The Larger The Fairer? Small Neural Networks Can Achieve Fairness for Edge Devices

Yi Sheng
George Mason University

Junhuan Yang
The University of New Mexico

Yiyu Shi
University of Notre Dame

Jingtong Hu
University of Pittsburgh

Yawen Wu
University of Pittsburgh

Kevin Mao
University of Pittsburgh

Weiwen Jiang
George Mason University

Lei Yang
The University of New Mexico

ABSTRACT

Along with the progress of AI democratization, neural networks are being deployed more frequently in edge devices for a wide range of applications. Fairness concerns gradually emerge in many applications, such as face recognition and mobile medical. One fundamental question arises: what will be the fairest neural architecture for edge devices? By examining the existing neural networks, we observe that larger networks typically are fairer. But, edge devices call for smaller neural architectures to meet hardware specifications.

To address these challenges, our proposed NAS framework leverages a dedicated designed reward function to balance fairness, accuracy, and hardware efficiency. Furthermore, we observed that the front layers (header) of neural networks will not affect fairness but only extract common features; while the intermediate feature maps in the end layers (tail) are quite different. Based on this observation, we develop a freezing method to accelerate the optimization without affecting the fairness. As a result, the training parameters and training time can be reduced, together with the reduction in the search space. The main contributions of this paper are as follows.

1 INTRODUCTION

With the continuous progress of AI democratization, we have witnessed the breakthrough of deep learning models deployed in the edge and mobile devices for AI applications, like mobile dermatology assistant [1], mobile eye cancer detection [2], comprehensive vital signs monitoring [3], and medical imaging and diagnostics [4]. To implement these models efficiently on devices, various model compression, accelerator design, and hardware/software co-design techniques [5–13] have been proposed to achieve both high accuracy and efficiency. Unfortunately, most of the existing AI system designs only pursue high overall accuracy and ignore fairness among diverse groups in the dataset. For example, [14] has pointed out the gender and skin-type bias in commercial AI systems. Examination of facial-analysis software shows an error rate of 0.8% for light-skinned men, 34.7% for dark-skinned women; [15] also pointed out similar racial disparity for Skin Image Search, which is an AI app that helps people identify skin conditions. It reports 70% accuracy for the whole dataset, but only 17% for dark skins.

Research efforts have been made in addressing the fairness issue [16]. However, they either focus on the model interpretability by modifying the neural network models to be fairer [17], or fairness-aware data collection [18]. While these works make important initial steps, achieving fairness on resource-constrained edge devices brings new challenges: neural networks need to be small enough to accommodate limited computation power and memory/storage space. However, as shown in Figure 1, we observed that larger neural network models generally have higher fairness, where the “unfairness score” is defined as the variation of the prediction accuracy among the diverse groups. Thereby, a fundamental question we are trying to answer is: can we identify small and fair neural networks to meet the hardware specifications? What is more, traditional methods manually fine-tune the models to achieve better fairness. In this work, we are trying to achieve fairness through automatic neural architecture search (NAS).

Although there have been various NAS frameworks [19–28], none of them have considered fairness as a goal. In this paper, we propose a novel “Fairness- and Hardware-aware NAS” framework, namely FaHaNa, to address these challenges. It integrates fairness as a part of the objective in a reinforcement learning (RL) based optimization process. Given a target hardware platform and a training dataset with diverse groups, FaHaNa searches for the neural architectures with the highest accuracy and the best fairness. Meanwhile, the latency can be guaranteed to meet the hardware specifications. To ensure fairness awareness, it seems straightforward to simply include a fairness metric together with accuracy to the existing NAS frameworks. However, this extra fairness metric can easily pull down good candidates (in terms of accuracy) in the search space, since they usually achieve high accuracy by catering to the majority group. Therefore, the NAS framework needs to ensure high fairness for diverse groups, while not compromising overall accuracy. In addition, NAS itself is known for lengthy search times.

To address these challenges, our proposed NAS framework leverages a dedicated designed reward function to balance fairness, accuracy, and hardware efficiency. Furthermore, we observed that the front layers (header) of neural networks will not affect fairness but only extract common features; while the intermediate feature maps in the end layers (tail) are quite different. Based on this observation, we develop a freezing method to accelerate the optimization without affecting the fairness. As a result, the training parameters and training time can be reduced, together with the reduction in the search space. The main contributions of this paper are as follows.

Figure 1: Fairness vs. model size on the existing neural networks: (a) larger networks within the same series have higher fairness; (b) increasing fairness along with larger models.
• **Framework.** To the best of our knowledge, FaHaNa is the first fairness-aware framework to explore fair neural architectures, which can further generate the optimal DNN architectures with the guaranteed latency on target hardware.

• **Acceleration.** We propose a freezing method to fix a part of the neural architecture and make use of the pre-trained parameters for common feature extraction, which significantly improves search efficiency without affecting the fairness.

• **Evaluation.** We have conducted a case study on medical AI (i.e., dermatological disease diagnosis) to evaluate FaHaNa. A dermatology dataset, including images with light skin (majority) and dark skin (minority), is built for evaluation.

Experimental results on the dermatology dataset evaluate the effectiveness of FaHaNa and the efficiency of the freezing method to accelerate the optimization process. First, compared with MnasNet, the network identified by FaHaNa (FaHaNa-Nets) can reduce the unfairness score from 0.4521 to 0.1973, meanwhile achieving 3.16% overall accuracy gain, 2.24× smaller model size, 2.11× and 3.15× latency reduction on Raspberry PI and Ondroid XU-4. Compared with a larger but fair model, MobileNetV2, FaHaNa-Nets can achieve 15.14% higher fairness and 0.23% higher accuracy, and the reductions of model size and latency are increased to 5.28×, 5.75×, and 5.79×. Second, the freezing method is effective to better explore the design space, reducing the search space from 10^{19} to 10^9 and accelerates the search process with 2.67× speedup. Last but not the least, FaHaNa is compatible with existing fairness techniques [18].

In the rest of the paper: Section 2 reviews the related background and provides the motivations; Section 3 defines the problem and presents our FaHaNa framework. Experimental results are shown in Section 4 and concluding remarks are given in Section 5.

## 2 RELATED WORK AND MOTIVATION

This section will provide our observations on the effects of neural architectures on fairness and review the related works.

**Observation 1:** Neural architectures affect fairness.

On the dermatology dataset, Figure 2 shows the unfairness score of different sets of neural architectures, including MobileNet, MnasNet, ProxylessNAS, and ResNet. The green bars and white bars represent the prediction accuracy of the majority (light skins) and minority (dark skins) in the dataset, respectively. The blue line shows the unfairness score on all models, which describes the variance in accuracy between the majority and minority groups. More specifically, the unfairness score varies from 0.4521 (MnasNet 0.5) to 0.1820 (ResNet-18) as reported in the figure. Results demonstrated that all these models have prejudice on the majority models, and each model has better fairness than its left-hand ones.

**Motivation 1:** Searching for a fair neural architecture.

The straightforward and commonly applied approach to address fairness issue is to balance data between the majority and minority groups [18] or learn fair representations between the protected and unprotected features [29]. However, there exists an inherent imbalance since data from the minority groups may not be easily collected due to objective reasons (e.g., a lack of medical professionals from marginalized communities). What’s worse, neural architecture acts as an equal or even more important role in fairness, and the effects of different network models may outweigh that by data balancing. Results in Figure 1(b) show the unfairness of different neural architectures on the training datasets with different amounts of minority data. We observe that even MnasNet 0.5 is trained on a dataset with 5× minority data (i.e., diamond for the smallest model), its unfairness score is still higher than ResNet-18 (0.2280 vs. 0.1820). This emphasizes the effects of the neural architecture on fairness and motivates us to conduct the fairness-aware architecture search.

**Observation 2:** Hardware specification affects fairness.

Table 1 reports the accuracy, unfairness score, and hardware performance of different neural network models. We run these models on Raspberry PI with a timing constraint of 1500ms. With such a hardware constraint, only SqueezeNet 1.0, MobileNetV3, and MnasNet 0.5 can meet the specification; however, the unfairness scores of MnasNet 0.5 and MobileNetV3 are 0.2196 and 0.0928 less than MobileNetV2’s score. Nevertheless, its latency violates the requirement. SqueezeNet 1.0 is much fairer, but its accuracy is as low as 15.65%. These results clearly demonstrate that fairness cannot be considered separately from hardware specifications.

**Motivation 2:** Making tradeoffs among fairness, accuracy, and hardware efficiency.

Fairness, accuracy, and hardware efficiency are equally important in edge AI applications, like medical AI [4, 30]. Losing any one of these characteristics will render the architecture useless (e.g., SqueezeNet has low accuracy, MobileNetV2 violates latency, and MnasNet 0.5 is less fair). Holistic optimization should be conducted on all these metrics.

Neural architecture search (NAS) methods have been developed to automatically identify neural architectures for maximum accuracy [19]. Together with the consideration of the hardware specifications, hardware-aware NAS [20, 21, 31] further explore the hardware design space, thus jointly identifying the best architecture and hardware designs. Decoupled from hardware, the multi-objective NAS (MONAS) [32] was proposed. Nevertheless, there is still a lack of NAS considering the fairness in the design objective. Straightforwardly integrating fairness into MONAS will reduce the reward of models with high accuracy but low fairness, and make the discrimination among models to be vague. As such, it potentially prolongs the search process for convergence. Furthermore, NAS itself is known for its lengthy search time. Therefore, a more efficient way for fairness-aware NAS is highly demanded.

**Observation 3:** Fairness is mostly affected by the tail.

---

**Table 1:** Different models with less than 30MB storage size running on Raspberry PI with timing constraint of 1500ms

| Model          | Latency (ms) | Storage (MB) | Accuracy | Unfairness Score | Meet Spec. |
|----------------|--------------|---------------|----------|------------------|------------|
| SqueezeNet 1.0 | 122.92       | 2.77          | 15.65%   | 0.2139           | ✓          |
| MobileNetV3    | 658.84       | 5.81          | 80.38%   | 0.3253           | ✓          |
| MnasNet 0.5    | 714.19       | 3.60          | 78.12%   | 0.4521           | ✓          |
| MobileNetV2    | 1939.40      | 8.51          | 81.05%   | 0.2325           | ×          |
| ProxylessNAS(G)| 3714.44      | 20.60         | 83.21%   | 0.2667           | ×          |
| MnasNet 1.0    | 3855.72      | 11.86         | 80.71%   | 0.2913           | ×          |
| ProxylessNAS(M)| 5241.51      | 10.70         | 81.27%   | 0.3094           | ×          |
To figure out how to accelerate the NAS process, we further investigated: how do the neural networks make different predictions for the minority or majority groups? Toward this, we compare the variation of intermediate features obtained by different groups after each layer in MobileNetV2. Results in Figure 3 show that the front layers (say before layer 12) have small variations. The visualizations of features after layer 2 and layer 13 are illustrated in Figure 3, where each row represents the intermediate feature corresponding to one input data, and the column corresponds to a specific neuron. Visualized pictures show that layer 2 has small variation because it has similar patterns in features from different groups, while layer 13 has different patterns. More sets of experiments on other networks have been conducted, and we obtain the same observation.

**Motivation 3:** Freezing the head and searching for the tail.

The above results demonstrate that the distinctions of groups are mainly contributed by the front layers; in other words, the front layer(s) extracts the common features which will not affect fairness. Based on the observation, we are inspired to freeze the header in the search process, and only search for the architecture of the tail.

### 3 FAHANA: PUT FAIRNESS, HARDWARE, NAS IN A HOLISTIC OPTIMIZATION LOOP

#### 3.1 Problem Definition

In this work, we study the fairness issue on the classification task in computer vision. This section will formally define the problem of "fairness-hardware-architectural co-optimization".

**Classification.** Given a dataset $D$, we define $C = \{c_1, c_2, \cdots, c_M\}$ as a set of $M$ classes, where each data $d_i \in D$ belongs to a class $c_j \in C$. That is, there exists a mapping function $f: f(d_i) = c_j$. A neural network $N$ is to build the mapping function from $D$ to $C$. On top of a training dataset, $N$ will learn a function $f'_N$ to approximate $f$. If $f(d_i) = f'_N(d_i)$, it is a correct prediction on data $d_i$; otherwise, it is an incorrect prediction. The accuracy $A(f'_N, D)$ describes the ratio of data in $D$ getting the correct prediction using model $N$.

**Diverse Groups.** For each data $d_i \in D$, in addition to its category feature (i.e., $C$), it may also have other inherent features, like the skin-color, race, sex, etc. For an inherent feature $I$, it can divide $D$ into $K$ groups: $D = \{D_{g_1}, D_{g_2}, \cdots, D_{g_K}\}$. The take the feature of skin color as an example, it can divide $D$ to 2 groups: light skin ($g_1$ = light) and dark skin ($g_2$ = dark). If the number of data in $D_{g_1}$ is less than that in $D_{g_2}$, i.e., $|D_{g_1}| < |D_{g_2}|$, then we call $D_{g_1}$ (e.g., dark skin) minority group in comparison with $D_{g_2}$ (e.g., light skin). Kindly note that the proposed method can support fairness for more than 2 diverse groups.

**Fairness.** For a model $N$ on data group $D_{g_k}$, its accuracy is $A(f'_N, D_{g_k})$. Based on the accuracy of all groups, we define the unfairness score $U$ of a model $N$ on dataset $D$ as based on L1-norm, which is $U(f'_N, D) = \sum_{g_k \in C} |A(f'_N, D_{g_k}) - A(f'_N, D)|$.

**Specification.** The specification contains two parts: software specification and hardware specification. The software specification is the requirement for prediction accuracy. Given an accuracy constraint $AC$, it requires the model $N$ to achieve accuracy $A(f'_N, D) \geq AC$. As to the hardware specification, we will be given a hardware device $H$ (e.g., Raspberry PI, a mobile phone, etc.), with the timing constraint $TC$. $L(H, N)$ represents the inference latency of running neural network $N$ on $H$. The hardware specification $S$ sets up hardware performance requirements, such as $L(H, N) \leq TC$.

**Problem Definition.** With the above definitions, we can formally define the "fairness-hardware-architectural co-optimization problem" as follows: Given a dataset $D$ with $M$ classes and an inherent feature $I$ dividing $D$ into $K$ groups, a hardware $H$, design specifications (e.g., timing constraint $TC$ and accuracy constraint $AC$), our objective is to automatically generate a neural architecture $N$, such that the accuracy $A(f'_N, D)$ can be maximized and the unfairness score $U(f'_N, D)$ can be minimized; meanwhile, accuracy $A(f'_N, D)$ and latency $L(H, N)$ can meet the design specifications.

#### 3.2 FAHAna Overview

**FAHAna Overview:** Figure 4 illustrates the overview of our proposed FAHAna framework. It is composed of four components: ➀ a recurrent neural network (RNN) based controller, ➋ a block-based search space, ➃ backbone architecture producer, ➄ performance evaluator and trainer. Specifically, ➀ the controller will guide the optimization process. From ➋ block-based search space, it will identify the searchable block in the backbone architecture (obtained by ➃ producer) to form a neural network $N$ (a.k.a., child network). Then, $N$ will be sent to the ➄ trainer to learn the function $f'_N$. It will be used for the inference on dataset $D$ and sub-group of $\{D_{g_1}, D_{g_2}, \cdots, D_{g_K}\}$ to obtain the accuracy $A(f'_N, D)$ and unfairness score $U(f'_N, D)$, respectively. Simultaneously, ➄ evaluator will get the latency $L(H, N)$ of $N$ on the given hardware $H$. Finally, a reward will be generated to update RNN in the controller. In the following section, we will introduce these components in detail.
The controller will iteratively predict the hyperparameters of a child network. In each iteration (a.k.a., episode), the controller will receive a reward to update the RNN network. The reward is generated based on the accuracy $A(f_{N_1}, D)$, unfairness score $U(f_{N_1}, D)$, and latency $L(H, N)$ (details see ② Evaluator and Trainer), which is formulated as below.

$$R = \alpha \cdot A(f_{N_1}, D) - \beta \cdot U(f_{N_1}, D) - 1$$

where $\alpha$, $\beta$ are two scaling factors that could be adjusted according to the specific demands on accuracy or fairness. Based on the reward, we employ reinforcement learning to update the controller. Specifically, we apply Monte Carlo policy gradient algorithm [33]:

$$\nabla J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \sum_{t=1}^{T} \gamma^{t-1} \nabla \theta \log \pi(a_t|a_{t-1}) (R_k - b)$$

where $m$ is the batch size and $T$ is the number of steps in each episode. Rewards are discounted by an exponential factor $\gamma$ and the baseline $b$ is the average exponential moving of rewards.

② Search Space: As shown in Figure 4, the search space is based on different basic computation blocks. Motivated by existing neural networks with the highest fairness (i.e., MobileNetV2 and ResNet-18 from Figure 2), we consider 4 types of basic blocks (MB, DB, RB, and CB). MB and DB are based on MobileNetV2 blocks with stride = 2 and stride = 1, respectively; RB is based on ResNet blocks; we also include CB based on the conventional convolution operation. All these 4 blocks have the same hyperparameters: channel numbers (CH1, CH2, and CH3) and kernel sizes (K). Kindly note that CH1 of one block b is determined by CH3 of block b’s precedence, while K, CH2 and CH3 are searchable. We also enable the skip operation in a block to make the flexibility on the depth of the neural network.

③ Backbone Architecture Producer: In the conventional NAS, each layer/block in the backbone architecture is searchable; however, motivated by Observation 3 in Section 2, we develop a producer to freeze the header of the backbone architecture. The challenge here is how to determine the blocks to be frozen or not, as shown in Figure 4 ③. To address this, the producer conducts 3 steps to determine the frozen blocks for a given backbone architecture.

First, a batch of minority data and majority data are streamed into a pre-trained backbone architecture, and we keep the feature maps in between layers. The second step compares the feature maps among all groups to obtain the feature variation using the L2-norm. Third, we fix a threshold $T$ by multiplying the maximum variation of all layers and a scaling factor $\gamma$, then search for the foremost layer L whose feature variation exceeds the threshold $T$. This is the splitting point, where all layers before L belong to the frozen blocks, while the rests (include L) belong to the searchable blocks.

For frozen blocks, in the optimization process, we will directly use the pre-trained parameters (i.e., weights) without training. Kindly note that in order to reduce the model size to meet the timing constraint, we can further replace the first layers as a convolution layer (which can be trained) and connect them with the frozen blocks to extract common features. For each searchable block, ① controller will determine a set of hyperparameters, including block type, from ② search space. After that, the producer will generate a child network $N$ for ④ evaluator and trainer. Experimental results will show the effectiveness of the proposed freezing method.

④ Evaluator and Trainer: After a child network is generated, it will be processed by the evaluator and trainer. The basic design concept is to accelerate the search process. To achieve this goal, we will first check whether the hardware specification can be met. If not, it will bypass the lengthy training procedure and directly generate the reward as -1 (see Equation 1). To further accelerate the evaluation and enable the automation of the optimization, we will test the performance of each block offline on the given hardware device $H$, based on which we can efficiently estimate the latency during the search process. For the finally identified neural network architecture, we will perform an end-to-end evaluation on the target devices.

If the hardware specification can be met, the searchable blocks in the child network $N$ will be trained to learn a function $f_{N_1}$ for dataset $D$. Then, the trained model will be applied to dataset $D$ and subgroups in $D$ to obtain the model accuracy $A(f_{N_1}, D)$ and the unfairness score $U(f_{N_1}, D)$.

4 EXPERIMENTS

FaHaNa, which has the demand to be run on mobile phones, is evaluated on a dermatology dataset for diagnosing the dermatological disease. Therefore, we apply two edge devices, Raspberry PI and Odroid XU-4, as our testbed. Results show that FaHaNa can improve the fairness without compromising accuracy, meanwhile, reducing the model size.

4.1. Experimental Setup

A. Dataset: A dermatology dataset is built based on patients’ images collected in the field and the open-access datasets including ISIC 2019 [34] for light-skin, Dermnet [35], and Atlas dermatology [36] for dark-skin. These images are utilized for a classification task with 5 dermatology diseases: Melanoma, Melanocytic nevus, Basal cell carcinoma, Dermatofibroma, and Squamous cell carcinoma.

B. FaHaNa settings: In the evaluation, both parameters $\alpha$ and $\beta$ of the RNN controller (Figure 4 ③) are set to be 1 with the objective to find a neural architecture with balanced accuracy and fairness. The ③ Producer takes MobileNetV2 as the backbone architecture; parameter $\gamma$ is set to 0.5 to select the frozen blocks. During the search process, we split the dataset into three sets: (1) training set with 60% images; (2) validation set with 20% images; and (3) test set with the rest 20% images. The number of episodes for reinforcement learning is set to 500. Finally, a series of neural architectures will be identified, denoted as FaHaNa-Nets.

C. Competitors and training settings: To evaluate FaHaNa-Nets, we select a set of state-of-the-art neural networks for comparison, including (1) the manually designed MobileNetV2 [37] and ResNet [38], and (2) the AutoML identified MobileNetV3 [39], ProxylessNAS [21] and MnasNet [20]. For a fair comparison, all the neural networks, including FaHaNa-Nets, are trained from scratch with the same hyperparameters on a GPU cluster with 48 RTX 3080: (1) learning rate starts from 0.1 with a decay of 0.9 in 20 steps, (2) 32 for the batch size, and (3) 500 epochs for training. In addition, we employ the multi-objective NAS (denoted as MONAS) [32] to evaluate the efficiency of the FaHaNa framework.

D. Edge devices: To compare the inference latency of FaHaNa-Nets and competitors, we employ two kinds of edge devices: (1) Raspberry PI Model B [40] with Broadcom BCM2711 equipping a 1.5 GHz quad-core ARM Cortex-A72 processor and 8 GB memory, and (2) Odroid XU-4 [41] with a Samsung Exynos 5422 equipping ARM Cortex-A15 and Cortex-A7 quad-core processor and 2 GB memory. The latency is obtained by deploying the trained models on both devices for inference using a vanilla PyTorch framework.
The Larger The Fairer? Small Neural Networks Can Achieve Fairness for Edge Devices

DAC, Design Automation Conference, 2022

4.2 Exploration by FaHaNa

In the first set of experiments, we demonstrate that FaHaNa-Nets can significantly push forward the Pareto frontiers among fairness, accuracy, and model size, compared with the competitors. The efficacy of FaHaNa’s search engine is also evaluated.

A. Best reward vs. model size: Figure 5 reports the design space exploration results. In Figure 5(a), the x-axis is the model size (i.e., number of parameters), and the y-axis is the reward calculated based on Equation 1. The ideal solution is located in the left corner, denoted as a star. For a clear demonstration, we only plot the architectures with less than 6M parameters.

In Figure 5(e), each circled point corresponds to a FaHaNa-Net and each green diamond is related to an existing network. The red and green lines plot the Pareto frontiers of FaHaNa-Nets and existing networks, respectively. From this figure, we observe that FaHaNa-Small on the left-top corner dominates all the existing neural networks in terms of reward and model size; while FaHaNa-Fair on the right-top corner achieves the highest fairness. These figures clearly show that FaHaNa can significantly push forward the Pareto frontiers in the reward and model size tradeoff.

B. Accuracy vs. Unfairness. We further investigate the Pareto frontier between fairness and accuracy by decomposing the reward in Figure 5(a). Results in Figure 5(b) consistently show that FaHaNa can push forward the Pareto frontier compared with the existing neural networks. More specifically, FaHaNa-Fair is the architecture that is the closest to the ideal solution. On the other hand, even FaHaNa-small has the smallest size, it can still dominate most of the existing neural architectures. These two architectures will be used for further detailed comparison.

C. Space and time. The efficiency and effectiveness of the freezing method are evaluated by comparing MONAS (with fairness added as one objective). We compare the search space and search time in Table 2. Two sets of experiments are carried out using a tight timing constraint and a relaxed timing constraint. Columns “Valid” show the ratio of the valid architectures (i.e., the reward is not equal to -1, see Equation 1) examined during NAS process.

There are several observations in Table 2. First, FaHaNa can significantly reduce the search space, from $10^{19}$ to $10^9$, compared with MONAS. Second, benefiting from the reduced search space, FaHaNa can search for more valid architectures. With the same number of episodes, the validation rates of MONAS and FaHaNa are increased from 27.50% to 71.05% and from 33.33% to 95.23% under tight TC and relaxed TC, respectively. This is because the freezing method can prune a lot of invalid neural architectures. Third, even with a higher validation rate (more architectures need to be trained), FaHaNa can still achieve 1.83× and 2.67× speedup. This is because the freezing method can reduce the number of parameters to accelerate the network training process. Overall, FaHaNa can shrink the search space to examine more valid networks for high-reward architectures; meanwhile, the search time can be significantly reduced.

4.3 FaHaNa-Nets vs. Existing Neural Architectures

Next, we compare FaHaNa-Nets against competitors with a given accuracy constraint (AC). We divide all neural architectures into two groups in terms of model size. Group G1 contains the small-size architectures with less than 4M parameters; other architectures belong to group G2. We select the architecture with the highest fairness from all the competitors in each group as the baseline: MobileNetV2 for G1 and ResNet-50 for G2. Table 3 reports the fairest trained model for each architecture, which is expected to meet a preset AC: 81% for G1 and 83% for G2. If the architecture cannot meet AC then we select the model with the highest accuracy for comparison. The parallel lines divide Table 3 into two parts: software metrics (left) and hardware metrics (right).

A. FaHaNa-Small has the smallest size and lowest latency: From Table 3, we have several observations. First, only MobileNetV2, ProxylessNAS(M), and FaHaNa-Small meet the AC of 81%. Second, FaHaNa-Small is the fairest architecture in G1. Compared with the baseline, MobileNetV2 with a 0.3252 unfairness score, MobileNetV2 can get 0.1973 which has a 15.14% improvement. Compared with other architectures, the unfairness improvement of FaHaNa-Small can reach up to 56.34% (i.e., MnasNet 0.5). Third, FaHaNa-Small has the minimum number of parameters; thus, it has the best hardware performance: 1.61M storage, 337.3ms latency on Raspberry Pi, and 736.22ms latency on Odriod. Compared with the baseline, it achieves 5.28× storage reduction, as well as 5.75× and 5.79× speedup on Raspberry Pi and Odriod respectively. These results, in response to our initial question, verified we can find a small neural network to achieve fairness for edge devices.

B. FaHaNa-Fair can achieve the highest fairness: FaHaNa-Fair is the fairest model in all competitors. Similar to the results in G1, FaHaNa-Fair achieves the lowest unfairness score in G2, 0.1755, compared with 0.1855 obtained by the baseline architecture ResNet-50. In addition, FaHaNa-Fair is 4.27× smaller than ResNet-50, achieving 1.75× and 3.14× speedup on edge devices.

C. Pareto frontier: Figure 6 further shows the comparison of Pareto frontiers in terms of the accuracy-unfairness tradeoff built by all models. Figure 6(a) and Figure 6(b) show the results of the models in G1 and G2, respectively. The stars in these figures refer to the ideal solutions. In Figure 6(a), the red points form the Pareto frontier of models with size < 4M; while the green points form the Pareto frontier of models with size ≥ 4M. The figures clearly show that FaHaNa can significantly push forward the Pareto frontiers among fairness and accuracy by decomposing the reward.

![Figure 5: Comparison between the existing neural networks and FaHaNa-Nets with the highest reward.](image)

![Figure 6: Pareto frontiers of the existing models, FaHaNa-Small, and FaHaNa-Fair in terms of accuracy and unfairness.](image)
Table 3: Comparison of the existing models and FaHaNa-Nets: Group 1 includes models with limited size (# of parameters) to be less than 4M and has the accuracy requirement of 81%; Group 2 includes larger models with the accuracy requirement of 83%.

| Group | Model         | # of Para. | Acc. Meet Acc. | Light | Dark | Unfairness Score | Fairness Comp. | Reward | Storage (MB) | Red. | Latency (ms) |
|-------|---------------|------------|----------------|-------|------|------------------|----------------|--------|---------------|------|--------------|
| G1    | MobileNetV2   | 2,230,277  | 81.05%         | ✓     |      | 81.27%           | 58.02%         | 0.2525 | 8.51 baseline | 1939.40 baseline | 2/246.53 baseline |
|       | ProxylessNAS(M) | 2,905,917  | 81.27%         | ✓     |      | 81.56%           | 50.62%         | 0.3094 | 70.70 0.79x   | 3241.51 0.37x   | 8784.53 0.49x   |
|       | MnasNet 0.5   | 943,917    | 78.12%         | ✓     |      | 78.54%           | 33.33%         | 0.4521 | 7.60 2.36x    | 714.19 2.72x   | 2312.05 1.84x   |
|       | MobileNetV3(S) | 1,522,981  | 80.38%         | ×     |      | 80.68%           | 18.15%         | 0.3253 | 5.81 1.46x    | 658.84 2.94x   | 1954.14 2.18x   |
|       | FaHaNa-Small  | 3,108,717  | 80.71%         | ×     |      | 80.98%           | 51.58%         | 0.2913 | 11.86 0.72x   | 3855.72 0.50x   | 7033.29 0.61x   |
| G2    | ResNet-50     | 25,518,277 | 83.81%         | ✓     |      | 83.98%           | 65.43%         | 0.1855 | 89.72 baseline | 1063.61 baseline | 5790.42 baseline |
|       | ResNet-18     | 11,179,077 | 83.08%         | ✓     |      | 83.28%           | 61.73%         | 0.2155 | 42.64 2.10x   | 425.90 2.50x   | 2029.22 2.03x   |
|       | ResNet-34     | 21,287,237 | 83.01%         | ✓     |      | 83.23%           | 59.26%         | 0.2397 | 81.20 1.10x   | 621.87 1.71x   | 2829.22 2.03x   |
|       | ProxylessNAS(G) | 5,399,493  | 83.21%         | ✓     |      | 83.46%           | 56.79%         | 0.2667 | 20.60 4.36x   | 3714.44 0.29x   | 9426.17 0.61x   |
|       | MobileNetV3(L) | 4,208,437  | 79.58%         | ×     |      | 80.00%           | 34.57%         | 0.4543 | 16.05 5.59x   | 2668.00 0.40x   | 4824.40 1.91x   |
|       | FaHaNa-Fair   | 5,502,469  | 84.06%         | ✓     |      | 84.22%           | 66.67%         | 0.1755 | 20.99 4.27x   | 606.80 1.75x   | 1833.76 3.14x   |

Table 4: FaHaNa-Nets can consistently achieve better fairness when data balancing [18] is applied for improving fairness.

| Model         | Acc. Unfair. | Acc. Impr. Unfair. | w/o balancing | w/ balancing |
|---------------|--------------|---------------------|---------------|-------------|
| MobileNetV2   | 81.05%       | 0.2352              | 82.24%        | 0.1985      |
| ProxylessNAS(M) | 81.27%       | 0.3094              | 81.53%        | 0.26%       |
| MnasNet 0.5   | 78.12%       | 0.4521              | 78.82%        | 0.70%       |
| MobileNetV3(S) | 80.38%       | 0.2535              | 80.55%        | 0.17%       |
| MnasNet 1.0   | 80.71%       | 0.2913              | 80.20% -0.51% | 0.1585      |
| FaHaNa-Small  | 81.28%       | 0.1973              | 82.02% 0.74%  | 0.1365      |

5 CONCLUSION

In this work, we have proposed a fairness- and hardware-aware NAS framework, FaHaNa, integrating fairness in NAS for the first time to design the fair neural architecture. On top of it, a freezing method has been proposed to accelerate the NAS process. As such, FaHaNa can identify a series of neural architectures forming a much better Pareto frontier on accuracy, fairness, and model size, compared to the existing neural architectures. Moreover, FaHaNa is compatible with the existing techniques for fairness improvement. Extensive experiments are carried out to evaluate FaHaNa, where architecture with 5x smaller size and 5x lower latency can be obtained for edge devices, meanwhile, achieving 15.14% higher fairness and not compromising overall accuracy, compared to MobileNetV2 which has the highest fairness in all examined competitors.

REFERENCES

[1] Using ai to help find answers to common skin conditions. https://blog.google/technology/health/ai-dermatology-preview-io-2021/.
[2] Cradle white eye detector. https://cradle-white-eye-detector.id/09/2023/.
[3] Vital signs monitoring for everyone, everywhere. www.binance.ai.
[4] G. A. Kaisiss et al. Secure, privacy-preserving and federated machine learning in medical imaging. Nature Machine Intelligence, 20(6):305–311, 2020.
[5] S. Han et al. Deep compression. Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149, 2015.
[6] C. Zhang et al. Optimizing fpga-based accelerator design for deep convolutional neural networks. In Proc. of FPGA, pp. 161–170. ACM, 2015.
[7] C. Hao et al. FPGA/DNN co-design: An efficient design methodology for IoT. DAC, Design Automation Conference, 2022.
[8] Y. Song et al. Dancing along battery: Enabling transformer with run-time re-configurability on mobile devices. In 2021 5th ACM/IEEE Design Automation Conference (DAC), pp. 1003–1008. IEEE, 2021.
[9] H. Peng et al. Optimizing fpga-based accelerator design for large-scale molecular similarity search (special session paper). In 2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD), pp. 1–7. IEEE, 2021.
[10] W. Jiang et al. Device-circuit-architecture co-exploration for computing-in-memory neural accelerators. IEEE Transactions on Computers, 70(4):995–1005, 2020.
[11] W. Jiang et al. Achieving super-linear speedup across multi-fpga for real-time dnn inference. *ACM Transactions on Embedded Computing Systems (TECS)*, 19(5s):67, 2019.

[12] X. Zhang et al. When neural architecture search meets hardware implementation: from hardware awareness to co-design. In 2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), pp. 25–30. IEEE, 2019.

[13] W. Jiang et al. Accuracy vs. efficiency: Achieving both through fpga-implementation aware neural architecture search. In *Proceedings of the 56th Annual Design Automation Conference* 2019, pp. 1–6, 2019.

[14] Gender and skin-type bias in commercial ai systems. [https://news.mit.edu/2018/study-finds-gender-skin-type-bias-artificial-intelligence-systems-0212](https://news.mit.edu/2018/study-finds-gender-skin-type-bias-artificial-intelligence-systems-0212).

[15] L. H. Kamulegeya et al. Using artificial intelligence on dermatology conditions in uganda: A case for diversity in training data sets for machine learning. *BioRxiv*, pp. 826057, 2019.

[16] N. Mehrabi et al. A survey on bias and fairness in machine learning. *ACM Computing Surveys (CSUR)*, 54(6):1–35, 2021.

[17] M. Choraś et al. Machine learning–the results are not the only thing that matters! what about security, explainability and fairness? In *International Conference on Computational Science*, pp. 615–628. Springer, 2020.

[18] K. Choi et al. Fair generative modeling via weak supervision. In *International Conference on Machine Learning*, pp. 1887–1898. PMLR, 2020.

[19] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. In *Proc. of ICLR*, 2017.

[20] M. Tan et al. Mnasnet: Platform-aware neural architecture search for mobile. In *Proc. of CVPR*, pp. 2820–2828, 2019.

[21] H. Cai et al. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018.

[22] Y. Zhang et al. Dian: Differentiable accelerator-network co-search towards maximal dnn efficiency. In 2021 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED), pp. 1–6. IEEE, 2021.

[23] Z. Yuan et al. Ms-nas: Multi-scale neural architecture search for medical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 388–397. Springer, 2020.

[24] S. Bian et al. Nsas: Optimizing secure inference via neural architecture search. *arXiv preprint arXiv:2001.11854*, 2020.

[25] W. Jiang et al. Hardware/software co-exploration of neural architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(12):4805–4815, 2020.

[26] L. Yang et al. Co-exploring neural architecture and network-on-chip design for real-time artificial intelligence. In 2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 85–90. IEEE, 2020.

[27] L. Yang et al. Co-exploration of neural architectures and heterogeneousasic accelerator designs targeting multiple tasks. In 2020 57th ACM/IEEE Design Automation Conference (DAC), pp. 1–6. IEEE, 2020.

[28] Q. Lu et al. On neural architecture search for resource-constrained hardware platforms. In *International Conference on Computer-Aided Design (ICCAD)*, pp. 1, ACM, 2019.

[29] Y. He et al. A geometric solution to fair representations. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pp. 279–285, 2020.

[30] E. Wu et al. How medical ai devices are evaluated: limitations and recommenda-
tions from an analysis of fda approvals. *Nature Medicine*, 27(4):582–584, 2021.

[31] W. Jiang et al. Standing on the shoulders of giants: Hardware and neural archi-
tecture co-search with hot start. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(11):4154–4165, 2020.

[32] C.-H. Hsu et al. Monas: Multi-objective neural architecture search using rein-
forcement learning. *arXiv preprint arXiv:1806.10332*, 2018.

[33] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[34] Skin lesion analysis. [https://challenge2019.isic-archive.com/](https://challenge2019.isic-archive.com/).

[35] Dermatology atlas. [http://www.atlasdermatologico.com.br/](http://www.atlasdermatologico.com.br/). Accessed Nov, 2021.

[36] M. Sandler et al. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proc. of CVPR*, pp. 4510–4520, 2018.

[37] A. Howard et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[38] Raspberry pi 4 model b. [https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/](https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/). Accessed Nov, 2021.

[39] Odroid-xu4. [https://www.hardkernel.com/](https://www.hardkernel.com/). Accessed Nov, 2021.