M	96.94	**58.54**	**0.62**	**0.33**	**0.30**	**1.23**	0.7
CIFAR-100, Adv	DARTS [7]	3.3M	60.49	31.87	26.94	26.50	10.63	37.28	0.5
	PDARTS [15]	3.4M	58.41	30.35	25.83	25.03	21.78	35.82	0.3
	SDARTS-ADV [43]	3.3M	60.22	32.59	27.05	26.31	10.18	37.33	1.3
	RACL [10]	3.6M	59.18	34.40	29.82	28.98	16.14	39.66	0.5
	AdvRush [9]	4.2M	58.73*	39.51*	30.15*	29.02	20.08	39.84	0.7
	WsrNet [12]	3.0M	57.81	28.08	23.27	22.90	6.72	33.18	4.0
	REP	3.6M	**60.55**	**41.19**	**32.44**	**32.14**	**22.64**	**42.25**	0.7
ImageNet, Adv	DARTS [7]	4.5M	50.58	17.45	10.07	9.36	41.16	16.80	0.5
	PDARTS [15]	4.9M	51.85	18.70	10.80	10.01	42.18	17.88	0.3
	SDARTS-ADV [43]	4.8M	51.25	17.79	9.84	9.11	41.48	16.51	1.3
	RACL [10]	5.2M	51.59	18.15	10.49	9.80	41.98	17.43	0.5
	AdvRush [9]	5.8M	51.54	18.42	10.74	10.04	42.30	17.78	0.7
	WsrNet [12]	4.4M	50.93	17.58	10.15	9.42	35.57	16.93	4.0
	REP	5.2M	**52.37**	**19.51**	**11.37**	**10.49**	**42.42**	**18.68**	0.7

"Adv" and "Std" denote the architectures are trained with adversarial training or standard training. The results with "*" mean that they are directly referred to the results in their original papers. HRS is calculated based on the natural accuracy and the adversarial robustness under the PGD attack.

parts, following the conventions in [9], [10] and [11], all the models are trained with two methods: adversarial training and standard training. In the third and fourth part, to measure the transferability of the architectures, we keep all the CNN and GNN architectures the same as themselves on CIFAR-10 and Cora, and then they are adversarially trained on CIFAR-100, ImageNet, CiteSeer, and PubMed. Please note that the fast adversarial training technique [31] is adopted to reduce the computational cost for the adversarial training on ImageNet.

Convolutional Neural Network: We report the comparison results in Table I, where the architecture of REP is searched on CIFAR-10. The robust search primitives are sampled by PGD attack. The best values are shown in bold.

As can be seen in Table I, REP achieves the best natural accuracy and adversarial robustness under all four adversarial attacks in the case of adversarial training on CIFAR-10. In the case of standard training on CIFAR-10, REP achieves the natural accuracy which is higher than that of RACL, but lower than the other peer competitors. However, REP outperforms all the peer competitors in terms of the adversarial robustness. Meanwhile, REP also achieves the state-of-the-art performance on CIFAR-100 and large-scale classification task, i.e., ImageNet. In addition, REP achieves the best HRS metric in all four cases, indicating that REP is still effective when considering robust metric beyond the performance under adversarial attacks.

To further validate the consistency of REP, we also perform experiments on both NAS-Bench-101 [58] and NATS-Bench [59]. REP is also integrated into DARTS to search for architectures. For peer competitors, please note that we directly run the open source code of them on both NAS-Bench-101 search space and the topology search space $\mathcal{S}_t$ of NATS-Bench. The experimental results are shown in Table II. As can be seen, REP achieves the best performance on both NAS-Bench-101 and NATS-Bench. Meanwhile, REP achieves significant improvements against DARTS. Specifically, the performance drop of DARTS is caused by the domination of the parameter-free operation, i.e., the "skip connection" operation, in the derived architecture [33]. As a result, both natural accuracy and adversarial robustness drop significantly because the derived architecture is filled with such parameter-free operation. In contrast, REP can alleviate such domination in the derived architecture. This is because REP introduces the distance metric $\psi(\alpha)$, which can increase the probability of selecting other kinds of operations

TABLE II
OVERALL RESULTS FOR CNN ON NAS-BENCH-101 AND NATS-BENCH BENCHMARKS

Models	Natural	Adversarial Attack			
		FGSM	PGD^{20}	$APGD_{CE}$	C&W
NAS-Bench-101					
DARTS [7]	83.28	50.93	39.41	38.68	5.71
PDARTS [15]	83.82	50.08	38.91	38.14	3.73
SDARTS-ADV [43]	83.52	50.18	39.10	38.11	3.90
AdvRush [9]	82.93	51.19	40.14	39.49	6.80
REP	**86.01**	**55.97**	**46.05**	**45.33**	**15.13**
NATS-Bench					
DARTS [7]	38.64	24.04	23.56	23.02	34.69
PDARTS [15]	73.29	45.81	41.32	40.93	50.08
SDARTS-ADV [43]	71.47	45.77	40.28	39.85	51.42
AdvRush [9]	59.31	35.58	33.26	32.74	51.42
REP	**84.11**	**58.47**	**51.08**	**50.48**	**54.14**

TABLE III
OVERALL RESULTS FOR GNN ON CORA, CITESEER, AND PUBMED

Dataset & Method	Models	Natural	Adversarial Attack			HRS(↑)	Search Cost (GPU days)
			Random	DICE	NE		
Cora, Adv	GCN [45]	82.73±0.42	78.99±0.71	79.29±0.61	81.09±0.12	80.97	-
	GAT [46]	83.89±0.56	81.09±0.78	80.60±0.92	82.58±0.26	82.21	-
	GraphSAGE [47]	82.58±0.43	80.49±0.36	80.98±0.87	81.12±0.48	81.77	-
	SANE [32]	81.42±0.52	79.10±0.64	79.18±0.78	81.01±0.39	80.28	0.0008
	G-RNA [35]	83.81±0.39	83.15±0.59	82.96±0.24	80.64±0.22	83.38	0.093
	REP	**84.46±0.82**	**83.56±0.46**	**83.52±0.41**	**83.89±0.90**	**83.99**	0.0015
Cora, Std	GCN [45]	83.40±0.30	80.26±0.52	80.64±0.77	81.98±0.30	82.00	-
	GAT [46]	83.05±0.28	78.31±0.76	79.81±0.62	82.02±0.20	81.40	-
	GraphSAGE [47]	83.26±0.77	79.33±0.40	80.30±0.60	81.16±0.44	81.75	-
	SANE [32]	82.28±0.19	79.55±0.32	79.59±0.44	80.86±0.54	80.91	0.0008
	G-RNA [35]	83.14±0.26	82.13±0.66	82.26±0.28	82.58±0.46	82.70	0.093
	REP	**83.52±0.29**	**82.55±0.77**	**82.47±0.61**	**83.15±0.34**	**82.99**	0.0015
CiteSeer, Adv	GCN [45]	72.85±0.50	70.06±0.44	71.06±0.79	72.15±0.83	71.94	-
	GAT [46]	74.75±0.80	71.49±0.52	71.49±0.42	73.73±0.37	73.08	-
	GraphSAGE [47]	74.79±0.54	73.24±0.57	73.42±0.40	73.36±0.46	74.10	-
	SANE [32]	74.70±0.45	72.33±0.35	72.49±0.31	74.27±0.44	73.58	0.0008
	G-RNA [35]	75.51±0.26	74.39±0.62	73.79±0.16	74.89±0.12	74.64	-
	REP	**75.91±0.43**	**74.43±0.46**	**73.97±0.34**	**75.06±0.24**	**74.93**	0.0015
PubMed, Adv	GCN [45]	86.35±0.15	82.01±0.32	81.58±0.19	82.93±0.07	83.90	-
	GAT [46]	85.28±0.20	80.53±0.61	80.14±0.36	81.74±0.16	82.63	-
	GraphSAGE [47]	86.09±0.15	85.55±0.18	85.10±0.28	85.97±0.11	85.59	-
	SANE [32]	86.49±0.27	85.78±0.27	85.43±0.22	86.30±0.18	85.96	0.0008
	G-RNA [35]	87.48±0.12	85.88±0.26	85.53±0.19	86.77±0.01	86.49	-
	REP	**87.79±0.17**	**87.19±0.12**	**86.78±0.19**	**87.68±0.13**	**87.28**	0.0015

"NE" is short for "node embedding" which denotes the node embedding attack.

against the parameter-free operation. Consequently, the architecture derived by REP is not dominated by the parameter-free operation. This is also the reason that REP + DARTS performs much better than DARTS does.

Graph Neural Network: The comparison results of GNN in four cases are reported in Table III. Because the time overhead of the training for GNN is much smaller than that of CNN, the average values of five times of training with standard deviation are reported. Moreover, we report the natural accuracy and adversarial robustness under random attack, DICE attack, and node embedding attack of the models.

In terms of the adversarial and standard training on Cora, REP outperforms all the competitors chosen. Moreover, when transferring the models directly to CiteSeer, REP can still deliver the state-of-the-art performance among the competitors, demonstrating the superior transferability. Meanwhile, REP still achieves the best performance on PubMed which is larger than Cora and CiteSeer. In addition, REP also achieves the highest HRS metric in all four cases, further demonstrating the effectiveness of REP on GNN.

Search Cost Analysis: Because the search cost is an important factor indicating the efficiency of NAS, the search cost of REP and peer competitors is also contained in overall results. As shown in Table I, the search cost of REP is 0.7 GPU days for CNN, without introducing much additional search cost. Furthermore, we present the computational cost of each step of REP in Table IV. As can be seen, the cost of the original search is the same as that of DARTS (about 0.5 GPU days). The cost of the robust evaluation is 0.18 GPU days for CNN. In addition, the cost of the probability enhancement search process is 0.07 GPU days.

Moreover, as shown in Table III, the search cost of REP is 0.0015 GPU days for GNN. Although additional search time is introduced by REP, the search cost of REP is considered practical because the training and evaluation for GNN is much more efficient than those of CNN. As shown in Table IV, the computational cost of the robust search primitives sampling is 0.0013 GPU days for GNN in which robustness evaluation accounts for 0.0005 GPU days. In summary, REP can effectively improve the natural accuracy and adversarial robustness without introducing much computational budget.

TABLE IV
THE SEARCH COST OF EACH STEP OF REP FOR CNN AND GNN

	Steps	Search Cost (GPU days)
CNN	Original Search	0.46
	Robustness Evaluation	0.18
	Probability Enhanced Search	0.07
GNN	Original Search	0.0008
	Robustness Evaluation	0.0005
	Probability Enhanced Search	0.0002

B. Comparison Under Different Attack Strengths

Ideally, the models are attacked by the predetermined attack strength of each type of adversarial attack as shown in Section V-A. However, the strength of adversarial attacks is actually uncertain in applications. In order to evaluate the adversarial robustness more comprehensively, we perform adversarial attacks with different attack strengths to attack both CNN and GNN models chosen.

Specifically, we choose the popular PGD attack with different attack iterations to attack CNN models on CIFAR-10. The attack iteration is set from one to 50 in experiments. The results are presented in Fig. 4(a), where the horizontal axis indicates the attack iterations and the vertical axis represents the adversarial robustness. As shown in Fig. 4(a), when the attack iteration

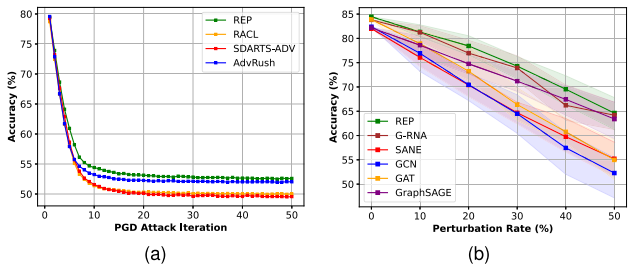


Fig. 4. Visualization results of models under different attack strengths. (a) CNNs on CIFAR-10. (b) GNNs on Cora.

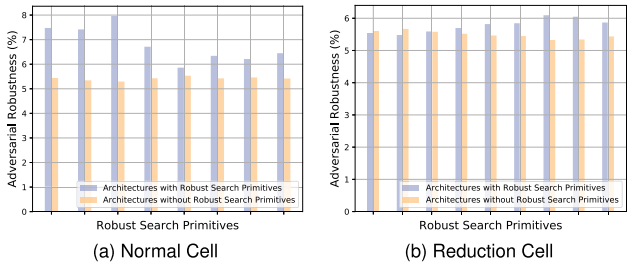


Fig. 5. The visualization of the average adversarial robustness under the PGD attack of the architectures sampled by REP.

of PGD attack is increasing, the performance of the models is decreasing due to the stronger attack. For each model, REP achieves the best adversarial robustness among all peer competitors under different attack iterations.

On the other hand, the DICE attack with different perturbation rates is performed to attack GNN models. In the experiment, the perturbation rate is set to 0%, 10%, 20%, 30%, 40%, and 50%. For each perturbation rate, we attack each model for 30 times and report the best, worst, and average performance, which are denoted as upper and lower boundaries of each color area, and the points on each line, respectively. The experimental results are shown in Fig. 4(b), where the horizontal axis indicates the perturbation rates and the vertical axis demonstrates the adversarial robustness. As can be seen, the mean performance of REP is the best among all the GNN models, showing that REP performs more robustly in face of the adversarial attack with the increasing attack strength.

C. Analysis on Robust Search Primitives Sampling Strategy

For an in-depth investigation for the effectiveness of robust search primitives sampled, we analyze the adversarial robustness of different architectures along with the robust search primitives sampled. In order to reflect the adversarial robustness of architectures with or without robust search primitives, the average adversarial robustness of both kinds of architectures are reported. The experimental results are shown in Fig. 5, where the vertical axis denotes the average adversarial robustness under the PGD attack of the architectures, and the horizontal axis denotes different search primitives contained by architectures.

As can be seen in Fig. 5(a) and (b), the architectures with robust search primitives often have relatively superior adversarial robustness, higher than the ones without robust search primitives. Meanwhile, giving that there are different combinations of search primitives in different architectures, we can come to the conclusion that the robust search primitives often have a positive effect on the adversarial robustness with different combinations of search primitives. Thus, the effectiveness of the sampled robust search primitives can be further justified.

TABLE V
THE EXPERIMENTAL RESULTS OF REP WITH DIFFERENT DIFFERENTIABLE NAS METHODS

Models	Params	Natural	Adversarial Attack			
			FGSM	PGD^{20}	$APGD_{CE}$	C&W
DARTS [7]	3.3M	85.17	58.74	50.45	48.32	41.25
DARTS + REP	3.6M	**87.57**	**62.45**	**54.37**	**53.70**	**48.02**
PDARTS [15]	3.4M	85.37	59.12	51.32	49.96	41.03
PDARTS + REP	4.1M	**87.89**	**62.48**	**52.80**	**51.90**	**53.93**
PCDARTS [42]	3.6M	85.93	57.88	48.37	47.75	39.25
PCDARTS + REP	3.6M	**88.27**	**62.48**	**53.68**	**52.86**	**43.30**

D. Ablation Study

The ablation study contains six parts in total. In particular, the first focuses on different NAS methods, to demonstrate the flexibility of REP as a plugin method. The second is about the different adversarial attacks for sampling robust search primitives, to investigate the effect of different adversarial attacks for robustness evaluation. The third is regarding the architecture pool construction methods, to justify the validity of the proposed architecture pool construction method in REP. The fourth aims at sets $\mathcal{B}_1$ and $\mathcal{B}_2$, to further explore the individual contribution of the sampled search primitives in $\mathcal{B}_1$ and $\mathcal{B}_2$. The fifth is about the distance metrics, to investigate the effectiveness of different distance metrics for regularization. The sixth places attention on different types of search primitives, to show the interpretability and the validity of the design of the probability enhancement search strategy of REP.

Different NAS Methods with REP: We choose three representative differentiable NAS methods: DARTS [7], PDARTS [15], and PCDARTS [42] to be integrated by REP. During the search process of these NAS methods, the architecture of each search epoch is recorded and evaluated, and then the robust search primitives are sampled by REP from these architectures. Finally, the resulting architectures are identified by REP and evaluated to do the comparison.

The experimental results of REP with three differentiable NAS methods are shown in Table V. In the column "Models", the baselines (DARTS, PDARTS, and PCDARTS) and their variants after adding REP (REP + DARTS, REP + PDARTS, and REP + PCDARTS) are presented. As can be seen, the natural accuracy is improved with the introduction of REP on the basis of all the baselines chosen. In addition, the adversarial robustness is improved under all adversarial attacks after REP is added. In conclusion, REP demonstrates the promising performance for different differentiable NAS methods as a plugin method.

Besides, we also perform REP on the single path one-shot (SPOS) NAS method [60]. In particular, we consider the choice blocks in the single path supernet of SPOS as search primitives, and REP is performed to sample robust search primitives in that supernet. The sampled robust search primitives are given larger

TABLE VI
THE EXPERIMENTAL RESULTS OF REP WITH THE ONE-SHOT NAS METHOD

Models	Params	Natural	Adversarial Attack			
			FGSM	PGD^{20}	$APGD_{CE}$	C&W
SPOS [60]	2.3M	81.74	55.83	49.13	48.40	43.19
SPOS + REP	2.2M	**82.76**	**56.03**	**51.27**	**50.83**	**49.40**

TABLE VII
THE EXPERIMENTAL RESULTS OF SAMPLING ROBUST SEARCH PRIMITIVES WITH DIFFERENT ADVERSARIAL ATTACKS

Models	Params	Natural	Adversarial Attack			
			FGSM	PGD^{20}	$APGD_{CE}$	C&W
DARTS [7]	3.3M	85.17	58.74	50.45	48.32	41.25
AdvRush [9]	4.2M	87.30*	60.87*	53.07*	51.83*	45.13
RobNet [44]	5.5M	82.79*	58.38*	52.74*	50.41	46.07
REP + FGSM	3.8M	86.42	61.38	53.26	52.71	**54.15**
REP + PGD	3.6M	**87.57**	**62.45**	**54.37**	**53.70**	48.02
REP + FGSM + PGD	3.7M	87.42	62.17	53.73	53.15	50.36

The results with "*" mean that they are referred directly to the experimental results in their original papers.

probability to be chosen in the search phase of SPOS. The results are shown in Table VI. As can be seen, both natural accuracy and adversarial robustness are improved after REP is integrated into SPOS, demonstrating the effectiveness of REP for the one-shot NAS method.

Different Adversarial Attacks for Sampling Robust Search Primitives: In this set of experiments, we perform REP in three cases. Specifically, the adversarial robustness is evaluated with: 1) only FGSM attack; 2) only PGD attack; 3) both FGSM and PGD attacks. In the third case, the adversarial robustness is calculated by averaging the evaluation results of FGSM and PGD attacks. The baseline NAS algorithm chosen to be plugged by REP is DARTS and the experimental results are shown in Table VII.

As can be seen in Table VII, the model "REP + PGD" achieves the best natural accuracy and adversarial robustness under all attacks except C&W. "REP + FGSM + PGD" achieves the second-best natural accuracy and adversarial robustness adversarial attacks. Meanwhile, "REP + FGSM" demonstrates the best adversarial robustness under C&W attack. According to above observations, we find that only using the stronger adversarial attack (PGD) for robust evaluation in REP can lead to higher adversarial robustness on most attacks (PGD, APGD, and FGSM). On the contrary, only using the weaker adversarial attack (FGSM) leads to lower adversarial robustness. In addition, the above situations show the opposite robustness under the C&W attack. This phenomenon can be explained by the cross-over mixture problem [61]. Specifically, when the adversarial attack becomes stronger (from REP + FGSM to REP + PGD), the cross-over mixture problem will become more serious. Meanwhile, the adversarial robustness under C&W attack is influenced by this problem relatively easily [61], comparing with that under FGSM, PGD, and APGD attacks. Consequently, the adversarial robustness under C&W attack diminishes.

Ablation Study of the Architecture Pool Construction Method: In this part, we conduct comparisons between the proposed construction method and the random sampling method. Please note that the comparisons are performed in three conditions where DARTS, PDARTS, and PCDARTS are chosen to be plugged by REP. In the random sampling process, the search primitives are all randomly selected at first from the search space to construct different architectures. In particular, there are 16 search primitives randomly selected to construct each architecture, and the total number of constructed architectures is kept the same as that recorded in the search process of DARTS, PDARTS or PCDARTS. Then, all the constructed architectures are adversarially trained to obtain their adversarial robustness. After that, the trained architectures are randomly sorted. Finally, the proposed sampling strategy of REP is adopted to sample robust search primitives in the sorted architectures. The comparison results are visualized in Fig. 6. As can be seen, no matter which differentiable NAS method is plugged by REP, the architecture pool construction method proposed in REP often outperforms the random sampling method in terms of all measurements.

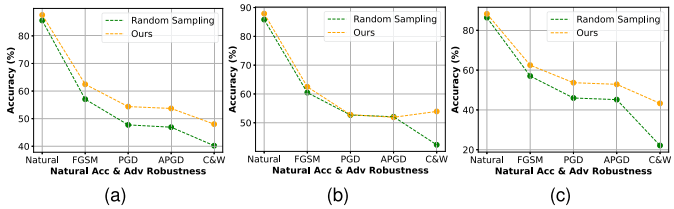


Fig. 6. Comparisons between the random sampling method and the architecture pool construction method proposed in REP. The differentiable NAS methods in which REP would plug are (a) DARTS, (b) PDARTS, and (c) PCDARTS.

TABLE VIII
THE EXPERIMENTAL RESULTS OF THE ABLATION STUDY ON THE SETS OF ROBUST SEARCH PRIMITIVES $\mathcal{B}_1$ AND $\mathcal{B}_2$

$\mathcal{B}_1$	$\mathcal{B}_2$	Natural	Adversarial Attack			
			FGSM	PGD	$APGD_{CE}$	C&W
×	×	85.17	58.74	50.45	48.32	41.25
×	✓	87.14	59.51	52.03	51.32	45.56
✓	×	87.22	59.39	51.87	51.18	41.32
✓	✓	**87.57**	**62.45**	**54.37**	**53.70**	**48.02**

Ablation Study of $\mathcal{B}_1$ and $\mathcal{B}_2$: In this part, the ablation study for $\mathcal{B}_1$ and $\mathcal{B}_2$ is conducted. The experimental results are shown in Table VIII, where the results from four cases are reported: 1) searching with none of $\mathcal{B}_1$ and $\mathcal{B}_2$ (original DARTS); 2) searching with $\mathcal{B}_2$ individually; 3) searching with $\mathcal{B}_1$ individually; 4) searching with both $\mathcal{B}_1$ and $\mathcal{B}_2$ (REP).

As can be observed from Table VIII, the natural accuracy values are all improved after introducing $\mathcal{B}_1$ or $\mathcal{B}_2$ individually or both $\mathcal{B}_1$ and $\mathcal{B}_2$ in the search process. This is because the distance metric $\psi(\alpha)$ can always play the regularization role no matter which set of search primitives ($\mathcal{B}_1$, $\mathcal{B}_2$ or the intersection of $\mathcal{B}_1$ and $\mathcal{B}_2$) are contained. Such regularization can prevent the architecture parameter α from overfitting on the validation set [33] and improve the natural accuracy of the searched architecture. In addition, the adversarial robustness under four adversarial attacks is also improved after introducing $\mathcal{B}_1$ or $\mathcal{B}_2$ individually or the intersection of $\mathcal{B}_1$ and $\mathcal{B}_2$. In particular, it reaches the best performance after searching with the intersection of $\mathcal{B}_1$ and $\mathcal{B}_2$. Consequently, the validity of the probability enhancement search strategy is further justified.

TABLE IX
THE EXPERIMENTAL RESULTS OF THE ABLATION STUDY IN TERMS OF DIFFERENT DISTANCE METRICS FOR REGULARIZATION

Metrics	Params	Natural	Adversarial Attack			
			FGSM	PGD	$APGD_{CE}$	C&W
None	3.3M	85.17	58.74	50.45	48.32	41.25
Cosine Distance	4.0M	86.77	60.81	51.73	51.11	45.81
KL Divergence	3.5M	87.35	60.30	53.08	52.68	45.51
Euclidean Distance (REP)	3.6M	**87.57**	**62.45**	**54.37**	**53.70**	**48.02**

The experiments are performed on CIFAR-10.

TABLE X
THE RESULTS OF THE ABLATION STUDY FOR ROBUST SEARCH PRIMITIVES AND OTHER SEARCH PRIMITIVES

Methods			Params	Natural	Adversarial Attack			
AdvT	Robust	Other			FGSM	PGD^{20}	$APGD_{CE}$	C&W
×	×	✓	2.8M	96.90	48.98	0.11	0.07	0.17
×	✓	×	1.7M	96.47	32.78	0.23	0.05	0.12
×	✓	✓	3.6M	**96.94**	**58.54**	**0.62**	**0.33**	**0.30**
✓	×	✓	2.8M	87.30	60.64	53.99	53.41	39.65
✓	✓	×	1.7M	85.41	59.40	50.81	49.96	33.97
✓	✓	✓	3.6M	**87.57**	**62.45**	**54.37**	**53.70**	**48.02**

Different Distance Metrics for Regularization: To explore the effectiveness of different distance metrics for regularization, we choose three kinds of distance metrics (i.e., euclidean distance, cosine distance, and KL divergence) to perform experiments. The experimental results are presented in Table IX.

As can be seen from Table IX, no matter which distance metric is adopted to the regularization, the natural accuracy and adversarial robustness are both improved. Meanwhile, the results of the euclidean distance are better than those of cosine distance and KL divergence in terms of both natural accuracy and adversarial robustness. This is also the reason for choosing the euclidean distance for regularization in this work. In addition, the KL divergence is shown to be slightly more effective than the cosine distance in regularization.

Robust Search Primitives & Other Search Primitives: We conduct ablation studies on robust search primitives and other search primitives in this part. Although CNN and GNN are used to handle different types of data (i.e., continuous image data and discrete graph data, respectively) and the robust strategy on image usually fails on graph-structured data [62], the architectures of both CNN and GNN can be represented in the same form, i.e., DAGs. Meanwhile, the robust search primitives are edges along with operations. Giving that REP focuses on the architectures and search primitives. Therefore, REP can be generalized across CNN and GNN no matter what kind of data they are used to handle.

Furthermore, we conduct empirical studies for the robust search primitives and other search primitives. For the CNN architectures derived by REP, we remove some search primitives in three cases so that the architecture remains: 1) Only other search primitives except robust ones; 2) Only robust search primitives; 3) Both robust and other search primitives. When removing a certain search primitive, we replace the operation of this search primitive with zero operation to achieve the deletion. All three architectures are trained with adversarial training and standard training with the same settings on CIFAR-10 and then evaluated.

The experimental results are demonstrated in Table X. To be more specific, "AdvT" indicates whether to perform adversarial training or not, "Robust" and "Other" denote whether to reserve robust search primitives and other search primitives or not in the first three columns of Table X.

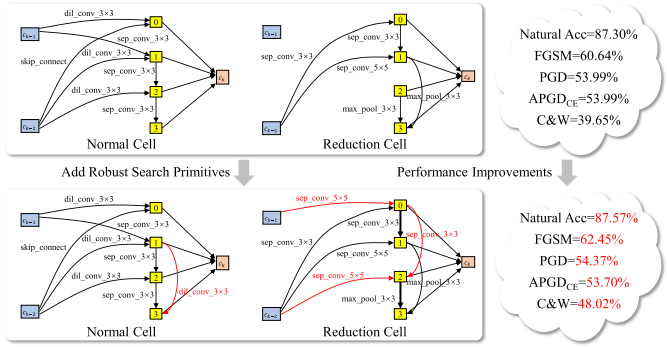


Fig. 7. The visualization of the CNN cells corresponding to the results in Table X. The cells in the upper part do not include robust search primitives sampled. The cells in the lower part includes the robust search primitives sampled. The robust search primitives are shown as red arrows.

First, we answer the research question that "*which elements lead to the improvement of adversarial robustness*". As shown in Table X, comparing with the architecture which does not include any robust search primitives (the first and fourth rows), both natural accuracy and adversarial robustness are improved after including robust search primitives (the third and sixth rows). In other words, the architecture with the sampled robust search primitives is considered more robust. Furthermore, we also visualize both architectures with their corresponding performance in Fig. 7. Specifically, the red arrows denote the robust search primitives sampled. When the architecture includes robust search primitives sampled, it will have higher natural accuracy and adversarial robustness. In summary, the sampled robust search primitives can lead to the improvement of adversarial robustness.

Second, for the validity of the design of probability enhancement search strategy, when only reserving robust search primitives in the search process (the second row and the fifth row), both natural accuracy and adversarial robustness of the architecture are inferior to those of the architecture in the third and sixth rows. Moreover, the architecture containing only robust search primitives even performs worse than the architectures (the first row and the fourth row) in terms of most measurements. This is because robust search primitives sampled are only a small part of all primitives as shown in Fig. 7. When searching for the architectures based on a small number of search primitives, the number of operations and edges significantly reduces in the searched architecture. Obviously, this will lead to the performance fall. Based on above analysis, the design of the proposed probability enhancement search strategy which searches architectures with both robust and other search primitives is shown to be reasonable.

VI. CONCLUSION AND DISCUSSION

The goal of this paper is to design a robust enhanced plugin method (REP), which can be flexibly used for multiple differentiable NAS methods to enhance the adversarial robustness. This goal is achieved by developing the robust search primitives sampling and the probability enhancement search strategy. The

proposed REP method is shown to be effective to both euclidean data and non-euclidean data on popular benchmark datasets, along with decent flexibility and interpretability. However, REP mainly focuses on differentiable NAS methods to search for cell-based architectures. In this case, only cells stacked to form a whole architecture can be searched, resulting in the generation of similar architectures. In future, we will place more efforts in improving REP for generating diverse architectures, in order to adapt REP to a wider range of real-world tasks. As for the potential negative social impact, the automation brought by REP may increase unemployment in the neural network design direction.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *J. Mach. Learn. Res.*, vol. 20, no. 1, pp. 1997–2017, 2019.

[3] B. Zoph and Q. Le, "Neural architecture search with reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=r1Ue8Hcxg

[4] Y. Gao et al., "HGNAS: Efficient architecture search for heterogeneous graph neural networks," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 9, pp. 9448–9461, Sep. 2023.

[5] Y. Gao et al., "GraphNAS : Distributed architecture search for graph neural networks," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 7, pp. 6973–6987, Jul. 2023.

[6] E. Real et al., "Large-scale evolution of image classifiers," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2902–2911.

[7] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *Proc. Int. Conf. Learn. Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=S1eYHoC5FX

[8] C. Devaguptapu, D. Agarwal, G. Mittal, P. Gopalani, and V. N. Balasubramanian, "On adversarial robustness: A neural architecture search perspective," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 152–161.

[9] J. Mok, B. Na, H. Choe, and S. Yoon, "AdvRush: Searching for adversarially robust neural architectures," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 12322–12332.

[10] M. Dong, Y. Li, Y. Wang, and C. Xu, "Adversarially robust neural architectures," 2020, *arXiv: 2009.00902*.

[11] R. Hosseini, X. Yang, and P. Xie, "DSRNA: Differentiable search of robust neural architectures," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 6196–6205.

[12] Z. Cheng, Y. Li, M. Dong, X. Su, S. You, and C. Xu, "Neural architecture search for wide spectrum adversarial robustness," in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 442–451.

[13] Z. Pan, L. Hu, W. Tang, J. Li, Y. He, and Z. Liu, "Privacy-preserving multi-granular federated neural architecture search–a general framework," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 3, pp. 2975–2986, Mar. 2023.

[14] C. Wang, B. Chen, G. Li, and H. Wang, "Automated graph neural network search under federated learning framework," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 10, pp. 9959–9972, Oct. 2023.

[15] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive DARTS: Bridging the optimization gap for NAS in the wild," *Int. J. Comput. Vis.*, vol. 129, no. 3, pp. 638–655, 2021.

[16] C. Yan et al., "ZeroNAS: Differentiable generative adversarial networks search for zero-shot learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 12, pp. 9733–9740, Dec. 2022.

[17] Y.-C. Gu et al., "DOTS: Decoupling operation and topology in differentiable architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 12311–12320.

[18] Y. Yang, S. You, H. Li, F. Wang, C. Qian, and Z. Lin, "Towards improving the consistency, efficiency, and flexibility of differentiable neural architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 6667–6676.

[19] C. Szegedy et al., "Intriguing properties of neural networks," 2013, *arXiv:1312.6199*.

[20] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. Int. Conf. Learn. Representations*, 2015.

[21] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proc. Int. Conf. Learn. Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=rJzIBfZAb

[22] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 2206–2216.

[23] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 39–57.

[24] D. Zügner and S. Günnemann, "Adversarial attacks on graph neural networks via meta learning," in *Proc. Int. Conf. Learn. Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=Bylnx209YX

[25] A. Bojchevski and S. Günnemann, "Adversarial attacks on node embeddings via graph poisoning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 695–704.

[26] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proc. ACM Workshop Artif. Intell. Secur.*, 2017, pp. 3–14.

[27] J. Cohen, E. Rosenfeld, and Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 1310–1320.

[28] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, and N. Usunier, "Parseval networks: Improving robustness to adversarial examples," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 854–863.

[29] D. Jakubovitz and R. Giryes, "Improving DNN robustness to adversarial attacks using jacobian regularization," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 514–529.

[30] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," in *Proc. Int. Conf. Learn. Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=rkZvSe-RZ

[31] E. Wong, L. Rice, and J. Z. Kolter, "Fast is better than free: Revisiting adversarial training," in *Proc. Int. Conf. Learn. Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=BJx040EFvH

[32] Z. Huan, Y. Quanming, and T. Weiwei, "Search to aggregate neighborhood for graph neural network," in *Proc. IEEE Int. Conf. Data Eng.*, 2021, pp. 552–563.

[33] A. Zela, T. Elsken, T. Saikia, Y. Marrakchi, T. Brox, and F. Hutter, "Understanding and robustifying differentiable architecture search," in *Proc. Int. Conf. Learn. Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=H1gDNyrKDS

[34] Z. Wang et al., "Towards transferable targeted adversarial examples," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 20534–20543.

[35] B. Xie et al., "Adversarially robust neural architecture search for graph neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 8143–8152.

[36] B. Li, E. Pan, and Z. Kang, "PC-Conv: Unifying homophily and heterophily with two-fold filtering," in *Proc. AAAI Conf. Artif. Intell.*, 2024, pp. 13437–13445.

[37] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009. [Online]. Available: https://www.cs.utoronto.ca/kriz/learning-features-2009-TR.pdf

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

[39] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI Mag.*, vol. 29, no. 3, pp. 93–93, 2008.

[40] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

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

[42] Y. Xu et al., "PC-DARTS: Partial channel connections for memory-efficient architecture search," in *Proc. Int. Conf. Learn. Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=BJlS634tPr

[43] X. Chen and C.-J. Hsieh, "Stabilizing differentiable architecture search via perturbation-based regularization," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 1554–1565.

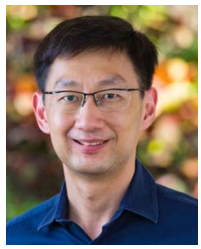
[44] M. Guo, Y. Yang, R. Xu, Z. Liu, and D. Lin, "When NAS meets robustness: In search of robust architectures against adversarial attacks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 631–640.

[45] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3844–3852.
[46] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=rJXMpikCZ
[47] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1025–1035.
[48] C. Guan et al., "AutoGL: A library for automated graph learning," in *Proc. Int. Conf. Learn. Representations Workshop*, 2021. [Online]. Available: https://openreview.net/forum?id=0yHwpLeInDn
[49] T. Pang, X. Yang, Y. Dong, H. Su, and J. Zhu, "Bag of tricks for adversarial training," in *Proc. Int. Conf. Learn. Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=Xb8xvrtB8Ce
[50] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc: 19th Int. Conf. Comput. Statist.* Paris France, Springer, 2010, pp. 177–186.
[51] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015.
[52] L. Rice, E. Wong, and Z. Kolter, "Overfitting in adversarially robust deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 8093–8104.
[53] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," 2017, *arXiv: 1708.04552*.
[54] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8026–8037.
[55] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," 2019, *arXiv: 1903.02428*.
[56] H. Kim, "Torchattacks: A pytorch repository for adversarial attacks," 2020, *arXiv: 2010.01950*.
[57] Y. Li, W. Jin, H. Xu, and J. Tang, "DeepRobust: A pytorch library for adversarial attacks and defenses," 2020, *arXiv: 2005.06149*.
[58] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter, "NAS-Bench-101: Towards reproducible neural architecture search," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7105–7114.
[59] X. Dong, L. Liu, K. Musial, and B. Gabrys, "NATS-Bench: Benchmarking NAS algorithms for architecture topology and size," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 7, pp. 3634–3646, Jul. 2022.
[60] Z. Guo et al., "Single path one-shot neural architecture search with uniform sampling," in *Proc. Eur. Conf. Comput. Vis.*, Springer, 2020, pp. 544–560.
[61] J. Zhang et al., "Attacks which do not kill training make adversarial learning stronger," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 11278–11287.
[62] L. Sun et al., "Adversarial attack and defense on graph data: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 8, pp. 7693–7711, Aug. 2023.

Yuqi Feng (Graduate Student Member, IEEE) received the BE degree in computer science and technology from Sichuan University, Chengdu, China, in 2022. He is currently working toward the PhD degree with the College of Computer Science, Sichuan University, Chengdu, China. His current research interests include neural architecture search and adversarial attacks.

Yanan Sun (Senior Member, IEEE) received the PhD degree in computer science from Sichuan University, Chengdu, China, in 2017. He is currently a professor with the College of Computer Science, Sichuan University, Chengdu, China. His research interests include evolutionary computation, neural networks, and their applications on neural architecture search. He lead a team winning the First place on the third unseen-data neural architecture search challenge in CVPR2023. Dr. Sun serves as an associate editor for *IEEE Transactions on Evolutionary Computation* and *IEEE Transactions on Neural Networks and Learning Systems*.

Gary G. Yen (Fellow, IEEE) received the PhD degree in electrical and computer engineering from the University of Notre Dame, Notre Dame, IN, USA, in 1992. In 1997, he was with the Structure Control Division, Air Force Research Laboratory, Albuquerque, NM, USA. He is currently a regents professor with the School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK, USA. His research interests include intelligent control, computational intelligence, conditional health monitoring, and signal processing and their industrial/defense applications.



Kay Chen Tan (Fellow, IEEE) received the BEng degree (First Class Hons.) and the PhD degree from the University of Glasgow, U.K., in 1994 and 1997, respectively. He is currently a chair professor of the Department of Computing, The Hong Kong Polytechnic University. He was the editor-in-chief of *IEEE Transactions on Evolutionary Computation*, and currently serves as an Editorial Board member of 10+ journals. He is currently the vice-president (Publications) of *IEEE Computational Intelligence Society*, an honorary professor with the University of Nottingham, in U.K., and the chief co-editor of Springer Book Series on Machine Learning: Foundations, Methodologies, and Applications.