

# FlatNAS: optimizing Flatness in Neural Architecture Search for Out-of-Distribution Robustness

Matteo Gambella, Fabrizio Pittorino, and Manuel Roveri

Dipartimento di Elettronica, Informazione e Bioingegneria,

Politecnico di Milano, Milan, Italy

Email: {matteo.gambella, fabrizio.pittorino, manuel.roveri}@polimi.it

**Abstract**—Neural Architecture Search (NAS) paves the way for the automatic definition of Neural Network (NN) architectures, attracting increasing research attention and offering solutions in various scenarios. This study introduces a novel NAS solution, called Flat Neural Architecture Search (FlatNAS), which explores the interplay between a novel figure of merit based on robustness to weight perturbations and single NN optimization with Sharpness-Aware Minimization (SAM). FlatNAS is the first work in the literature to systematically explore flat regions in the loss landscape of NNs in a NAS procedure, while jointly optimizing their performance on in-distribution data, their out-of-distribution (OOD) robustness, and constraining the number of parameters in their architecture. Differently from current studies primarily concentrating on OOD algorithms, FlatNAS successfully evaluates the impact of NN architectures on OOD robustness, a crucial aspect in real-world applications of machine and deep learning. FlatNAS achieves a good trade-off between performance, OOD generalization, and the number of parameters, by using only in-distribution data in the NAS exploration. The OOD robustness of the NAS-designed models is evaluated by focusing on robustness to input data corruptions, using popular benchmark datasets in the literature.

**Index Terms**—Neural Architecture Search (NAS), Once-For-All Network (OFA), Constrained optimization, Sharpness-Aware Minimization (SAM), Out-of-Distribution (OOD) robustness

## I. INTRODUCTION

Deep learning models are widely spread nowadays [1]. However, the definition of deep neural network (NN) architectures is typically a complex and time-consuming task [2], often requiring a high expertise in the field [2]. Neural Architecture Search (NAS) [3], a promising method of AutoML [2], aims at automating the design of a NN exploring different architectural configurations given a NN topology, a task to be accomplished, and a dataset used for training and validation to provide the optimal architecture.

Most of works on NAS focus on the design of NN architectures while taking into account target hardware. NAS solutions in this field fall into the area of Hardware-Aware NAS [4]. An emerging direction of these NAS solutions regards the use of constraints in the NAS exploration [5]. A constrained exploration is crucial in relevant scenarios such as TinyML [6] and privacy-preserving deep learning solutions [5].

The traditional workflow of NAS, in both the constrained and non-constrained settings, deals with data that are sampled from a single dataset, resulting in NN architectures that are trained or fine-tuned on a single data distribution. However, in real-world scenarios of modern machine and deep learning,

mismatches of test and training data distributions are often observed, with deep learning models encountering significant performance drop in Out-Of-Distribution (OOD) scenarios [7], [8]. This property undermines the trustworthiness of systems depending on such models and potentially threatens the safety of their users. A crucial challenge in this direction is how much NN architectures are robust to changes on the distribution on which they are trained. Out-of-distribution robustness considers how to design the NN architectures resilient to various forms of data shift at test time, with extensive efforts being devoted to improving OOD robustness [9] through algorithmic enhancements. A natural algorithmic improvement for OOD robust NN architectures is to use NN optimizers that look for flat regions in the loss landscape of NNs, such as SAM and its variants [10]–[12], as flatness-enhancing algorithms have been shown to optimize robustness to weight perturbation [13]. In fact, the geometrical structure of the loss landscape of NNs has been a key topic of study for several decades [13]–[15] and there is accumulating evidence of the connection between the flatness of minima found by optimization algorithms like stochastic gradient descent (SGD) and the generalization performance of the NN [16]–[19]. Some works concentrate on the role of flatness-optimizing algorithms for OOD generalization [20]–[22], highlighting that this is a promising direction for algorithmic improvement.

As it is difficult to define principled guidelines for designing more robust NN architectures, NAS is a natural candidate for automating their design. Many works suggest that the existing NAS solutions, and more in general AutoML methods, search for models achieving optimal performance, while neglecting their robustness [23], [24]. Furthermore, [25] explores the correlation between model complexity and robustness, highlighting that in a given architectural family, increasing the number of network parameters can lead to enhanced robustness. This suggests that networks constrained by the number of parameters are especially prone to degradation when faced with OOD data. These considerations highlight the need for a unified NAS framework that simultaneously optimizes all relevant aspects along these dimensions, i.e., increase the accuracy, increase the robustness, and reduce the number of parameters of the NN. Currently, the NAS solutions addressing robustness are mainly designed to optimize the adversarial robustness of the NNs [26]–[31], which is a somehow related but different task with respect to OOD robustness.

In particular, some of these NAS algorithms account for the flatness of the input loss landscape of the network [30], [31].

The goal of our work is to develop Flat Neural Architecture Search (FlatNAS), a Neural Architecture Search algorithm, extending Constrained Neural Architecture Search (CNAS) [5] by exploiting its constrained exploration while searching for flat regions in the loss landscape of NNs for enhanced generalization capabilities and OOD robustness. In particular, the explored architectures are constrained in their number of parameters (this case is of particular interest as smaller NNs are thought to be in general less robust [25]). In order to effectively account for flat regions, we introduce a novel metric,  $R(x, \sigma)$ , which assesses the capability of a given NN architecture  $x$  to maintain accuracy under parameter perturbations of intensity  $\sigma$ , while optimizing single NN with SAM [10], [11] instead of traditional SGD. We highlight that our robustness metric pertains to OOD robustness rather than adversarial robustness, and that our approach aims to design NNs with superior performance in OOD tasks by using exclusively in-distribution data in the NAS search process. The novel contributions of our research are twofold:

- the development of the first NAS algorithm specifically tailored to investigate flat regions within the optimization landscape of NNs with constrained number of parameters;
- the introduction of a new metric designed to evaluate the robustness of NN architectures during NAS optimization.

The rest of this paper is organized as follows. Section II introduces the related works regarding NAS solutions for robustness. Section III provides the background. Section IV introduces the FlatNAS framework developed in this study, while in Section V the experimental results aiming at evaluating the effectiveness of FlatNAS are shown. Conclusions are finally drawn in Section VI. To facilitate comparisons and reproducibility, the source code of FlatNAS is released to the scientific community as a public repository.<sup>1</sup>

## II. RELATED LITERATURE

This section explores NAS solutions enhancing the robustness of NN architectures. We initially review the existing body of work in NAS that addresses adversarial robustness. Then, we focus on studies that optimize the flatness of the loss landscape with respect to input data. Finally, we present the first example of NAS specifically tailored for OOD generalization.

In the field of NAS addressing adversarial robustness, RNAS [28] is a NAS solution derived from DARTS [32] designed to optimize the balance between accuracy and adversarial robustness by implementing a regularization process that computes the similarity between outputs from natural and adversarial data. For computational efficiency, RNAS employs noise examples as proxies for adversarial data during the NAS search phase. Differently, ROBNET [29], utilizes an Once-For-All (OFA) supernet [33] in order to identify NNs that exhibit optimal adversarial robustness. It achieves this goal by sampling architectures from the supernet and subsequently

fine-tuning these candidate architectures over a few epoch, to evaluate their accuracy when subjected to adversarial attacks. Similarly, NADAR [26], based on DARTS, introduces a constrained optimization approach aimed at enhancing robustness within a specified computational budget measured in floating point operations (FLOPS). This method expands the backbone networks by integrating dilation networks, which are additional layers designed to augment robustness, considering their resulting complexity overhead. Finally, RACL [27] presents a DARTS-based technique for optimizing robustness through a regularization process approximating the Lipschitz constant derived from the architecture parameters.

A different perspective is provided in ADVRUSH [30] and NA-DARTS [31] whose goal is to address the problem of finding architectures with inherent robustness by targeting a smooth input loss landscape. More specifically, ADVRUSH, derived from DARTS, features an objective function that combines top1 accuracy on clean validation data and a smoothness-based regularization term. This regularization aims to optimize the curvature degree (flatness) of the input loss landscape evaluated using the largest eigenvalue of the Hessian matrix. In contrast, NA-DARTS, another DARTS-based approach, differs from previous methods by assessing robustness through alterations in NN architecture, such as substituting a convolution with a skip connection. This solution aims to optimize the flatness of the NN architecture by employing an objective function focused on performance across the NN architecture of neighboring configurations.

Finally, NAS-OOD [24] is the first example of NAS method designed specifically for OOD generalization. Given a predefined set of source domains, the aim of the OOD task is to discover the optimal network architecture that can generalize well to the unseen target domain. This DARTS-based approach optimizes the architecture on its performance on synthetically generated OOD data. A data generator is trained to create novel domain data by maximizing losses across various neural architectures, with NAS objective being to find parameters that effectively minimize these losses. The data generator and the neural architecture undergo simultaneous optimization. We highlight that the generator is designed to create images with different background patterns, textures, and colors. Differently, in our work we do not make any assumption about the type of perturbations considered and perform the NAS search only using in-distribution data. This study considers a general OOD generalization task and is not concentrated on OOD robustness to image corruptions and perturbations. Also, our method could be integrated with the proposed data generator.

## III. BACKGROUND

### A. Neural Architecture Search

NAS solutions can be classified based on three distinct dimensions [34].

The first dimension is the *Search Space*, defining the architecture representation. The most straightforward is the *entire-structured Search Space*, depicted layer-wise with each node representing a layer. Motivated by handcrafted architectures

<sup>1</sup><https://github.com/AI-Tech-Research-Lab/CNAS>

with repeated motifs [35], the *cell-based Search Space* defines each node as a cell (also called a block), representing a group of layers. Other variations include hierarchical cells or morphism-based identity transformations between layers.

The second dimension is the *Search Strategy*, determining how to explore the search space. Common strategies include reinforcement learning, gradient optimization, and evolutionary algorithms where genetic methods, especially NSGA-II [36], are widely used. NSGA-II generates offspring using a specific crossover and mutation, selecting the next generation based on fitness through nondominated sorting and crowding distance comparison. It is a multi-objective algorithm, optimizing different figures of merit and effectively handling constraints [37], that can be managed by redefining the objective function. The simplest scheme is:

$$\phi(x) = f(x) + pG(x) \quad (1)$$

where  $\phi(x)$  is the joint objective,  $f(x)$  is the original objective without constraints,  $p$  is the penalty term, and  $G(x)$  is a function deciding whether to apply the penalty by accounting for the constraints. The penalty can be static (a constant value) or dynamic (adapted during the evolutionary process).

The third dimension is the *Performance Estimation Strategy*, which estimates the performance without fully training every NN architecture in the search space (that would be computationally prohibitive). A popular strategy is weight sharing, with Once-For-All (OFA) [33] being a state-of-the-art example. OFA trains a comprehensive supernet of many network configurations once and evaluates candidate networks by fine-tuning from the supernet weights during NAS. The OFA supernet is trained, before the NAS search, using Progressive Shrinking, starting with the largest NN and progressively fine-tuning to support smaller sub-networks sharing weights. Another approach involves surrogate models such as Gaussian Process and Radial Basis Function. For instance, MSuNAS [36] proposes *adaptive-switching*, selecting the best accuracy predictor among four surrogate models based on a correlation metric named Kendall's Tau in each NAS iteration.

### B. Sharpness-Aware Minimisation (SAM)

Sharpness-Aware Minimization (SAM) has emerged as a significant advancement in the field of machine learning, particularly for deep neural networks, thanks to its ability to reduce generalization error by minimizing a sharpness measure that reflects the geometry of the loss landscape [10]. Traditionally, training of DNNs has focused on minimizing empirical loss, often leading to overfitting and convergence to sharp minima [19]. SAM addresses this issue by seeking flat minima through a min-max optimization problem, involving two forward-backward computations for each update. This approach achieved remarkable results in training various deep neural networks [12], [38].

## IV. THE PROPOSED FLATNAS

This section, introducing and detailing FlatNAS, is organized as follows: Section IV-A provides the problem formulation, Section IV-B provides an overall description of FlatNAS,

Section IV-C presents the figure of merit accounting for classification accuracy and robustness on OOD generalization, and finally Section IV-D introduces the figure of merit accounting for the number of parameters and the related constraint.

### A. Problem formulation

FlatNAS addresses the challenge of selecting a NN architecture by simultaneously optimizing classification accuracy and robustness to weight perturbations, while satisfying a constraint on the maximum number of parameters. FlatNAS can be defined as the following optimization problem:

$$\begin{aligned} & \text{minimize } \mathcal{G}(F_A(x), R(x, \sigma), F_P(x)) \\ & \text{s. t. } F_P(x) < \overline{F}_P, \\ & \quad x \in \Omega_x \end{aligned} \quad (2)$$

where  $\mathcal{G}$  is a multi-objective optimization function,  $x$  and  $\Omega_x$  represent a candidate NN architecture and the search space of the NN exploration, respectively; the metrics  $F_A(x)$  and  $R(x, \sigma)$  calculate the top-1 accuracy and the robustness of the architecture  $x$ , respectively.  $F_P(x)$  represents the number of parameters in architecture  $x$ , and  $\overline{F}_P$  denotes the upper limit on the allowable number of parameters. The optimization problem outlined in Eq. (2) is addressed by the FlatNAS framework, which is detailed in what follows.

### B. The overall view of FlatNAS

The FlatNAS framework, illustrated in Fig. 1, operates as follows. It receives a dataset  $\mathcal{DS}$ , which includes a training set for training candidate networks and a validation set for their validation. The framework utilizes a type of OFA supernet to select a set of candidate networks  $\Omega_x$ , defining the *Search Space* of NAS. Additionally, FlatNAS incorporates the constraint  $\overline{F}_P$  and the associated penalty in the objective function optimized by the *Search Strategy* to limit the number of parameters. Integral components of the framework are the intensity of the perturbation  $\sigma$ , used by the *Evaluator* to assess robustness, and the weight  $\alpha$ , used to combine classification performance and robustness.

The *Evaluator* module receives a NN architecture from the set  $\Omega_x$  and trains it by using the  $\mathcal{DS}$  dataset and the SAM optimizer. Subsequently, it assesses the NN robustness. The resulting output is an *archive*, i.e., a collection of tuples  $\langle x, F_A(x), F_P(x), R(x, \sigma) \rangle$ , representing the NN configurations and their corresponding figures of merit, i.e.,  $F_A(x)$  for accuracy,  $F_P(x)$  for the number of parameters, and  $R(x, \sigma)$  for robustness. Initially, a representative subset of OFA  $\Omega_x$  is sampled before starting the search. The number  $N_{start}$  of selected NN architectures is determined by the user and it is typically set to 100. Therefore, the initial size of the archive is equal to  $N_{start}$ .

In FlatNAS, the optimization problem outlined in Eq. (2) is reformulated to implicitly incorporate constraints:

$$\begin{aligned} & \text{minimize } \mathcal{G}(S_{AR}(x, \alpha, \sigma), F_{CP}(x, \overline{F}_P, p)) \\ & \text{s. t. } x \in \Omega_x \end{aligned} \quad (3)$$

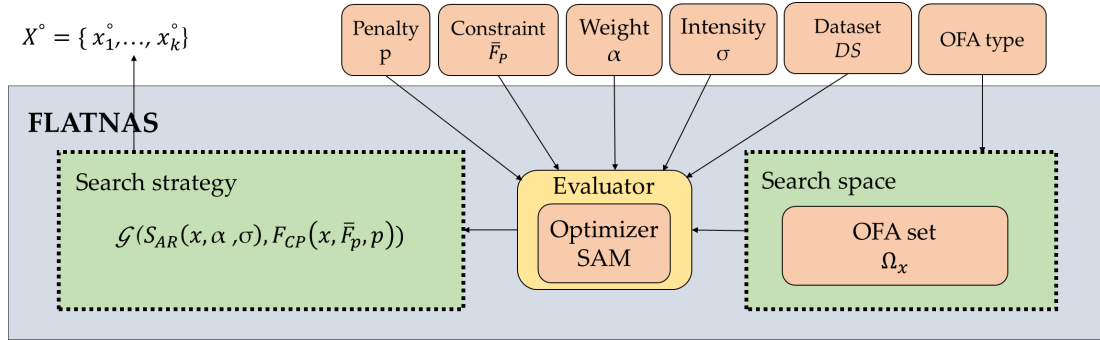


Fig. 1: The proposed FlatNAS framework, which is composed of a Search Space, Evaluator, and a Search Strategy module.

where  $S_{AR}(x, \alpha, \sigma)$  is the predicted value of  $F_{AR}(x, \alpha, \sigma)$  as estimated by a surrogate model, and  $F_{CP}(x, \bar{F}_P, p)$  accounts jointly for the number of parameters of the architecture and the related constraint. In particular,  $F_{AR}(x, \alpha, \sigma)$  is a novel figure of merit, which will be introduced in Section IV-C, accounting for both the classification accuracy and the robustness of model  $x$ . It utilizes a weight  $\alpha$  to balance the importance of these two factors with the intensity of the perturbation governed by the parameter  $\sigma$ .

The core of FlatNAS is the *Search Strategy* module. It adopts the NSGA-II genetic algorithm [36] to solve the bi-objective problem introduced in Eq. (3), by optimizing the objectives  $S_{AR}(x, \alpha, \sigma)$  and  $F_{CP}(x, \bar{F}_P, p)$ . The search process is iterative: at each iteration, the surrogate model, computing  $S_{AR}(x, \alpha, \sigma)$ , is chosen employing a mechanism called *adaptive-switching*, the same method used by MSuNAS [36], which selects the best surrogate model according to a correlation metric (i.e., Kendall’s Tau). The surrogate models are trained by using the *archive* as the dataset. We highlight that  $F_{CP}(x, \bar{F}_P, p)$  is computed analytically as shown in Section IV-D, hence a surrogate model is not needed. Then, a ranking of the candidate NNs, based on  $S_{AR}(x, \alpha, \sigma)$  and  $F_{CP}(x, \bar{F}_P, p)$ , is computed and a new set of candidates is obtained by the genetic algorithm and forwarded to the *Evaluator*. The *Evaluator* updates the *archive*, which becomes available for evaluation in the next iteration. At the end of the search, FlatNAS returns the set of the  $k$  NN architectures  $X^o = \{x_1^o, \dots, x_k^o\}$  characterized by the best trade-off among the objectives, where  $k$  is a user-specified value.

### C. The figure of merit $F_{AR}(x, \alpha, \sigma)$

In this section, we define the novel figure of merit of FlatNAS accounting jointly for the accuracy and the robustness.

To achieve this goal, we use a rescaling-invariant notion of flatness used in [13], [17]. Given a weight configuration  $w(x) \in \mathbb{R}^N$  (i.e., the weight relative to architecture  $x$ ), we define the *robustness*  $R(x, \sigma)$  as the average training error difference with respect to  $E_{\text{train}}(w(x))$  when perturbing the weight configuration  $w(x)$  by a (multiplicative) noise proportional to a parameter  $\sigma$ , i.e.,

$$R(x, \sigma) = \mathbb{E}_z E_{\text{train}}(w(x) + \sigma z \odot w(x)) - E_{\text{train}}(w(x)), \quad (4)$$

where  $\odot$  denotes the element-wise product and the expectation is over normally distributed  $z \sim \mathcal{N}(0, I_N)$ . In our experiments we set  $\sigma = 0.05$  (in principle varying the parameter  $\sigma$  can be informative [13], however a single value already shows a good correlation with generalization [17]). In practice, we compute  $R(x, \sigma)$  as the average value of  $E_{\text{train}}(w(x) + \sigma z \odot w(x)) - E_{\text{train}}$  over a user-defined number of samples  $M$  from  $z$  (in our case  $M = 20$ ).

Finally, we define  $F_{AR}(x, \alpha, \sigma)$  as the weighted sum of two different terms as follows:

$$F_{AR}(x, \alpha, \sigma) = \alpha(1 - F_A(x)) + \gamma(1 - \alpha)R(x, \sigma),$$

$$\gamma = \left( \sum_{i=1}^{N_k} F_A(x_i) \right) / \left( \sum_{i=1}^{N_k} R(x_i, \sigma) \right) \quad (5)$$

where  $F_A(x)$  is the top-1 classification accuracy on the validation set, while  $\alpha$  is a user-defined parameter that controls the relative significance of optimizing the two components, and  $\gamma$  is the ratio between the average value of the top1 accuracy and the average value of the robustness of the architectures in the *archive* at iteration  $k$  of the NAS with  $N_k$  the cardinality of the *archive* at iteration  $k$ . This latter term balances the order of magnitude between the top1 accuracy and the robustness in order to let  $\alpha$  work effectively.

### D. The figure of merit $F_{CP}(x, \bar{F}_P, p)$

In this section, we introduce the definition of the figure of merit  $F_{CP}(x, \bar{F}_P, p)$  accounting for the number of parameters and the constraints related.

A viable approach to incorporate constraints into a figure of merit is the one used in [5]. Given a constraint on the maximum number of parameters  $\bar{F}_P$  and a penalty factor  $p$ , a possible figure of merit  $F_{CP}$  accounting for the constraint is:

$$F_{CP}(x, \bar{F}_P, p) = F_P(x) + \max(0, F_P(x) - \bar{F}_P)p \quad (6)$$

The figure of merit  $F_{CP}(x, \bar{F}_P, p)$  imposes a penalty when the number of parameters in a model exceeds the maximum allowable limit. Conversely, no penalty is applied (as indicated by the max operator returning zero) when the constraint is satisfied. The extent of the penalty is proportional to the degree of constraint violation and the value of the constant  $p$ .

### E. The use of Sharpness-Aware Minimization (SAM)

In order to enhance flatness optimization at the single NN level, instead of SGD we use SAM [10] for optimizing NN architectures  $x$  during the NAS exploration. We will see that the use of SAM has a non-trivial effect on the NAS architectures space exploration, resulting not only in an optimization of the figure of merit  $F_{AR}(x, \alpha, \sigma)$  and consequently on more robust models on OOD data, but also in qualitatively different architectures. These aspects are detailed in the experimental results that are introduced in the next Section.

## V. EXPERIMENTAL RESULTS

This section describes the experimental results aiming at assessing the effectiveness of FlatNAS.

### A. Out-Of-Distribution (OOD) datasets

Numerous studies have shown that datasets with corrupted versions of natural data, such as images affected by noise or more complex distortions, can be utilized to evaluate model robustness [7], [9]. In particular, [7] introduces the first comprehensive benchmarks aimed specifically at robustness, concentrating on various types of corruption and perturbation. The CIFAR-10-C and CIFAR-100-C benchmarks [7], where the C at the end stands for the corrupted version of the original datasets, consist of a wide range of image corruptions that have been demonstrated to reflect robustness against certain real-world data variations. These benchmarks are designed to challenge models trained on the CIFAR-10 and CIFAR-100 datasets by providing a more demanding, held-out test set. Our experiments employ the original, clean datasets CIFAR-10 and CIFAR-100, along with their corrupted versions, CIFAR-10-C and CIFAR-100-C, which include 15 different types of distortions. Since the corrupted datasets are available only with a resolution of 32 while we consider NNs with different resolutions, we reproduced the corrupted data for each resolution evaluated. We highlight that simply resizing the images of the original corrupted datasets would not be equivalent. More information about the specific distortions and an illustration of the transformations, can be found in [7] and Fig. 3, respectively.

### B. Experimental setup

The setup of the experiments is the following:

- the first objective is  $F_{AR}(x, \alpha, \sigma)$  and the second objective is  $F_{CP}(x, \bar{F}_P, p)$ ;
- the initial number of samples populating the archive  $N_{start}$  is set to 100 and the total number of NAS iterations is set to 30 (at each iteration a set of 8 new candidate NNs is added to the archive);
- the OFA supernet is based on MobileNetV3 and the hyperparameters of  $\Omega_x$  refer to resolution (range from 128 to 224 with step size 4), depth (2, 3, or 4), kernel size (3, 5, or 7) and expansion rate (3, 4, or 6);
- we use learning rate 0.1, momentum 0.9, batch size 128 and weight decay 0.0005 for the SGD optimizer;

- we use an adaptive SAM optimizer [11] with the same hyperparameters used for SGD, and parameter  $\rho = 2$  as empirically suggested in [11].

The set of optimal architectures  $X^\circ$  from each NAS search contains the architecture with the best trade-off between  $F_{AR}(x, \alpha, \sigma)$  and  $F_{CP}(x, \bar{F}_P, p)$ . We refer to these networks as  $FlatNAS_{\alpha X \sigma Y}$ , where  $X$  is the value of  $\alpha$ , the weight of terms in  $F_{AR}$ , and  $Y$  is the value of  $\sigma$ , the perturbation parameter used in computing  $R(x, \sigma)$ .

For comparative purposes, we compare FlatNAS with the basic version of CNAS, which uses SGD and the figure of merit  $F_A(x)$ , rather than SAM and the novel metric  $F_{AR}(x)$ . Taking CNAS as a baseline aims to assess the FlatNAS framework effectiveness in identifying robust and high-performing NNs with respect to standard NAS solutions that do not specifically enhance robustness.

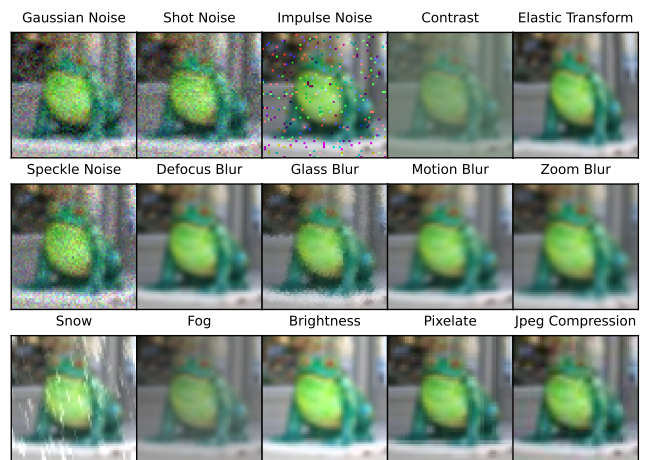


Fig. 2: Image corruptions from the CIFAR-10-C dataset. The figure shows the 15 noise types with level of severity set to 3 with image resolution 64.

### C. Analysis of the results

A summary of experimental results on CIFAR-10 and CIFAR-100 (and their respective corrupted versions CIFAR-10-C and CIFAR-100-C) is reported in Tables I and II. In each table we show a comparison between FlatNAS at different  $\alpha$ -values and CNAS (trained with SGD), with respect to top1 accuracy  $F_A(x)$ , the robustness  $R(x, \sigma)$ , the figure of merit  $F_{AR}$ , the number of parameters (*Params*), the number of multiply-accumulate operations (*MACs*) and the mean Corruption Error (*mCE*). To ensure a fair benchmarking of CNAS against other FlatNAS networks on OOD datasets, we show additional results in which at the end of the CNAS exploration the resulting NN model is optimized with SAM. This step isolates the positive impact of the SAM optimizer, hence allowing us to more accurately evaluate the capability of FlatNAS in identifying the most OOD-robust architecture.

The *mCE* metric that we report in Tables I and II refers to the average value of the Corruption Error mediated over all corruption types and their intensities. As suggested in [7], it is



calculated as  $CE_c = \sum_{s=1}^5 E_{s,c}$ , where  $s$  is the severity level and  $c$  the corruption type. In this way model corruption robustness is summarized by averaging the 15 Corruption Error values, i.e.,  $CE = \{CE_{\text{Gaussian Noise}}, CE_{\text{Shot Noise}}, \dots, CE_{\text{JPEG}}\}$ , resulting in the *mean CE (mCE)*. We do not subtract from the *mCE* the original classification error, which would result in the *relative mCE*, as our goal is the *mCE* over architectures.

To show all the information that may be hidden in the single *mCE* value, in Fig. 3 we show the *CE* mediated over all corruptions as a function of the intensity value, providing a good summary of the main results showcasing enhanced robustness for NN models found by FlatNAS with respect to CNAS. In Figs. 4 and 5 we report a detailed comparison between the performance of the NNs models found by FlatNAS and CNAS for each distortion type present in CIFAR-10-C and CIFAR-100-C, as a function of the intensity value.

TABLE I: Results of FlatNAS and CNAS on CIFAR10 and CIFAR10-C.

Model	$1 - F_A(x)$	$R(x, \sigma)$	mCE	Params	MACs
FlatNAS $\alpha=0.1\sigma=0.05$	9.81	7.85	35.26	5.63	261.90
FlatNAS $\alpha=0.5\sigma=0.05$	9.34	2.11	26.58	4.34	173.63
FlatNAS $\alpha=0.9\sigma=0.05$	8.62	6.18	30.19	4.40	267.94
CNAS (SAM)	8.60	2.18	31.48	4.50	252.20
CNAS	9.61	17.51	34.58	4.50	252.20

TABLE II: Results of FlatNAS and CNAS on CIFAR100 and CIFAR100-C.

Model	$1 - F_A(x)$	$R(x, \sigma)$	mCE	Params	MACs
FlatNAS $\alpha=0.1\sigma=0.05$	31.62	8.42	56.35	4.72	254.66
FlatNAS $\alpha=0.5\sigma=0.05$	27.74	6.46	48.71	5.15	338.45
FlatNAS $\alpha=0.9\sigma=0.05$	27.24	6.8	51.66	6.85	376.22
CNAS (SAM)	29.11	7.72	50.76	5.19	295.61
CNAS	29.59	26.59	53.93	5.19	295.61

In Tables I and II, and in Fig.3, we can see that FlatNAS is able to reduce significantly the *mCE* values without incurring in a significant loss in the accuracy  $F_A(x)$ , while maintaining a comparable number of parameters, which is our principal result. Notice that this holds for every value of the corruption intensity. Meanwhile, NN models found by FlatNAS with  $\alpha = 0.9$  yield comparable results to models obtained with CNAS and then trained with SAM, indicating that in this case FlatNAS is not effective in finding the architecture with the highest robustness. With  $\alpha = 0.1$ ,  $R(x, \sigma)$  is the main term optimized in  $F_{AR}(x, \alpha, \sigma)$ , resulting in NNs with worse  $F_A(x)$  score, resulting in a degradation of the mCE. We see that in Figs. 4 and 5, where we isolate the error on each corruption type, the general trend observed in Fig.3 is respected. In particular, on the most difficult distortion (i.e., yielding high values of corruption errors), FlatNAS achieves excellent improvements in robustness with respect to CNAS. For instance from Fig. 4 we see that on CIFAR-10-C, at the lowest corruption intensity value, on Gaussian Noise FlatNAS

scores 26% of corruption error and CNAS 63%, while on Impulse Noise they achieve 31% and 67% respectively. In Fig. 3 and Tables I and II we also show results varying the  $\alpha$  value in the figure of merit  $F_{AR}(x, \alpha, \sigma)$  defined in Eq. (5). We observe that  $\alpha = 0.5$  gives better mCE (and therefore OOD robustness). This indicates that both terms in  $F_{AR}(x, \alpha, \sigma)$  are equally important for optimizing OOD robustness.

It is worth noting that NN models identified by FlatNAS are in general different with respect the ones found by CNAS. In particular, FlatNAS consistently employs a diverse range of kernel sizes (3, 5, 7) across both datasets, with a noticeable inclusion of larger kernels (7). CNAS also utilizes a range of kernel sizes but does not show as clear a pattern in the preference for larger kernels across datasets.

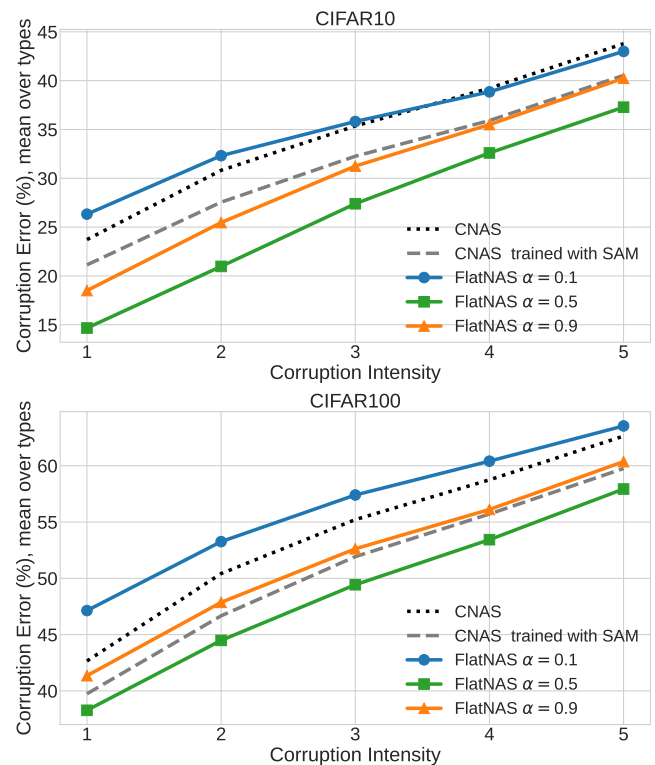


Fig. 3: Comparison of mean NN corruption errors on corrupted datasets in function of the perturbation intensity, for FlatNAS at different  $\alpha$  values and CNAS. (Upper panel: NNs trained on CIFAR-10 and evaluated on CIFAR-10-C; lower panel: same as upper panel but for CIFAR-100.)

## VI. CONCLUSIONS

In this work, we have addressed the timely issue of NAS for OOD robustness. We have focused on parameter-constrained NNs, which are expected to particularly suffer the issue of robustness. Our results indicate that sharpness-aware methods are successful in enhancing OOD robustness in NAS, modifying in a non-trivial manner the NAS exploration and resulting in qualitatively different and more robust architectures. Future works will examine different OOD tasks, OFAs, application scenarios, and the role of architectural perturbations.

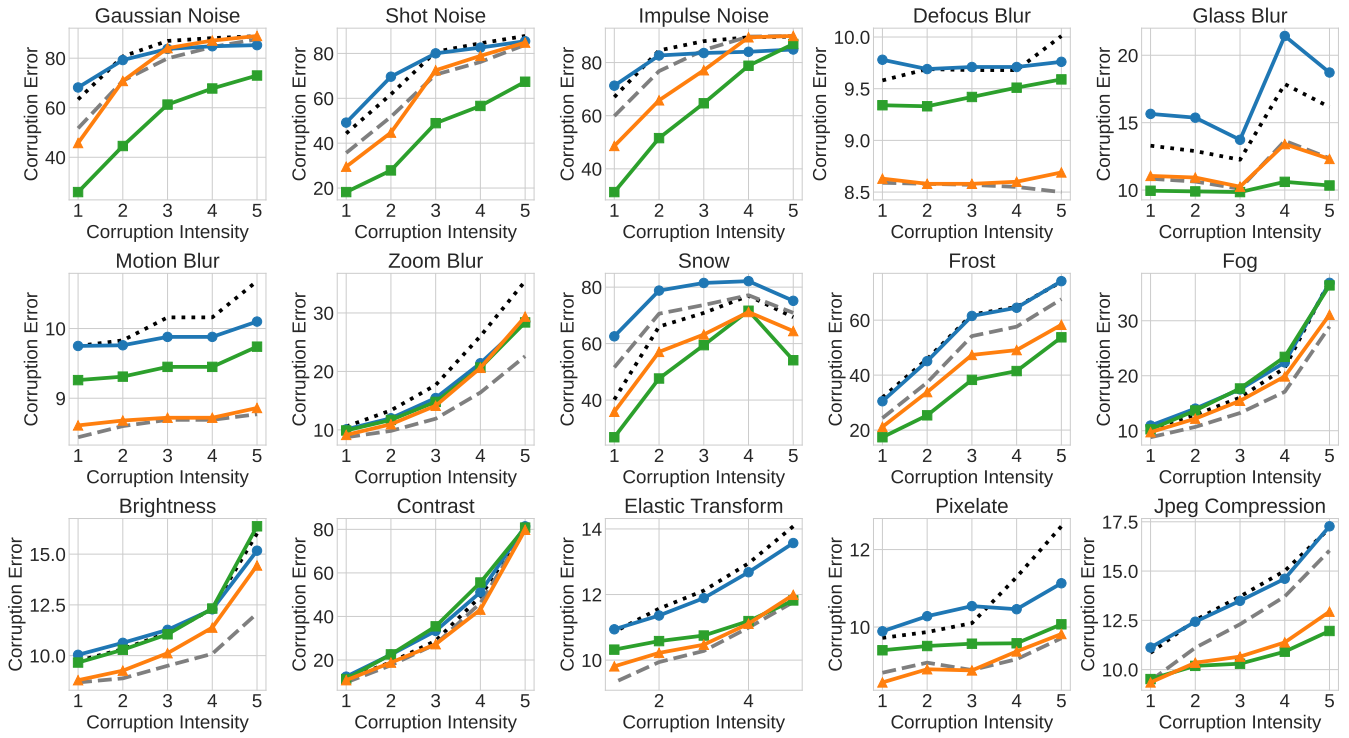


Fig. 4: Final generalization profiles on CIFAR-10-C of NNs trained on CIFAR-10, in function of the perturbation intensity and for each corruption type. Dotted lines correspond to the best NN model found by CNAS and trained with SGD, dashed lines correspond to the latter NN model trained with SAM, and full lines correspond to the best model found by FlatNAS (the dot symbol corresponds to  $\alpha = 0.1$ , the square to  $\alpha = 0.5$  and the triangle to  $\alpha = 0.9$ ).

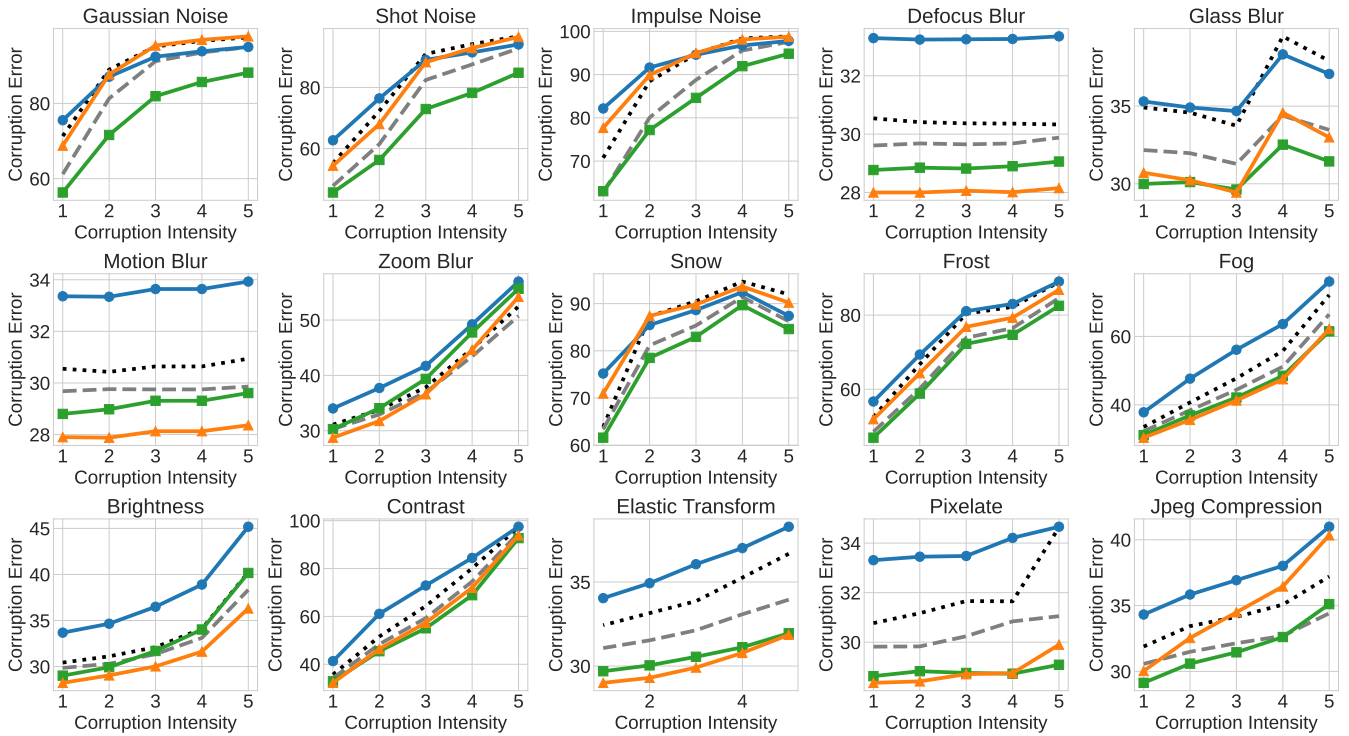


Fig. 5: Same as Fig. 4 but for CIFAR-100-C.

## REFERENCES

- [1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–44, 05 2015.
- [2] X. He, K. Zhao, and X. Chu, "Automl: A survey of the state-of-the-art," *Knowledge-Based Systems*, vol. 212, p. 106622, 2021.
- [3] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, X. Chen, and X. Wang, "A comprehensive survey of neural architecture search: Challenges and solutions," *arXiv preprint arXiv:2006.02903*, 2020.
- [4] H. Benmezziane, K. El Maghraoui, H. Ouarnoughi, S. Niar, M. Wistuba, and N. Wang, "Hardware-Aware Neural Architecture Search: Survey and Taxonomy," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*. Montreal, Canada: International Joint Conferences on Artificial Intelligence Organization, Aug. 2021, pp. 4322–4329.
- [5] M. Gambella, A. Falchetta, and M. Roveri, "CNAS: Constrained Neural Architecture Search," in *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. Prague, Czech Republic: IEEE, Oct. 2022, pp. 2918–2923. [Online]. Available: <https://ieeexplore.ieee.org/document/9945080/>
- [6] M. Roveri, "Is tiny deep learning the new deep learning?" in *Computational Intelligence and Data Analytics*, R. Buyya, S. M. Hernandez, R. M. R. Kovvur, and T. H. Sarma, Eds. Singapore: Springer Nature Singapore, 2023, pp. 23–39.
- [7] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," in *International Conference on Learning Representations*, 2019.
- [8] D. Hendrycks, A. Zou, M. Mazeika, L. Tang, D. Song, and J. Steinhardt, "Pxm: Dreamlike pictures comprehensively improve safety measures," in *NeurIPS 2021 Workshop on Distribution Shifts: Connecting Methods and Applications*, 2021. [Online]. Available: [https://openreview.net/forum?id=WeUg\\_KpkFtt](https://openreview.net/forum?id=WeUg_KpkFtt)
- [9] K. Kirchheim, M. Filax, and F. Ortmeier, "PyTorch-OOD: A Library for Out-of-Distribution Detection based on PyTorch," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. New Orleans, LA, USA: IEEE, Jun. 2022, pp. 4350–4359.
- [10] P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur, "Sharpness-aware minimization for efficiently improving generalization," in *International Conference on Learning Representations*, 2021.
- [11] J. Kwon, J. Kim, H. Park, and I. K. Choi, "Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 5905–5914.
- [12] Y. Yue, J. Jiang, Z. Ye, N. Gao, Y. Liu, and K. Zhang, "Sharpness-aware minimization revisited: Weighted sharpness as a regularization term," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ser. KDD '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 3185–3194. [Online]. Available: <https://doi.org/10.1145/3580305.3599501>
- [13] F. Pittorino, C. Lucibello, C. Feinauer, G. Perugini, C. Baldassi, E. Demyanenko, and R. Zecchina, "Entropic gradient descent algorithms and wide flat minima," in *International Conference on Learning Representations*, 2021. [Online]. Available: <https://openreview.net/forum?id=xjXg0bnoDms>
- [14] S. Hochreiter and J. Schmidhuber, "Flat minima," *Neural Computation*, vol. 9, no. 1, pp. 1–42, 1997.
- [15] B. L. Annesi, C. Lauditi, C. Lucibello, E. M. Malatesta, G. Perugini, F. Pittorino, and L. Saglietti, "Star-shaped space of solutions of the spherical negative perceptron," *Phys. Rev. Lett.*, vol. 131, p. 227301, Nov 2023. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevLett.131.227301>
- [16] C. Baldassi, F. Pittorino, and R. Zecchina, "Shaping the learning landscape in neural networks around wide flat minima," *Proceedings of the National Academy of Sciences*, vol. 117, no. 1, pp. 161–170, 2020.
- [17] Y. Jiang, B. Neyshabur, H. Mobahi, D. Krishnan, and S. Bengio, "Fantastic generalization measures and where to find them," 2019.
- [18] C. Lucibello, F. Pittorino, G. Perugini, and R. Zecchina, "Deep learning via message passing algorithms based on belief propagation," *Machine Learning: Science and Technology*, vol. 3, no. 3, p. 035005, Jul 2022. [Online]. Available: <https://dx.doi.org/10.1088/2632-2153/ac7d3b>
- [19] F. Pittorino, A. Ferraro, G. Perugini, C. Feinauer, C. Baldassi, and R. Zecchina, "Deep networks on toroids: Removing symmetries reveals the structure of flat regions in the landscape geometry," in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., vol. 162. PMLR, 17–23 Jul 2022, pp. 17759–17781. [Online]. Available: <https://proceedings.mlr.press/v162/pittorino22a.html>
- [20] D. Li, Z. Teng, Q. Li, and Z. Wang, "Sharpness-aware minimization for out-of-distribution generalization," in *Neural Information Processing*, B. Luo, L. Cheng, Z.-G. Wu, H. Li, and C. Li, Eds. Singapore: Springer Nature Singapore, 2024, pp. 555–567.
- [21] J. Cha, S. Chun, K. Lee, H.-C. Cho, S. Park, Y. Lee, and S. Park, "Swad: Domain generalization by seeking flat minima," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 22405–22418.
- [22] T. Kim, S. Lim, and K. Song, "Sufficient invariant learning for distribution shift," 2023.
- [23] R. Pang, Z. Xi, S. Ji, X. Luo, and T. Wang, "On the Security Risks of AutoML," Oct. 2021, arXiv:2110.06018 [cs].
- [24] H. Bai, F. Zhou, L. Hong, N. Ye, S.-H. G. Chan, and Z. Li, "NAS-OoD: Neural Architecture Search for Out-of-Distribution Generalization," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. Montreal, QC, Canada: IEEE, Oct. 2021, pp. 8300–8309.
- [25] C. Devaguptapu, D. Agarwal, G. Mittal, P. Gopalani, and V. N. Balasubramanian, "On Adversarial Robustness: A Neural Architecture Search perspective," Aug. 2021, arXiv:2007.08428 [cs, stat].
- [26] Y. Li, Z. Yang, Y. Wang, and C. Xu, "Neural Architecture Dilation for Adversarial Robustness," Aug. 2021, arXiv:2108.06885 [cs].
- [27] M. Dong, Y. Li, Y. Wang, and C. Xu, "Adversarially Robust Neural Architectures," Feb. 2023, arXiv:2009.00902 [cs].
- [28] X. Zhu, J. Li, Y. Liu, and W. Wang, "Robust Neural Architecture Search," Apr. 2023, arXiv:2304.02845 [cs].
- [29] M. Guo, Y. Yang, R. Xu, Z. Liu, and D. Lin, "When NAS Meets Robustness: In Search of Robust Architectures against Adversarial Attacks," Mar. 2020, arXiv:1911.10695 [cs, stat].
- [30] J. Mok, B. Na, H. Choe, and S. Yoon, "AdvRush: Searching for Adversarially Robust Neural Architectures," Aug. 2021, arXiv:2108.01289 [cs].
- [31] X. Wang, S. Cao, M. Li, and K. M. Kitani, "Neighborhood-Aware Neural Architecture Search," Oct. 2021, arXiv:2105.06369 [cs].
- [32] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable Architecture Search," Apr. 2019, arXiv:1806.09055 [cs, stat].
- [33] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-All: Train One Network and Specialize it for Efficient Deployment," Apr. 2020, arXiv:1908.09791 [cs, stat].
- [34] M. Wistuba, A. Rawat, and T. Pedapati, "A Survey on Neural Architecture Search," Jun. 2019, arXiv:1905.01392 [cs, stat].
- [35] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for MobileNetV3," Nov. 2019, arXiv:1905.02244 [cs]. [Online]. Available: <http://arxiv.org/abs/1905.02244>
- [36] Z. Lu, K. Deb, E. Goodman, W. Banzhaf, and V. N. Boddeti, "NSGANetV2: Evolutionary Multi-Objective Surrogate-Assisted Neural Architecture Search," Jul. 2020, arXiv:2007.10396 [cs]. [Online]. Available: <http://arxiv.org/abs/2007.10396>
- [37] M. S. Mahbub, "A comparative study on constraint handling techniques of nsgaii," in *2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*, 2020, pp. 1–5.
- [38] J. Zhuang, B. Gong, L. Yuan, Y. Cui, H. Adam, N. C. Dvornek, sekhar tatikonda, J. s Duncan, and T. Liu, "Surrogate gap minimization improves sharpness-aware training," in *International Conference on Learning Representations*, 2022.