Pattern Recognition 131 (2022) 108864

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

DARTSRepair: Core-failure-set guided DARTS for network robustness to common corruptions

Xuhong Ren[a,1], Jianlang Chen[b,1], Felix Juefei-Xu[c], Wanli Xue[a,*], Qing Guo[d], Lei Ma[e,b,f], Jianjun Zhao[b], Shengyong Chen[a]

[a] School of Computer Science and Engineering, Tianjin University of Technology, Tianjin, China
[b] Kyushu University, Japan
[c] Alibaba Group, USA
[d] Nanyang Technological University, Singapore
[e] University of Alberta, Canada
[f] Alberta Machine Intelligence Institute, Canada

ARTICLE INFO

Article history:
Received 3 January 2022
Revised 7 June 2022
Accepted 16 June 2022
Available online 18 June 2022

MSC:
68T10

PACS:
07.05.Mh

Keywords:
Network architecture search
Core-failure-set selection
Robustness enhancement
Differentiable architecture search

ABSTRACT

Network architecture search (NAS), in particular the differentiable architecture search (DARTS) method, has shown a great power to learn excellent model architectures on the specific dataset of interest. In contrast to using a fixed dataset, in this work, we focus on a different but important scenario for NAS: how to refine a deployed network's model architecture to enhance its robustness with the guidance of a few collected and misclassified examples that are degraded by some real-world unknown corruptions having a specific pattern (*e.g.*, noise, blur, *etc.*.). To this end, we first conduct an empirical study to validate that the model architectures can be definitely related to the corruption patterns. Surprisingly, by just adding a few corrupted and misclassified examples (*e.g.*, 10^3 examples) to the clean training dataset (*e.g.*, 5.0×10^4 examples), we can refine the model architecture and enhance the robustness significantly. To make it more practical, the key problem, *i.e.*, how to select the proper failure examples for the effective NAS guidance, should be carefully investigated. Then, we propose a novel *core-failure-set guided DARTS* that embeds a K-center-greedy algorithm for DARTS to select suitable corrupted failure examples to refine the model architecture. We use our method for DARTS-refined DNNs on the clean as well as 15 corruptions with the guidance of four specific real-world corruptions. Compared with the state-of-the-art NAS as well as data-augmentation-based enhancement methods, our final method can achieve higher accuracy on both corrupted datasets and the original clean dataset. On some of the corruption patterns, we can achieve as high as over 45% absolute accuracy improvements.

1. Introduction

In the process of neural network design, different tasks usually require the guidance of human experts' prior domain knowledge and repeated trial and error to obtain a high-performance model. This makes the design cost of a good neural network architecture rather expensive. The emergence of neural architecture search (NAS) makes it possible to automate such a design process. As a classic NAS algorithm, differentiable architecture search (DARTS) [1], while improving the performance of the model, reduces the architecture search time from thousands of GPU days [2,3] to 1.5 GPU days on the CIFAR-10 dataset. In many of the recent NAS-related studies, the focus has mostly been put on how to improve the model accuracy (most commonly for image classification tasks) [4–6] as well as model training and searching efficiency [7–10]. However, one important aspect of an ideal DNN may be slightly overlooked, and that is the robustness improvement of the DNN models as a result of NAS. The DNN model robustness is a vitally important characteristic, especially for models that are deployed in the safety- and mission-critical environments.

It is a known fact that (almost) all the high-performance DNN models that are naively trained with standard training datasets might suffer from robustness issues easily [11]. Specifically, they are prone to make erroneous decisions when presented with less ideal data (*e.g.*, data that are perturbed by common real-world but

[*] Corresponding author.
E-mail address: xuewanli@email.tjut.edu.cn (W. Xue).

[1] Xuhong Ren and Jianlang Chen are co-first authors and contribute equally to this work.

https://doi.org/10.1016/j.patcog.2022.108864
0031-3203/© 2022 Elsevier Ltd. All rights reserved.

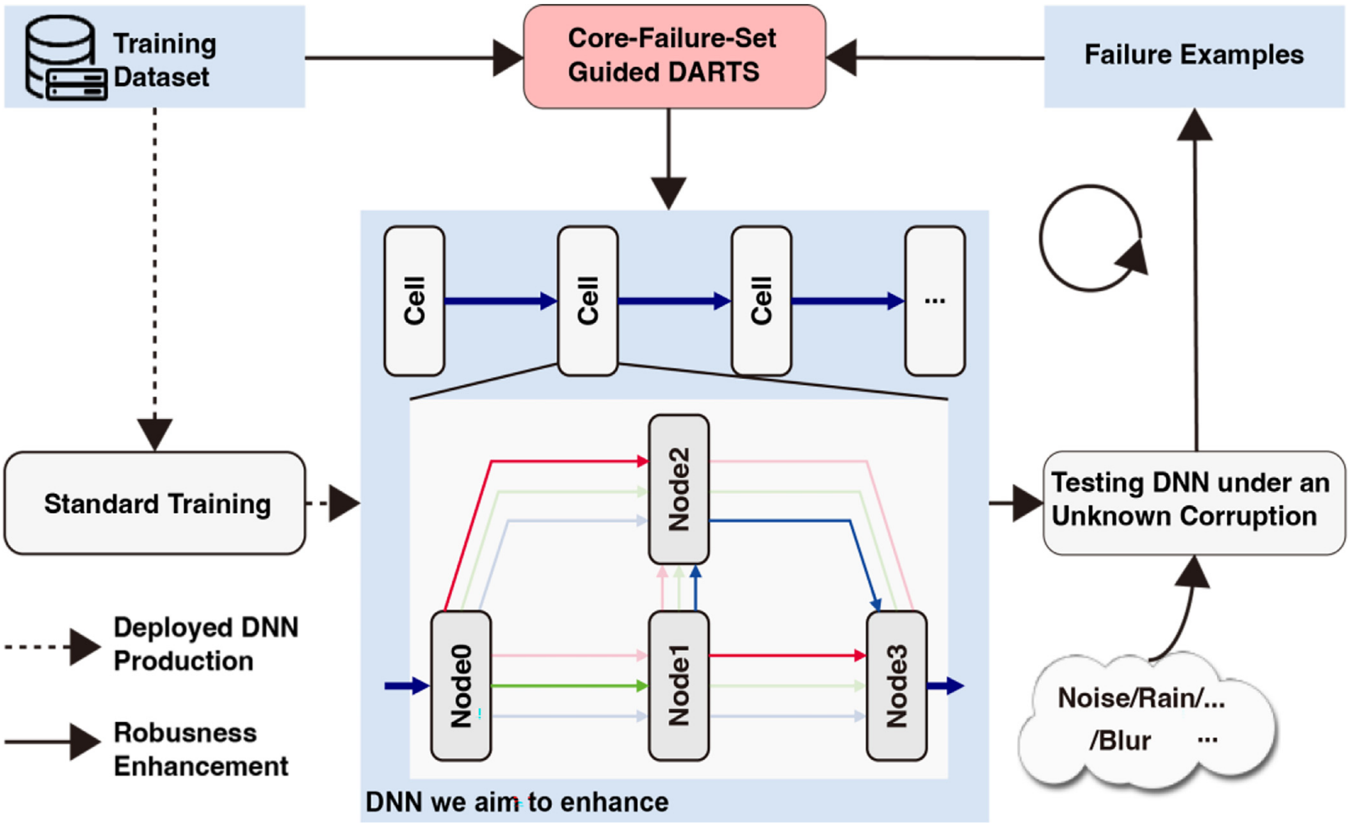


Fig. 1. Pipeline of the proposed core-failure-set guided DARTS (*i.e.*, CF-DARTS). The intuitive idea is to use a few collected failure examples to guide the network architecture search process to enhance the deployed DNN's robustness.

unknown in advance corruption patterns), letting alone the examples that are adversarially crafted.

Therefore, in this paper, we will take a deeper dive into studying the robustness aspect of NAS. In particular, we take a special focus on studying how to refine a deployed DNN model's architecture for enhancing its robustness with the guidance of a few limited collected and misclassified examples that are degraded by some unknown but specific corruption patterns such as noise, blur, *etc.*, through the DARTS procedure. The intuitive idea and the whole pipeline are presented in Fig. 1. As will be presented in a later section, our empirical study has validated that the model architectures are definitely related to the corruption patterns. We have made a surprising while interesting observation that by merely adding a few corrupted and misclassified examples (*e.g.*, 10^3 examples) into the clean training dataset (*e.g.*, 5.0×10^4 examples), we can already refine the model architecture and significantly enhance the model robustness. For a more in-depth investigation of how to select the proper failure examples for the effective NAS guidance under a more practical setting, we propose a novel *core-failure-set guided DARTS* that embeds a K-center-greedy algorithm for DARTS to select suitable corrupted failure examples to refine the model architecture. We have evaluated our method for DARTS-refined DNNs on the clean as well as 15 corruptions with the guidance of four specific real-world corruptions, respectively. Compared with the state-of-the-art NAS methods as well as data-augmentation-based enhancement methods, our final method can achieve higher accuracy on all corrupted datasets and the original clean dataset. In particular, on some of the corruptions, we can achieve over 45% absolute accuracy improvements. To the best of our knowledge, this work is among the very first attempts to investigate the novel aspect of DARTS for improving network robustness under the guidance of a few failure examples.

2. Related work

Network architecture search (NAS). Compared with manual architecture designing [12–14], NAS allows model designers to obtain a better model without professional domain knowledge and repeated trial and error. Early NAS algorithms are usually based on evolutionary algorithms [3] and reinforcement learning [2], which often require a lot of search time. DARTS [1] provides a new way of thinking, which relaxes the search space to make it continuous so that the searching process can be performed based on the gradient. This method can greatly reduce the time of architecture search so that the final architecture can be obtained within one GPU-day and has excellent performance. Following DARTS, GDAS [15] proposes the differentiable architecture sampler, in which only one of the sub-graphs sampled need to be optimized at one training iteration, so as to reduce the search time efficiently. PC-DARTS [16] proposes channel sampling and edge normalization technologies to solve the problem that DARTS requires large memory and computing when searching for models. P-DARTS [17] bridges the depth gap between the search net and the evaluation net in DARTS by gradually increasing the search depth. RobDARTS [18] studies the failure mode of DARTS by looking at the largest eigenvalue Hessian matrix of the verification loss of the architecture and improves the robustness of DARTS based on the analysis. More recently, Tian et al. [19] add a loss term for DARTS to alleviate the influence of the discretization of searching space. Hu et al. [20] propose to reduce the degree of weight sharing of DARTS. As a result, the method benefits the more stable and accurate prediction. Guo et al. [21] accelerate the network architecture searching by avoiding the evaluation of candidate networks. Xue et al. [22] regard the automatic network architecture search as a combinatorial optimization problem on the search space and search strategies. The above methods aim at improving the accuracy and search speed of DARTS, while our method intends to make the models searched from DARTS more robust.

Network robustness enhancement. The robustness of the deep neural network refers to the characteristic that the neural network can still maintain the normal input-output relationship when the input information or the neural network itself has limited disturbances. Through the adversarial attack algorithm, some adversarial samples with only a few differences can be automatically generated [23,24], which makes the neural network make wrong judgments [25]. Also, some common perturbations can also make the neural network make mistakes, such as pictures that are affected by overexposure, out of focus, bad weather, *etc.*[26]. To improve the robustness of the neural network, He et al. [27] propose parametric-noise-injection (PNI), which improves the model's robustness against adversarial attacks and the accuracy of perturbed data by performing trainable Gaussian noise injection in activation and weights. Rusak et al. [11] use additive Gaussian and Speckle noise to adjust the training method and improve the performance of the model on common corruptions data. Schneider et al. [28] uses corrupted samples to correct the statistics of batch normalization and remove the covariate shift caused by common corruptions, thereby improving the DNN's accuracy for the corruption dataset. Compared with the above data-driven optimization methods, our method starts from a new angle and focuses on improving the model's robustness to corruption data by optimizing the model's architecture itself.

Subset selection methods. Our paper is inspired by active learning. For neural network training, due to the high cost of the manual labeling process, active learning is proposed, which can select a portion of the image from the collected data for annotation, thereby reducing the cost of labeling and still making the model with higher accuracy. Sener and Savarese [29] define active learning as a core-set selection problem. This is the first time that the core-set selection method has been applied to DNNs. Before this, the core-set selection method was applied to the core vector machine (CVM) [30] to accelerate the training of SVM on large-scale data sets. Har-Peled et al. [31] applied coreset to k-median and k-means clustering. Different from the above methods, it is an early exploration that we apply the core-set method to the refining of DARTS and improved the DNN's robustness to common corruptions.

3. Methodology

3.1. Preliminary of DARTS

Differentiable architecture search (DARTS) [1] is an impressive one-shot NAS method. It jointly optimizes the network's weights and model architectures represented by a supernet on the specific dataset of interest. As all model architectures are based on the basic building unit (*i.e.*, cell) and share the network weights inherited from the supernet, they do not need to be retrained during the searching process, leading to efficient architecture search. In contrast to the most of existing NAS-related works, our work explores how to use a few available failure examples to help robust network architecture search. For better understanding, we first review DARTS:

Architecture search space. DARTS is to search the architecture of the basic building unit, *i.e.*, cell, and construct the whole network by stacking them. In particular, each cell is a direct acyclic graph containing N nodes and can be represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Each node $\mathrm{V}_i \in \mathcal{V}$ corresponds to a feature map in the network while each edge $\mathrm{E}_{i\to j} \in \mathcal{E}$ is an operation (*e.g.*, $\mathrm{O}_{i\to j}$) that maps the node V_i to another node V_j. The operation $\mathrm{O}_{i\to j}$ is selected from an operation set $\mathcal{O} = \{\mathrm{O}^k | k = 1, \ldots, K\}$ including convolution, max pooling, average pooling, identity, and zero representing for empty operation. Then, for each edge, we set a parameter $\alpha_k^{i\to j}$ to represent the contribution of the kth operation in $\mathcal{O}$ to a well-performed architecture. The set $\boldsymbol{\alpha} = \{\alpha_k^{i\to j}|, i, j = 1, \ldots, N, k = 1, \ldots, K\}$ thus represents an architecture in the search space. NAS is to optimize $\boldsymbol{\alpha}$ for higher performance.

Searching algorithm. Given a training dataset $\mathcal{D}_{\text{train}}$ and a validation dataset $\mathcal{D}_{\text{val}}$, DARTS aims to solve two problems as follows,

$$\mathbf{W} = \arg\min_{\mathbf{W}'} \mathcal{L}(\mathbf{W}', \boldsymbol{\alpha}, \mathcal{D}_{\text{train}}), \quad \text{and} \tag{1}$$

$$\boldsymbol{\alpha} = \arg\min_{\boldsymbol{\alpha}'} \mathcal{L}(\mathbf{W}, \boldsymbol{\alpha}', \mathcal{D}_{\text{val}}), \tag{2}$$

where $\mathcal{L}(\cdot)$ is the cross-entropy loss function for classification. $\boldsymbol{\alpha}$ is a set of continuous variables representing the weights of all candidate operations in the supernet. DARTS uses the bilevel optimization to solve the two problems with approximated architecture gradients. The final architecture consists of the operations with the maximum weights in $\boldsymbol{\alpha}$ and the weights are retrained on the $\mathcal{D}_{\text{train}}$ instead of inheriting from the supernet. Please refer to [1] for more details. We finally represent a DNN by $\phi(\mathbf{W}, \boldsymbol{\alpha})$ with $\mathbf{W}$ and $\boldsymbol{\alpha}$ for its weights and architecture, respectively.

3.2. Problem formulation and empirical study

After training a deep model, it is highly desirable that it is robust to different corruptions that may happen in the real world. Nevertheless, in practice, it is rather difficult to train such a perfect deep model due to the inevitable distribution shifting between the training dataset and real-world examples in the wild, as well as the less effective model architectures. As a result, even state-of-the-art deep models do not always predict correctly when unknown or unseen specific corruptions appear [26]. A practical and potentially feasible solution is to enhance the deep model's robustness with a few collected failure examples that may be degraded by specific corruptions, and enhance the model to be robust to similar corruption patterns while not harming its capability of handling the clean and other corrupted data. Quite a few existing works address this problem via advanced data augmentation methods, *e.g.*, AugMix [32] and CutMix [33], while ignoring the influence of model architectures. In this work, we handle this problem from the angle by searching for better model architectures with the guidance of a few collected failure examples. Note that, in contrast to the general robustness enhancement that can use pre-collected large-scale degraded examples to refine the model architecture, this task focuses on a more realistic scenario where the specific corruption pattern is unknown in advance and only a few collected failure examples are available. In the following, we give the definition of this task and provide some intuitive studies to reveal the feasibility and challenges of this idea.

Problem formulation. We first train a DNN $\phi(\mathbf{W}, \alpha)$ on original datasets $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{val}}$, and evaluate it on the original testing dataset $\mathcal{D}_{\text{test}}$. Then, we deploy it for real-world applications with the assumption the distribution of real-world examples is the same as the training dataset. However, there can be numerous real-world corruptions [26] making the assumption invalid. When we fed ϕ with the degraded examples with a specific corruption, we inevitably get failure examples denoted as $\mathcal{D}_{\text{fail}}$. Our goal is to refine the architecture of ϕ with the guidance of a small subset of $\mathcal{D}_{\text{fail}}$ (*i.e.*, $\mathcal{C}_{\text{fail}}$), and enhance the model to generalize to similar corruption while not harming the accuracy on $\mathcal{D}_{\text{test}}$ and robustness to other corruptions. We argue the reasons for using a small subset of $\mathcal{D}_{\text{fail}}$ for NAS as follows: ❶ A small subset of the failure examples would not result in extra high overhead for the process of the network architecture search, and thus it is more flexible for our

Table 1
Preliminary experiment on CIFAR-10 dataset. A initial DNN's architecture is refined with the guidance of $\mathcal{C}^{\text{gn}}_{\text{fail}}$ and retrained on $\mathcal{D}_{\text{train}}$. $\mathcal{D}_{\text{test}}$ and $\mathcal{D}^{\text{gn}}_{\text{test}}$ are the original testing dataset of CIFAR-10 and its degraded counterpart via Gaussian noise.

Initial DNN			DARTS-10^3GN			DARTS-5×10^3GN			DARTS-10^4GN		
$\mathcal{D}_{\text{test}}$	$\mathcal{D}^{\text{gn}}_{\text{test}}$	$\mathcal{D}^{\text{gn}}_{\text{fail}}/\mathcal{C}^{\text{gn}}_{\text{fail}}$	$\mathcal{D}_{\text{test}}$	$\mathcal{D}^{\text{gn}}_{\text{test}}$	$\mathcal{D}^{\text{gn}}_{\text{fail}}/\mathcal{C}^{\text{gn}}_{\text{fail}}$	$\mathcal{D}_{\text{test}}$	$\mathcal{D}^{\text{gn}}_{\text{test}}$	$\mathcal{D}^{\text{gn}}_{\text{fail}}/\mathcal{C}^{\text{gn}}_{\text{fail}}$	$\mathcal{D}_{\text{test}}$	$\mathcal{D}^{\text{gn}}_{\text{test}}$	$\mathcal{D}^{\text{gn}}_{\text{fail}}/\mathcal{C}^{\text{gn}}_{\text{fail}}$
96.26	32.58	0.00	96.73	33.06	14.06	96.58	35.57	15.16	96.77	34.89	11.82

method for real-world applications. ❷ As analyzed in the following parts, with limited $\mathcal{D}_{\text{fail}}$, searching robust model architecture does not keep a positive relationship with the number of failure examples.

Observations. To understand the above problem clearly, we conduct a preliminary experiment on the CIFAR-10 dataset. Specifically, given a pre-trained deep neural network (DNN) optimized by DARTS on the CIFAR-10, we evaluate it on the testing examples corrupted by a specific corruption, *e.g.*, Gaussian noise subset of CIFAR-10-C denoted as $\mathcal{D}^{\text{gn}}_{\text{test}}$ [26]. As shown in the 'initial DNN' column of Table 1, we show that numerous testing examples are misclassified under this common corruption since the accuracy of original testing data reduces from 96.26% to 32.58%. We collect all failure examples as the set $\mathcal{D}^{\text{gn}}_{\text{fail}}$ having around 30,000 examples. Then, we randomly select 1,000 failure examples from the $\mathcal{D}^{\text{gn}}_{\text{fail}}$ as the set $\mathcal{C}^{\text{gn}}_{\text{fail}}$. Moreover, we combine $\mathcal{C}^{\text{gn}}_{\text{fail}}$ and the original training dataset $\mathcal{D}_{\text{train}}$ as well as validation dataset $\mathcal{D}_{\text{val}}$ to refine the initial DNN's architecture via Eqs. (1) and (2). After that, the architecture is fixed and retrained via $\mathcal{D}_{\text{train}}$. Since the failure examples are only used in the architecture search stage, there is no overfitting risk on the model weights for those failure examples $\mathcal{C}^{\text{gn}}_{\text{fail}}$. To avoid potential doubt, we compare the robustness of the original and refined DNNs on both $\mathcal{D}^{\text{gn}}_{\text{test}}$ and $\mathcal{D}^{\text{gn}}_{\text{fail}}$ without $\mathcal{C}^{\text{gn}}_{\text{fail}}$, *i.e.*, $\mathcal{D}^{\text{gn}}_{\text{fail}}/\mathcal{C}^{\text{gn}}_{\text{fail}}$. We also conduct this process with larger $\mathcal{C}^{\text{gn}}_{\text{fail}}$, *i.e.*, 5×10^3 and 10^4 examples. All results are summarized in Table 1.

Overall, we observe that: ❶ Even a few corrupted examples (*i.e.*, $|\mathcal{C}^{\text{gn}}_{\text{fail}}| = 1,000$ that is much smaller than $|\mathcal{D}_{\text{train}}| = 50,000$) help search more robust model architecture to a specific corruption, that is, the accuracy on $\mathcal{D}^{\text{gn}}_{\text{fail}}$ increases from 0.0% to 14.06% while the accuracy on the original testing dataset (*i.e.*, $\mathcal{D}_{\text{test}}$) becomes even higher. ❷ Different to the common understanding, the accuracy on $\mathcal{D}^{\text{gn}}_{\text{fail}}$ does not increase with the increasing of $|\mathcal{C}^{\text{gn}}_{\text{fail}}|$. We understand this can be due to the relationship between $\mathcal{C}^{\text{gn}}_{\text{fail}}$ and $\mathcal{D}_{\text{train}}$ as well $\mathcal{D}_{\text{val}}$ playing a key role for searching robust model architectures. Hence, a more advanced core-failure-aware network architecture search method should be developed to select effective failure examples from $\mathcal{D}^{\text{gn}}_{\text{fail}}$.

3.3. Core-failure-set-guided DARTS

According to the above analysis, we aim to identify a critical subset (*i.e.*, $\mathcal{C}_{\text{final}}$) from the $\mathcal{D}_{\text{fail}}$ to make the searched model architecture be more robust to the specific corruption. Specifically, given an initial DNN ϕ with its weights $\mathbf{W}$ and architecture $\boldsymbol{\alpha}$, we add an extra step before searching

$$\mathcal{C}_{\text{fail}} = \arg\min_{\mathcal{C}'_{\text{fail}}} |\mathcal{L}(\phi, \mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{fail}}) - \mathcal{L}(\phi, \mathcal{D}_{\text{train}} \cup \mathcal{C}'_{\text{fail}})|, \quad \text{subject to} \quad |\mathcal{C}'_{\text{fail}}| \leq B \tag{3}$$

where $\mathcal{C}_{\text{fail}}$ is the selected failure examples for robust architecture searching and B denotes the upper bound of the number of $\mathcal{C}_{\text{fail}}$. Intuitively, Eq. (3) is a core-set loss function [29] that encourages the classification loss on $\mathcal{D}_{\text{train}} \cup \mathcal{C}_{\text{fail}}$ to be similar with that on $\mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{fail}}$, letting selected a few failure examples can cover situations of all failure examples when combining with $\mathcal{D}_{\text{train}}$. Then, we split $\mathcal{C}_{\text{fail}}$ into two equal subsets denoted as $\mathcal{C}_{\text{fail-t}}$ and $\mathcal{C}_{\text{fail-v}}$, respectively. Moreover, the Eqs. (1) and (2) for searching algorithm are modified as

$$\mathbf{W} = \arg\min_{\mathbf{W}'} \mathcal{L}(\mathbf{W}', \boldsymbol{\alpha}, \mathcal{D}_{\text{train}} \cup \mathcal{C}_{\text{fail-t}}), \quad \text{and} \tag{4}$$

$$\boldsymbol{\alpha} = \arg\min_{\boldsymbol{\alpha}'} \mathcal{L}(\mathbf{W}, \boldsymbol{\alpha}', \mathcal{D}_{\text{val}} \cup \mathcal{C}_{\text{fail-v}}). \tag{5}$$

Then, the key problem is how to solve Eq. (3) effectively. Since not all labels of the examples in $\mathcal{D}_{\text{fail}}$ are given, we cannot naively solve Eq. (3) via the gradient decent. Alternatively, as demonstrated in [34] and [29], this problem has an upper bound and is equivalent to solving the K-center problem.

K-center greedy for $\mathcal{C}_{\text{fail}}$. As demonstrated in [34], minimizing the core-set objective function is equivalent to the k-center problem, that is, choosing B center examples to let the largest distance between an example in $\mathcal{D}_{\text{train}}$ and its nearest center be minimized. We can formulate it as

$$\mathcal{C}_{\text{fail}} = \arg\min_{\mathcal{C}'_{\text{fail}}} \max_{\mathbf{X}_i \in \mathcal{C}'_{\text{fail}}} \min_{\mathbf{X}_j \in \mathcal{D}_{\text{train}} \cup \mathcal{C}'_{\text{fail}}} \text{dist}(\mathbf{X}_i, \mathbf{X}_j), \quad \text{subject to} \quad |\mathcal{C}'_{\text{fail}}| \leq B, \tag{6}$$

where $\text{dist}(\mathbf{X}_i, \mathbf{X}_j)$ denotes the distance between $\mathbf{X}_i$ and $\mathbf{X}_j$. Here, we adopt the feature of the layer before softmax of the initial DNN and their L_2-norm as the distance function. Although this problem is NP-hard, we can solve it efficiently via a greedy method [29] and show the details in Algorithm 1.

Algorithm 1: $\mathcal{C}_{\text{fail}}$ selection via K-center greedy.

```
Input: $\mathcal{D}_{\text{train}}$, the upper bound $B$, an initial DNN $\phi$ with its
       $\mathbf{W}_0$ and architecture $\boldsymbol{\alpha}_0$, and $\mathcal{D}_{\text{fail}}$.
Output: $\mathcal{C}_{\text{fail}}$.
Initialize $\mathcal{C}_{\text{fail}}$ as empty ;
for 1 to $B$ do
    $\mathbf{X} = \arg\max_{\mathbf{X}_i \in \mathcal{D}_{\text{fail}}} \min_{\mathbf{X}_j \in \mathcal{D}_{\text{train}}} \text{dist}(\mathbf{X}_i, \mathbf{X}_j)$;
    $\mathcal{C}_{\text{fail}} = \mathcal{C}_{\text{fail}} \cup \mathbf{X}$;
end
```

3.4. Implementation details

Dataset configuration. Given a dataset including $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{val}}$ and $\mathcal{D}_{\text{test}}$, *e.g.*, CIFAR-10 dataset [35], we first train a model on $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{val}}$ via DARTS as the initial DNN denoted as the ϕ_0 with weights and architecture as $\mathbf{W}_0$ and $\boldsymbol{\alpha}_0$. Then, we build a degraded testing dataset by adding a specific corruption to $\mathcal{D}_{\text{test}}$, which is denoted as $\mathcal{D}^{\text{xx}}_{\text{test}}$ where xx is the name short for the corruption. We evaluate the ϕ_0 on $\mathcal{D}^{\text{xx}}_{\text{test}}$ and collect all failure examples as $\mathcal{D}^{\text{xx}}_{\text{fail}}$. Our main goal is to enhance the robustness of ϕ_0 for the corruption 'xx' while not harming the effectiveness of handling other corruptions.

Evaluation configuration. We use Algorithm 1 to obtain the core failure set $\mathcal{C}^{\text{xx}}_{\text{fail}}$ from $\mathcal{D}^{\text{xx}}_{\text{fail}}$ with $B = 1000$ examples included in $\mathcal{C}^{\text{xx}}_{\text{fail}}$. Then, we refine ϕ_0 based on Eqs. (4) and (5) and the datasets $\mathcal{D}_{\text{train}} \cup \mathcal{C}_{\text{fail-t}}$ and $\mathcal{D}_{\text{val}} \cup \mathcal{C}_{\text{fail-v}}$, and we can get a refined network ϕ_1

with updated weights $\mathbf{W}_1$ and α_1. We denote the above process as an iteration to refine the targeted DNN ϕ_0. Actually, we can further refine $\mathbf{W}_1$ and α_1 for a second iteration refinement and get $\mathbf{W}_2$ and α_2. In practice, we only perform the refinement once for the weights and architecture optimization due to the computational expensive methods. In each iteration, the optimization of the architecture and weights requires several epochs (*i.e.*, we set 20 epochs in our work) as done in the standard mini-batch-based training process. Specifically, in each epoch, we first randomly sample a batch from the training dataset (*i.e.*, $\mathcal{D}_{\text{train}} \cup \mathcal{C}_{\text{fail-t}}$) and use the Eq. (4) to update the network weights while fixing the architecture parameters α. Then, we randomly sample a batch from the validation dataset (*i.e.*, $\mathcal{D}_{\text{val}} \cup \mathcal{C}_{\text{fail-v}}$) and employ Eq. (5) to optimize the architecture parameters while the weights stay the same. We conduct the above min-batch-based optimization for around 520 times for each epoch, and perform 20 epoches totally for ϕ_0's refinement. We denote our method, *i.e.*, *core-failure-set guided DARTS*, as CF-DARTS. Moreover, we implement a more advanced version of CF-DARTS by combining the core failure set $\mathcal{C}^{\text{xx}}_{\text{fail}}$ with $\mathcal{D}_{\text{train}}$ to retrain the DNN, and we denote this version as CF-DARTSE. Finally, we evaluate and validate our method to observe the accuracy of refined model (*i.e.*, $\boldsymbol{\alpha}_1$) on $\mathcal{D}^{\text{xx}}_{\text{fail}}/\mathcal{C}^{\text{xx}}_{\text{fail}}$ meaning the dataset $\mathcal{D}^{\text{xx}}_{\text{fail}}$ excluding $\mathcal{C}^{\text{xx}}_{\text{fail}}$, which is also represented as 'xx$_{\text{fail}}$' in Table 2, *e.g.*, 'Gauss$_{\text{fail}}$' for $\mathcal{D}^{\text{gn}}_{\text{fail}}/\mathcal{C}^{\text{gn}}_{\text{fail}}$.

4. Experimental results

4.1. Setups

Datasets. Our experiments are performed on CIFAR-10 [35], CIFAR-10-C [26], Tiny-ImageNet [36], and Tiny-ImageNet-C [26]. CIFAR-10 consisting of 60,000 32x32 color images with 10 classes and each class has 6,000 images, including 50,000 training and 10,000 testing images. CIFAR-10-C is an extension of CIFAR-10, adding 15 corruptions to the original testing dataset of CIFAR-10. The corruption names are shown in Tables 2 and 6. Similarly, we can also set up the Tiny-ImageNet [36] and Tiny-ImageNet-C datasets [26]. Then, we retrain the network architectures searched on CIFAR-10 and test them on Tiny-ImageNet and Tiny-ImageNet-C to validate the generalization. We validate our method by regarding four different common corruptions, *i.e.*, Gaussian noise (gn), pixelate (pl), fog (fg), and glass blur (gb), as the specific corruptions mentioned in Section 3.2. Our objective is to enhance the robustness of an initial DNN to the four corruptions, respectively, while not harming the accuracy of the original testing dataset of CIFAR-10 and other corruption datasets.

Metrics. In our experiment, we choose top-1 accuracy as the metric, that is, the proportion of the correct model judgment among the total number of testing images.

Baseline methods. We consider the following baselines: ❶ To validate the effectiveness of the K-center greedy based $\mathcal{C}^{\text{xx}}_{\text{fail}}$ selection, we first set the random selection strategy as a baseline. Specifically, we randomly select $\mathcal{C}^{\text{xx}}_{\text{fail}}$ from $\mathcal{D}^{\text{xx}}_{\text{fail}}$ and perform the same steps as done in Section 3.4. Then, we can have two variants, *i.e.*, RF-DARTS and RF-DARTSE, corresponding to our two versions, CF-DARTS and CF-DARTSE, respectively. In addition, we augment training examples via a operation set (*i.e.*, {rotation, solarization, shear transformation, translation, auto-contrast adjustment, equalization}) and randomly select 1000 examples to replace the failure examples $\mathcal{C}_{\text{fail}}$ in Eqs. (4) and (5), and conduct the same architecture search process as detailed in Section 4.1. We get two variants denoted as the augmentation-set guided DARTS (AU-DARTS) and AU-DARTSE. ❷ To validate the advantages of the proposed method over the popular data-augmentation-based robustness enhancement, we select the following state-of-the-art baseline methods: CutMix [33], AugMix [32], and DeepRepair [37]. Specifically,

Table 2
Top-1 accuracy of the initial DNN and refined DNNs on the original CIFAR-10 testing dataset (*i.e.*, 'Clean' column), the specific corruption (*i.e.*, Gauss$_{\text{fail}}$), and other 14 corruptions. DeepRepair, AugMix, and CutMix are data-augmentation-based robustness enhancements. AU-DARTS, RF-DARTS, and CF-DARTS are our methods based on the augmentation, random failure set selection, and core failure set selection strategies, respectively. AU/RF/CF-DARTSE retrains the refined architecture via the original training dataset and selected failure or augmented examples. All methods are evaluated five times, and the average accuracy, as well as the standard deviations, are reported in the first and second row of each method, respectively.

		Noise			Blur				Weather				Digital			
	Clean	Gauss$_{\text{fail}}$	Shot	Impulse	Defocus	Glass	Motion	Zoom	Snow	Frost	Fog	Bright	Contrast	Elastic	Pixel	Jpeg
Org. DNN	96.26	0.00	45.18	51.53	87.49	56.43	82.11	81.71	87.58	80.46	91.30	94.96	82.05	88.31	76.30	82.67
DeepRepair	95.25	29.26	46.09	40.83	80.66	50.56	78.08	78.07	79.09	81.51	90.31	88.63	87.35	76.12	66.11	63.69
	±0.86	±7.06	±4.99	±3.93	±0.85	±3.48	±1.36	±1.09	±0.90	±0.94	±0.41	±0.68	±0.58	±0.92	±0.82	±1.37
AugMix	89.75	50.35	67.58	71.6	87.62	61.06	83.98	85.98	82.81	81.28	89.88	90.18	89.61	82.73	69.22	73.90
	±1.02	±3.11	±1.74	±2.54	±1.48	±2.00	±1.81	±1.61	±2.37	±3.51	±1.38	±1.65	±1.39	±1.17	±2.56	±1.79
CutMix	95.70	21.99	53.82	50.61	84.36	62.58	80.60	78.22	85.52	81.80	89.35	94.06	77.63	85.54	75.49	79.04
	±0.16	±0.90	±0.98	±3.77	±0.80	±3.13	±1.12	±1.32	±0.60	±0.89	±0.22	±0.06	±0.47	±0.90	±1.75	±1.47
AU-DARTS	97.15	17.19	50.99	50.87	88.09	67.90	84.86	81.31	88.58	84.07	91.76	94.04	83.06	88.63	78.62	82.65
	±0.09	±4.82	±2.74	±3.14	±0.96	±0.84	±0.60	±1.49	±0.22	±0.49	±0.28	±0.08	±0.90	±0.35	±1.25	±0.74
AU-DARTSE	97.04	58.37	71.74	78.00	92.79	72.15	90.76	89.47	91.14	87.15	93.80	95.97	87.34	91.22	81.59	84.97
	±0.17	±3.43	±1.62	±0.70	±0.26	±1.70	±0.39	±0.17	±0.18	±0.13	±0.05	±0.10	±0.34	±0.24	±0.55	±0.12
RF-DARTS	96.80	17.07	49.66	52.24	87.94	66.58	85.00	82.27	88.82	84.05	91.90	95.58	83.97	89.74	79.43	84.02
	±0.27	±2.24	±1.53	±4.37	±0.81	±3.17	±0.51	±1.09	±0.96	±1.42	±0.69	±0.27	±1.41	±0.26	±1.99	±0.27
RF-DARTSE	97.48	82.87	87.42	66.50	88.84	77.15	85.85	83.51	90.98	90.25	92.63	96.42	84.87	90.74	81.15	85.82
	±0.10	±1.43	±1.02	±5.23	±0.59	±2.49	±0.72	±1.27	±0.36	±0.85	±0.20	±0.15	±1.60	±0.21	±1.03	±0.48
CF-DARTS	96.83	18.27	50.21	49.25	87.74	68.03	84.96	82.67	89.35	84.67	92.08	95.66	83.69	89.92	78.96	84.04
	±0.20	±2.86	±1.62	±4.03	±0.32	±2.76	±0.37	±0.83	±0.39	±1.13	±0.36	±0.21	±0.76	±0.20	±1.86	±0.94
CF-DARTSE	97.35	87.77	90.40	75.93	89.08	76.26	86.88	84.21	91.01	89.66	92.98	96.39	84.71	91.00	80.62	85.28
	±0.10	±2.08	±0.83	±5.93	±0.74	±2.51	±0.51	±1.45	±0.50	±1.21	±0.39	±0.09	±0.91	±0.22	±0.59	±0.62

CutMix augments training examples by cutting and pasting parts of the images while mixing the labels proportionally. AugMix stochastically samples a series of the operations from {rotation, solarization, shear transformation, translation, auto-contrast adjustment, equalization} to augment the input image multiple times and conduct mix on transformed images. DeepRepair is an extension of the AugMix by adding a style-transfer method to the augmentation set, which uses the failure examples as the references to guide the transformations of the training data. ❸ We also extend our method to more recent NAS methods and see whether the proposed method can benefit them and outperform their original versions. Specifically, we take PC-DARTS [16], DARTSPT [38], and RobDARTS [18]. PC-DARTS promotes searching efficiency, and DARTSPT focuses on architecture selection. In particular, RobDARTS is to enhance DARTS that yields degenerate architectures with very poor test performance.

4.2. Validation and comparison results

Validation of proposed methods. In this part, we validate our methods, *i.e.*, CF-DARTS and CF-DARTSE, by taking the Gaussian noise (gn) as the specific corruption, *i.e.*, 'xx=gn' in Section 3.4. To demonstrate the effectiveness of the core failure set selection method in Algorithm 1, we can simply replace the selected core-failure set with the randomly selected examples and augmented examples, thus get four baselines, *i.e.*, RF-DARTS, RF-DARTSE, AU-DARTS, and AU-DARTSE (See more details in Section 3.4). Here, we report the accuracy on the original testing dataset (*i.e.*, 'Clean' column), $\mathcal{D}^{\text{gn}}_{\text{fail}}/\mathcal{C}^{\text{gn}}_{\text{fail}}$ (*i.e.*, 'Gauss$_{\text{fail}}$' column), and 14 corrupted testing datasets in Table 2. All methods are evaluated five times and the averaged results, as well as standard deviation, are reported.

Moreover, we further present the accuracy variations of the DNNs searched by DARTS, CF-DARSTS, and CF-DARTSE during the training process. Specifically, we perform CF-DARTS, CF-DARTSE, and DARTS one by one on the same server and record the accuracy of the searched DNNs on the clean and corrupted testing datasets during the training process. Here, we consider three corruptions, *i.e.*, Gaussian noise, shot noise, and impulse noise. CF-DARTS and CF-DARTSE are guided by a few failure examples selected from $\mathcal{D}^{\text{gn}}_{\text{fail}}$. As shown in Fig. 2, we see that the DNNs searched by our two methods (*i.e.*, CF-DARTS and CF-DARTSE) outperform the one based on the basic DARTS along most of the training iterations under the three corruptions. Besides, the results of the DNNs on the clean dataset are almost the same. Overall, the experiments demonstrate that our two methods can enhance DNN's robustness significantly while not harming the capability on the clean data.

According to Table 2, we observe that: ❶ Our method CF-DARTS does enhance the robustness of the initial DNN to Gassusian noise since the accuracy on Gauss$_{\text{fail}}$ increases from 0.00% to 18.27%. Moreover, the accuracy of the clean data and most of the other corruptions do not reduce and become even higher, demonstrating that the proposed method is able to enhance the robustness to a specific corruption while not harming the accuracy of clean data and robustness against other corruptions. ❷ CF-DARTSE significantly outperforms CF-DARTS on Gauss$_{\text{fail}}$ and also achieves much better scores on other datasets even if we just add 1000 failure examples for retraining the refined architecture. It demonstrates that failure data itself benefits further robustness enhancement. ❸ CF-DARTS/DARTSE achieve better results across all cases than RF-DARTS/DARTSE and AU-DARTS/DARTSE on the Gauss$_{\text{fail}}$ and also achieve higher accuracy on most the other corrupted datasets. The results show that the selected failure examples can guide the robustness enhancement of the targeted DNN more effectively than randomly selected or augmented examples, which infers the effectiveness of our K-center greedy selection.

Comparison to data-augmentation-based methods. Following the experimental setups in Section 4.2, we further compare our method with the pure data-augmentation-based robustness enhancements, *i.e.*, DeepRepair [37], AugMix [32], and CutMix [33]. We use their default setups and retrain the initial DNN with their respective augmentation strategies. As presented in Table 2 and Fig. 3, we see that: ❶ Compared with CF-DARTSE, DeepRepair, AugMix, and CutMix can also improve the robustness against the specific corruption, *i.e.*, Gaussian noise, with the accuracy on Gauss$_{\text{fail}}$ from 0.00% to 50.35%, 29.26% and 21.99%, respectively. Nevertheless, they lead to slightly worse results on clean datasets and most the corrupted datasets. In particular, AugMix reduces the accuracy of clean images from 96.26% to 89.75%. ❷ Our method CF-DARTSE outperforms the three methods significantly on most of the datasets with large margins, demonstrating that combining architecture refinement and failure example-enriched dataset can help enhance robustness better than data augmentation-based methods.

Comparison to other NAS methods. We equip our method to more recent NAS methods and see whether the proposed method can benefit them and outperform their original versions. We follow the same process in Section 3.4. Specifically, given a NAS method, we first search and train a DNN (*i.e.*, ϕ_0) and test it on different corrupted datasets. Then, given a few failure examples containing a specific corruption (*e.g.*, Gaussian noise), we use our method to refine the architecture and enhance the robustness against the specific corruption. For each NAS method, we follow the setups in Section 4.1 and try four variants, *i.e.*, RF-'**', RF-'**'E, CF-'**', and CF-'**'E where '**' denotes the name of the NAS method. Specifically, we validate the effectiveness of our method on PC-DARTS [16], DARTSPT [38], and RobDARTS [18], respectively. As shown in Fig. 5, for all of the three NAS methods, our methods (*i.e.*, CF-'**' and CF-'**'E) enhance their robustness significantly and outperform the basic variants (*i.e.*, RF-'**' and RF-'**'E) with large margins, which further demonstrates the effectiveness of the proposed framework.

4.3. Results of different corruptions

In addition to the Gaussian noise corruption, we can further validate our method by considering other corruptions. However, it takes huge computing resources and training time to test all corruptions. Hence, we randomly select the other three corruptions from 'Blur', 'Digital', and 'Weather' types for validation, *i.e.*, the glass blur, pixelate, and fog corruptions. The results are reported in Table 3. The results further demonstrate the effectiveness of our method: ❶ Both methods, *i.e.*, CF-DARTS and CF-DARTSE, on all three corruptions let refined DNNs outperform the original DNN significantly on all datasets including the clean testing dataset and other 14 corrupted datasets. ❷ CF-DARTSE using the 1000 failure examples to retrain the refined architectures increases the accuracy on all datasets significantly.

4.4. Generalization to Tiny-ImageNet/Tiny-ImageNet-C

In the subsection, we aim to validate whether the neural networks searched on the training dataset of CIFAR-10 and the selected core-failure examples can generalize to other datasets, *i.e.*, Tiny-ImageNet and Tiny-ImageNet-C. Specifically, as the method introduced in Section 3.3, we can get refined architecture by searching on the training and validation datasets of CIFAR-10 (*i.e.*, $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{val}}$) and the selected core-failure-set (*i.e.*, $\mathcal{C}^{\text{xx}}_{\text{fail-t}}$ and $\mathcal{C}^{\text{xx}}_{\text{fail-v}}$) where xx denotes the name short for specific corruptions. Here, we retrain the searched architecture based on the training dataset of Tiny-ImageNet. Then, we further evaluate the searched

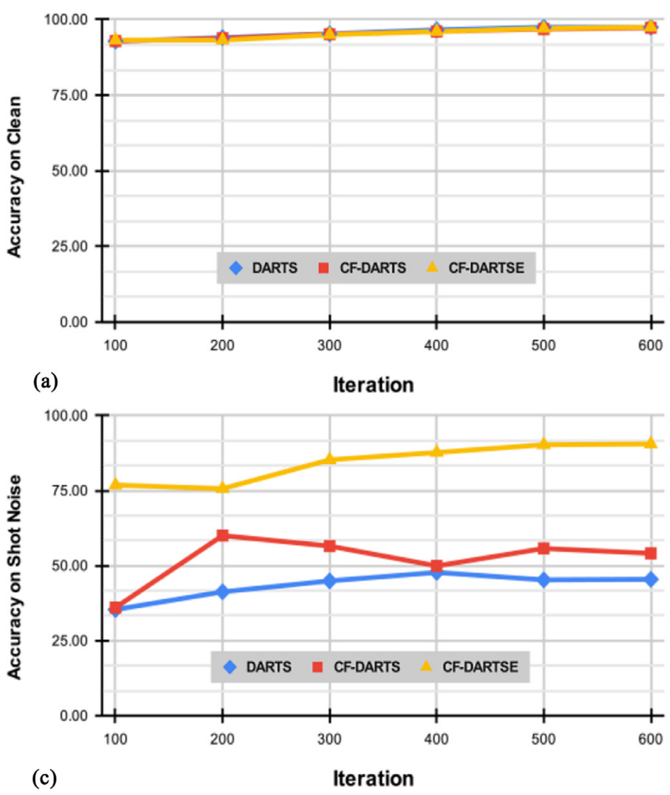


Fig. 2. Accuracy of DARTS, CF-DARTS, and CF-DARTSE on the datasets of clean (*i.e.*, (a)), Gaussian$_{\text{fail}}$ (*i.e.*, (b)), shot noise (*i.e.*, (c)), and impulse noise (*i.e.*, (d)) along the training iterations.

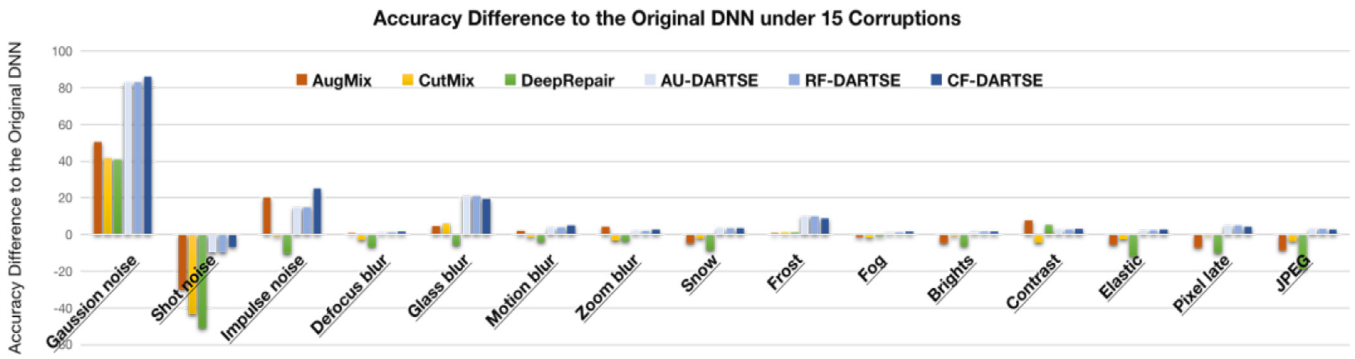


Fig. 3. Accuracy difference between enhanced and original DNN via AugMix, CutMix, DeepRepair, AU-DARTSE, RF-DARTSE, and CF-DARTSE, respectively. That is, we use the scores of each row in Table 2 to minus the first row.

architecture on the testing dataset of Tiny-ImageNet and its fifteen corrupted datasets. We report the results in Table 4 and have the following observations: ❶ Compared with the original DNN, the CF-DARTS-searched architectures achieve much higher accuracy on both clean and corrupted datasets the most times. For example, the architecture searched on $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{val}}$ of CIFAR-10 and the selected core-failure-set $\mathcal{C}^{\text{gn}}_{\text{fail}}$ under Gaussian noise (gn) has the accuracy of 65.52% on the raw Tiny-ImageNet dataset (*i.e.*, the 'Clean' column), which is significantly higher than the original DNN with the accuracy of 62.63%. Moreover, it also achieves much higher accuracy on all corrupted datasets except the glass blur. We have similar observations on other searched architectures under pixelate, fog, and glass blur. ❷ Compared with the architecture searched by baseline method RF-DARTS, CF-DARTS gets higher accuracy on most of the corrupted datasets. For example, for the architecture searched with the core-failure-set under Gaussian noise (gn) corruption, CF-DARTS outperforms RF-DARTS on the clean and fifteen corrupted datasets. Overall, the experiments demonstrate that our method can search architectures with high generalizations, which can generalize to a totally different dataset.

4.5. Influence of failure examples

In previous experiments, we fix the size of core failure set $\mathcal{C}_{\text{fail}}$ as 1000 examples (*i.e.*, $B = 1,000$). Here, we further discuss its influence by setting $B = \{1,000, 5,000, 10,000, 15,000, 20,000\}$ and report the results of RF-DARTS and CF-DARTS in Table 6 (Top). We have the following observations: ❶ the accuracy on all datasets does not increase as the size of $\mathcal{C}_{\text{fail}}$ becomes larger, which is different from the common understanding of training model weights where the accuracy keeps a positive relationship with the size of the training dataset, since we just use $\mathcal{C}_{\text{fail}}$ for architecture refine-

Table 3
Top-1 accuracy of the initial DNN and refined DNNs with our CF-DARTS and CF-DARTSE on the original testing dataset (*i.e.*, 'Clean' column), the specific corruptions (*i.e.*, $\text{Gauss}_{\text{fail}}$, $\text{Pixel}_{\text{fail}}$, and $\text{Glass}_{\text{fail}}$), and other 14 corruptions.

		Noise			Blur				Weather				Digital			
	Clean	$\text{Gauss}_{\text{fail}}$	Shot	Impulse	Defocus	Glass	Motion	Zoom	Snow	Frost	Fog	Bright	Contrast	Elastic	Pixel	Jpeg
Org. DNN	96.26	0.00	45.18	51.53	87.49	56.43	82.11	81.71	87.58	80.46	91.30	94.96	82.05	88.31	76.30	82.67
CF-DARTS	96.83	18.27	50.21	49.25	87.74	68.03	84.96	82.67	89.35	84.67	92.08	95.66	83.69	89.92	78.96	84.04
CF-DARTSE	**97.35**	**87.01**	**90.40**	**75.93**	**89.08**	**76.26**	**86.88**	**84.21**	**91.01**	**89.66**	**92.98**	**96.39**	**84.71**	**91.00**	**80.62**	**85.28**
	Clean	Gauss	Shot	Impulse	Defocus	$\text{Glass}_{\text{fail}}$	Motion	Zoom	Snow	Frost	Fog	Bright	Contrast	Elastic	Pixel	Jpeg
Org. DNN	96.26	32.58	45.18	51.53	87.49	0.00	82.11	81.71	87.58	80.46	91.30	94.96	82.05	88.31	76.30	82.67
CF-DARTS	96.84	48.33	59.26	57.51	87.45	49.86	85.22	80.85	89.88	84.71	91.82	95.80	83.56	89.94	79.44	84.33
CF-DARTSE	**97.15**	**63.85**	**69.19**	**63.07**	**90.74**	**82.06**	**87.23**	**85.78**	**91.34**	**90.73**	**92.89**	**95.88**	**86.28**	**91.43**	**83.65**	**84.95**
	Clean	Gauss	Shot	Impulse	Defocus	Glass	Motion	Zoom	Snow	Frost	Fog_{fail}	Bright	Contrast	Elastic	Pixel	Jpeg
Org. DNN	96.26	32.58	45.18	51.53	87.49	56.43	82.11	81.71	87.58	80.46	0.00	94.96	82.05	88.31	76.30	82.67
CF-DARTS	97.22	**43.34**	**55.14**	53.48	89.42	**72.68**	86.39	84.01	90.70	87.40	93.27	96.11	85.92	90.13	79.01	83.06
CF-DARTSE	**99.32**	32.14	45.85	**53.97**	**92.67**	70.11	**90.47**	**88.20**	**92.50**	**88.45**	**97.05**	**98.40**	**92.77**	**92.72**	**79.88**	**85.93**
	Clean	Gauss	Shot	Impulse	Defocus	Glass	Motion	Zoom	Snow	Frost	Fog	Bright	Contrast	Elastic	$\text{Pixel}_{\text{fail}}$	Jpeg
Org. DNN	96.26	32.58	45.18	51.53	**87.49**	56.43	82.11	81.71	87.58	80.46	91.30	94.96	**82.05**	88.31	0.00	82.67
CF-DARTS	96.55	39.45	52.73	52.03	86.79	60.50	83.00	82.10	88.62	83.91	91.35	95.22	81.19	88.50	31.43	83.95
CF-DARTSE	**97.52**	**43.80**	**56.28**	**55.01**	87.29	**65.49**	**84.12**	**82.94**	**90.53**	**85.44**	**91.65**	**96.55**	81.52	**89.94**	**87.62**	**85.85**

Table 4
Top-1 accuracy of the original DNN and refined DNNs on the raw Tiny-ImageNet testing dataset (*i.e.*, 'Clean' column) and respective 15 corruptions. Note that, the refined DNNs' architectures are searched on the training and validation datasets of CIFAR-10 and the selected failure set. RF-DARTS and CF-DARTS are our methods based on the random failure set selection and core failure set selection strategies, respectively.

			Noise			Blur				Weather				Digital			
		Clean	Gauss	Shot	Impulse	Defocus	Glass	Motion	Zoom	Snow	Frost	Fog	Bright	Contrast	Elastic	Pixel	Jpeg
Org. DNN		62.63	23.55	29.06	24.14	22.44	23.44	29.02	24.82	31.50	31.15	28.94	39.29	13.53	35.63	44.02	42.58
Researched DNN on $\mathcal{D}_{\text{fail}}^{\text{gn}}/\mathcal{C}_{\text{fail}}^{\text{gn}}$	RF-DARTS	64.06	25.09	30.44	25.42	21.75	24.17	28.00	25.29	31.30	31.93	29.69	38.47	13.99	36.32	44.90	44.07
	CF-DARTS	65.52	25.43	31.46	26.35	23.47	23.00	29.83	26.36	32.99	33.86	32.04	40.88	14.92	37.62	46.38	45.10
Researched DNN on $\mathcal{D}_{\text{fail}}^{\text{gb}}/\mathcal{C}_{\text{fail}}^{\text{gb}}$	RF-DARTS	64.26	25.75	30.95	25.84	21.66	23.75	27.06	24.75	31.90	32.49	29.38	39.85	13.80	36.03	45.78	44.69
	CF-DARTS	62.08	29.67	35.28	28.42	23.95	28.28	33.12	27.79	36.09	36.02	31.57	42.83	15.73	41.50	48.21	50.20
Researched DNN on $\mathcal{D}_{\text{fail}}^{\text{fg}}/\mathcal{C}_{\text{fail}}^{\text{fg}}$	RF-DARTS	64.59	25.42	31.42	27.08	24.25	23.87	30.89	27.29	33.71	34.23	30.97	41.19	14.50	37.52	46.10	45.38
	CF-DARTS	66.00	26.14	32.15	26.90	24.31	23.72	29.89	26.83	33.19	34.56	31.76	42.95	14.57	37.70	46.77	45.20
Researched DNN on $\mathcal{D}_{\text{fail}}^{\text{pl}}/\mathcal{C}_{\text{fail}}^{\text{pl}}$	RF-DARTS	63.40	23.95	29.86	24.72	23.14	22.54	29.09	25.81	30.77	30.77	29.19	39.75	14.11	35.42	43.79	43.27
	CF-DARTS	64.04	25.13	30.37	26.02	23.10	23.17	29.07	25.28	31.72	32.72	29.91	40.91	14.19	35.70	44.25	43.06

Table 5

Top-1 accuracy of the initial DNN and refined DNNs with RF-‘**’, RF-‘**’E, CF-‘**’ and CF-‘**’E on the original testing dataset (*i.e.*, ‘Clean’ column), the specific corruptions (*i.e.*, $\text{Gauss}_{\text{fail}}$), and other 14 corruptions. ‘**’ represents the names of different NAS methods, *i.e.*, PC-DARTS [16], DARTSPT [38], and RobDARTS [18].

		Noise			Blur				Weather				Digital			
	Clean	$\text{Gauss}_{\text{fail}}$	Shot	Impulse	Defocus	Glass	Motion	Zoom	Snow	Frost	Fog	Bright	Contrast	Elastic	Pixel	Jpeg
PCDARTS	97.00	0.00	49.38	58.08	84.24	67.69	79.86	79.00	90.01	85.78	92.01	95.63	85.42	86.90	73.39	81.22
RF-PCDARTS	96.68	13.30	51.85	49.83	84.33	63.04	81.68	79.37	88.41	84.00	91.41	95.08	83.91	87.34	70.83	81.45
RF-PCDARTSE	97.01	84.32	88.05	73.28	85.00	71.55	82.04	79.90	89.86	88.08	92.06	95.94	83.15	88.08	74.15	82.39
CF-PCDARTS	96.84	13.76	52.60	51.53	84.96	64.90	81.66	80.03	88.75	84.97	91.61	95.51	83.33	87.46	71.80	80.99
CF-PCDARTSE	**96.95**	**88.69**	**90.48**	**77.66**	**85.20**	**72.65**	**82.45**	**81.20**	**89.86**	**88.86**	**92.59**	**96.83**	**84.84**	**88.30**	**74.32**	**82.98**
DARTSPT	97.41	0.00	48.23	54.18	89.81	76.03	86.06	84.07	90.32	86.16	92.26	96.25	86.59	90.29	80.53	83.59
RF-DARTSPT	97.29	12.55	48.07	54.76	87.26	68.04	85.16	80.85	89.59	84.04	92.41	96.25	85.37	89.48	79.39	83.56
RF-DARTSPTE	97.64	89.43	92.84	84.64	88.59	75.42	86.70	82.91	90.24	88.11	93.12	96.50	86.56	90.49	78.34	84.37
CF-DARTSPT	97.32	14.74	48.80	56.25	88.12	65.35	85.73	82.35	90.14	84.60	92.85	96.12	86.57	89.88	79.57	83.45
CF-DARTSPTE	**97.75**	**90.75**	**92.85**	**86.12**	**90.09**	**75.89**	**87.29**	**84.98**	**91.24**	**88.49**	**93.55**	**96.59**	**86.83**	**90.88**	**83.64**	**85.97**
RobDARTS	96.96	0.00	51.14	55.98	86.81	65.88	83.41	81.48	90.10	85.88	92.99	96.00	87.43	89.38	75.06	83.45
RF-RobDARTS	97.20	6.83	41.95	46.16	88.24	68.23	86.44	82.30	89.40	84.75	92.36	95.85	85.41	90.22	80.40	84.11
RF-RobDARTSE	97.32	87.86	91.14	84.77	89.13	74.49	86.69	82.84	90.81	87.42	92.44	96.33	85.51	90.64	81.51	85.62
CF-RobDARTS	97.30	12.73	48.95	52.15	88.73	72.21	86.29	82.85	90.39	85.77	93.25	95.97	85.40	90.39	77.14	83.85
CF-RobDARTSE	**97.88**	**88.55**	**91.80**	**86.49**	**90.23**	**77.42**	**87.43**	**83.99**	**91.66**	**89.08**	**93.68**	**96.75**	**86.92**	**91.51**	**81.98**	**85.73**

Table 6

Top: Top-1 accuracy of the initial DNN and refined DNNs via RF-DARTS and CF-DARTS with different sizes of $\mathcal{C}_{\text{fail}}$ on the original testing dataset (*i.e.*, ‘Clean’ column), the specific corruptions (*i.e.*, $\text{Gauss}_{\text{fail}}$), and other 14 corruptions. Bottom: Top-1 accuracy of another sophisticated pre-trained DNN (*i.e.*‘Org. DNN-v2’) and its refined counterparts through our RF-DARTS and CF-DARTS on the original testing dataset (*i.e.*, ‘Clean’ column), the specific corruptions (*i.e.*, $\text{Gauss}_{\text{fail}}$), and other 14 corruptions.

		Noise			Blur				Weather				Digital			
	Clean	$\text{Gauss}_{\text{fail}}$	Shot	Impulse	Defocus	Glass	Motion	Zoom	Snow	Frost	Fog	Bright	Contrast	Elastic	Pixel	Jpeg
Org. DNN	96.26	0.00	45.18	51.53	87.49	56.43	82.11	81.71	87.58	80.46	91.30	94.96	82.05	88.31	76.30	82.67
RF-DARTS(1000)	96.73	14.06	43.94	56.96	86.48	66.47	85.24	79.33	87.88	82.02	92.03	95.47	85.17	89.75	77.79	83.40
CF-DARTS(1000)	96.62	**20.05**	53.55	46.09	87.67	71.93	84.93	81.43	89.11	85.26	91.62	95.41	84.51	89.96	78.23	84.55
RF-DARTS(5000)	96.58	15.16	46.73	51.53	86.36	65.85	84.44	80.68	87.36	82.27	91.50	95.23	84.41	89.46	77.13	83.74
CF-DARTS(5000)	96.53	**22.37**	53.76	48.24	84.32	66.72	82.81	75.54	87.64	82.53	91.36	95.26	82.33	89.19	78.07	84.30
RF-DARTS(10000)	96.77	11.82	46.97	51.20	87.93	65.81	85.03	83.41	89.02	84.13	91.96	95.39	82.67	89.85	80.61	83.43
CF-DARTS(10000)	96.76	**16.59**	53.72	50.16	87.59	67.71	84.07	82.21	89.63	85.22	91.70	95.76	82.61	89.59	77.03	84.02
RF-DARTS(15000)	96.50	10.25	44.75	54.16	85.37	67.35	82.56	78.70	87.65	81.94	90.65	94.95	81.94	88.69	78.04	83.80
CF-DARTS(15000)	96.93	**14.46**	51.82	55.26	88.46	70.39	85.63	82.14	89.00	83.28	91.90	95.69	84.88	90.05	79.30	83.37
RF-DARTS(20000)	96.97	12.12	49.52	50.07	88.90	67.40	87.01	84.20	89.44	84.61	92.33	95.82	83.58	90.26	78.12	84.66
CF-DARTS(20000)	96.51	**14.21**	55.70	51.85	87.58	69.35	85.75	80.83	88.58	82.92	91.44	95.42	83.30	89.61	77.99	84.02
Org. DNN-v2	97.36	0.00	47.21	54.43	87.81	66.25	84.30	82.18	89.41	84.17	92.63	96.17	84.23	89.51	78.16	83.53
RF-DARTS	95.91	18.35	49.82	49.34	83.09	63.82	78.70	77.90	86.61	82.78	91.06	94.64	82.58	86.17	70.74	80.36
CF-DARTS	96.24	**19.12**	51.72	59.53	83.28	65.13	79.60	78.77	86.84	82.93	90.72	94.81	79.51	87.40	74.06	81.55
RF-DARTSE	95.95	73.07	81.81	65.57	82.80	66.64	79.29	78.36	87.91	86.09	90.10	94.75	80.19	86.29	73.27	80.89
CF-DARTSE	96.40	**83.25**	88.07	75.41	85.17	72.95	82.39	80.95	88.31	88.02	90.76	95.10	82.45	88.18	74.61	83.16

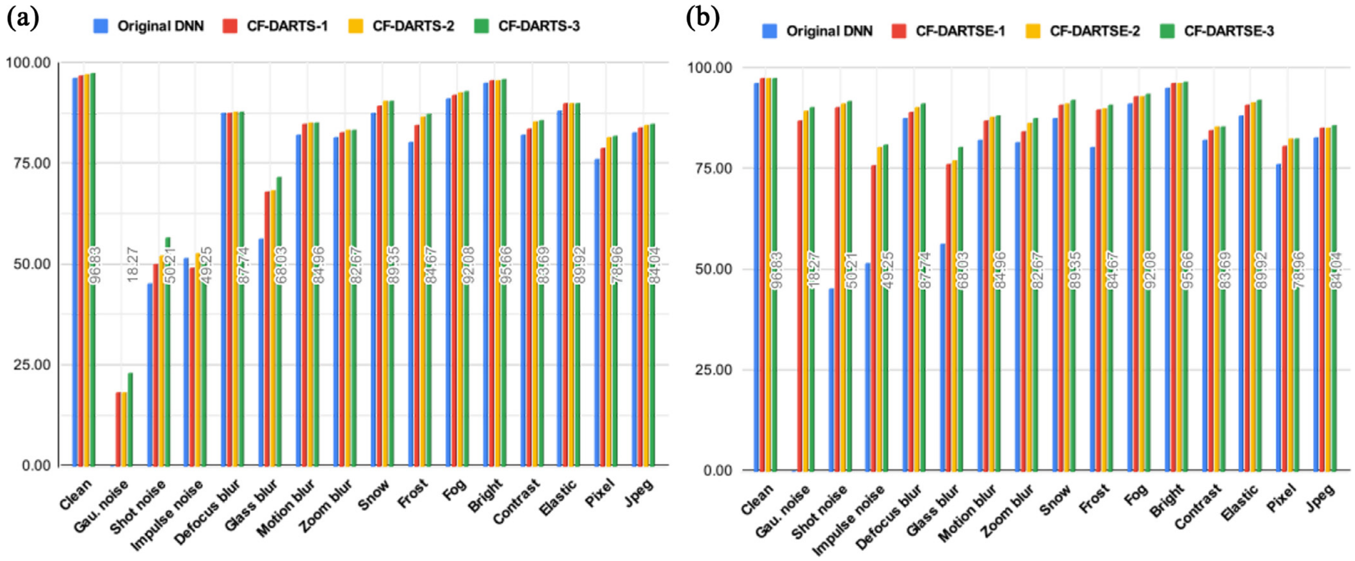


Fig. 4. Accuracy comparison of CF-DARTS(E)-{1,2,3} and the original DNN. CF-DARTS(E)-{1,2,3} are used the refine the ϕ_0 with the guidance of Gaussian noise corruption. (a) displays the results of CF-DARTS-{1,2,3} and (b) represents the results of CF-DARTSE-{1,2,3}.

Table 7
Time cost of CF-DARTS.

	Total	Core-Failure-Set Search	Network Architecture Search	Network Retraining
Avg. Time (Hours)	44.1472	0.4732	12.1700	31.5040
Std. Dev.	1.5049	0.0314	0.0935	1.4598
Ratio	100.0 %	1.0 %	27.6 %	71.4 %

ment but not for model weight retraining. ❷ although the size of $\mathcal{C}_{\text{fail}}$ is different, CF-DARTS always outperforms RF-DARTS, further demonstrating the effectiveness of the proposed core-failure-set selection method.

4.6. Influences of initial networks

In our experiment, we found that the architecture searched by the original DARTS does not always perform well. To eliminate the possible impact of deviation of the accuracy of the initial network, we generated five additional architectures by DARTS and selected the best model as our 'Org. DNN-v2'. As shown in Table 6 (Bottom), compared with the 'Org. DNN', the new version with more refinements achieved much better accuracy on the clean testing dataset and 14 corrupted datasets. Even then, our methods (*i.e.*, CF-DARTS and CF-DARTSE) still enhance the robustness against the specific corruption (*i.e.*, $\text{Gauss}_{\text{fail}}$) and other 14 corruptions significantly with obvious advantages over RF-DARTS and RF-DARTSE.

4.7. Influences of iteration numbers

As detailed in Section 3.4, our method can be conducted for several iterations where each iteration contains the optimization of the architecture and weights with 20 epochs. In the previous experiments, we set the iteration number as one. Here, we study the influence of the iteration numbers beyond one by taking the Gaussian noise (gn) as the specific corruption (*i.e.*, 'xx=gn' in Section 3.4). Specifically, we try the CF-DARTS and CF-DARTSE for robustness enhancement of the original DNN ϕ_0 with the iteration numbers as 1, 2, and 3, respectively, which are denoted as CF-DARTS(E)-{1,2,3}. We show the results in Fig. 4 and see that: *First*, all CF-DARTS methods enhance the robustness of ϕ_0 against the Gaussian noise as well as other corruption types. *Second*, all CF-DARTS methods do not harm the accuracy of the clean images. *Third*, the accuracy of enhanced DNNs becomes higher as the iteration number is larger under almost all corruptions.

4.8. Cost analysis

As detailed in Section 3.4, our method consists of three processes, that is, core-failure set search, network architecture search, and network retraining. We report the average time cost of the whole process in Table 7 and observe that the core-failure-set search only accounts for 1.0% of the whole cost while the network architecture search and retraining take 99.0% time cost, which demonstrates that our method has a limited effect on the time cost of the original DARTS.

5. Conclusion

In this paper, we have investigated how to refine a deployed DNN model's architecture for enhancing its robustness with the guidance of a few collected and misclassified examples that are degraded by some unknown but specific corruption patterns in the wild. We have made a surprising and interesting observation that by merely adding a few corrupted and misclassified examples into the clean training dataset, we can already refine the model architecture and significantly enhance the model robustness. We have further proposed a novel *core-failure-set guided DARTS* that embeds a K-center-greedy algorithm for DARTS to select suitable corrupted failure examples to refine the model architecture. Compared with the raw NAS method as well as the SOTA data-augmentation-based enhancement methods, our final method can achieve higher accuracy on both corrupted datasets and the original clean dataset. In particular, on some of the corruptions, we can achieve over 45% absolute accuracy improvements. Although achieving progress, the method does not fully utilize the failure patterns in collected failure examples and deliveries them to the training data. In the future, we will further extend the proposed method by combining it with the data augmentation methods to fill the gap. Moreover, we could extend the method to video data and other visual intelligent tasks.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

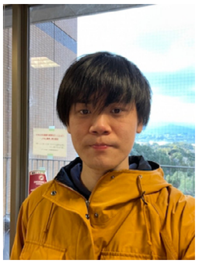
Acknowledgement

This work was supported in part by the National Natural Science Foundation of China (61906135, 62020106004, 92048301); and in part by the Tianjin Science and Technology Plan Project (20JCQNJC01350). This research was also supported in part by Canada CIFAR AI Chairs Program, Amii RAP program, the Natural Sciences and Engineering Research Council of Canada (NSERC No.RGPIN-2021-02549, No.RGPAS-2021-00034, No.DGECR-2021-00019), as well as JSPS KAKENHI Grant No.JP20H04168, No.JP21H04877, JST-Mirai Program Grant No.JPMJMI20B8.

References

[1] H. Liu, K. Simonyan, Y. Yang, DARTS: differentiable architecture search, in: International Conference on Learning Representations, 2019.
[2] B. Zoph, V. Vasudevan, J. Shlens, Q.V. Le, Learning transferable architectures for scalable image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 8697–8710.
[3] E. Real, A. Aggarwal, Y. Huang, Q.V. Le, Regularized evolution for image classifier architecture search, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 4780–4789.
[4] M. Tan, Q. Le, EfficientNet: rethinking model scaling for convolutional neural networks, in: International Conference on Machine Learning, PMLR, 2019, pp. 6105–6114.
[5] H. Pham, Z. Dai, Q. Xie, Q.V. Le, Meta pseudo labels, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 11557–11568.
[6] A. Wongpanich, H. Pham, J. Demmel, M. Tan, Q. Le, Y. You, S. Kumar, Training efficientnets at supercomputer scale: 83% imagenet top-1 accuracy in one hour, in: 2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2021, pp. 947–950, doi:10.1109/IPDPSW52791.2021.00146.
[7] X. Dong, M. Tan, A.W. Yu, D. Peng, B. Gabrys, Q.V. Le, AutoHAS: efficient hyperparameter and architecture search, arXiv preprint arXiv:2006.03656(2020).
[8] Y. Guo, Y. Chen, Y. Zheng, P. Zhao, J. Chen, J. Huang, M. Tan, Breaking the curse of space explosion: towards efficient NAS with curriculum search, in: International Conference on Machine Learning, PMLR, 2020, pp. 3822–3831.
[9] S. Yan, B. Fang, F. Zhang, Y. Zheng, X. Zeng, M. Zhang, H. Xu, HM-NAS: efficient neural architecture search via hierarchical masking, in: Proceedings of the IEEE International Conference on Computer Vision Workshops, 2019. pp. 0–0
[10] D. Stamoulis, R. Ding, D. Wang, D. Lymberopoulos, B. Priyantha, J. Liu, D. Marculescu, Single-path NAS: designing hardware-efficient convnets in less than 4 h, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2019, pp. 481–497.
[11] E. Rusak, L. Schott, R.S. Zimmermann, J. Bitterwolf, O. Bringmann, M. Bethge, W. Brendel, A simple way to make neural networks robust against diverse image corruptions, in: A. Vedaldi, H. Bischof, T. Brox, J.-M. Frahm (Eds.), Computer Vision – ECCV 2020, 2020, pp. 53–69.
[12] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, Adv. Neural Inf. Process. Syst. 25 (2012).
[13] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.
[14] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: International Conference on Learning Representations, 2015.
[15] X. Dong, Y. Yang, Searching for a robust neural architecture in four GPU hours, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 1761–1770.
[16] Y. Xu, L. Xie, X. Zhang, X. Chen, G.-J. Qi, Q. Tian, H. Xiong, {PC}-{darts}: partial channel connections for memory-efficient architecture search, in: International Conference on Learning Representations, 2020.
[17] X. Chen, L. Xie, J. Wu, Q. Tian, Progressive differentiable architecture search: bridging the depth gap between search and evaluation, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 1294–1303.
[18] A. Zela, T. Elsken, T. Saikia, Y. Marrakchi, T. Brox, F. Hutter, Understanding and robustifying differentiable architecture search, in: International Conference on Learning Representations, 2020.
[19] Y. Tian, C. Liu, L. Xie, J. jiao, Q. Ye, Discretization-aware architecture search, Pattern Recognit. 120 (2021) 108186.
[20] Y. Hu, X. Wang, L. Li, Q. Gu, Improving one-shot NAS with shrinking-and-expanding supernet, Pattern Recognit. 118 (2021) 108025.
[21] Q. Guo, X.-J. Wu, J. Kittler, Z. Feng, Differentiable neural architecture learning for efficient neural networks, Pattern Recognit. 126 (2022) 108448.
[22] C. Xue, M. Hu, X. Huang, C.-G. Li, Automated search space and search strategy selection for autoML, Pattern Recognit. 124 (2022) 108474.
[23] Q. Guo, F. Juefei-Xu, X. Xie, L. Ma, J. Wang, B. Yu, W. Feng, Y. Liu, Watch out! Motion is blurring the vision of your deep neural networks, Advances in Neural Information Processing Systems (NeurIPS), 2020.
[24] B. Tian, F. Juefei-Xu, Q. Guo, X. Xie, X. Li, Y. Liu, AVA: adversarial vignetting attack against visual recognition, in: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 2021.
[25] N. Carlini, D. Wagner, Towards evaluating the robustness of neural networks, in: 2017 IEEE Symposium on Security and Privacy (SP), IEEE, 2017, pp. 39–57.
[26] D. Hendrycks, T. Dietterich, Benchmarking neural network robustness to common corruptions and perturbations, ICLR, 2019.
[27] Z. He, A.S. Rakin, D. Fan, Parametric noise injection: trainable randomness to improve deep neural network robustness against adversarial attack, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019.
[28] S. Schneider, E. Rusak, L. Eck, O. Bringmann, W. Brendel, M. Bethge, Improving robustness against common corruptions by covariate shift adaptation, Adv. Neural Inf. Process. Syst. 33 (2020) 11539–11551.
[29] O. Sener, S. Savarese, Active learning for convolutional neural networks: a core-set approach, in: International Conference on Learning Representations, 2018.
[30] I.W. Tsang, J.T. Kwok, P.-M. Cheung, Core vector machines: fast SVM training on very large data sets, J. Mach. Learn. Res. 6 (13) (2005) 363–392. http://jmlr.org/papers/v6/tsang05a.html
[31] S. Har-Peled, A. Kushal, Smaller coresets for k-median and k-means clustering, in: Proceedings of the Twenty-First Annual Symposium on Computational Geometry, in: SCG '05, 2005, pp. 126–134.
[32] D. Hendrycks, N. Mu, E.D. Cubuk, B. Zoph, J. Gilmer, B. Lakshminarayanan, AugMix: a simple data processing method to improve robustness and uncertainty, in: Proceedings of the International Conference on Learning Representations (ICLR), 2020.
[33] S. Yun, D. Han, S.J. Oh, S. Chun, J. Choe, Y. Yoo, CutMix: regularization strategy to train strong classifiers with localizable features, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 6023–6032.
[34] G.W. Wolf, Facility location: concepts, models, algorithms and case studies. series: contributions to management science, Int. J. Geogr. Inf. Sci. 25 (2) (2011) 331–333.
[35] A. Krizhevsky, Learning Multiple Layers of Features from Tiny Images, 2012.
[36] Y. Le, X. Yang, Tiny imagenet visual recognition challenge, CS 231N 7 (7) (2015) 3.
[37] B. Yu, H. Qi, Q. Guo, F. Juefei-Xu, X. Xie, L. Ma, J. Zhao, DeepRepair: style-guided repairing for deep neural networks in the real-world operational environment, IEEE Trans. Reliab. (2021) 1–16.
[38] R. Wang, M. Cheng, X. Chen, X. Tang, C.-J. Hsieh, Rethinking architecture selection in differentiable NAS, in: International Conference on Learning Representations (ICLR), 2021.

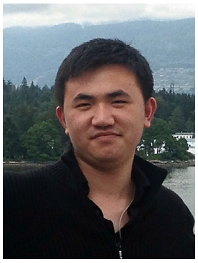
Xuhong Ren received her BS degree in Electronic and Information Engineering from the North China Institute of Aerospace Engineering in 2011, ME degree in computer application technology from the College of Computer and Information Technology, China Three Gorges University in 2014. From 2014 to 2020, she has worked as a lecturer in the North China Institute of Aerospace Engineering, and now is a doctoral student of Tianjin University of Technology, her research interests include computer vision, machine learning and tracking.



Jianlang Chen received a Bachelor's degree in Computer Science from Beijing University of Technology in 2018. He is a master student in the Information Science and Electrical Engineering faculty of Kyushu University. He is generally interested in deep learning, computer vision, neural architecture search, etc.

Felix Juefei-Xu received the PhD degree in Electrical and Computer Engineering from Carnegie Mellon University (CMU), Pittsburgh, PA, USA. Prior to that, he received the MS degree in Electrical and Computer Engineering and the MS degree in Machine Learning from CMU, and the BS degree in Electronic Engineering from Shanghai Jiao Tong University (SJTU), Shanghai, China. Currently, he is a Research Scientist with Alibaba Group, Sunnyvale, CA, USA, with research focus on a fuller understanding of deep learning where he is actively exploring new methods in deep learning that are statistically efficient and adversarially robust. He also has broader interests in pattern recognition, computer vision, machine learning, optimization, statistics, compressive sensing, and image processing. He is the recipient of multiple best/distinguished paper awards, including IJCB'11, BTAS'15-16, ASE'18, and ACCV'18.

Wanli Xue is a lecturer of the school of computer science and engineering at Tianjin University of Technology. He received the BS degree in pure and applied mathematics from Tianjin Polytechnic University in 2009 and PhD in technology of computer application from Tianjin University in 2019. His research interests include visual tracking, images stitching.



Qing Guo received his BS degree in Electronic and Information Engineering from the North China Institute of Aerospace Engineering in 2011, ME degree in computer application technology from the College of Computer and Information Technology, China Three Gorges University in 2014, and the PhD degree in computer application technology from the School of Computer Science and Technology, Tianjin University, China. He was a research fellow with the Nanyang Technological University, Singapore, from Dec. 2019 to Sep. 2020. He is currently a Wallenberg-NTU Presidential Postdoctoral Fellow with the Nanyang Technological University, Singapore. His research interests include computer vision, AI security, and image processing.

Lei Ma is currently an associate professor and Canada CIFAR AI chair at the University of Alberta, Canada. He also holds a research fellow position, co-leading Intelligent Software Engineering Lab of Kyushu University Japan and honorably affiliated with Alberta Machine Intelligence Institute. He received his PhD and ME from The University of Tokyo, and BE from Shanghai Jiao Tong University. His recent research centers around the interdisciplinary fields of Software Engineering (SE) and Trustworthy AI with a special focus on the quality and reliability assurance of machine learning and AI Systems. Many of his work were published in top-tier software engineering and AI venues (e.g., TSE, ICSE, FSE, ASE, ISSTA, ICML, NeurIPS, ACM MM, AAAI, IJCAI, ECCV, CAV). He is a recipient of more than 10 prestigious academic awards, including 3 ACM SIGSOFT Distinguished Paper Awards.

Jianjun Zhao received his BS degree in computer science from Tsinghua University, China, in 1987 and his PhD degree in computer science from Kyushu University, Japan, in 1997. He then joined the Department of Computer Science and Engineering, Fukuoka Institute of Technology, Japan, as an assistant professor and was promoted to associate professor in 2000. Since November 2005, he was a professor in the School of Software and then the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. Since April 2016, he has joined the School of Information Science and Electrical Engineering, Kyushu University, as a professor. His main research interests include program analysis and verification, AI quality assurance, automatic programming, software testing, and programming language design.

Shengyong Chen received the PhD degree in robot vision from the City University of Hong Kong, Honk Kong, in 2003. From 2006 to 2007, he was with the University of Hamburg, Hamburg, Germany. He is currently a Professor with the Tianjin University of Technology, Tianjin, China. His current research interests include computer vision, robotics, and image analysis.