IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 20 August 2022, accepted 6 October 2022, date of publication 13 October 2022, date of current version 18 October 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3214312

RESEARCH ARTICLE

RoHNAS: A Neural Architecture Search Framework With Conjoint Optimization for Adversarial Robustness and Hardware Efficiency of Convolutional and Capsule Networks

ALBERTO MARCHISIO[1], (Graduate Student Member, IEEE), VOJTECH MRAZEK[2], (Member, IEEE), ANDREA MASSA[3], BEATRICE BUSSOLINO[3], (Member, IEEE), MAURIZIO MARTINA[3], (Senior Member, IEEE), AND MUHAMMAD SHAFIQUE[4], (Senior Member, IEEE)

[1]Embedded Computing Systems Group, Institute of Computer Engineering, Technische Universität Wien (TU Wien), 1040 Vienna, Austria
[2]Evolvable Hardware Research Group, Faculty of Information Technology, Brno University of Technology, 60190 Brno, Czech Republic
[3]VLSI Laboratory, Department of Electronics and Telecommunications, Politecnico di Torino, 10129 Turin, Italy
[4]eBrain Laboratory, Division of Engineering, New York University Abu Dhabi, Abu Dhabi, United Arab Emirates

Corresponding author: Alberto Marchisio (alberto.marchisio@tuwien.ac.at)

This work was supported in part by the Doctoral College Resilient Embedded Systems through the TU Wien's Faculty of Informatics and the UAS Technikum Wien; in part by the NYUAD's Research Enhancement Fund (REF) Award on "eDLAuto: An Automated Framework for Energy-Efficient Embedded Deep Learning in Autonomous Systems;" in part by the NYUAD Center for Artificial Intelligence and Robotics (CAIR) funded by Tamkeen under the NYUAD Research Institute under Award CG010; in part by Czech Science Foundation under Grant GA22-02067S; and in part by the Ministry of Education, Youth and Sports of the Czech Republic through the e-INFRA CZ under Grant 90140.

ABSTRACT Neural Architecture Search (NAS) algorithms aim at finding efficient Deep Neural Network (DNN) architectures for a given application under given system constraints. DNNs are computationally-complex as well as vulnerable to adversarial attacks. In order to address multiple design objectives, we propose *RoHNAS*, a novel NAS framework that jointly optimizes for adversarial-robustness and hardware-efficiency of DNNs executed on specialized hardware accelerators. Besides the traditional convolutional DNNs, *RoHNAS* additionally accounts for complex types of DNNs such as Capsule Networks. For reducing the exploration time, *RoHNAS* analyzes and selects appropriate values of adversarial perturbation for each dataset to employ in the NAS flow. Extensive evaluations on multi - Graphics Processing Unit (GPU) - High Performance Computing (HPC) nodes provide a set of Pareto-optimal solutions, leveraging the tradeoff between the above-discussed design objectives. For example, a Pareto-optimal DNN for the CIFAR-10 dataset exhibits 86.07% accuracy, while having an energy of 38.63 mJ, a memory footprint of 11.85 MiB, and a latency of 4.47 ms.

INDEX TERMS Adversarial robustness, energy efficiency, latency, memory, hardware-aware neural architecture search, evolutionary algorithm, deep neural networks, capsule networks.

I. INTRODUCTION

Among the Machine Learning algorithms, Deep Neural Networks (DNNs) have shown state-of-the-art performance in a wide variety of applications [1], [2]. Finding an efficient DNN architecture for a given application through a Neural Architecture Search (NAS) is a very complex optimization problem, which involves a huge number of parameters and typically extremely long exploration time [3]. The search space becomes even bigger when employing NAS algorithms

The associate editor coordinating the review of this manuscript and approving it for publication was Chun-Hao Chen.

for new brain-inspired types of DNNs, such as the Capsule Networks (CapsNets) [4]. Such CapsNets, and more in general advanced DNN models, aim at providing high learning capabilities. However, these advancements in DNN architectures come with multiple design challenges:

1) *High computational complexity:* DNNs need specialized hardware accelerators to be deployed and executed at the edge, where the resources are constrained [5].
2) *Security:* DNN classifiers can be fooled by adversarial attacks, which are small and imperceptible perturbations added to the inputs [6]. Such a threat is extremely dangerous for safety-critical applications [7]. Furthermore, integrating means for security during NAS is a challenging problem, but can enable robust DNN designs [8], [9], as compared to the regular DNN design flow.

Hence, the problem is: *how to design complex DNNs in an energy-efficient and robust way through an automated multi-objective NAS framework?*

A. LIMITATIONS OF STATE-OF-THE-ART AND SCIENTIFIC CHALLENGES

Traditionally, the adversarial robustness of a given DNN is investigated a posteriori, i.e., once the DNN is already designed. The hardware efficiency of a DNN implemented on a given hardware accelerator is also a metric that is typically analyzed a posteriori, thus challenging the feasibility of its implementation on resource-constrained neuromorphic and/or IoT devices. We perform a motivational case study to analyze the adversarial accuracy[1] and memory footprint of different DNNs, illustrating their adversarial robustness and complexity. We apply the Projected Gradient Descent (PGD) attack [10] with $\varepsilon = 0.0001$ to the LeNet [11], the ResNet-20 [12], the CapsNet [4], and the DeepCaps [13], trained for the CIFAR-10[2] dataset [14]. The results in Fig. 1 show that the LeNet [11], which is relatively small and shallow, is hardware efficient due to its low memory footprint, but relatively more vulnerable to attacks. A more complex DNN such as the ResNet-20 [12] has a higher memory footprint but it also exhibits higher adversarial accuracy than the LeNet. Interestingly, the DeepCaps [13], despite having a smaller memory footprint than the ResNet-20, is also relatively more robust against adversarial attacks. *The goal of this paper is to integrate these diverse yet important objectives in a NAS framework to obtain Pareto-optimal solutions that explore the potential tradeoffs between different design objectives like computational complexity, memory, energy, latency, and/or security.*

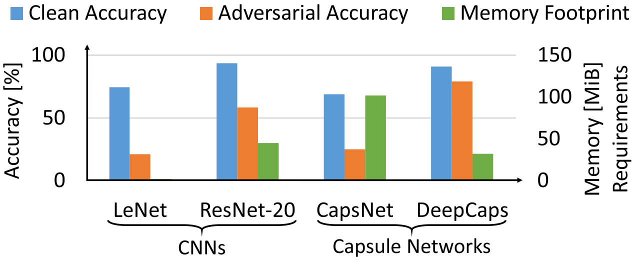


FIGURE 1. Adversarial robustness to the PGD attack vs. memory footprint of LeNet, CapsNet, ResNet-20, and DeepCaps for the CIFAR-10 dataset.

Including the DNN security into the optimization goals of the NAS is a challenging task, because, besides the challenges in its representation in the design framework, it might lead to a massive search space explosion due to several additional factors and extremely time-consuming training and evaluations of numerous candidate solutions. A wide variety of adversarial attacks have been proposed in the literature [15], and it is extremely complex to evaluate the adversarial robustness to different attack algorithms. A recent study in [16] proposed a method evaluating the DNN robustness to the PGD attack [10] as the optimization goal of the NAS. *On the contrary, our work performs joint optimizations for the adversarial robustness and hardware efficiency both, thereby leading to the increased complexity of the optimization problem, as well as large training time for evaluating the DNN robustness.* Moreover, it is challenging to model, implement and evaluate the hardware execution of different DNNs and CapsNets (including convolutional layers, fully-connected layers, and dynamic routing) in the NAS design flow.

B. OUR NOVEL CONTRIBUTIONS

To address the above-discussed challenges, we propose the novel *RoHNAS* framework (see Figure 2) that integrates multiple optimization objectives (like hardware efficiency and adversarial robustness) for diverse types of DNNs, like Convolutional Neural Networks (CNNs) and CapsNets. *RoHNAS* employs the following key mechanisms:

1) For architectural model flexibility and fast hardware estimation, we deploy analytical models of the layers and operations of DNNs and CapsNets, as well as their mapping and execution on specialized accelerators (**Section III-A**).
2) To speed-up the robustness evaluation, we analyze and choose the values of the adversarial perturbations, which provide valuable differences when performing the NAS with DNNs subjected to such adversarial perturbations (**Section III-B**).
3) We develop a specialized evolutionary algorithm, based on the principles of the Non dominated Sorting Genetic Algorithm II (NSGA-II) method [17], to perform a multi-objective Pareto-frontier selection, with conjoint optimization for adversarial robustness, energy, memory, and latency of DNNs. (**Section III-C**)

[1] We refer to the *adversarial accuracy* as the DNN test accuracy obtained when applying the adversarial attacks to every test example, i.e., by giving adversarial examples as input to the DNN.

[2] Performing numerous experiments for analyses and evaluation, constituting many NAS rounds on complex DNNs with CIFAR-10 dataset already took several weeks to months on our multi-GPU HPC node. Therefore, testing for bigger dataset is out of our currently available computational power and memory resource. Nevertheless, we believe that these findings are highly valuable, and would scale to bigger datasets as well.

4) To reduce the overall training time, we devise a fast evaluation methodology for DNNs trained for a limited number of epochs (**Section IV-C**), while the Pareto-optimal solutions are evaluated after full-training, to obtain the exact results (**Section IV-D**).

Implementation and Validation Contributions: We have implemented our *RoHNAS* using the TensorFlow library [18], and evaluated more than 900 DNNs for the MNIST, Fashion-MNIST and CIFAR-10 datasets. Extensive validations are performed on Nvidia's multi-V100 Graphics Processing Unit (GPU) High Performance Computing (HPC) Nodes requiring weeks to months of experimentation time.

Open-Source Contribution: For reproducible research, we release the code of the *RoHNAS* framework at https://github.com/ehw-fit/rohnas.

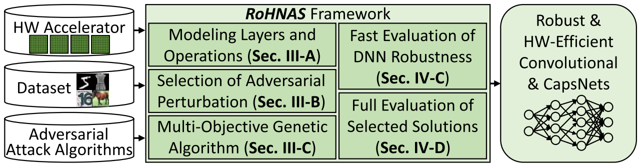

HW Accelerator
Dataset
Adversarial Attack Algorithms
RoHNAS Framework
Modeling Layers and Operations (Sec. III-A)
Selection of Adversarial Perturbation (Sec. III-B)
Multi-Objective Genetic Algorithm (Sec. III-C)
Fast Evaluation of DNN Robustness (Sec. IV-C)
Full Evaluation of Selected Solutions (Sec. IV-D)
Robust & HW-Efficient Convolutional & CapsNets

FIGURE 2. Overview of our *RoHNAS* framework.

II. BACKGROUND AND RELATED WORKS

A. ADVERSARIAL ATTACKS

DNNs are now deployed for a wide variety of applications, including safety-critical ones such as Autonomous Driving [2], Medicine [19], and Finance [20]. Despite their performance, DNNs have severe security flaws, as adversarial attacks can fool DNNs with small input perturbations [6]. Many studies [15], [21] have shown that DNNs are vulnerable to carefully crafted inputs designed to fool them. Very small imperceptible perturbations added to the data can completely change the output of the DNN model [22].

It is essential for the attacker to *minimize the added adversarial perturbation to avoid its detection*. Formally, given an original input x with a target classification label c with a DNN model $m()$, the problem of generating an adversarial example x^* can be formulated as a constrained optimization problem [15]:

$$\begin{aligned} x^* &= \arg\min_{x^*} \mathcal{D}(x, x^*), \\ & s.t.\ m(x) = c,\ m(x^*) = c^*,\ c \neq c^* \end{aligned} \tag{1}$$

where $\mathcal{D}$ is the distance between two images and the optimization objective is to minimize this adversarial perturbation to make it stealthy. x^* is considered as an adversarial example if and only if $m(x) \neq m(x^*)$ and the perturbation is bounded ($\mathcal{D}(x, x^*) < \epsilon$, where $\epsilon \geqslant 0$).

Goodfellow et al. [23] proposed the fast gradient sign method (FGSM) to generate adversarial examples by exploiting the gradient of the model w.r.t. the input images, towards the direction of the highest loss. Afterward, Madry et al. [10] and Kurakin et al. [24] proposed two different versions of the projected gradient descent (PGD) attack, an iterative version of the FGSM that introduces a perturbation α to multiple smaller steps. After each iteration, the PGD projects the generated image into a ball with a radius ϵ, keeping the perturbation size small. It is a white-box attack and has both the targeted and untargeted versions. The algorithm consists of the following iteration:

$$x_i^* = x_{i-1}^* - proj_\varepsilon(\alpha \cdot sign(\nabla_x loss(\theta, x, t))) \tag{2}$$

Further details about different types of adversarial attacks and defenses can be found in comprehensive surveys such as [6] and [15]. Moreover, recent works attempted to improve the DNN robustness against adversarial attacks by hash-based deep compression [25] or approximate computing [26], thus requiring significant hardware design overhead.

B. CONVOLUTIONAL AND CAPSULE NETWORK HARDWARE

A wide variety of hardware architectures has been proposed for accelerating the execution of DNN inference [27], [28], focusing on improving the performance and energy-efficiency through compression, dedicated operation mapping, and specialized hardware design. Recently, hardware architectures for CapsNets have been proposed [29]. CapsNets layers require the execution of operations, such as dynamic routing, that are not supported by traditional DNN accelerators but crucial to detect changes in the compositional structure of the inputs [30].

CapsNets, firstly proposed by the Google Brain's team [31], are elaborated DNN models in which the neurons are grouped together in vector form to compose the *capsules*. Each neuron of a capsule encodes spatial information, while the vector's length encodes the probability of the entity being present. While the first architecture proposed in [4] is composed of only three layers, recently deeper CapsNet models were proposed [13], [32]. The main components of a CapsNet are the following:

- **Convolutional (Conv) Layer:** The CapsNets need one or more traditional Conv layers to be applied at the beginning of the network.
- **Convolutional or Fully-Connected (FC) Capsule Layers:** A generic CapsNet can contain some Conv capsule layers, whose principle of operation is identical to that of traditional Conv layers. However, the convolution is performed between the capsules rather than neurons, and the activation function needs to be the *squash* operation (Eq. 3), which constraints the length of the capsule vectors in the range [0,1].

$$y = \frac{|x|^2}{(1 + |x|)^2} \frac{x}{|x|} \tag{3}$$

A CapsNet needs necessarily to be ended by a FC capsule layer, which mimics a traditional FC layer, but operating with capsules.

- **Dynamic Routing:** It is possible, but not necessary, to perform a *dynamic routing* between two adjacent capsule layers. The dynamic routing [4] is an iterative algorithm which associates *coupling coefficients* to the capsules predictions. The coupling coefficients of the capsules predicting the same result with greater confidence are maximized.

Figure 3 shows a simple three-layers CapsNet, as presented in [4]. CapsNets need to be completed by a *reconstruction network*, consisting of FC layers or Transposed Conv layers, to reconstruct the input image.

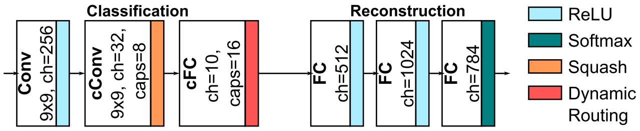


FIGURE 3. **Architectural diagram of the CapsNet model of [4].**

The aforementioned CapsAcc [33] accelerator (see Figure 4) has been proposed to efficiently deploy the CapsNets in hardware, adapting to the specific needs of the dynamic routing and the squash operation. CapsAcc differs from other DNN accelerators for its Activations unit, which can apply the squash function in addition to the traditional Rectified Linear Unit (ReLU) and Softmax functions. Moreover, a Routing Buffer is inserted to store the partial results generated during the execution of the dynamic routing algorithm. Dedicated scratchpad memories are employed to minimize the energy consumption at runtime [34].

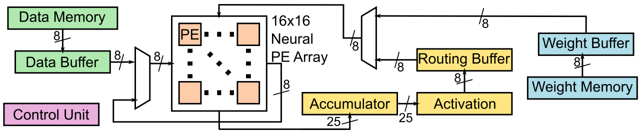


FIGURE 4. **Architectural diagram of the CapsAcc accelerator of [33].**

C. HARDWARE-AWARE NAS AND ROBUST NAS

Traditional NAS algorithms [3], [35], [36] have aimed at finding a high accurate DNN model for a given task, i.e., the DNN model which provides the highest accuracy on a given dataset. For example, the Efficient Neural Architecture Search (ENAS) algorithm [3] has generated a new architecture with 55.6 perplexity on the Penn Treebank [37] dataset. Recently, the interest in hardware efficiency has been growing, leading to designing Hardware-Aware NAS (HA-NAS) methodologies [38]. The main difference between traditional NAS and HA-NAS algorithms is that the latter also consider the hardware-deployment efficiency of candidate models, e.g., in terms of energy consumption, latency, or memory footprint. Among the related works, there exist mainly three types of heuristic search algorithms for the HA-NAS, which are (1) evolutionary algorithms, (2) reinforcement learning, and (3) differentiable NAS. The Accuracy-and-Performance-aware Neural Architecture Search (APNAS) [39], which is based on reinforcement learning, extends the ENAS algorithm by including the performance of DNNs executed in hardware in the optimization objectives of the NAS. AttentiveNAS [40] jointly optimizes the DNNs' accuracy and the computational complexity in terms of Mega FLoating Point Operations (MFLOPs). MnasNet [41] takes as an objective the inference latency and measures it by executing the candidate models on mobile phones. In [42], an extended search space is used, which includes architecture parameters, quantization, and hardware parameters, precisely the tiling factors. Targeting the Field Programmable Gate Arrays (FPGAs), the FPGA-implementation aware Neural Architecture Search (FNAS) algorithm [43] uses an analytical model to consider the latency only. HotNAS [44] targets energy efficiency by including model compression in the search space and supporting hardware for compressed models. During the candidate selection, the Single Path One-Shot (SPOS) NAS [45] applies latency and FLoating Point Operations (FLOPs) constraints. HURRICANE [46] generates a search space tailored to a specific hardware platform, considering the FLOPs and number of parameters, and their effect on the latency. The Differentiable NAS (DNAS) framework [47], in which the search space is represented by a stochastic super net, explores a layer-wise space where each layer of the CNN corresponds to a different block, and the learning is conducted by training the super net. These works are primarily for the CNN models, and cannot handle Capsule Networks.

On the other hand, recent works have also proposed NAS methodologies to achieve high robustness against adversarial attacks. In [16], a *supernet* containing all the possible architectures in the search space is trained. Then, subnetworks are sampled from the supernet and evaluated in terms of accuracy and robustness to adversarial attacks. In [48], the search space is expanded to include some combinations of layers that have been proven to be particularly effective against adversarial attacks. However, all the works that focus on NAS for adversarial attacks have not yet considered the hardware efficiency aspects as conjoint optimization objectives. Moreover, these works are primarily for the CNN models, and cannot handle CapsNets. Recently, NASCaps [49] has proposed a NAS methodology for CapsNets based on an evolutionary algorithm, but it cannot handle robustness challenges, and does not explore the tradeoffs between hardware efficiency and adversarial robustness.

Our RoHNAS framework distinguishes from the previous works because it combines for the first time hardware efficiency and robustness to adversarial perturbations as joint optimization goals for the NAS, and targets both CNN and CapsNets models.

III. RoHNAS FRAMEWORK

Our evolutionary algorithm-based NAS methodology performs a multi-objective search. It automatically searches for inherently robust yet hardware-efficient DNN models by selecting Pareto-optimal candidates in terms of robustness, energy, latency, and memory footprint. The search space comprises both CNNs and CapsNets. The workflow of our

RoHNAS framework is shown in Figure 5, and is explained in detail in the following subsections.

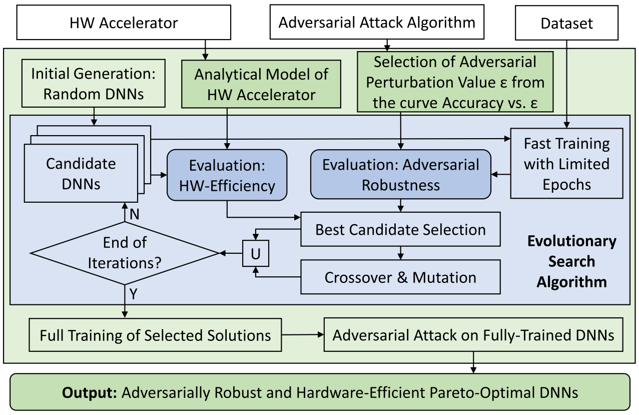

HW Accelerator
Adversarial Attack Algorithm
Dataset
Initial Generation: Random DNNs
Analytical Model of HW Accelerator
Selection of Adversarial Perturbation Value ε from the curve Accuracy vs. ε
Candidate DNNs
Evaluation: HW-Efficiency
Evaluation: Adversarial Robustness
Fast Training with Limited Epochs
N
Best Candidate Selection
End of Iterations?
U
Crossover & Mutation
Evolutionary Search Algorithm
Y
Full Training of Selected Solutions
Adversarial Attack on Fully-Trained DNNs
Output: Adversarially Robust and Hardware-Efficient Pareto-Optimal DNNs

FIGURE 5. Overview of our *RoHNAS* framework and its key functionalities.

The framework's inputs are the hardware accelerator, the algorithm for generating the adversarial attack, and the dataset. After modeling analytically the hardware accelerator, the appropriate values of the adversarial perturbation to employ in the search are selected. This process, as will be described more in detail in Section III-B, consists of analyzing the accuracy vs. adversarial perturbation curve, and focusing on the high variation region which corresponds to the highest slope of the curve. After selecting the values of the adversarial perturbation to employ in the search, the evolutionary search algorithm (based on the principles of the NGSA-II genetic algorithm [17]) performs an iterative exploration through crossover, mutation, and best DNN candidate selection based on the objectives. To speed up the process, during the evolutionary algorithm, the adversarial robustness is evaluated after a fast training, i.e., for DNNs trained with a limited number of epochs, where its number is determined based on the Pearson Correlation Coefficient [50]. Towards generating exact robustness results, the set of Pareto-optimal DNN models are fully-trained, and the robustness against the adversarial attack on fully-trained DNNs is evaluated.

A. LAYER AND OPERATION MODELING

The *RoHNAS* framework models each layer through a *layer descriptor*, which contains all the relevant architectural parameters necessary to describe a generic DNN layer using a position-based representation. As shown in Figure 6, a layer descriptor contains all the information to construct its related layer, such as layer type, input feature map (IFM) size, input channels, input capsules, kernel size, stride size, output feature map (OFM) size, output channels, and output capsules. Using these parameters, it is possible to build many different types of CNN or CapsNet layers. Moreover, such a modular representation can easily be extended to support different layer types. Multiple layer descriptors, together with information on extra skip connections and resizing of the inputs, form a *genotype*, which allows describing various CNN and CapsNet architectural models.

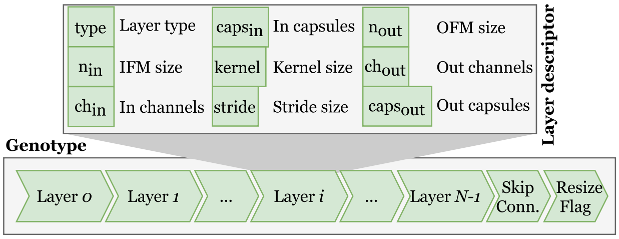


FIGURE 6. Genotype structure. IFM stands for input feature map, while OFM means ouput feature map.

To estimate the execution requirements of a DNN model on a specialized DNN hardware accelerator (e.g., CapsAcc [33] or Tensor Processing Unit (TPU) [51]), it is necessary to know its underlying hardware characteristics, for instance:

- T, the clock period;
- $Load\_Weights$, the number of clock cycles necessary to load the weights into the Processing Element (PE) array;
- $P_{PEarray}$, the power consumed by the PE array, here estimated with the Synopsys Design Compiler tool;
- E_{memory}, the energy required for one memory access, here estimated with the CACTI-P tool [52];

Knowing these parameters makes it possible to estimate the latency, energy consumption, and memory footprint of a DNN model analytically. From the dimensions of the layers of a given DNN, we assess:

- w_l, the number of weights in a layer;
- s_l, the number of values to be summed to obtain an output value, for each layer;
- f_l, the number of feature maps to be multiplied by the same weight, for each layer;
- c_l, the number of clock cycles needed to process a layer.

Given the model and the hardware features, the number of groups of weights loaded into the array ($w_{PEarray}$) and the number of memory accesses (m_acc) can be determined through Equations 4 and 5, respectively. By computing the clock cycles (see Eq. 6), it is possible to estimate the latency and energy consumption, which, in conjunction with the memory footprint, form the set of hardware parameters computed through Eq. 7.

$$w_{PEarray} = \left\lceil \frac{w_l}{16 \cdot \min(16, s_l)} \right\rceil \tag{4}$$

$$m\_acc = \begin{cases} 256, & \text{if } f_l = 1 \\ 16 \cdot \max(s_l - 15, 1), & \text{otherwise} \end{cases} \tag{5}$$

$$c_l = w_l \cdot w_{PEarray} + f_l \tag{6}$$

$$\begin{aligned} latency &= \sum_{l \in L} c_l \cdot T \\ energy &= \left\lceil \frac{m\_acc}{128} \right\rceil \cdot E_{memory} \\ &\quad + \sum_{l \in L} c_l \cdot T \cdot P_{PEarray} \\ memory\ footprint &= \sum_{l \in L} w_l \end{aligned} \tag{7}$$

The model has been validated by comparing the results with the hardware implementation of the CapsAcc [33]. Recent studies in [49] have also shown that the above-discussed parameters and such analytical models are sufficient to accurately estimate the latency, energy, and memory footprint of a given DNN model. In the following, we discuss the efficacy of our analytical models by comparing the estimated values with the real values of latency, energy, and memory requirements. By comparing our analytical model with the real implementation of the CapsNet [4] on CapsAcc [33], our model provides accurate estimations of latency and memory footprint, and underesimates the energy consumption by around 25%. Such a difference might be due to other elements of the hardware implementation (e.g., interconnection overhead) that are not considered by the analytical model. Despite this underestimation, the fidelity of our models is high, i.e., all candidates have similar underestimation trend, so the selection of the candidates would not be affected by this underestimation of analytical models. Please note that our main focus was to have fast estimation with high fidelity.

B. DESIGN SPACE REDUCTION BY SELECTING AN APPROPRIATE ADVERSARIAL PERTURBATION VALUE

Since the design space can potentially explode by considering several types and strengths of adversarial perturbations, the *RoHNAS* framework restricts the design space by automatically selecting the values of adversarial perturbations to be used in the NAS for a given dataset. Algorithm 1 summarizes the proposed procedure. For each element of the testing dataset, the adversarial example is generated through the PGD algorithm [10] (line 4). Note, here we use PGD for illustrative reasons, and other adversarial attack algorithms can be integrated into our *RoHNAS* framework. The parameter ε determines the amount of adversarial perturbation. When considering the variation of the accuracy w.r.t. ε, as we will show in Section IV-B, the region in which the slope is highest is in the middle of the graph, which corresponds to half of the clean accuracy, i.e., $\frac{Acc_0}{2}$ when considering that Acc_0 is the clean accuracy. By exploiting this intuition, our algorithm selects ε_{NAS}, which is the value of adversarial perturbation that provides the closest accuracy to the desired value of $\frac{Acc_0}{2}$. The selected value of ε_{NAS} is employed in the *One EPS* search, which optimizes for the robustness against one value of perturbation. Moreover, aiming at covering a wider spectrum of adversarial perturbation range, the *Two EPS* search is devised. ε_{low} and ε_{high} are selected (lines 10-11), and the NAS is conducted by optimizing for the adversarial accuracy with both values.

C. MULTI-OBJECTIVE EVOLUTIONARY ALGORITHM

The selection of the Pareto-optimal solutions in the *RoHNAS* framework is based on the principles of the NSGA-II algorithm [17]. The main core of the search algorithm is iterated for g times, where each iteration g_i represents a *generation*. Each generation g_i consists of a set of *parent candidates* P_i, from which the *offspring candidates* Q_i are generated.

Algorithm 1 Adversarial Perturbation Selection

```
Input: Deep Neural Network: $N$;
Test Dataset: $\mathcal{D} = \bigcup_j X_j$;
Adversarial Perturbation Budget:
$\varepsilon_i \in \mathcal{E} = [\varepsilon_{MIN}, \varepsilon_{MAX}]$;
Output: Perturbation to apply for the NAS: $\varepsilon_{NAS}$;
1  $Acc_0 = Accuracy(N(D))$;
2  for $i \in < \mathcal{E} >$ do
3      for $j \in < \mathcal{D} >$ do
4          $X'_{ij} = PGD(N, \varepsilon_i, X_j)$;
5      end
6      $\mathcal{D}'_i = \bigcup_j X'_{ij}$;
7      $Acc_i = Accuracy(N(\mathcal{D}'_i))$;
8  end
9  $\varepsilon_{NAS} = \varepsilon_i \ : \ Acc_i \approx \frac{Acc_0}{2}$;
10 $\varepsilon_{low} \approx \frac{\varepsilon_{NAS}}{10}$;
11 $\varepsilon_{high} \approx 3 \cdot \varepsilon_{NAS}$;
```

At each generation g_i, the offsprings are generated from the parents via *crossover* and *mutation*. To perform the crossover operation, two parents P_a and P_b are randomly selected from the whole set of parent candidates. The genotypes of P_a and P_b are then pseudo-randomly splitted in two parts, obtaining four genotypes: $P_{a,1}$, $P_{a,2}$, $P_{b,1}$ and $P_{b,2}$. Two offsprings are then obtained concatenating the four genotypes as follows:

$$Q_a = P_{a,1} \& P_{b,2} \qquad Q_b = P_{b,1} \& P_{a,2} \tag{8}$$

To perform a mutation, a random parameter of a random layer descriptor is selected and modified. In particular, the kernel size, the stride, the skip connections, and the number of output capsules can be affected. When the generation of the offsprings is complete, it is necessary to check the validity of the solutions and in case remove the invalid candidates.

After the crossover and mutation processes, the set of candidates is the union of the parents and the offsprings sets. To select the best candidates, that will then be the parents in the next generation, the solutions are divided into a series $F_1, F_2, \ldots, F_N$ of Pareto-fronts, where F_1 is the best Pareto-front. The next-generation parents' set P_{i+1} is filled with the solutions from the best Pareto-front. To obtain the chosen number of candidates, it may be necessary to select only a certain number of solutions from a Pareto-front (e.g., F_3 in Figure 7). In this case, the Pareto-front's solutions are sorted by *crowding distance*, and the best ones are picked.

IV. EVALUATION OF THE RoHNAS FRAMEWORK

A. EXPERIMENTAL SETUP

The flow of our experiments and the tools used to implement the *RoHNAS* framework are summarized in Fig. 8. The PGD adversarial attack algorithm [10] has been implemented with the CleverHans library [53]. The hardware model has been implemented using the open-source NASCaps library [49],

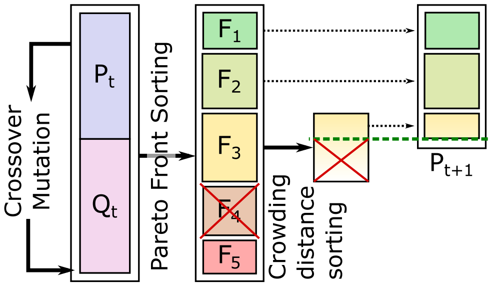


FIGURE 7. One iteration of the NSGA-II algorithm.

which is based on the CapsAcc architecture [33] synthesized using the Synopsys Design Compiler tool, with a 45nm technology node and a clock period of 3ns. The training and testing of the DNNs, implemented in TensorFlow [18] have been running on the GPU-HPC computing nodes equipped with four NVIDIA Tesla V100-SXM2 GPUs. Note that, our experiments were running for 2,000 GPU hours with our fast evaluation method and 8,000 GPU hours for the final training and PGD attack evaluation. Without such exploration time reductions, or by considering more complex optimization problems (e.g., larger datasets or deeper DNN models), the exploration time would have lasted several GPU months.

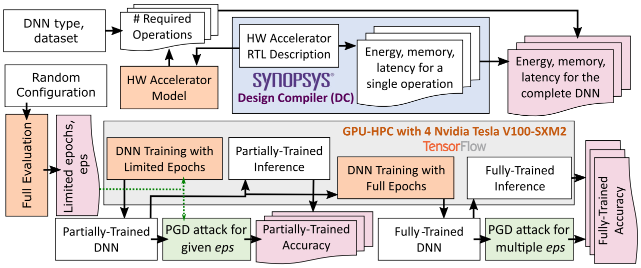

DNN type, dataset
# Required Operations
HW Accelerator RTL Description
Energy, memory, latency for a single operation
Energy, memory, latency for the complete DNN
Random Configuration
HW Accelerator Model
SYNOPSYS
Design Compiler (DC)
GPU-HPC with 4 Nvidia Tesla V100-SXM2
TensorFlow
Full Evaluation
Limited epochs, eps
DNN Training with Limited Epochs
Partially-Trained Inference
DNN Training with Full Epochs
Fully-Trained Inference
Fully-Trained Accuracy
Partially-Trained DNN
PGD attack for given eps
Partially-Trained Accuracy
Fully-Trained DNN
PGD attack for multiple eps

FIGURE 8. Tool-flow and setup for conducting the experiments.

The search algorithm is initialized with a random population of 10 elements, running for a maximum of 20 iterations of the genetic loop. The offspring population size is 10, and the mutation probability is 10%. Each convolutional layer can be composed of a 3 × 3, 5 × 5, or 9 × 9 kernel, with a stride of either 1 or 2. The channels and capsule dimensions can both span between 1 and 64.

B. SELECTION OF ADVERSARIAL PERTURBATION FOR THE NAS

The amount of adversarial perturbation is a key parameter to be selected for performing the NAS. Following the procedure described in Section III-B, the Pareto-optimal DNNs of the NASCaps library [49] have been tested under the PGD attack [10], with different values of the adversarial perturbation ε. The results reported in Fig. 9 show that, as expected, the higher ε is, the lower the DNNs' accuracy drops. The selected values for the NAS are reported in Table 1. The selection process follows the procedure described in Algorithm 1. The *One EPS* column refers to the search using a single value of ε, while the *Two EPS* column refer to a search conducted with two different values of ε, which are called ε_{low} and ε_{high}. Note, a simple dataset like the MNIST requires a relatively high adversarial perturbation to impact the DNN robustness. On the other hand, on a more complex dataset like the CIFAR-10, a smaller perturbation is already sufficient to misclassify a certain set of inputs.

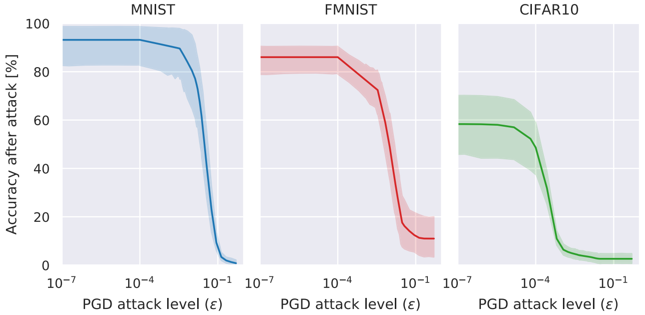


FIGURE 9. Analysis of the DNN robustness under the PGD attack, with different adversarial perturbation values, for MNIST, Fashion-MNIST, and CIFAR-10.

TABLE 1. Selected values of the adversarial perturbation ε for the NAS, for MNIST, Fashion-MNIST and CIFAR-10 datasets. There are also reported the values of ε_{low} and ε_{high} for the *Two EPS* search, which will be used for comparison in Section IV-D.

	Two EPS ε_{low}	One EPS ε	Two EPS ε_{high}
MNIST	3e-3	3e-2	1e-1
F-MNIST	1e-3	1e-2	3e-2
CIFAR-10	3e-5	3e-4	1e-3

C. RoHNAS RESULTS WITH FAST DNN ROBUSTNESS EVALUATION

As discussed in Section III, to reduce the exploration time, our algorithm trains the DNNs only for a limited number of epochs, which results in a fast robustness evaluation. The similarity w.r.t. the full-training robustness has been measured through the Pearson Correlation Coefficient [50], using the procedure described in [49]. The choice of 10 training epochs for the CIFAR-10 dataset and 5 epochs for the Fashion-MNIST and MNIST datasets leverages the tradeoff between a high correlation and low training time.

The results of the *RoHNAS - One EPS* with fast robustness evaluation are reported in Fig. 10. The earliest generation of the algorithm produces sub-optimal DNN solutions, while most Pareto-optimal solutions are found in the latest generation. Note that, for the *RoHNAS* evaluated on the CIFAR-10 dataset, the latest generations find DNNs that are less robust to the PGD attack, but still belong to the Pareto-frontier due to the low energy consumption (see pointer ①). Note that, as highlighted by pointer ②, several candidate DNNs found in the earliest generations are highly vulnerable to the PGD attack and are automatically discarded by the Pareto-frontier selection.

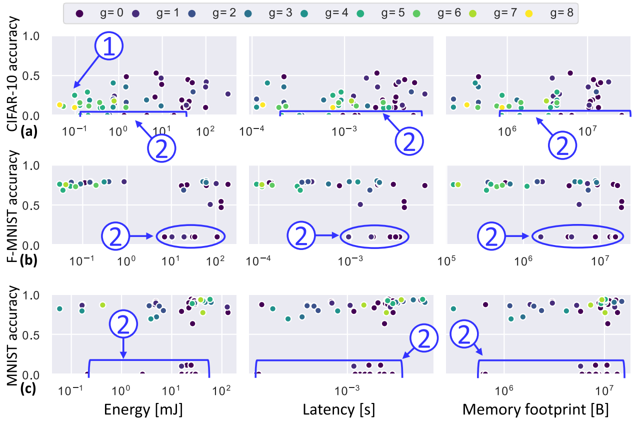


FIGURE 10. ***RoHNAS*' fast evaluation of DNN robustness under PGD attack, showing tradeoffs w.r.t. energy, latency and memory footprint. (a) Results for CIFAR-10. (b) Results for Fashion-MNIST. (c) Results for MNIST.**

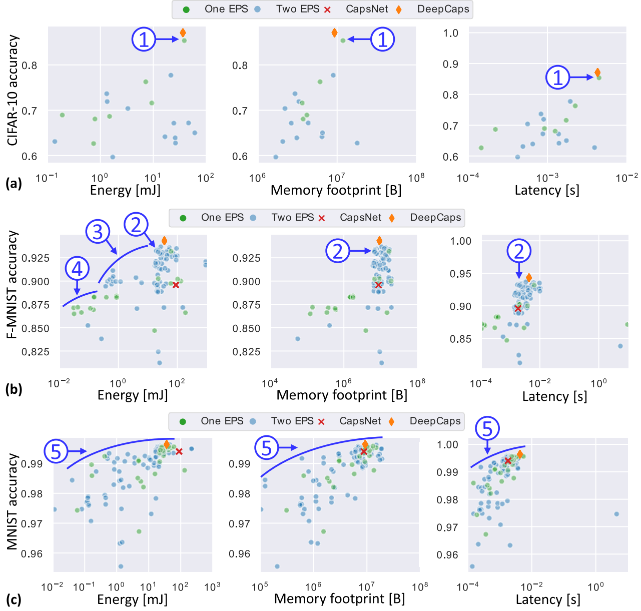


FIGURE 11. ***RoHNAS*' exact robustness evaluation of Pareto-optimal DNN solutions under the PGD attack, showing tradeoffs w.r.t. hardware-efficiency. (a) Results for CIFAR-10. (b) Results for Fashion-MNIST. (c) Results for MNIST.**

D. RoHNAS EXACT RESULTS FOR PARETO-OPTIMAL DNNs

The Pareto-optimal DNNs that are selected at the previous stage have been *fully-trained* to obtain an exact robustness evaluation. The DNNs for the MNIST and Fashion-MNIST datasets have been trained for 100 epochs, while 300 epochs of training has been used for the DNNs targeting the CIFAR-10 dataset. The results reported in Fig. 11 show tradeoffs between the design objectives. As highlighted by pointer ① in Fig. 11, a Pareto-optimal solution found by the *RoHNAS* framework for the CIFAR-10 dataset achieves 86.07% accuracy while having an energy consumption of 38.63 mJ, a memory footprint of 11.85 MiB, and a latency of 4.47 ms. Similarly, the solution for the Fashion-MNIST dataset pointed in ② reaches an accuracy of 93.40% while having 6.40 ms latency, 61.19 mJ energy, and 16.82 MiB memory. Note that, while the *Two EPS* search finds Pareto-optimal solutions in the middle range of energy (see pointer ③), other interesting low-energy solutions are found by the *One EPS* search, as indicated in pointer ④. The Pareto-optimal DNNs's search for MNIST covers a more heterogeneous range of values, leveraging tradeoffs between different objectives (see pointer ⑤).

The *RoHNAS* framework has been compared with other state-of-the-art DNN and CapsNet architectures, and NAS methodologies that include capsule layers in the search space. Fig. 12 shows the comparison between our *RoHNAS* framework (*One EPS* setting), NASCaps [49], CapsNet [4] and DeepCaps [13]. For the MNIST dataset, the Pareto-optimal solutions generated with the *RoHNAS* framework are particularly robust for a high range of perturbation ε (see pointer ①). Indeed, the accuracy starts dropping at around one order of magnitude higher ε than NASCaps (see pointer ②). For the Fashion-MNIST, the robustness behavior of the Pareto-optimal DNNs selected with the *RoHNAS* frameweork is closely related to the CapsNet. Instead, for the CIFAR-10 dataset, the *RoHNAS* DNNs' behavior is similar to the DeepCaps for low values of ε (see pointer ③), while a Pareto-optimal *RoHNAS* solution offer a respectable robustness also with higher adversarial perturbation (see pointer ④).

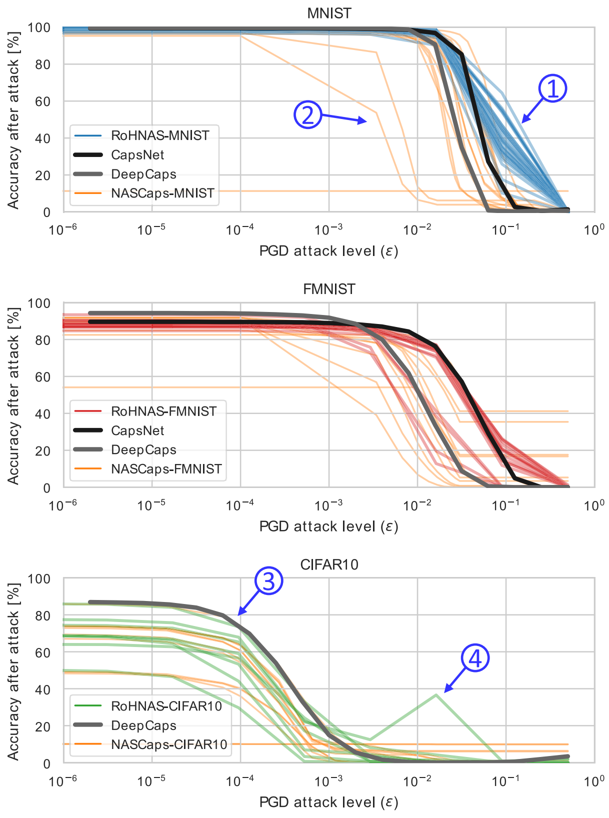


FIGURE 12. **Evaluation of the *RoHNAS* framework with the *One EPS* setting, compared to other state-of-the-art architectures and NAS algorithms.**

The evaluation of the *RoHNAS* framework with the *Two EPS* setting is shown in Fig. 13. Compared to the *One EPS* setting, the NAS produces different levels of robustness w.r.t. ε for the MNIST and Fashion-MNIST datasets (see pointer ① in Fig. 13). However, for the CIFAR-10 dataset, the *Two EPS* search leads to worse results than the *One EPS* counterpart (see pointer ②).

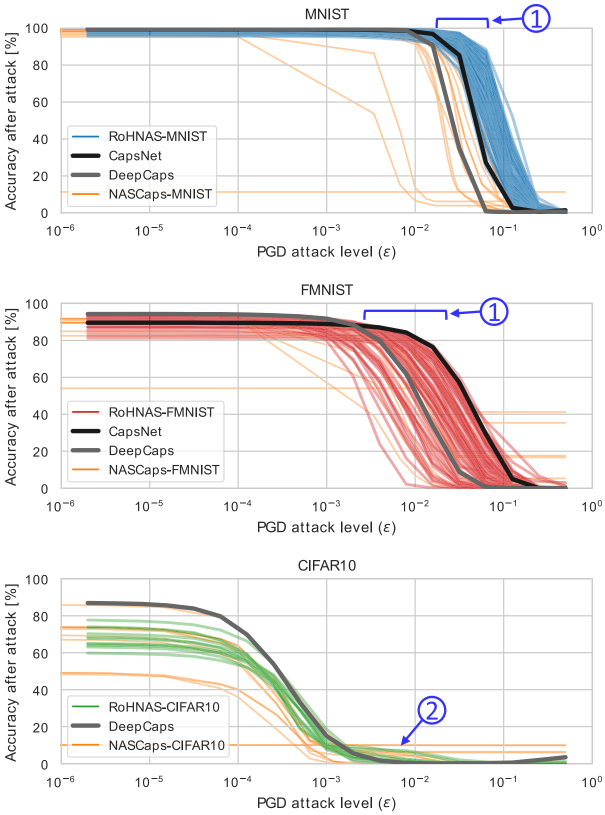


FIGURE 13. Evaluation of the *RoHNAS* framework with the *Two EPS* setting, compared to other state-of-the-art architectures and NAS algorithms.

E. RoHNAS RESULTS VS. RANDOM SEARCH

The *RoHNAS* framework based on the evolutionary search algorithm has been compared to a modified version using random search. The results of the fast DNN robustness evaluation for the *Two EPS* configuration are shown in Figures 14 and 15, where Fig. 14 shows the accuracy measured when the adversarial perturbation value for the PGD attack is ε_{high}, while Fig. 15 uses ε_{low}. Pointer ① in Fig. 14 indicates the Pareto frontier for the CIFAR-10 dataset obtained by random search, which is outperformed by several candidate DNN models found using the NSGA-II algorithm of our *RoHNAS* framework (see pointer ② in Fig. 14). For the Fashion-MNIST dataset, some candidate DNNs found using the random search show high hardware efficiency, but the solution generated through the NSGA-II algorithm indicated by pointer ③ shows higher robustness. Also for the MNIST dataset, the NSGA-II generates solutions that have better tradeoffs between the objectives, compared to using random search (see pointer ④).

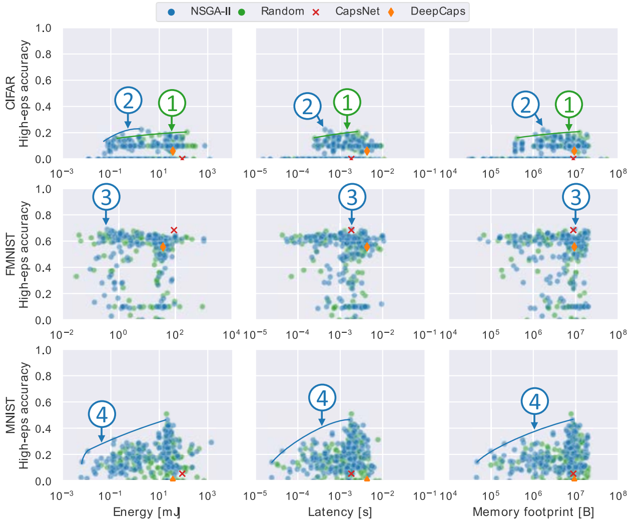


FIGURE 14. *RoHNAS*' fast evaluation of DNN robustness under PGD attack in the *Two EPS* setting using the ε_{high} value, compared to the solutions found with random search, showing tradeoffs w.r.t. energy, latency, and memory footprint.

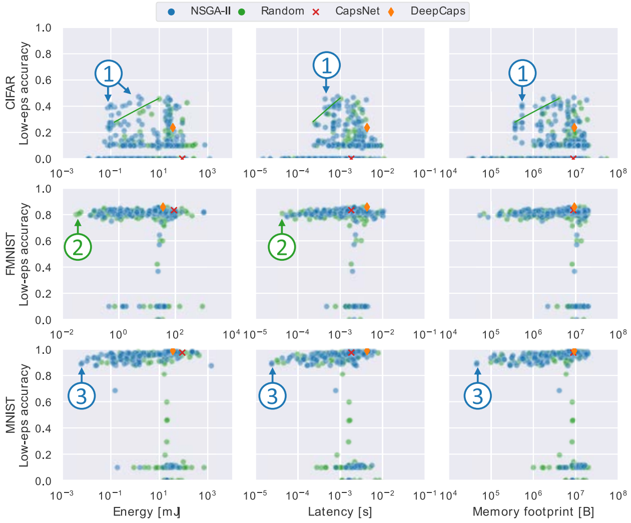


FIGURE 15. *RoHNAS*' fast evaluation of DNN robustness under PGD attack in the *Two EPS* setting using the ε_{low} value, compared to the solutions found with random search, showing tradeoffs w.r.t. energy, latency, and memory footprint.

Similar observations can be made when the adversarial perturbation value for the PGD attack is ε_{low}. As indicated by pointer ① in Fig. 15, several candidate DNN models for the CIFAR-10 dataset found using the NSGA-II algorithm have better tradeoffs than the Pareto-frontier obtained with random search. However, for the Fashion-MNIST dataset, the solutions with high hardware efficiency (especially low energy and low latency) are found by random search (see pointer ②). On the other hand, the NSGA-II algorithm generates solutions with higher hardware efficiency for the MNIST dataset (see pointer ③).

The Pareto-optimal DNNs selected using random search for the *Two EPS* setting are *fully-trained* and compared to the results of the *RoHNAS* framework in Fig. 16. As highlighted by pointer ①, the Pareto-optimal solutions generated by both algorithms are robust for a high range of perturbation values ε. However, the key differences can be observed in some curves belonging to the *RoHNAS* search, which exhibit higher robustness than the curves obtained with the random search (see pointer ② in Fig. 16).

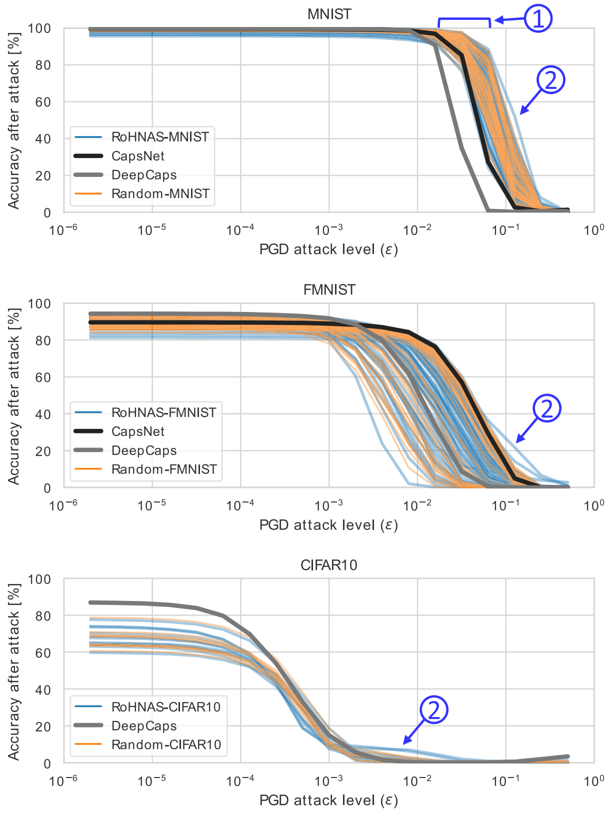


FIGURE 16. **Evaluation of the *RoHNAS* framework with the *Two EPS* setting, compared to the random search.**

Our framework supports the integration of different search techniques. Therefore, depending upon the requirements of a system, different search techniques can be run and the best possible solutions can be picked. However, this will lead to a higher experimentation time. Therefore, we recommend using the NSGA-II search algorithm that outperforms the random search in most of the cases.

V. CONCLUSION

In this paper, we proposed *RoHNAS*, a novel framework for the Neural Architecture Search, jointly optimizing for the hardware efficiency (latency, energy, and memory footprint) and robustness against adversarial attacks. Our optimizations for reducing the search space and the exploration time allow finding a set of CNNs and CapsNets, which are Pareto-optimal w.r.t. the above-discussed objectives, in a fast fashion. In our experiments, 900 different DNN models have been evaluated, using 2,000 GPU hours with our fast training settings. Thanks to our *RoHNAS* framework, the deployment of robust DNNs in resource-constrained IoT/neuromorphic edge devices is made possible. We open-source our framework at https://github.com/ehw-fit/rohnas.

REFERENCES

[1] M. Capra, B. Bussolino, A. Marchisio, G. Masera, M. Martina, and M. Shafique, "Hardware and software optimizations for accelerating deep neural networks: Survey of current trends, challenges, and the road ahead," *IEEE Access*, vol. 8, pp. 225134–225180, 2020, doi: 10.1109/ACCESS.2020.3039858.

[2] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *J. Field Robot.*, vol. 37, no. 3, pp. 362–386, Apr. 2020, doi: 10.1002/rob.21918.

[3] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *Proc. 35th Int. Conf. Mach. Learn. (ICML)*, vol. 80, J. G. Dy and A. Krause, Eds., Stockholm, Sweden, Jul. 2018, pp. 4092–4101. [Online]. Available: http://proceedings.mlr.press/v80/pham18a.html

[4] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Proc. Adv. Neural Inf. Process. Syst. 30, Annu. Conf. Neural Inf. Process. Syst.*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., Long Beach, CA, USA, Dec. 2017, pp. 3856–3866. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/2cad8fa47bbef282bad bb8de5374b894-Abstract.html

[5] A. Marchisio, M. A. Hanif, F. Khalid, G. Plastiras, C. Kyrkou, T. Theocharides, and M. Shafique, "Deep learning for edge computing: Current trends, cross-layer optimizations, and open research challenges," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Miami, FL, USA, Jul. 2019, pp. 553–559, doi: 10.1109/ISVLSI.2019.00105.

[6] M. Shafique, M. Naseer, T. Theocharides, C. Kyrkou, O. Mutlu, L. Orosa, and J. Choi, "Robust machine learning systems: Challenges, Current trends, perspectives, and the road ahead," *IEEE Des. Test*, vol. 37, no. 2, pp. 30–57, Apr. 2020, doi: 10.1109/MDAT.2020.2971217.

[7] C.-H. Cheng, F. Diehl, G. Hinz, Y. Hamza, G. Nüehrenberg, M. Rickert, H. Ruess, and M. Truong-Le, "Neural networks for safety-critical applications—Challenges, experiments and perspectives," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, J. Madsen and A. K. Coskun, Eds., Dresden, Germany, Mar. 2018, pp. 1005–1006, doi: 10.23919/DATE.2018.8342158.

[8] S. Dave, A. Marchisio, M. A. Hanif, A. Guesmi, A. Shrivastava, I. Alouani, and M. Shafique, "Special session: Towards an agile design methodology for efficient, reliable, and secure ML systems," *CoRR*, vol. abs/2204.09514, pp. 1–14, Apr. 2022.

[9] M. Shafique, A. Marchisio, R. V. W. Putra, and M. A. Hanif, "Towards energy-efficient and secure edge AI: A cross-layer framework ICCAD special session paper," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, Munich, Germany, Nov. 2021, pp. 1–9, doi: 10.1109/ICCAD51958.2021.9643539.

[10] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proc. 6th Int. Conf. Learn. Represent. (ICLR)*, Vancouver, BC, Canada, Apr./May 2018, pp. 1–28. [Online]. Available: https://openreview.net/forum?id=rJzIBfZAb

[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998, doi: 10.1109/5.726791.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA: IEEE Computer Society, Jun. 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.

[13] J. Rajasegaran, V. Jayasundara, S. Jayasekara, H. Jayasekara, S. Seneviratne, and R. Rodrigo, "DeepCaps: Going deeper with capsule networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*. Long Beach, CA, USA: Computer Vision Foundation, Jun. 2019, pp. 10725–10733. [Online]. Available: http://openaccess.thecvf.com/content_CVPR_2019/html/Rajasegaran_DeepCaps_Going_Deeper_With_Capsule_Networks_CVPR_2019_paper.html

[14] A. Krizhevsky, ‘‘Learning multiple layers of features from tiny images,’’ Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009.

[15] X. Yuan, P. He, Q. Zhu, and X. Li, ‘‘Adversarial examples: Attacks and defenses for deep learning,’’ *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9, pp. 2805–2824, Sep. 2019, doi: 10.1109/TNNLS.2018.2886017.

[16] M. Guo, Y. Yang, R. Xu, Z. Liu, and D. Lin, ‘‘When NAS meets robustness: In search of robust architectures against adversarial attacks,’’ in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*. Seattle, WA, USA: Computer Vision Foundation, Jun. 2020, pp. 628–637. [Online]. Available: https://openaccess.thecvf.com/content_CVPR_2020/html/Guo_When_NAS_Meets_Robustness_In_Search_of_Robust_Architectures_Against_CVPR_2020_paper.html

[17] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, ‘‘A fast and elitist multiobjective genetic algorithm: NSGA-II,’’ *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Aug. 2002, doi: 10.1109/4235.996017.

[18] M. Abadi et al., ‘‘TensorFlow: A system for large-scale machine learning,’’ in *Proc. 12th USENIX Symp. Operating Syst. Design Implement. (OSDI)*, K. Keeton and T. Roscoe, Eds. Savannah, GA, USA: USENIX Association, Nov. 2016, pp. 265–283. [Online]. Available: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi

[19] C. Barata and J. S. Marques, ‘‘Deep learning for skin cancer diagnosis with hierarchical architectures,’’ in *Proc. IEEE 16th Int. Symp. Biomed. Imag. (ISBI)*, Venice, Italy, Apr. 2019, pp. 841–845, doi: 10.1109/ISBI.2019.8759561.

[20] R. Zanc, T. Cioara, and I. Anghel, ‘‘Forecasting financial markets using deep learning,’’ in *Proc. IEEE 15th Int. Conf. Intell. Comput. Commun. Process. (ICCP)*, S. Nedevschi, R. Potolea, and R. R. Slavescu, Eds., Cluj-Napoca, Romania, Sep. 2019, pp. 459–466, doi: 10.1109/ICCP48234.2019.8959715.

[21] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, ‘‘Intriguing properties of neural networks,’’ in *Proc. 2nd Int. Conf. Learn. Represent. (ICLR)*, Y. Bengio and Y. LeCun, Eds., Banff, AB, Canada, Apr. 2014, pp. 1–10.

[22] J. J. Zhang, K. Liu, F. Khalid, M. A. Hanif, S. Rehman, T. Theocharides, A. Artussi, M. Shafique, and S. Garg, ‘‘Building robust machine learning systems: Current progress, research challenges, and opportunities,’’ in *Proc. 56th Annu. Design Autom. Conf.*, Las Vegas, NV, USA, Jun. 2019, p. 175, doi: 10.1145/3316781.3323472.

[23] I. J. Goodfellow, J. Shlens, and C. Szegedy, ‘‘Explaining and harnessing adversarial examples,’’ in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, Y. Bengio and Y. LeCun, Eds., San Diego, CA, USA, May 2015, pp. 1–11.

[24] A. Kurakin, I. J. Goodfellow, and S. Bengio, ‘‘Adversarial examples in the physical world,’’ in *Proc. 5th Int. Conf. Learn. Represent. (ICLR)*, Toulon, France, Apr. 2017, pp. 1–14. [Online]. Available: https://openreview.net/forum?id=HJGU3Rodl

[25] Q. Liu, T. Liu, Z. Liu, Y. Wang, Y. Jin, and W. Wen, ‘‘Security analysis and enhancement of model compressed deep learning systems under adversarial attacks,’’ in *Proc. 23rd Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Y. Shin, Ed., Jeju, South Korea, Jan. 2018, pp. 721–726, doi: 10.1109/ASPDAC.2018.8297407.

[26] A. Guesmi, I. Alouani, K. N. Khasawneh, M. Baklouti, T. Frikha, M. Abid, and N. Abu-Ghazaleh, ‘‘Defensive approximation: Securing CNNs using approximate computing,’’ in *Proc. 26th ACM Int. Conf. Architectural Support Program. Lang. Operating Syst.*, T. Sherwood, E. D. Berger, and C. Kozyrakis, Eds. Virtual Event, USA, Apr. 2021, pp. 990–1003, doi: 10.1145/3445814.3446747.

[27] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, ‘‘Efficient processing of deep neural networks: A tutorial and survey,’’ *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017, doi: 10.1109/JPROC.2017.2761740.

[28] M. Capra, B. Bussolino, A. Marchisio, M. Shafique, G. Masera, and M. Martina, ‘‘An updated survey of efficient hardware architectures for accelerating deep convolutional neural networks,’’ *Future Internet*, vol. 12, no. 7, p. 113, Jul. 2020. [Online]. Available: https://www.mdpi.com/1999-5903/12/7/113

[29] A. Marchisio, V. Mrazek, M. A. Hanif, and M. Shafique, ‘‘FEECA: Design space exploration for low-latency and energy-efficient capsule network accelerators,’’ *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 29, no. 4, pp. 716–729, Apr. 2021, doi: 10.1109/TVLSI.2021.3059518.

[30] S. R. Venkatraman, A. Anand, S. Balasubramanian, and R. R. Sarma, ‘‘Learning compositional structures for deep learning: Why routing-by-agreement is necessary,’’ *CoRR*, vol. abs/2010.01488, pp. 1–11, Oct. 2020.

[31] G. E. Hinton, A. Krizhevsky, and S. D. Wang, ‘‘Transforming auto-encoders,’’ in *Artificial Neural Networks and Machine Learning—ICANN 2011* (Lecture Notes in Computer Science), vol. 6791, T. Honkela, W. Duch, M. A. Girolami, and S. Kaski, Eds. Espoo, Finland: Springer, Jun. 2011, pp. 44–51, doi: 10.1007/978-3-642-21735-7_6.

[32] K. Sun, L. Yuan, H. Xu, and X. Wen, ‘‘Deep tensor capsule network,’’ *IEEE Access*, vol. 8, pp. 96920–96933, 2020, doi: 10.1109/ACCESS.2020.2996282.

[33] A. Marchisio, M. A. Hanif, and M. Shafique, ‘‘CapsAcc: An efficient hardware accelerator for CapsuleNets with data reuse,’’ in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, J. Teich and F. Fummi, Eds., Florence, Italy, Mar. 2019, pp. 964–967, doi: 10.23919/DATE.2019.8714922.

[34] A. Marchisio, V. Mrazek, M. A. Hanif, and M. Shafique, ‘‘DESCNet: Developing efficient scratchpad memories for capsule network hardware,’’ *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 9, pp. 1768–1781, Sep. 2021, doi: 10.1109/TCAD.2020.3030610.

[35] B. Zoph and Q. V. Le, ‘‘Neural architecture search with reinforcement learning,’’ in *Proc. 5th Int. Conf. Learn. Represent. (ICLR)*, Toulon, France, Apr. 2017, pp. 1–16. [Online]. Available: https://openreview.net/forum?id=r1Ue8Hcxg

[36] D. Stamoulis, R. Ding, D. Wang, D. Lymberopoulos, B. Priyantha, J. Liu, and D. Marculescu, ‘‘Single-path NAS: Designing hardware-efficient convnets in less than 4 hours,’’ in *Machine Learning and Knowledge Discovery in Databases* (Lecture Notes in Computer Science), vol. 11907. Würzburg, Germany: Springer, Sep. 2019, pp. 481–497, doi: 10.1007/978-3-030-46147-8_29.

[37] M. P. Marcus and M. A. Marcinkiewicz, and B. Santorini, ‘‘Building a large annotated corpus of English: The Penn treebank,’’ *Comput. Linguistics*, vol. 19, no. 2, pp. 313–330, 1993. [Online]. Available: https://www.aclweb.org/anthology/J93-2004

[38] L. Sekanina, ‘‘Neural architecture search and hardware accelerator co-search: A survey,’’ *IEEE Access*, vol. 9, pp. 151337–151362, 2021, doi: 10.1109/ACCESS.2021.3126685.

[39] P. Achararit, M. A. Hanif, R. V. W. Putra, M. Shafique, and Y. Hara-Azumi, ‘‘APNAS: Accuracy-and-performance-aware neural architecture search for neural hardware accelerators,’’ *IEEE Access*, vol. 8, pp. 165319–165334, 2020, doi: 10.1109/ACCESS.2020.3022327.

[40] D. Wang, M. Li, C. Gong, and V. Chandra, ‘‘AttentiveNAS: Improving neural architecture search via attentive sampling,’’ in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*. New York, NY, USA: Computer Vision Foundation, Jun. 2021, pp. 6418–6427. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2021/html/Wang_AttentiveNAS_Improving_Neural_Architecture_Search_via_Attentive_Sampling_CVPR_2021_paper.html

[41] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, ‘‘MnasNet: Platform-aware neural architecture search for mobile,’’ in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*. Long Beach, CA, USA: Computer Vision Foundation, Jun. 2019, pp. 2820–2828. [Online]. Available: http://openaccess.thecvf.com/content_CVPR_2019/html/Tan_MnasNet_Platform-Aware_Neural_Architecture_Search_for_Mobile_CVPR_2019_paper.html

[42] Q. Lu, W. Jiang, X. Xu, Y. Shi, and J. Hu, ‘‘On neural architecture search for resource-constrained hardware platforms,’’ *CoRR*, vol. abs/1911.00105, pp. 1–8, Oct. 2019.

[43] W. Jiang, X. Zhang, E. H.-M. Sha, L. Yang, Q. Zhuge, Y. Shi, and J. Hu, ‘‘Accuracy vs. efficiency: Achieving both through FPGA-implementation aware neural architecture search,’’ in *Proc. 56th Annu. Design Autom. Conf.*, Las Vegas, NV, USA, Jun. 2019, p. 5, doi: 10.1145/3316781.3317757.

[44] W. Jiang, L. Yang, S. Dasgupta, J. Hu, and Y. Shi, ‘‘Standing on the shoulders of giants: Hardware and neural architecture co-search with hot start,’’ *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 11, pp. 4154–4165, Nov. 2020, doi: 10.1109/TCAD.2020.3012863.

[45] Z. Guo, X. Zhang, H. Mu, W. Heng, Z. Liu, Y. Wei, and J. Sun, ‘‘Single path one-shot neural architecture search with uniform sampling,’’ in *Computer Vision—ECCV 2020* (Lecture Notes in Computer Science), vol. 12361. Glasgow, U.K.: Springer, Aug. 2020, pp. 544–560, doi: 10.1007/978-3-030-58517-4_32.

[46] L. L. Zhang, Y. Yang, Y. Jiang, W. Zhu, and Y. Liu, ‘‘Hardware-aware one-shot neural architecture search in coordinate ascent framework,’’ *CoRR*, vol. abs/1910.11609, pp. 1–10, Apr. 2019.

[47] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, "FBNet: Hardware-aware efficient convnet design via differentiable neural architecture search," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*. Long Beach, CA, USA: Computer Vision Foundation, Jun. 2019, pp. 10734–10742. [Online]. Available: http://openaccess.thecvf.com/content_CVPR_2019/html/Wu_FBNet_Hardware-Aware_Efficient_ConvNet_Design_via_Differentiable_Neural_Architecture_Search_CVPR_2019_paper.html

[48] S. Kotyan and D. V. Vargas, "Evolving robust neural architectures to defend from adversarial attacks," in *Proc. Workshop Artif. Intell. Saf., 29th Int. Joint Conf. Artif. Intell., 17th Pacific Rim Int. Conf. Artif. Intell. (IJCAI-PRICAI)*, vol. 2640, H. Espinoza, J. McDermid, X. Huang, M. Castillo-Effen, X. C. Chen, J. Hernández-Orallo, S. Ó. Héigeartaigh, and R. Mallah, Eds., Yokohama, Japan, Jan. 2021, pp. 1–8. [Online]. Available: http://ceur-ws.org/Vol-2640/paper_1.pdf

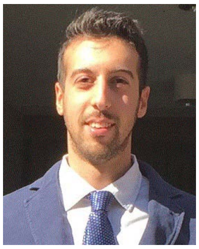
[49] A. Marchisio, A. Massa, V. Mrazek, B. Bussolino, M. Martina, and M. Shafique, "NASCaps: A framework for neural architecture search to optimize the accuracy and hardware efficiency of convolutional capsule networks," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, San Diego, CA, USA, Nov. 2020, pp. 114:1–114:9, doi: 10.1145/3400302.3415731.

[50] K. Pearson, "Note on regression and inheritance in the case of two parents," *Proc. Roy. Soc. London*, vol. 58, pp. 240–242, Jan. 1895. [Online]. Available: http://www.jstor.org/stable/115794

[51] N. P. Jouppi et al., "In-datacenter performance analysis of a tensor processing unit," in *Proc. 44th Annu. Int. Symp. Comput. Archit. (ISCA)*, Toronto, ON, Canada, Jun. 2017, pp. 1–12, doi: 10.1145/3079856.3080246.

[52] S. Li, K. Chen, J. H. Ahn, J. B. Brockman, and N. P. Jouppi, "CACTI-P: Architecture-level modeling for SRAM-based structures with advanced leakage reduction techniques," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*. San Jose, CA, USA: IEEE Computer Society, Nov. 2011, pp. 694–701, doi: 10.1109/ICCAD.2011.6105405.

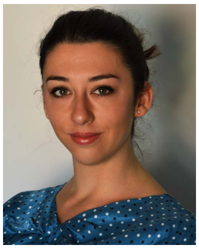
[53] I. J. Goodfellow, N. Papernot, and P. D. McDaniel, "cleverhans V0.1: An adversarial machine learning library," *CoRR*, vol. abs/1610.00768, pp. 1–5, Oct. 2016.



ALBERTO MARCHISIO (Graduate Student Member, IEEE) received the B.Sc. and M.Sc. degrees in electronic engineering from the Politecnico di Torino, Turin, Italy, in October 2015 and April 2018, respectively. He is currently pursuing the Ph.D. degree with the Computer Architecture and Robust Energy-Efficient Technologies (CARE-Tech.) Laboratory, Institute of Computer Engineering, Technische Universität Wien (TU Wien), Vienna, Austria, under the supervision of Dr. Muhammad Shafique. He has coauthored more than 20 papers in prestigious international conferences and journals. His research interests include hardware and software optimizations for machine learning, brain-inspired computing, VLSI architecture design, emerging computing technologies, robust design, and approximate computing for energy efficiency. He received the honorable mention at the Italian National Finals of Maths Olympic Games, in 2012, and the Richard Newton Young Fellow Award, in 2019.

VOJTECH MRAZEK (Member, IEEE) received the Ing. and Ph.D. degrees in information technology from the Faculty of Information Technology, Brno University of Technology, Czech Republic, in 2014 and 2018, respectively. He was a Visiting Postdoctoral Researcher at the Department of Informatics, Institute of Computer Engineering, Technische Universität Wien (TU Wien), Vienna, Austria. He is currently an Assistant Professor at the Evolvable Hardware Group, Faculty of Information Technology. His research interests include approximate computing, genetic programming, and machine learning. He has authored or coauthored over 40 conference/journal articles focused on approximate computing and evolvable hardware. He received several awards for his research in approximate computing, including the Joseph Fourier Award, in 2018, for research in computer science and engineering.

ANDREA MASSA received the B.Sc. degree in electrical and electronic engineering from the University of Cagliari, Italy, in October 2017, and the M.Sc. degree in electronic engineering from the Politecnico di Torino, Turin, Italy, in July 2020. He is currently a Hardware and Software Designer at Computer Company, Turin. His research interests include the fields of image processing, computer architecture, machine learning, deep neural networks, and genetic algorithms. His previous research work focused on image processing techniques and image registration algorithms applied to real-time automatic camera calibration in the W7-X fusion reactor at the Max Planck Institute for Plasma Physics, Greifswald, Germany, and it was carried out within the activities of the University of Cagliari in different EUROfusion Tasks and Grants. In 2020, his M.Sc. thesis work focused on the development of a hardware-aware neural architecture search framework which resulted into a paper that has been accepted for the IEEE/ACM International Conference on Computer-Aided Design (ICCAD 2020).



BEATRICE BUSSOLINO (Member, IEEE) received the B.Sc. and M.Sc. degrees in electronic engineering from the Politecnico di Torino, Turin, Italy, in October 2017 and October 2019, respectively, where she is currently pursuing the Ph.D. degree in electrical, electronics and communications engineering under the supervision of Prof. Maurizio Martina. Her current research interests include the field of machine learning and deep neural networks (DNNs) in particular. The focus of her research activity is the development of on-chip architectures for the edge deployment of DNNs. In 2020, she received the Richard Newton Young Fellow Award and won the DAC Young Fellow Poster Presentation Award.

MAURIZIO MARTINA (Senior Member, IEEE) received the M.S. and Ph.D. degrees in electrical engineering from the Politecnico di Torino, Italy, in 2000 and 2004, respectively. He is currently a Full Professor with the VLSI-Laboratory Group, Politecnico di Torino. His research interests include computer architecture and VLSI design of architectures for digital signal processing, video coding, communications, networking, artificial intelligence, machine learning, and event-based processing. He edited one book and published three book chapters on VLSI architectures and digital circuits for video coding, wireless communications, and error correcting codes. He has more than 100 scientific publications and is the coauthor of two patents. He is currently an Associate Editor of IEEE Transactions on Circuits and Systems—I: Regular Papers. He had been part of an Organizing and Technical Committee of several international conferences, including BioCAS 2017, ICECS 2019, and AICAS 2020. Currently, he is the Counselor of the IEEE Student Branch at the Politecnico di Torino and a Professional Member of IEEE HKN.

MUHAMMAD SHAFIQUE (Senior Member, IEEE) received the Ph.D. degree in computer science from the Karlsruhe Institute of Technology (KIT), Germany, in 2011. Afterwards, he established and led a highly recognized research group at KIT for several years and conducted impactful collaborative research and development activities across the globe. In October 2016, he joined the Faculty of Informatics, Institute of Computer Engineering, Technische Universität Wien (TU Wien), Vienna, Austria, as a Full Professor in computer architecture and robust, energy-efficient technologies. Since September 2020, he has been with New York University (NYU) Abu Dhabi, United Arab Emirates, where he is currently a Full Professor and the Director of the eBrain Laboratory and a Global Network Professor at the Tandon School of Engineering, NYU-New York City, USA. He is also a Co-PI/an Investigator in multiple NYUAD Centers, including the Center of Artificial Intelligence and Robotics (CAIR), the Center of Cyber Security (CCS), the Center for InTeractIng urban nEtworkS (CITIES), and the Center for Quantum and Topological Systems (CQTS). His research interests include AI & machine learning hardware and system-level design, brain-inspired computing, quantum machine learning, cognitive autonomous systems, wearable healthcare, energy-efficient systems, robust computing, hardware security, emerging technologies, FPGAs, MPSoCs, and embedded systems. His research has a special focus on cross-layer analysis, modeling, design, and optimization of computing and memory systems. The researched technologies and tools are deployed in application use cases from Internet of Things (IoT), smart cyber-physical systems (CPS), and ICT for development (ICT4D) domains. He has given several keynotes, invited talks, tutorials, and organized many special sessions at premier venues. He has served as the PC chair, the general chair, the track chair, and a PC member for several prestigious IEEE/ACM conferences. He holds one U.S. patent, has coauthored six books, more than ten book chapters, more than 350 papers in premier journals and conferences, and more than 50 archive articles. He received the 2015 ACM/SIGDA Outstanding New Faculty Award, the AI 2000 Chip Technology Most Influential Scholar Award, in 2020 and 2022, the ASPIRE AARE Research Excellence Award, in 2021, six gold medals, and several best paper awards and nominations at prestigious conferences. He is a Senior Member of IEEE Signal Processing Society (SPS) and a member of the ACM, SIGARCH, SIGDA, SIGBED, and HIPEAC.