RoCo-NAS: Robust and Compact Neural Architecture Search

Vahid Geraeinejad*, Sima Sinaei†‡, Mehdi Modarressi*¶, Masoud Daneshtalab†§
* School of Electrical and Computer Engineering, University of Tehran, Iran,†Mälardalen University, Sweden,
‡RISE Research Institute of Sweden, ¶ School of Computer Science, IPM, Iran, §Tallinn University of Technology, Estonia
{geraeinejad, modarressi}@ut.ac.ir, {sima.sinaei, masoud.daneshtalab}@mdh.se

***Abstract*—Deep model compression has been studied widely, and state-of-the-art methods can now achieve high compression ratios with minimum accuracy loss. Recent advances in adversarial attacks reveal the inherent vulnerability of deep neural networks to slightly perturbed images called adversarial examples. Since then, extensive efforts have been performed to enhance deep networks' robustness via specialized loss functions and learning algorithms. Previous works suggest that network size and robustness against adversarial examples contradict on most occasions. In this paper, we investigate how to optimize compactness and robustness to adversarial attacks of neural network architectures while maintaining the accuracy using multi-objective neural architecture search. We propose the use of previously generated adversarial examples as an objective to evaluate the robustness of our models in addition to the number of floating-point operations to assess model complexity i.e. compactness. Experiments on some recent neural architecture search algorithms show that due to their limited search space they fail to find robust and compact architectures. By creating a novel neural architecture search (RoCo-NAS), we were able to evolve an architecture that is up to 7% more accurate against adversarial samples than its more complex architecture counterpart. Thus, the results show inherently robust architectures regardless of their size. This opens up a new range of possibilities for the exploration and design of deep neural networks using automatic architecture search.**

I. Introduction

Deep neural networks (DNNs) are vulnerable to adversarial attacks, where the natural data is perturbed with human-inconspicuous, carefully crafted noises [1]. To mitigate this pitfall, extensive efforts have been devoted to adversarial defense mechanisms, where the main focus has been on specialized adversarial learning algorithms [2], [3], loss/regularization functions [4], [5], as well as image preprocessing [6]–[8]. Yet, few studies have explored the intrinsic influence of network architecture on network resilience to adversarial perturbations. Although the importance of architectures in adversarial robustness has emerged in the experiments of several previous work [9], [10], more comprehensive study on the role of network architectures in robustness remains needed. In this work, we focus on gaining systematical understanding of adversarial robustness in neural networks from an architectural perspective.

Network architecture search (NAS) and adversarial samples have rarely appeared together. Regarding adversarial samples, they were discovered in 2013 when DNNs were shown to behave strangely for nearly the same images [11]. Afterwards, a series of vulnerabilities were found [12]–[14]. Such attacks can also be easily applied to real world scenarios [1], [15] which confers a big problem for current deep neural networks' applications. Currently, there is not any known learning algorithm or procedure that can consistently defend against adversarial attacks.

Regarding NAS, the automatic design of architectures has been of wide interest for many years. The aim is to develop methods that do not need specialists in order to be applied to a different application. This would confer not only generality but also ease of use. Many researchers have developed unique approaches [16]–[18] to improve the accuracy besides decreasing the complexity and computational cost. Current state-of-the-art on image classification and object detection are developed using NAS, which shows that NAS plays an important role in solving standard learning tasks. Previous works [10], [19], [20] discussed that network complexity and robustness against adversarial examples contradict on most occasions; therefore, compressed models commonly exhibit poor robustness.

In this paper, we propose the use of NAS to tackle this issue. In other words, architecture search will be employed to find accurate and compact neural networks with higher robustness than their size equivalent architectures. This is based on the principle that robustness of neural networks can be evaluated by using adversarial attacks as a function evaluation. We hypothesize that if there is a solution in a given architecture search space, by considering the robustness as one of the design objectives, the search algorithm would be able to find the more robust solution. This is not only a blind search for a cure. The best architectures found should also hint which structures and procedures provide robustness for neural networks. Therefore, it would be possible to use the results of the search to further understand how to improve the search methodology as well as design yet more robust ones. The paper is structured as follows: Section II presents a background on the neural architecture search (NAS) and existing researches on adversarial attacks and defense. Our proposed method is discussed in Section III. Section IV presents the experimental results. Finally, the paper is concluded in Section V and section VI is dedicated to acknowledgements.

II. Related Works

In this section, we briefly introduce the literature of network architecture search, adversarial attacks, and available defense

Preprint submitted to International Joint Conference on Neural Networks (IJCNN) 2021

techniques.

A. Neural Architecture Search

The aim of the NAS is to automatically design network architectures to replace conventional handcrafted ones. Representative techniques are divided into reinforcement learning (RL) [18], [21]–[24] and evolutionary algorithm (EA) based approaches [25]–[32], with a few methods falling outside these two categories. On one hand, in reinforcement learning approaches, architectures are created from a sequence of actions which are afterwards rewarded proportionally to the crafted architecture's accuracy. On the other hand, in evolutionary computation based methods, small changes in the architecture (mutations) and recombinations (crossover) are used to create new architectures. All architectures are evaluated based on their accuracy with some of the best architectures chosen to continue to the next generation. Our search method, RoCO-NAS, is based on an evolutionary method. We augment the original encoding of NSGA-Net [33] and also introduce multi-objective based selection scheme, including compactness and robustness.

B. Generating Adversarial Example

An adversarial example is a slightly perturbed image, recognizable by human observer, generated by an adversary with the goal of producing a wrong output from the correct target class. An adversarial example x' is generated as $x' = x + \epsilon$, where x is the original sample and ϵ is the added perturbation. The aim of an adversary is to find x' to deceive network f such that

$$f(x) \neq f(x') \wedge \Delta(x', x) \leqslant \epsilon \tag{1}$$

where $\Delta(x', x)$ indicates the difference between the original image and the adversarial example. Generally, the strength of the adversaries are limited by ϵ i.e. the amount of change they are allowed to apply to the original sample in order to generate adversarial examples. Δ is commonly defined by a L_p-norm distance metric. Three of the most popular choices for L_p are:

L_0 : For image samples, this metric limits the number of pixels that the adversary is allowed to perturb in order to generate an adversarial example. The amount of perturbation on each pixel is unlimited.

L_2 : This metric is the Euclidean distance between the initial sample x and the adversarial example x'. In this method the adversary is allowed to tweak any pixel of the image as long as the L_2 distance is smaller than a certain value.

L_∞ : This metric limits the amount of perturbation that an adversary is allowed to apply to each pixel. The adversary may alter any number of pixels as long as the amount of perturbation for each pixel is smaller than a predefined value.

We have used several different attack algorithms to evaluate the models in our experiments. These attacks generate their examples based on different metrics, one pixel attack [14] uses L_0 norm, DeepFool [12] uses L_2 norm and L_∞ norm attacks such as FGSM [2], BIM [1] and PGD [3]. Carlini and Wagner attack (C&W) [34] supports all of the aforementioned metrics.

C. Defensive Techniques

Recently, many defensive techniques have been developed in order to improve the robustness of neural networks against adversarial attacks. Silva et al. [35] provided a comprehensive study of the existing defense methods that separates them into three groups:

1) Gradient Masking: The aim of the gradient masking techniques is to make gradient based DNNs robust against adversarial examples by hiding useful gradient information from adversaries; therefore, they will not be able to easily alter an input data to create an effective adversarial example [36], [37]. These techniques produce a loss function that is smooth around the training data points. Gradient masking defense techniques are weak against black-box or attacks that are based on approximate gradients [38], [39].

2) Adversarial Example Detection: Adversarial example detection techniques are designed to recognize perturbed input images from natural ones [40]–[43]. This task is usually done by training a sub-network or designing a separate detector. After detecting an adversarial example, these methods may chose whether to ignore the example altogether or transform it into a recognizable input for the network.

3) Robust Optimization: Robust optimization is referred to the techniques that modify the optimization function by adding knowledge about adversarial examples in the objective function or adding uncertainty to model layers [35].

Adversarial training is the most popular type of robust optimization. This technique extends the training dataset by attaching predefined adversarial examples and their ground truth label to it [2], [10], [11]. The goal is that the trained model will be able to find the ground truth label, once it is given a new adversarial example. Despite their popularity, adversarial training methods mainly suffer from a common problem. Considering the wide range of existing adversarial attacks, it is not feasible to include every attack type in the training; hence, adversarially trained models tend to be biased towards the type of attack they are trained with. Plus, these methods mainly train the network weight regardless of any attention to the architecture. To resolve the former issue, Madry et al. proposed to use PGD attack in adversarial training. They have argued that PGD-based adversarial training is able to increase the robustness of the network against any other attack. Their results show a superior increase in robustness when it is implemented on MNIST based models compared to CIFAR-10 based models.

Recently, few cutting edges methods have taken advantage of NAS to find robust neural network architectures against adversarial examples [44]–[46]. Kotyan et al. [44] proposed an evolutionary based NAS method called robust architecture search (RAS) to find robust models against adversarial examples. Their results show that RAS is more capable than other NAS methods in finding robust models. Guo et al. [45] Incorporated PGD adversarial training with one-shot NAS [47] to generate a family of models called RobNet which show relatively superior robustness on various datasets, such as CIFAR, SVHN, and Tiny-ImageNet while using less

parameters than their DenseNet and ResNet counterparts. As opposed to [44], our robust optimization defense method takes model complexity and natural accuracy into consideration by utilizing multi-objective NAS and unlike [45] does not employ any adversarial training in the process. In this work, we answer the following question: "Can the complex topology of a neural network provide adversarial robustness without any form of adversarial training?". In other word, we try to understand adversarial robustness purely from an architectural perspective.

III. Robust and Compact Neural Architecture Search

Execution of DNNs on embedded devices are often constrained by limitations in hardware resources in terms of power consumption, latency constraints, and available memory. Hence, the design of DNNs is required to balance multiple, possibly competing, objectives. On a separate note, the deployment of CNNs also calls for attention to their robustness. Despite their impressive predictive powers, the state-of-the-art CNNs remain to commonly suffer from fragility to adversarial attacks, and they are required to be designed robustly.

Often, when several design objectives are needed to be considered simultaneously, there would not exist a single solution that is able to lead optimally in all desired criteria, especially when objectives contradict each other. Under such conditions, a set of solutions that provides the entire trade-off information between the objectives is more desirable. This enables a designer to analyze the importance of each objective, depending on the application, and to choose a suitable solution on the trade-off frontier. A solution is considered to be Pareto Optimal point, if and only if any other solution to the problem does not dominate it. We propose RoCo-NAS, a genetic-based architecture search algorithm to automatically generate a set of DNN architectures that approximate the Pareto-front between performance, complexity, and robustness for vision tasks. The rest of this section describes the multi-objective optimization problem, our search space and encoding scheme, and the main components of RoCo-NAS in detail.

A. Exploration Method

To obtain an efficient trade-off solutions for multi-objective optimization, two main approaches exist: i) classical point-based methods like a weighted sum of the objective functions (e.g., $af_1(x) + (1-a)f_2(x)$); and ii) population-based methods like genetic algorithms. Population-based strategies offer a flexible approach to find multiple efficient trade-off solutions in one execution. The efficiency obtained by such parallelism cannot be compared by point-based methods such as weighted combinations. Recent studies show that population-based methods can successfully solve problems with billions of variables [48]. In contrast, classical point-based methods, like branch-and-bound, cannot handle even hundreds of variables.

Multi-Objective Selection: In mathematical terms, a multi-objective optimization problem is formulated as:

$$\min_x \{f_1(x), ..., f_M(x)\} \tag{2}$$

where each f_i is an objective that we are going to optimize, and x is the neural network architecture solution. For the aforementioned problem, given solutions x_1 and x_2, x_1 is said to dominate x_2 (i.e., $x_1 \leq x_2$) if:

1) x_1 is no worse than x_2 for all objectives ($f_i(x_1) \leq f_i(x_2) \forall i \in \{0, \ldots, M\}$
2) x_1 is better than x_2 in at least one objective ($\exists i \in \{0, ..., M\} | f_i(x_1) < f_i(x_2)$)

Therefore, a solution x_i is non-dominated if these conditions hold for all x_j and $j \neq i$.

The core of RoCo-NAS is a selection criterion that leverages non-dominated solutions. Specifically, given a population of network architectures $\{x_1, \ldots, x_n\}$ and their fitness functions $\{f_1(x_i), \ldots, f_M(x_i)\}$, the ranking and selection procedure consists of two stages: (1) non-dominated solutions are selected over dominated solutions; (2) explicitly ranking of solutions that are diverse w.r.t. the trade-off between the objectives higher than solutions that are "crowded" on the trade-off front, i.e., how close a given solution is to its neighbors in the objective space. We used the NSGA algorithm presented by Deb et al. [49] to adapt non-domination ranking and crowdedness measurements. The non-domination ranking indicates the front number that a solution belongs to; these fronts are composed of the set of non-dominated solutions at the current search iteration.

Elitist Selection: Using this approach, the best solution (in terms of objective values, which are accuracy, compactness, and robustness in our problem) in the parent population is always passed to the next population. This policy allows the previous best solution to have a chance for sharing its genetic information with the next generation, while there is no risk for losing the information to the newly generated child population.

B. Search Space and Encoding

Most of the hand-crafted CNN models are designed by combining the same optimized computational block (e.g. DenseNet [50], ResNet [51], etc. computational blocks) that consists of layer-wise computations. However, RoCo-NAS is adapted from NSGA-Net that constructs architectures by a combination of several optimized computational blocks, also referred to as a phase. Though RoCo-NAS operates to optimize individual computational blocks, the performance of each architecture solution is assessed based on the entire model i.e. several distinct interconnected computational blocks.

RoCo-NAS encodes the operations of each network as $x_o = (x_o^{(1)}, x_o^{(2)}, ..., x_o^{(n_p)})$ where n_p is the total number of computational blocks. Each $x_o^{(i)}$ encodes a directed acyclic graph with maximum of n_o nodes (computational units) that describes the operation within a phase encoded by a binary string. In this approach, nodes are basic computational units that carry out the same sequence of operations i.e. a 3×3 convolution followed by batch-normalization and ReLU activation. This encoding scheme offers a compact representation

of the network architectures in search space, yet is flexible enough that many of the computational blocks in hand-crafted networks can be encoded. The total number of architectures in our search space is:

$$\gamma_x = n_p \times 2^{n_o(n_o-1)/2+1} \tag{3}$$

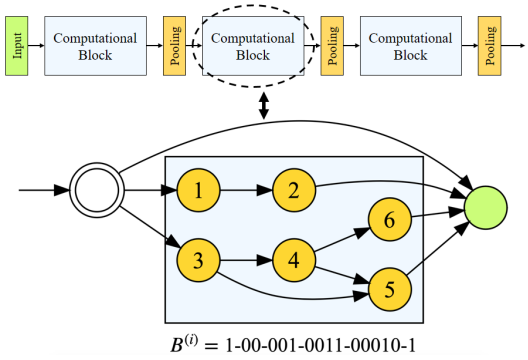

Fig. 1. The search space from RoCo-NAS. Including computation blocks and a detailed representation of a sample computational block, its corresponding nodes and encoding.

Fig. 1 illustrates RoCo-NAS search space including three computational blocks, each consist of a distinct combination of computational units. The figure also shows a possible variation of the nodes inside a computation block and their corresponding encoding by the approach presented in [33].

C. Search Procedure

RoCo-NAS is an iterative process in which initial solutions are made gradually better as a group, called population. In every iteration, the same number of new network architectures (offspring) are generated from parents, which are selected from the last iteration in the population. Both parents and offspring compete for remaining and reproduction (becoming a parent) in the next iteration. There exists flexibility in the generating of the initial population. It may be generated randomly or forced to select the hand-crafted network architectures (according to prior knowledge). The procedure for RoCo-NAS search algorithm has been shown in Algorithm 1.

The goal of exploration is to discover diverse ways of connecting nodes to form a phase (computational block). Genetic operations, crossover, and mutation offer an effective means to realize this goal.

Crossover selects two members of the population (parents) and uses them to create a new population member (offspring) by inheriting computational blocks from parents. The idea of the crossover is to preserve frequent parents' shared computational blocks.

Mutation operation randomly changes the number of nodes or connections in a computational block. Mutation ensures generating a diverse set of network architectures and avoids the local optimum problem. To escape creating a completely random architecture, only one computational block is allowed to be mutated at a time.

Algorithm 1: RoCo-NAS procedure using Elitist Evolutionary Algorithm (NSGA)

Requires: N : size of population
T : Maximum number of generations.
Ensure: Non-dominated individuals in P_{t+1}.
Step 1. *Initialization:* Generate a random initial population of Neural Network Architecture P_0 and create an empty child set $Q_0.T \leftarrow 0$
Step 2. *Fitness assignment:* $P_{t+1} \leftarrow P_t \cup Q_t$, and then calculate the fitness values for Robustness, Compactness and accuracy of the individuals (Network Architectures) in P_{t+1}.
Step 3. *Truncation:* Reduce size of P_{t+1} by keeping best N network architecture according to their fitness values.
Step 4. *Termination:* If $t = T$, output non-dominated individuals (network architectures) in P_{t+1} and terminate.
Step 5. *Selection:* Select neural network architectures from P_{t+1} for mating.
Step 6. *Variation:* Apply crossover and mutation operations to generate Q_{t+1}. $t \leftarrow t + 1$ and go to **Step 2**.

D. Fitness function Evaluation

In addition to accuracy, we have considered robustness and compactness as the second and third objectives of RoCo-NAS.

1) Compactness: In order to generate more compact models, computational complexity, defined as the number of floating-point operations (FLOPs) conducted during a forward pass in an architecture, is considered as an objective.

2) Robustness: There exists a wide variety of adversarial attacks for neural networks; therefore, choosing an attack to evaluate robustness is an ambiguous task. Kotyan et al. [52] proposed a method that uses attacks with different levels of both L_0 and L_∞ metrics to evaluate the robustness of neural networks. This method suggests that adversarial examples created with this duality (L_0 and L_∞), as opposed to other combination of metrics, can correctly assess the robustness of neural networks. To evaluate the robustness of our models during the search process, we have used this method to generate a set of adversarial examples. This set is created using various thresholds (1,3,5 and 10) for L_0 and L_∞ metrics and target networks such as DenseNet, LeNet, Resnet and VGG-16 to be inclusive among a wide range of neural network architectures. Table I summarizes our robustness evaluation adversarial set based on their attack type, target network, threshold and the number of successful samples. Generating the robustness evaluation set before NAS accelerates our procedure and also ensures that the robustness of our models is not biased toward certain attacks.

The objectives of the RoCo-NAS for each network architecture solution a are defined as accuracy ($err_{natural}(a)$), compactness ($FLOPs(a)$) and robustness ($err_{adv}(a)$). The

TABLE I
DETAILS OF GENERATED EVALUATION ADVERSARIAL SAMPLES.

Model	L_0				L_∞				Total
	1	3	5	10	1	3	5	10	
DenseNet	133	178	201	285	76	85	93	143	**1194**
LeNet	444	529	571	651	274	279	289	329	**3366**
ResNet	220	304	367	478	91	102	129	190	**1881**
VGG-16	440	506	541	627	289	296	307	344	**3350**
Total	**1237**	**1517**	**1680**	**2041**	**730**	**762**	**818**	**1006**	**9791**

Number of adversarial samples generated using different attack types, thresholds and target networks.

aim of the fitness function is to evaluate the models based on minimizing all the objectives.

IV. EXPERIMENTAL RESULTS

In this section, we demonstrate the results of using multi-objective neural architecture search on robustness, compactness and natural accuracy by evaluating the generated models from our proposed method. First, we will describe our implementation settings and our baseline configuration. Then, we will compare the models from our proposed method with the baseline models in terms of their model complexity, robustness and natural accuracy.

A. Implementation Details

Search Parameters: Table II shows NSGA-Net's fixed parameters in all of our experiments. Our search method explores different architectures by changing the number of computational units in each computational block and their connections. Our architectures have 3 computational blocks and each block contains the maximum of 6 computational units or nodes. Our search algorithm is executed for 20 generations and models are trained for 25 for epochs before getting evaluated during the search process.

TABLE II
NSGA-NET NAS PARAMETERS

Parameter	Value
Computational blocks	3
Maximum computational units	6
Genetic algorithm generations	20
Genetic algorithm population	40
Training epochs	25

Post-search Optimization: After the search procedure, architectures are sorted using pareto-optimal optimization to find optimal solutions regarding the objectives. Then, Each model is trained with the batch size as 128 and initial channels as 32 for 120 epochs to improve classification accuracy. The following evaluations are done using these trained models.

Hardware Description: We have used a PC equipped with 12-core Intel® Xeon® Silver 4109T CPU at 4.3GHz, 16GB of system memory, and a GeForce GTX 1080 Ti GPU to conduct our experiments.

B. Baseline-NAS Analysis

As opposed to RoCo-NAS with a separate robustness objective, our baseline-NAS is implemented using only accuracy and compactness objectives. Our models are capable of generating architectures comparable to state-of-the-art NAS methods such as Proxyless [53], ENAS [18], DARTS [26] and SMASH [54]. For this comparison, we have selected an optimal model from the set of our baseline-NAS solutions considering both FLOPs and CIFAR-10 accuracy objectives. As illustrated in Fig. 2, our baseline-NAS architecture exhibits comparable CIFAR-10 classification accuracy while using fewer number of parameters. The cost of our search algorithm is approximately 9.5 GPU days.

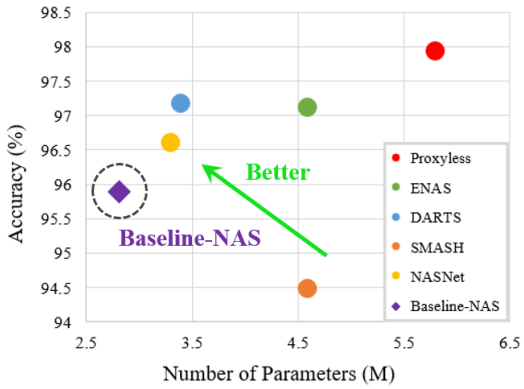

Fig. 2. Accuracy *vs* number of parameters of our baseline-NAS compared to other state-of-the-art NAS methods on CIFAR-10.

C. RoCo-NAS: Compression, Robustness and Accuracy

We have designed our proposed search algorithm using multi-objective optimization to reduce model complexity and improve robustness and natural accuracy of the models. Our goal here is to evaluate the models from our proposed method and compare them with our baseline NAS models. In order to compare different sets of models, generated by different search configurations, we have selected the top N computationally light models, calculated by the number of FLOPs, from the set of pareto-optimal solutions, where $N \in \{1, 3, 5\}$.

Model Complexity: We have used two different metrics to compare model complexity including the number of FLOPs and the number of parameters.

We have measured computational complexity by measuring the number of FLOPs in our search algorithm. Generally, as it is illustrated in Fig. 3, the top models from RoCo-NAS are more computationally intensive than our baseline models.

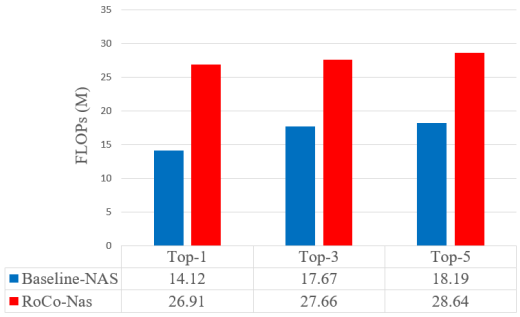

Fig. 3. Comparing computational complexity of the top models from baseline-NAS and RoCo-NAS using number of FLOPs.

We have also measured the number of parameters to compare the size of our models. Fig. 4 shows a comparison between the size of top models from baseline-NAS and RoCo-NAS. Similar to computational complexity, the models form our baseline-NAS are more compact than RoCo-NAS. The top models from RoCo-NAS are up to $8\times$ larger than our baseline-NAS.

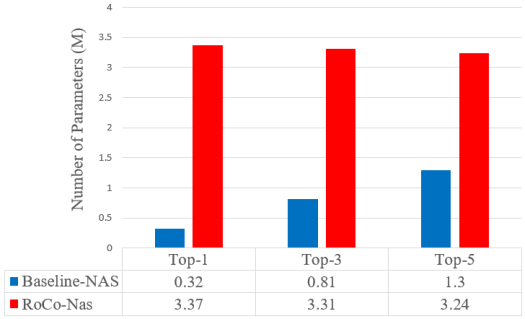

Fig. 4. Comparing size of the top models from baseline-NAS and RoCo-NAS using number of parameters.

Robustness We have compared the robustness of our models using our generated adversarial dataset and a number of well-known attacks such as PGD, FGSM, BIM, Carlini & Wagner and DeepFool.

As shown in Table III, RoCo-NAS is able to find more robust models against our L_0 and L_∞ based generated adversarial examples even if it is at the cost of model complexity. The question that is later discussed in Section IV-D is whether the model complexity is the only contributing factor to the higher robustness of RoCo-NAS models.

TABLE III
ACCURACY OF THE TOP MODELS FROM BASELINE-NAS AND ROCO-NAS AGAINST OUR L_0 AND L_∞ BASED GENERATED ADVERSARIAL EXAMPLES.

Configuration	Top-1	Top-3	Top-5
Baseline NAS	75.77%	78.65%	79.55%
RoCo-NAS	82.15%	81.89%	81.91%

We have previously demonstrated how RoCo-NAS generates more complex models compared to our baseline-NAS. Seeing that the models from RoCo-NAS are also more robust is an indication of a clear trade-off between model complexity and robustness.

As it is illustrated in the Fig. 5, we have used a weak variation of PGD attack with $\epsilon = 1$ to compare the robustness of our models displayed with different colors depending on their configuration based on the number of their FLOPs. The red curved line that is an approximation of the RoCo-NAS models is located on the top-right corner of the figure which shows their high robustness considering their model complexity (higher number of FLOPs).

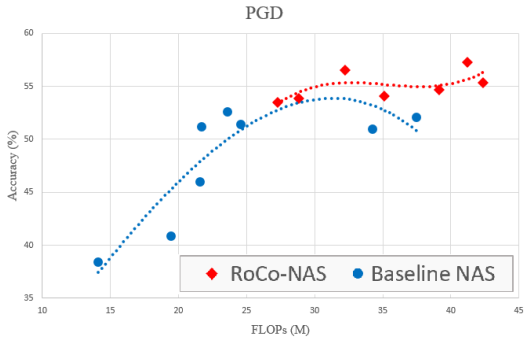

Fig. 5. Comparing the robustness of baseline-NAS and RoCo-NAS against PGD attack with $\epsilon = 1$.

We have repeated the same experiment with four other attacks including different variations of FGSM with $\epsilon = 0.1, 0.2, 0.3$, BIM, DeepFool and C&W. As it is evident in the Fig. 6, they all follow a similar pattern and point to the same conclusions we have previously discussed. The models from RoCo-NAS are generally more robust and their robustness is proportional to the number of FLOPs i.e. model complexity.

Natural Accuracy: The natural accuracy of our models is evaluated using validation images from CIFAR-10 dataset. Table IV shows that the top models from RoCo-NAS outperform the models from our baseline-NAS by a small yet evident margin. The fact that the models from RoCo-NAS are generally larger suggests that network size is directly related to natural accuracy.

TABLE IV
ACCURACY OF THE TOP MODELS FROM BASELINE-NAS AND ROCO-NAS SEARCH CONFIGURATIONS MEASURED BY CIFAR-10 DATASET.

Configuration	Top-1	Top-3	Top-5
Baseline-NAS	92.80%	94.50%	94.97%
RoCo-NAS	95.80%	95.90%	95.78%

D. Robustness analysis of different configurations

We have compared two different models of equivalent complexity (relatively equal number of FLOPs) from RoCo-NAS and our baseline-NAS configurations regarding their robustness and the results are presented in the Table V. Even

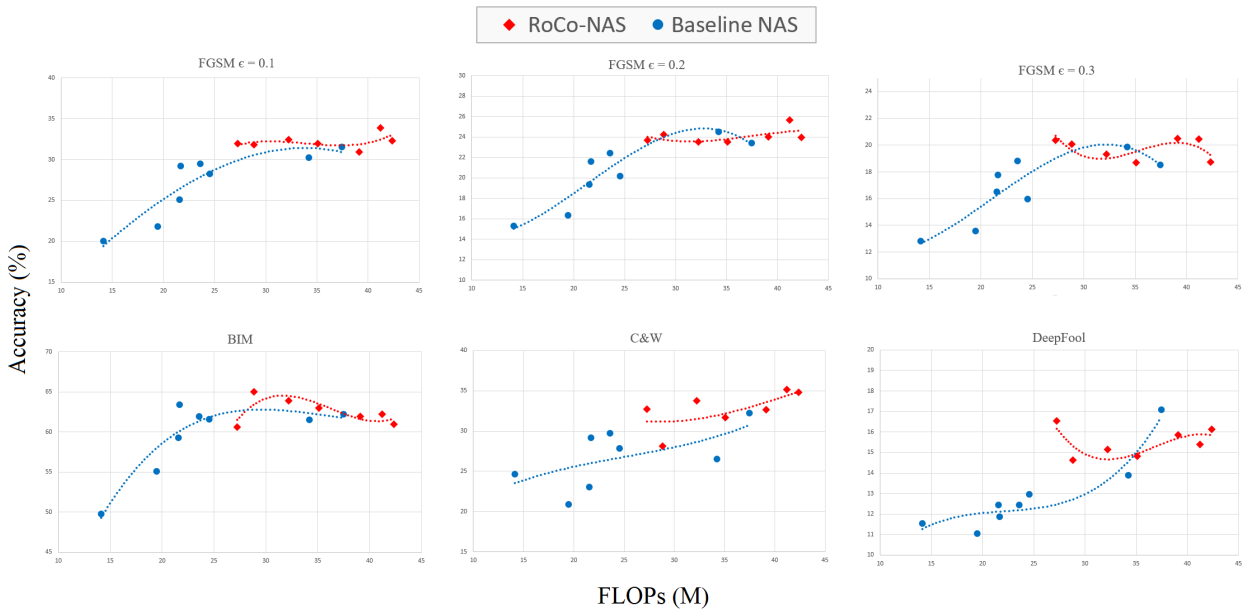

Fig. 6. Comparing the robustness of baseline-NAS and RoCo-NAS models using FGSM with different values for ϵ, BIM, C&W and DeepFool attacks based on the number of FLOPs.

though the selected model from our baseline-NAS is slightly larger (measured by the number of parameters) and more computational intensive (measured by the number of FLOPs), the model from RoCo-NAS is up to 7% more robust against C&W and PGD attacks.

TABLE V
ROBUSTNESS COMPARISON OF MODELS WITH EQUIVALENT COMPLEXITY FROM DIFFERENT SEARCH CONFIGURATIONS

Configuration	Parameters	FLOPs	PGD	C&W
Baseline-NAS	3.27M	34.22M	50.96%	26.55%
RoCo-NAS	3.13M	32.22M	56.56%	33.78%

The superior robustness of the less complex architecture found by our proposed method, RoCo-NAS, suggests that this configuration is able to find models that are inherently robust to adversarial examples, regardless of the model complexity.

V. CONCLUSION

We proposed a robust architecture search framework as a solution for developing and researching robust and compact models. This method is based on using adversarial attacks together with accuracy and compactness as evaluation functions in NAS. Here, we propose the RoCo-NAS, which has used genetic-based multi-objective optimization. Results with our proposed method showed those novel robust and compact architectures exist. In fact, the evolved architectures achieved compact results comparable with state-of-the-art models and were capable of improving the robustness compared to the more complex baseline models while not having any special training or defense. In other words, the obtained architecture are inherently robust.

VI. ACKNOWLEDGEMENT

This work has been conducted in the project ”ICT programme” supported by the European Union through the European Social Fund and KKS DPAC.

REFERENCES

[1] A. Kurakin, I. J. Goodfellow, and S. Bengio, “Adversarial examples in the physical world,” in *5th International Conference on Learning Representations (ICLR)*, 2019.

[2] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” in *3rd International Conference on Learning Representations (ICLR)*, 2015.

[3] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks,” in *6th International Conference on Learning Representations (ICLR)*, 2018.

[4] H. Kannan, A. Kurakin, and I. Goodfellow, “Adversarial Logit Pairing,” in *Conference on Neural Information Processing Systems (NIPS)*, 2018.

[5] H. Zhang, Y. Yu, J. Jiao, E. P. Xing, L. E. Ghaoui, and M. I. Jordan, “Theoretically principled trade-off between robustness and accuracy,” in *36th International Conference on Machine Learning (ICML)*, 2019.

[6] C. Xie, Z. Zhang, A. L. Yuille, J. Wang, and Z. Ren, “Mitigating adversarial effects through randomization,” in *6th International Conference on Learning Representations (ICLR)*, 2018.

[7] C. Xie, Y. Wu, L. V. D. Maaten, A. L. Yuille, and K. He, “Feature denoising for improving adversarial robustness,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019.

[8] Y. Yang, G. Zhang, D. Katabi, and Z. Xu, “ME-Net: Towards effective adversarial robustness with matrix estimation,” in *36th International Conference on Machine Learning (ICML)*, 2019.

[9] C. Xie and A. Yuille, “Intriguing properties of adversarial training at scale,” in *8th International Conference on Learning Representations (ICLR)*, 2019.

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

[11] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *2nd International Conference on Learning Representations (ICLR)*, 2014.

[12] S. M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016.

[13] S. M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[14] J. Su, D. V. Vargas, and K. Sakurai, "One Pixel Attack for Fooling Deep Neural Networks," *IEEE Transactions on Evolutionary Computation*, 2019.

[15] A. Athalye, L. Engstrom, A. Ilyas, and K. Kevin, "Synthesizing robust adversarial examples," in *35th International Conference on Machine Learning (ICML)*, 2018.

[16] S. Yan, B. Fang, F. Zhang, Y. Zheng, X. Zeng, M. Zhang, and H. Xu, "HM-NAS: Efficient neural architecture search via hierarchical masking," in *Proceedings of the International Conference on Computer Vision Workshop (ICCVW)*, 2019.

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

[18] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient Neural Architecture Search via parameter Sharing," in *35th International Conference on Machine Learning (ICML)*, 2018.

[19] P. Nakkiran, "Adversarial robustness may be at odds with simplicity," *arXiv preprint arXiv:1901.00532*, 2019.

[20] S. Gui, H. Wang, H. Yang, C. Yu, Z. Wang, and J. Liu, "Model compression with adversarial robustness: A unified optimization framework," in *Advances in Neural Information Processing Systems*, 2019.

[21] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning Transferable Architectures for Scalable Image Recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018.

[22] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," in *5th International Conference on Learning Representations (ICLR)*, 2017.

[23] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, Oxford, 1989.

[24] T. Wei, C. Wang, Y. Rui, and C. W. Chen, "Network morphism," in *33rd International Conference on Machine Learning (ICML)*, 2016.

[25] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, 2002.

[26] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *7th International Conference on Learning Representations (ICLR)*, 2019.

[27] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz, "Boa: The bayesian optimization algorithm," in *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - Volume 1*, ser. GECCO'99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, p. 525–532.

[28] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015.

[29] L. Xie and A. Yuille, "Genetic CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[30] R. Kozma, C. Alippi, Y. Choe, and F. C. Morabito, *Artificial intelligence in the age of neural networks and brain computing*, 2018.

[31] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," in *6th International Conference on Learning Representations (ICLR)*, 2018.

[32] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Conference on Artificial Intelligence (AAAI)*, 2019.

[33] Z. Lu, I. Whalen, V. Boddeti, Y. Dhebar, K. Deb, E. Goodman, and W. Banzhaf, "NSGA-Net: Neural architecture search using multi-objective genetic algorithm," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, 2019.

[34] N. Carlini and D. Wagner, "Towards Evaluating the Robustness of Neural Networks," in *Proceedings - IEEE Symposium on Security and Privacy*, 2017.

[35] S. H. Silva and P. Najafirad, "Opportunities and challenges in deep learning adversarial robustness: A survey," *arXiv preprint arXiv:2007.00753*, 2020.

[36] Y. Song, S. Nowozin, N. Kushman, T. Kim, and S. Ermon, "PixelDefend: Leveraging generative models to understand and defend against adversarial examples," in *6th International Conference on Learning Representations (ICLR)*, 2018.

[37] N. Papernot and P. McDaniel, "Extending defensive distillation," *arXiv preprint arXiv:1705.05264*, 2017.

[38] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM Asia Conference on Computer and Communications Security (ASIA CCS)*, 2017.

[39] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *35th International Conference on Machine Learning (ICML)*, 2018.

[40] J. Lu, T. Issaranon, and D. Forsyth, "SafetyNet: Detecting and Rejecting Adversarial Examples Robustly," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[41] A. N. Bhagoji, D. Cullina, C. Sitawarin, and P. Mittal, "Dimensionality Reduction as a Defense against Evasion Attacks on Machine Learning Classifier," 2018.

[42] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. McDaniel, "On the (statistical) detection of adversarial examples," *arXiv preprint arXiv:1702.06280*, 2017.

[43] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," in *Network and Distributed Systems Security Symposium (NDSS)*, 2017.

[44] S. Kotyan and D. V. Vargas, "Towards evolving robust neural architectures to defend from adversarial attacks," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO)*, 2020.

[45] M. Guo, Y. Yang, R. Xu, Z. Liu, and D. Lin, "When NAS Meets Robustness: In Search of Robust Architectures Against Adversarial Attacks," in *Proceedings of the 33th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

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

[47] G. Bender, P. J. Kindermans, B. Zoph, V. Vasudevan, and Q. Le, "Understanding and simplifying one-shot architecture search," in *35th International Conference on Machine Learning (ICML)*, 2018.

[48] K. Deb and C. Myburgh, "A population-based fast algorithm for a billion-dimensional resource allocation problem with integer variables," *European Journal of Operational Research*, 2017.

[49] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2000.

[50] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[51] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016.

[52] S. Kotyan and D. Vasconcellos Vargas, "Adversarial robustness assessment: Why both l_0 and l_∞ attacks are necessary," *arXiv e-prints*, pp. arXiv–1906, 2019.

[53] H. Cai, L. Zhu, and S. Han, "Proxylessnas: Direct neural architecture search on target task and hardware," in *7th International Conference on Learning Representations (ICLR)*, 2019.

[54] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "Smash: one-shot model architecture search through hypernetworks," *arXiv preprint arXiv:1708.05344*, 2017.