HW-NAS-BENCH: HARDWARE-AWARE NEURAL ARCHITECTURE SEARCH BENCHMARK

Anonymous authors
Paper under double-blind review

ABSTRACT

HardWare-aware Neural Architecture Search (HW-NAS) has recently gained tremendous attention for automating the design of DNNs to be deployed into more resource-constrained daily life devices. Despite their promising performance, developing optimal HW-NAS solutions can be prohibitively challenging as it requires cross-disciplinary knowledge in the algorithm, micro-architecture, and device-specific compilation. First, to construct the hardware cost to be incorporated into the NAS process, existing works mostly adopt either pre-collected cost look-up tables or device-specific hardware cost models. The former can be time-consuming due to the needed learning about the device's compilation method and how to set up the measurement pipeline, while the latter is often a barrier for non-hardware experts like NAS researchers. Both of them limit the development of HW-NAS innovations and impose a barrier-to-entry to non-hardware experts. Second, similar to generic NAS, it can be notoriously difficult to benchmark HW-NAS algorithms due to the required significant computational resources and the differences in their adopted search space, hyperparameters, and hardware devices. To this end, we develop HW-NAS-Bench, the first public dataset for HW-NAS research which aims to democratize HW-NAS research to non-hardware experts and make HW-NAS research more reproducible and accessible. To design HW-NAS-Bench, we carefully collected the measured/estimated hardware performance (e.g., energy cost and latency) of all the networks in the search space of both NAS-Bench-201 and FBNet, considering six hardware devices that fall into three categories (i.e., commercial edge devices, FPGA, and ASIC). Furthermore, we provide a comprehensive analysis of the collected measurements in HW-NAS-Bench to provide insights for HW-NAS research. Finally, we demonstrate exemplary user cases where HW-NAS-Bench (1) allows non-hardware experts to perform HW-NAS by simply querying our pre-measured dataset and (2) verify that dedicated device-specific HW-NAS can indeed often provide optimal accuracy-cost trade-offs. All the codes and data will be released publicly upon acceptance.

1 INTRODUCTION

The recent performance breakthroughs of deep neural networks (DNNs) have attracted an explosion of research in designing efficient DNNs, aiming to bring powerful yet power-hungry DNNs into more resource-constrained daily life devices for enabling various DNN-powered intelligent functions. Among them, HardWare-aware Neural Architecture Search (HW-NAS) has emerged as one of the most promising techniques as it can automate the process of designing optimal DNN structures for the target applications, each of which often corresponds to a different hardware device and requires a different hardware efficiency metric (e.g., prioritizes latency or energy). For example, HW-NAS in (Wu et al., 2019) develops a differentiable neural architecture search (DNAS) framework and discovers state-of-the-art (SOTA) DNNs balancing both accuracy and hardware efficiency, by incorporating a loss that consists of both the cross-entropy loss that leads to better accuracy and the latency loss that penalizes the network's latency on a target device.

Despite the promising performance achieved by SOTA HW-NAS, there exist paramount challenges that limit the development of HW-NAS innovations. First, HW-NAS requires the collection of hardware efficiency data corresponding to (all) the networks in the search space. To do so, current practice either pre-collects these data to construct a hardware-cost look-up table or adopts device-

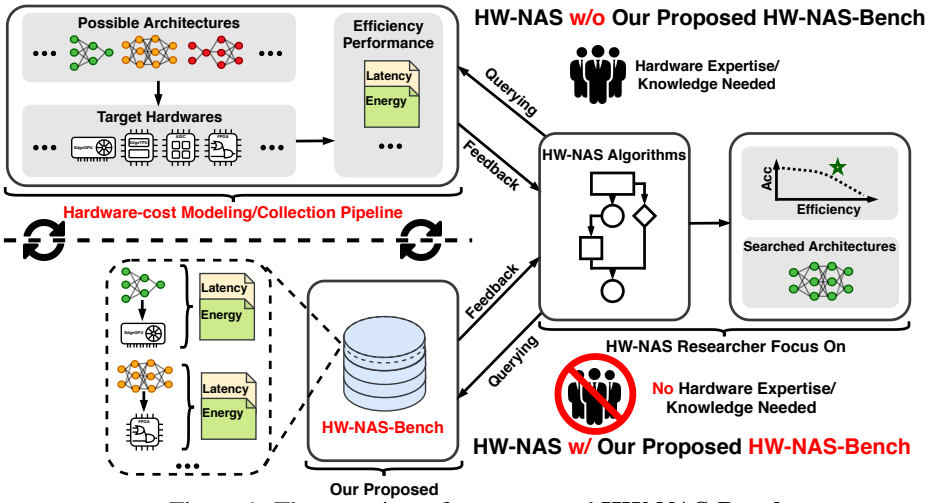

Figure 1: The overview of our proposed **HW-NAS-Bench**

specific hardware-cost estimators/models, both of which can be time consuming to obtain and impose a barrier-to-entry to non-hardware experts. This is because it requires knowledge about device-specific compilation and properly setting up the hardware measurement pipeline for hardware-cost data collection. Second, similar to generic NAS, it can be notoriously difficult to benchmark HW-NAS algorithms due to the required significant computational resources and the differences in their (1) hardware devices, which are specific for HW-NAS, (2) adopted search space, and (3) hyperparameters. Such a difficulty is even higher for HW-NAS considering the numerous choices of hardware devices, each of which can favor very different network structures even under the same target hardware efficiency, as discussed in (Chu et al., 2020). While the number of floating-point operations (FLOPs) have been commonly used to estimate the hardware-cost, many works have pointed out that DNNs with fewer FLOPs are not necessarily faster or more efficient (Wu et al., 2019). For example, NasNet-A has a comparable complexity in terms of FLOPs as MobileNetV1 yet can have a slower latency than the latter due to NasNet-A's adopted hardware-unfriendly structure.

It is thus imperative to address the aforementioned challenges in order to make HW-NAS more accessible and reproducible to unfold HW-NAS's full potential. Note that although pioneering NAS benchmark datasets (Ying et al., 2019; Dong & Yang, 2020; Klyuchnikov et al., 2020; Siems et al., 2020; Dong et al., 2020) have made a significant step towards providing a unified benchmark dataset for generic NAS works, they all either merely provide latency on server-level GPUs (e.g., GTX 1080Ti) or do not provide hardware-cost metrics on real hardware, limiting their applicability to HW-NAS (Wu et al., 2019; Wan et al., 2020; Cai et al., 2018) which primarily targets commercial edge devices, FPGA, and ASIC. To this end, as shown in Figure 1, we develop **HW-NAS-Bench** and make the following contributions in this paper:

- We have developed **HW-NAS-Bench**, the first public dataset for HW-NAS research aiming to (1) democratize HW-NAS research to non-hardware experts and (2) provide a unified benchmark for HW-NAS to make HW-NAS research more reproducible and accessible, covering two SOTA NAS search spaces including NAS-Bench-201 and FBNet, with the former being the most popular NAS search space and the latter having been shown to be the most hardware friendly.
- We provide hardware-cost data collection pipelines for six commonly used hardware devices that fall into three categories (i.e., commercial edge devices, FPGA, and ASIC), in addition to the measured/estimated hardware-cost (e.g., energy cost and latency) on these devices for all the networks in the search space of both NAS-Bench-201 and FBNet.
- We conduct comprehensive analysis of the collected data in HW-NAS-Bench, such as studying the correlation between collected hardware-cost and accuracy-cost trade-offs of all the networks on the six hardware devices, which can provide insights to not only HW-NAS researchers but also DNN accelerator designers. Other researchers can extract other useful insights from HW-NAS-Bench that have not been discussed in this work.
- We demonstrate exemplary user cases to show (1) how HW-NAS-Bench can be easily used by non-hardware experts to develop HW-NAS solutions by simply querying the collected data in our HW-NAS-Bench and (2) that dedicated device-specific HW-NAS can indeed

often provide optimal accuracy-cost trade-offs, demonstrating the great necessity of HW-NAS benchmark frameworks like our proposed HW-NAS-Bench.

2 RELATED WORKS

2.1 HARDWARE-AWARE NEURAL ARCHITECTURE SEARCH

Driven by the growing demand for efficient DNN solutions, HW-NAS has been proposed to automate the search for efficient DNN structures under the target efficiency constraints. For example, (Tan et al., 2019; Howard et al., 2019; Tan & Le, 2019) adopts reinforcement learning based NAS with a multi-objective reward consisting of both task performance and efficiency, achieving promising results yet suffering from prohibitive search time/cost. In parallel, (Wu et al., 2019; Wan et al., 2020; Cai et al., 2018; Stamoulis et al., 2019) explore the design space in a differentiable manner following (Liu et al., 2018) and significantly improve the search efficiency. The promising performance of HW-NAS has motivated a tremendous interest in applying it to more diverse applications (Fu et al., 2020; Chen et al., 2019; Marchisio et al., 2020) paired with target hardware devices, e.g., EdgeTPU (Xiong et al., 2020) and NPU (Lee et al., 2020), in addition to the widely explored mobile phones.

As discussed in (Chu et al., 2020), different hardware devices can favor very different network structures under the same hardware efficiency metric, and the optimal network structure can differ significantly when considering different application driven hardware efficiency metrics on the same hardware devices. As such, it would ideally lead to the optimal accuracy-efficiency trade-offs if the HW-NAS design is dedicated for the target device and hardware efficiency metrics. However, this requires a good understanding of both device-specific compilation and hardware-cost characterization, imposing a barrier-to-entry to non-hardware experts, such as many NAS researchers, and thus limiting the development of optimal HW-NAS results for numerous applications, each of which often corresponds to a different application driven hardware efficiency and adopts different types of hardware devices. As such, our proposed HW-NAS-Bench will make HW-NAS more friendly to NAS researchers, who are often non-hardware experts, as it consists of comprehensive hardware-cost data in a wide range of hardware devices for all the networks in two SOTA NAS search spaces, expediting the development of HW-NAS innovations.

2.2 NEURAL ARCHITECTURE SEARCH BENCHMARKS.

The importance and difficulty of NAS reproducibility and benchmarking has recently gained increasing attention. Pioneering efforts include (Ying et al., 2019; Dong & Yang, 2020; Klyuchnikov et al., 2020; Siems et al., 2020; Dong et al., 2020). Specifically, NAS-Bench-101 (Ying et al., 2019) develops the first large-scale, open-source architecture dataset for NAS, in which they provide the ground truth test accuracy on CIFAR-10 (Krizhevsky et al., 2009) of all the architectures (i.e., 423k) in their search space. Later, NAS-Bench-201 (Dong & Yang, 2020) further extends NAS-Bench-101 to support more NAS algorithm categories (e.g., differentiable algorithms) and more datasets (e.g., CIFAR-100 (Krizhevsky et al., 2009) and ImageNet16-120 (Chrabaszcz et al., 2017)). Most recently, NAS-Bench-301 (Siems et al., 2020) and NATS-Bench (Dong et al., 2020) were developed to support benchmarking on larger search spaces. However, all of these works either merely provide latency on the server-level GPU (e.g., GTX 1080Ti) or do not consider hardware efficiency metrics on real hardware, limiting their applicability to HW-NAS (Wu et al., 2019; Wan et al., 2020; Cai et al., 2018) that primarily target the commercial edge devices, FPGA, and ASIC. This has motivated us to develop our HW-NAS-Bench, aiming to make HW-NAS more accessible and reproducible.

3 THE PROPOSED HW-NAS-BENCH FRAMEWORK

3.1 HW-NAS-BENCH'S CONSIDERED SEARCH SPACES

To ensure a wide applicability, our HW-NAS-Bench considers two representative NAS search spaces: (1) NAS-Bench-201's cell-based search space and (2) FBNet search space. Both contributes valuable aspects to ensure our goal of constructing a comprehensive HW-NAS benchmark framework. Specifically, the former enables HW-NAS-Bench to naturally integrate the ground truth ac-

curacy data of all NAS-Bench-201's considered network architectures, while the latter ensures that HW-NAS-Bench includes the most commonly recognized hardware friendly search space.

NAS-Bench-201 Search Space. Inspired from the search space used in the most popular cell-based NAS, NAS-Bench-201 adopts a fixed cell search space, where each architecture consists of a predefined skeleton with a stack of the searched cell that is represented as a densely-connected directed acyclic graph (DAG). Specifically, it considers 4 nodes and 5 representative operation candidates for the operation set, and varies the feature map sizes and the dimensions of the final Fully-connected layers to handle its considered three datasets (i.e., CIFAR-10, CIFAR-100 (Krizhevsky et al., 2009), and ImageNet16-120 (Chrabaszcz et al., 2017)), leading to a total of $3 \times 5^6 = 46875$ architectures. Training log and accuracy are provided for each architecture. However, NAS-Bench-201 can not be directly used for HW-NAS as it only includes theoretical cost metrics (i.e., FLOPs and the number of parameters (#Params)) and latency on a server-type GPU (i.e., GTX 1080Ti). **HW-NAS-Bench enhances NAS-Bench-201 by providing all the 46875 architectures' measured/estimated hardware-cost on six hardware devices, that are primarily targeted by SOTA HW-NAS works**.

FBNet Search Space. FBNet (Wu et al., 2019) constructs a layer-wise search space with a fixed macro-architecture, which defines the number of layers and the input/output dimensions of each layer and fixes first and the last three layers with the remaining layers to be searched. In this way, networks in FBNet (Wu et al., 2019) search space have more regular structure than those in NAS-Bench-201, and have been shown to be more hardware friendly (Fu et al., 2020; Ma et al., 2018). The 9 considered re-defined cell candidates and 22 unique positions lead to a total of $9^{22} \approx 10^{21}$ unique architectures. While HW-NAS researchers can develop their search algorithms on top of FBNet (Wu et al., 2019) search space, tedious efforts are required to collect the hardware-cost look-up tables or models on each target devices. **HW-NAS-Bench provides measured/estimated hardware-cost on six hardware devices for all the 10^{21} architectures of FBNet search space, aiming to make HW-NAS research more friendly to non-hardware experts**.

3.2 Hardware-cost Collection Pipeline and The Considered Devices

To collect hardware-cost data for all the architectures in both NAS-Bench-201 and FBNet spaces, we construct a generic hardware-cost collection pipeline (see Figure 2) to automate the process. Specifically, the pipeline mainly consists of the target devices and corresponding implementation tools (e.g., compilers), where it takes all the networks as its inputs, compile the networks to (1) convert them into the device's required execution format and (2) optimize the execution flow, the latter of which aims to optimize the hardware performance on the target devices. For example, for collecting hardware-cost in Edge GPU, we first set the device in the Max-N mode to fully make use of all available resources following (Wofk et al., 2019), and then set up the embedded power rail monitor (Texas Instruments Inc.) to obtain real-measured latency and energy via sysfs (Patrick Mochel and Mike Murphy.) averaged over 50 runs. We can see that **the hardware-cost collection process requires various hardware domain knowledge including machine learning development frameworks, device compilation, embedded systems, and device measurements**.

Next, we briefly introduce the six considered hardware devices and the specific configuration required to collect hardware-cost data using each device.

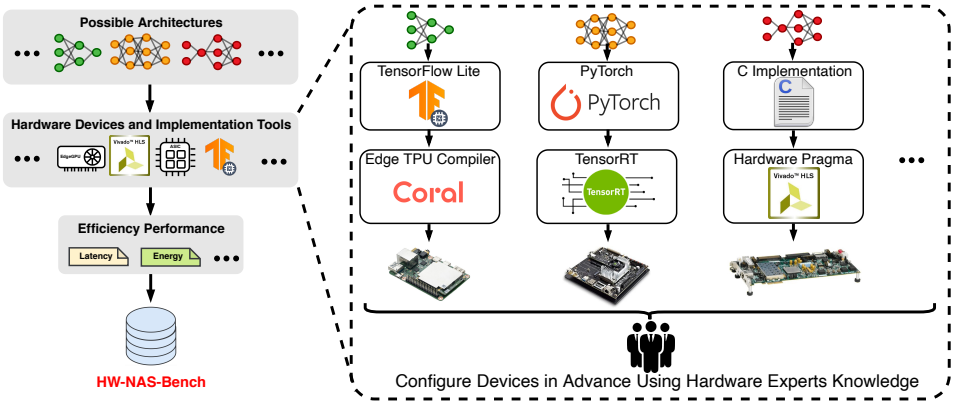

Figure 2: Illustrating the hardware-cost collection pipeline applicable to various hardware devices.

Table 1: Important details about the six hardware devices considered by our HW-NAS-Bench

Devices	Edge GPU	Raspi 4	Edge TPU	Pixel 3	ASIC-Eyeriss	FPGA
Collected Metrics	Latency (ms) Energy (mJ)	Latency (ms)	Latency (ms)	Latency (ms)	Latency (ms) Energy (mJ)	Latency (ms) Energy (mJ)
Collecting Method	Measured	Measured	Measured	Measured	Estimated	Estimated
Runtime Environment	TensorRT	TensorFlow Lite	Edge TPU Runtime	TensorFlow Lite	Accelergy+Timeloop / DNN-Chip Predictor	Vivado HLS
Customizing Hardware?	✗	✗	✗	✗	✓	✓
Category	Commercial Edge Devices				ASIC	FPGA

Edge GPU: NVIDIA Edge GPU Jetson TX2 (Edge GPU) is a commercial device with a 256-core Pascal GPU and a 8GB LPDDR4 targeting IoT applications (NVIDIA Inc., a). When plugging Edge GPU into the above hardware-cost collection pipeline, we first compile the network architectures in both NAS-Bench-201 and FBNet spaces to (1) convert them to the TensorRT format and (2) optimize the inference implementation within NVIDIA's recommended TensorRT runtime environment, and then execute them in Edge GPU to measure the consumed energy and latency.

Raspi 4: Raspberry Pi 4 (Raspi 4) is the latest Raspberry Pi device (Raspberry Pi Limited.), consisting of a Broadcom BCM2711 SoC and a 4GB LPDDR4. To collect hardware-cost operating on it, we compile the architecture candidates to (1) convert them into TensorFlow Lite (TFLite) (Abadi et al., 2016) format (2) optimize the implementation using the official interpreter (Google LLC., 2020) in Raspi 4, which will be configured in advance.

Edge TPU: Edge TPU Dev Board (Google LLC., a) is a dedicated ASIC accelerator developed by Google for targeting Artificial Intelligence (AI) inference for edge applications. Similar to the case when using Raspi 4, all the architectures are converted into TFLite format. After that, an Edge TPU compiler will be used to convert the pre-built TFLite model into a more compressed format which is compatible to the pre-configured Edge TPU runtime environment in the Dev Board.

Pixel 3: Pixel 3 is one of the latest Pixel mobile phones (Google LLC., e) that are widely used as the target platform by recent NAS works (Xiong et al., 2020; Howard et al., 2019; Tan et al., 2019). To collect hardware-cost in Pixel 3, we first convert all the architectures into TFLite, then use TFLite's official benchmark binary files to obtain the latency, when configuring the Pixel 3 to only use big cores for reducing the measurement variance as in (Xiong et al., 2020; Tan et al., 2019).

ASIC-Eyeriss: For collecting hardware-cost data in ASIC, we consider a SOTA ASIC accelerator (Chen et al., 2016). Specifically, we adopt SOTA ASIC accelerator performance simulators (1) Accelergy (Wu et al., 2019)+Timeloop (Parashar et al., 2019) and (2) DNN-Chip Predictor (Zhao et al., 2020), which automatically identify the optimal algorithm-to-hardware mapping methods for each architecture and then provide the estimated hardware-cost when being executed in Eyeriss.

FPGA: FPGA is a widely adopted AI acceleration platform featuring higher hardware flexibility and decent hardware efficiency. To collect hardware-cost in this platform, we first develop a SOTA chunk based pipeline structure (Zhang et al., 2018; Shen et al., 2017) implementation, compile all the architectures using standard Vivado HLS toolflow (Xilinx Inc., a), and then obtain the hardware-cost on a Xilinx ZC706 board with Zynq XC7045 SoC (Xilinx Inc., b).

More details about the pipeline for each device are provided in the Appendix D.

Note that to estimated the hardware-cost of networks in the FBNet search space (Wu et al., 2019) when being executed on the commercial category of edge devices (i.e., edge GPU, Raspi 4, edge TPU, and Pixel 3) in our HW-NAS-Bench, we sum up hardware-cost of all unique blocks (i.e., "block" in the FBNet space (Wu et al., 2019)) within the network architectures. To validate that

Table 2: Two types of correlation coefficients (larger means more related) between the measured hardware-cost of the whole architectures and the approximated hardware-cost based on 100 randomly sampled architectures from the FBNet search space.

Correlation Coefficient Types	Datasets	Latency on Edge GPU	Energy on Edge GPU	Latency on Raspi 4	Latency on Edge TPU	Latency on Pixel 3
Pearson Correlation Coefficient	CIFAR-100	0.9200	0.9116	0.9219	0.4935	0.9324
	ImageNet	0.8634	0.9640	0.9897	0.7153	0.9162
Kendall Rank Correlation Coefficient	CIFAR-100	0.7373	0.7240	0.7470	0.3551	0.8593
	ImageNet	0.7111	0.8379	0.9163	0.5806	0.8064

such an approximation is close to real-measured results, we conduct the experiments summarized in Table 2, which calculates two types of correlation coefficients between the measured and the approximated hardware-cost based on 100 randomly sampled architectures from the FBNet search space. We can see that our approximated costs are highly correlated with the real-measured ones, except for the case on EdgeTPU, which we conjecture is caused by our adopted in-house EdgeTPU compiler (Google LLC., c). More visualization results can be found in Appendix A.

4 ANALYSIS ON HW-NAS-BENCH

In this section, we provide analysis and visualization of the hardware-cost and corresponding accuracy data (the latter only for architectures in NAS-Bench-201) for all the architectures in the two considered search spaces. Specifically, our analysis and visualization confirm that (1) commonly used theoretical hardware-cost metrics such as FLOPs do not correlate well with measured/estimated hardware-cost; (2) hardware-cost of the same architectures can differ a lot when executed on different devices; and (3) device-specific HW-NAS is necessary because optimal architectures resulting from HW-NAS targeting on one device can perform poorly in terms of hardware efficiency when being executed on another device.

4.1 CORRELATION BETWEEN COLLECTED HARDWARE-COST AND THEORETICAL ONES

To confirm whether commonly used theoretical hardware-cost metrics align with real measured/estimated ones, we summarize the calculated correlation between the collected hardware-cost in our HW-NAS-Bench and the theoretical metrics (i.e., FLOPs and #Params), based on data for all the architectures in both search spaces on all the six considered hardware devices where a total of four different datasets are involved.

As summarized in Tables 3 - 4, commonly used theoretical hardware-cost metrics (i.e., FLOPs and #Params) do not always correlate well with measured/estimated hardware-cost for the architectures in both the NAS-Bench-201 and FBNet spaces. For example, there exists at least one coefficient <0.5 on all devices, especially for the cases with measured hardware-cost on commonly considered edge platforms including Edge GPU, Edge TPU, and ASIC-Eyeriss. As such, HW-NAS based on theoretical hardware-cost might lead to sub-optimal results, motivating HW-NAS benchmark frameworks like our HW-NAS-Bench. Note that we consider the Kendall Rank Correlation Coefficients (Abdi, 2007), which is a commonly used correlation coefficient in both recent NAS and NAS benchmark frameworks (You et al., 2020; Siems et al., 2020; Yang et al., 2020).

Table 3: Kendall Rank Correlation Coefficient between real measured/estimated hardware-cost and theoretical ones considering the **NAS-Bench-201 search space**, where coefficients <0.5 are **bolded**.

Dataset	Metrics	Edge GPU		Raspi 4	Edge TPU	Pixel 3	ASIC-Eyeriss		FPGA	
		Latency	Energy	Latency	Latency	Latency	Latency	Energy	Latency	Energy
CIFAR-10	FLOPs	**0.3571**	**0.4064**	0.7394	**0.1847**	0.6823	**0.4178**	0.5359	0.8313	0.8313
	#Params	**0.3571**	**0.4064**	0.7394	**0.1847**	0.6823	**0.4178**	0.5359	0.8313	0.8313
CIFAR-100	FLOPs	**0.3589**	**0.4073**	0.7384	**0.1851**	0.6844	**0.4197**	0.5360	0.8313	0.8313
	#Params	**0.3589**	**0.4073**	0.7384	**0.1851**	0.6844	**0.4197**	0.5360	0.8313	0.8313
ImageNet16-120	FLOPs	**0.3544**	**0.3868**	0.6303	**0.2635**	0.7017	**0.4166**	0.5363	0.9205	0.9205
	#Params	**0.3544**	**0.3868**	0.6303	**0.2635**	0.7017	**0.4166**	0.5363	0.9205	0.9205

Table 4: Kendall Rank Correlation Coefficient between real measured/estimated hardware-cost and theoretical ones considering the **FBNet search space**, where coefficients <0.5 are **bolded**.

Dataset	Metrics	Edge GPU		Raspi 4	Pixel 3	ASIC-Eyeriss		FPGA	
		Latency	Energy	Latency	Latency	Latency	Energy	Latency	Energy
CIFAR-100	FLOPs	**0.0149**	**0.1564**	0.7713	0.8092	0.8490	0.7854	0.8710	0.8710
	#Params	**-0.0733**	**0.0202**	**0.4910**	**0.3734**	**0.4297**	0.6455	0.5151	0.5151
ImageNet	FLOPs	**0.4633**	0.6094	0.7531	0.7678	0.8935	0.7970	0.8643	0.8643
	#Params	**0.0985**	**0.1840**	**0.2318**	**0.2357**	**0.3202**	**0.4140**	**0.4198**	**0.4198**

4.2 CORRELATION AMONG COLLECTED HARDWARE-COST ON DIFFERENT DEVICES

To check how much the hardware-cost of the same architectures on different devices correlate, we visualize the correlation between the hardware-cost collected from every two paired devices base on data for all the architectures on both the NAS-Bench-201 and FBNet search spaces with each of the architecture being associated with 9 different hardware-cost metrics.

The visualization in Figures 3 - 4 indicates that hardware-cost of the same network architectures can differ a lot when being executed on different devices. More specifically, the correlation coefficients can be as small as -0.00 (e.g., Edge GPU latency vs. ASIC-Eyeriss energy), which is a result of the large difference in (1) their underlying hardware architectures and (2) the amount of available resources. Thus, the resulting architecture of HW-NAS targeting one device might perform poorly when being executed on other device, motivating device-specific HW-NAS, and it is crucial to develop comprehensive hardware-cost datasets like our HW-NAS-Bench to enable fast development and optimal results of HW-NAS for different applications.

4.3 OPTIMAL ARCHITECTURES ON DIFFERENT HARDWARE DEVICES

To confirm the necessity of performing device-specific HW-NAS from another perspective, we summarize all architectures' test accuracy vs. hardware-cost on ImageNet16-120 of NAS-Bench-201 and focus on the architectures with the optimal accuracy-cost trade-offs.

As shown in Figure 5, such optimal architectures for different devices are not the same. For example, the optimal architectures on Edge GPU (marked as red points) can perform poorly in terms of hardware efficiency in other devices, especially in ASIC-Eyeriss and Edge TPU whose hardware-cost have exactly the smallest correlation coefficient with the hardware-cost measured in the Edge GPU, which is shown in Figure 3. Again, this set of analysis and visualization confirms that HW-NAS targeting on one device can perform poorly in terms of hardware efficiency when being executed on another device, thus motivating the necessity of device-specific HW-NAS.

5 USER CASE: BENCHMARK SOTA HW-NAS ALGORITHMS

In this section, we will demonstrate the user cases of our HW-NAS-Bench to show (1) how non-hardware experts can use it to develop HW-NAS solutions by simply querying the hardware-cost data (2) dedicated device-specific HW-NAS can indeed often lead to optimal accuracy-cost trade-offs, again showing the need for HW-NAS benchmark frameworks like our HW-NAS-Bench to enable more optimal HW-NAS solutions via device-specific HW-NAS.

Benchmark Setting. We adopt a SOTA HW-NAS algorithm, ProxylessNAS (Cai et al., 2018) for this experiment. As an example to use our HW-NAS-Bench, we use ProxylessNAS to search over the FBNet (Wu et al., 2019) space on CIFAR-100 (Krizhevsky et al., 2009), when targeting different devices in our HW-NAS-Bench by simply querying the corresponding device's measured/estimated hardware-cost, which has negligible overhead as compared with the NAS algorithm itself, without the need for hardware expertise or knowledge during the whole HW-NAS.

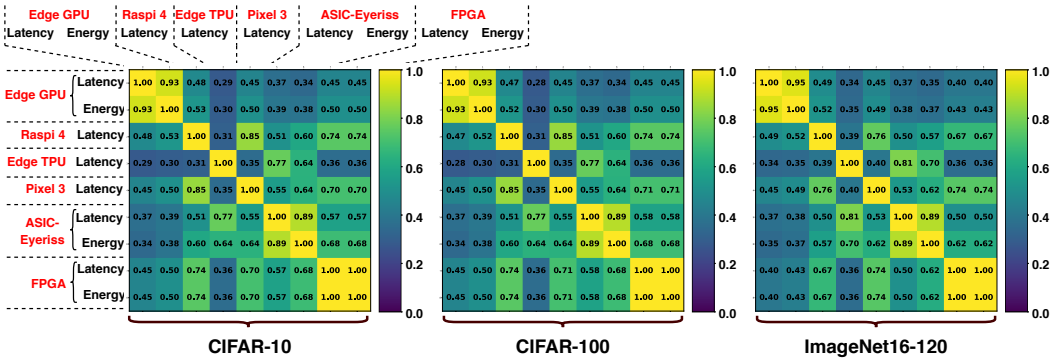

Figure 3: Kendall Rank Correlation Coefficient between real measured/estimated hardware-cost in different devices considering the **NAS-Bench-201 search space**.

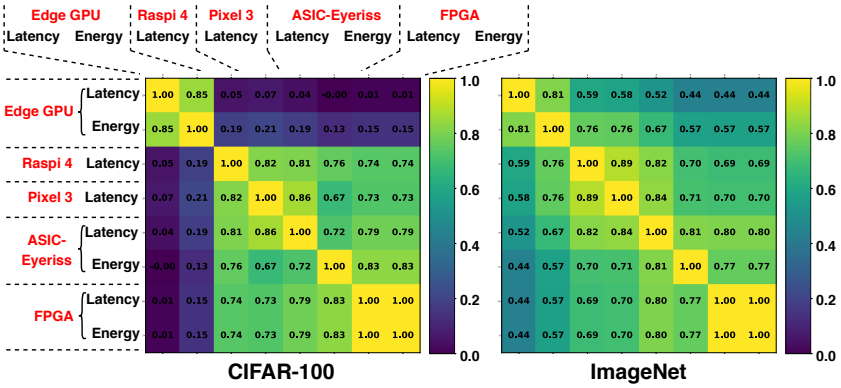

Figure 4: Kendall Rank Correlation Coefficient between real measured/estimated hardware-cost in different devices considering the **FBNet search space**.

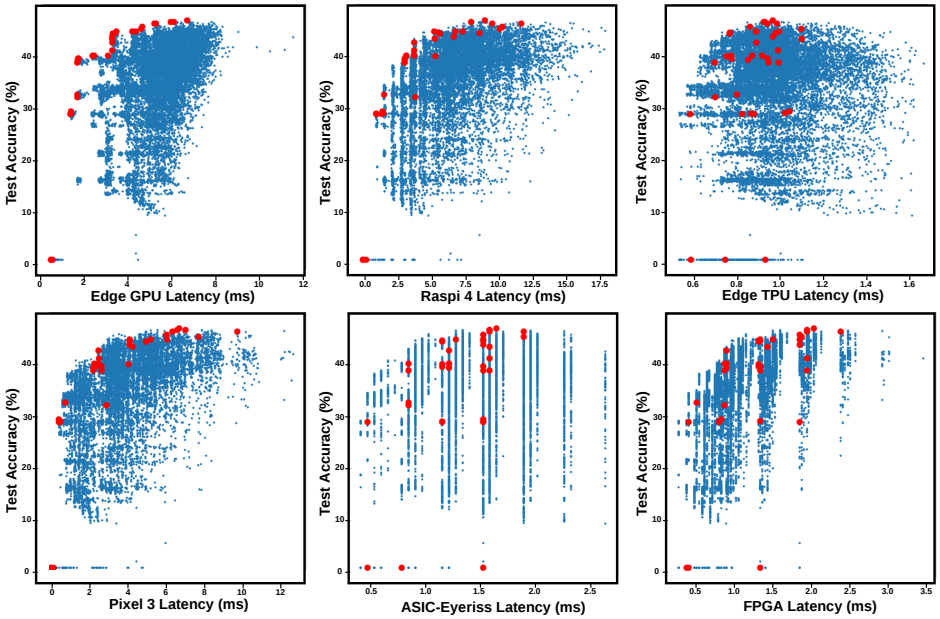

Figure 5: Accuracy vs. hardware-cost on different devices considering NAS-Bench-201, where points in red denote the architectures with the optimal "accuracy on ImageNet16-120 vs. smallest latency measured on Edge GPU" trade-offs of which the architectures represent the ground truth of HW-NAS targeting Edge GPU.

5.1 Optimal Architectures Resulting from Device-specific HW-NAS

Table 5: Inference performance comparison of optimal architectures resulting from HW-NAS-Bench when targeting different hardware devices.

Targeting Device in HW-NAS	Top-1 Acc.(%)	Latency on Edge GPU (ms)	Latency on Raspi 4 (ms)	Latency on FPGA (ms)
Edge GPU	74.11	**9.96**	31.01	20.19
Raspi 4	73.46	13.88	**22.91**	15.39
FPGA	73.51	20.65	25.43	**13.96**

Table 5 illustrates that the searched architectures achieve the lowest latency among all architectures, when the target devices of HW-NAS are the same as the one used to measure the architecture's on-device inference latency. Specifically, when being executed on Edge GPU, the architecture targeting on Raspi 4 during HW-NAS leads to about a 50% higher latency while the architecture targeting on FPGA during HW-NAS introduces over a 100% higher latency, than the architecture specifically target on Edge GPU during HW-NAS, under the same inference accuracy. This set of experiment

show that non-hardware experts can easily use our HW-NAS-Bench to develop optimal HW-NAS solutions, while demonstrating the need of device-specific HW-NAS solutions.

6 CONCLUSION

We have developed HW-NAS-Bench, the first public dataset for HW-NAS research aiming to (1) democratize HW-NAS research to non-hardware experts and (2) provide a unified benchmark for HW-NAS to make HW-NAS research more reproducible and accessible. Our HW-NAS-Bench covers two representative NAS search spaces, provides all network architectures' hardware-cost data on six commonly used hardware devices that fall into three categories (i.e., commercial edge devices, FPGA, and ASIC). Furthermore, we conduct comprehensive analysis of the collected data in HW-NAS-Bench, aiming to provide insights to not only HW-NAS researchers but also DNN accelerator designers. Finally, we demonstrate exemplary user cases of HW-NAS-Bench to show (1) how HW-NAS-Bench can be easily used by non-hardware experts to develop HW-NAS solutions by simply querying the collected data and (2) that dedicated device-specific HW-NAS can indeed often provide optimal accuracy-cost trade-offs, demonstrating the great necessity of HW-NAS benchmark frameworks like our proposed HW-NAS-Bench.

REFERENCES

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In 12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16), pp. 265–283, 2016.

Hervé Abdi. The kendall rank correlation coefficient. Encyclopedia of Measurement and Statistics. Sage, Thousand Oaks, CA, pp. 508–510, 2007.

Junjie Bai, Fang Lu, Ke Zhang, et al. Onnx: Open neural network exchange. `https://github.com/onnx/onnx`, 2020.

Samik Basu, Mahasweta Ghosh, and Soma Barman. Raspberry pi 3b+ based smart remote health monitoring system using iot platform. In Proceedings of the 2nd International Conference on Communication, Devices and Computing, pp. 473–484. Springer, 2020.

Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. In Noise reduction in speech processing, pp. 1–4. Springer, 2009.

Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. arXiv preprint arXiv:1812.00332, 2018.

Wuyang Chen, Xinyu Gong, Xianming Liu, Qian Zhang, Yuan Li, and Zhangyang Wang. Fasterseg: Searching for faster real-time semantic segmentation. arXiv preprint arXiv:1912.10917, 2019.

Y. Chen, T. Krishna, J. Emer, and V. Sze. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. JSSC 2017, 52(1):127–138, 2017.

Yu-Hsin Chen, Joel Emer, and Vivienne Sze. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. ACM SIGARCH Computer Architecture News, 44 (3):367–379, 2016.

Yu-Hsin Chen, Tushar Krishna, Joel Emer, and Vivienne Sze. Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks. In IEEE International Solid-State Circuits Conference, ISSCC 2016, Digest of Technical Papers, pp. 262–263, 2016.

François Chollet et al. Keras. `https://keras.io`, 2015.

Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. arXiv preprint arXiv:1707.08819, 2017.

Grace Chu, Okan Arikan, Gabriel Bender, Weijun Wang, Achille Brighton, Pieter-Jan Kindermans, Hanxiao Liu, Berkin Akin, Suyog Gupta, and Andrew Howard. Discovering multi-hardware mobile models via architecture search. arXiv preprint arXiv:2008.08178, 2020.

Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In International Conference on Learning Representations (ICLR), 2020. URL https://openreview.net/forum?id=HJxyZkBKDr.

Xuanyi Dong, Lu Liu, Katarzyna Musial, and Bogdan Gabrys. Nats-bench: Benchmarking nas algorithms for architecture topology and size. arXiv preprint arXiv:2009.00437, 2020.

Yonggan Fu, Wuyang Chen, Haotao Wang, Haoran Li, Yingyan Lin, and Zhangyang Wang. Autogan-distiller: Searching to compress generative adversarial networks. arXiv preprint arXiv:2006.08198, 2020.

Lukas Geiger and Plumerai Team. Larq: An open-source library for training binarized neural networks. Journal of Open Source Software, 5(45):1746, January 2020. doi: 10.21105/joss.01746. URL https://doi.org/10.21105/joss.01746.

Google LLC. Edge TPU Compiler, a. https://coral.ai/docs/dev-board/get-started/, accessed 2020-09-01.

Google LLC. Edge TPU Code Examples, b. https://coral.ai/examples/#code-examples, accessed 2019-11-21.

Google LLC. Edge TPU Compiler, c. https://coral.ai/docs/edgetpu/compiler/#system-requirements, accessed 2020-09-01.

Google LLC. Edge TPU FAQ, d. https://coral.ai/docs/edgetpu/faq/, accessed 2019-11-21.

Google LLC. Pixel 3, e. https://g.co/kgs/pVRc1Y, accessed 2020-09-01.

Google LLC. TensorFlow Lite: Deploy machine learning models on mobile and IoT devices, f. https://www.tensorflow.org/lite, accessed 2019-11-21.

Google LLC. Tflite python quickstart. https://www.tensorflow.org/lite/guide/python, 2020.

Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In Proceedings of the IEEE International Conference on Computer Vision, pp. 1314–1324, 2019.

Nikita Klyuchnikov, Ilya Trofimov, Ekaterina Artemova, Mikhail Salnikov, Maxim Fedorov, and Evgeny Burnaev. Nas-bench-nlp: Neural architecture search benchmark for natural language processing, 2020.

Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009.

Jaeseong Lee, Duseok Kang, and Soonhoi Ha. S3nas: Fast npu-aware neural architecture search methodology. arXiv preprint arXiv:2009.02009, 2020.

Chaojian Li, Tianlong Chen, Haoran You, Zhangyang Wang, and Yingyan Lin. Halo: Hardware-aware learning to optimize. In Proceedings of the European Conference on Computer Vision (ECCV), September 2020.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. arXiv preprint arXiv:1806.09055, 2018.

Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In Proceedings of the European Conference on Computer Vision (ECCV), pp. 116–131, 2018.

Alberto Marchisio, Andrea Massa, Vojtech Mrazek, Beatrice Bussolino, Maurizio Martina, and Muhammad Shafique. Nascaps: A framework for neural architecture search to optimize the accuracy and hardware efficiency of convolutional capsule networks. arXiv preprint arXiv:2008.08476, 2020.

NVIDIA Inc. NVIDIA Jetson TX2, a. https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-tx2/, accessed 2020-09-01.

NVIDIA Inc. Tensorrt, b.

NVIDIA Inc. Benchmark tx2 performance in googlenet with tensorrt, c.

Angshuman Parashar, Priyanka Raina, Yakun Sophia Shao, Yu-Hsin Chen, Victor A Ying, Anurag Mukkara, Rangharajan Venkatesan, Brucek Khailany, Stephen W Keckler, and Joel Emer. Timeloop: A systematic approach to dnn accelerator evaluation. In 2019 IEEE international symposium on performance analysis of systems and software (ISPASS), pp. 304–315. IEEE, 2019.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In Advances in neural information processing systems, pp. 8026–8037, 2019.

Patrick Mochel and Mike Murphy. sysfs - The filesystem for exporting kernel objects. https://www.kernel.org/doc/Documentation/filesystems/sysfs.txt, accessed 2019-11-21.

Raspberry Pi Limited. Pixel 3. https://www.raspberrypi.org/products/raspberry-pi-4-model-b/, accessed 2020-09-01.

Yongming Shen, Michael Ferdman, and Peter Milder. Maximizing cnn accelerator efficiency through resource partitioning. In Proceedings of the 44th Annual International Symposium on Computer Architecture, ISCA '17, pp. 535–547, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450348928. doi: 10.1145/3079856.3080221. URL https://doi.org/10.1145/3079856.3080221.

Mennatullah Siam, Mostafa Gamal, Moemen Abdel-Razek, Senthil Yogamani, Martin Jagersand, and Hong Zhang. A comparative study of real-time semantic segmentation for autonomous driving. In Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pp. 587–597, 2018.

Julien Siems, Lucas Zimmer, Arber Zela, Jovita Lukasik, Margret Keuper, and Frank Hutter. Nas-bench-301 and the case for surrogate benchmarks for neural architecture search. arXiv preprint arXiv:2008.09777, 2020.

Dimitrios Stamoulis, Ruizhou Ding, Di Wang, Dimitrios Lymberopoulos, Bodhi Priyantha, Jie Liu, and Diana Marculescu. Single-path nas: Designing hardware-efficient convnets in less than 4 hours. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 481–497. Springer, 2019.

Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. arXiv preprint arXiv:1905.11946, 2019.

Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2820–2828, 2019.

Texas Instruments Inc. INA3221 Triple-Channel, High-Side Measurement, Shunt and Bus Voltage Monitor. http://www.ti.com/product/INA3221, accessed 2019-11-21.

Alvin Wan, Xiaoliang Dai, Peizhao Zhang, Zijian He, Yuandong Tian, Saining Xie, Bichen Wu, Matthew Yu, Tao Xu, Kan Chen, et al. Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12965–12974, 2020.

Chien-Yao Wang, Hong-Yuan Mark Liao, Ping-Yang Chen, and Jun-Wei Hsieh. Enriching variety of layer-wise learning information by gradient combination. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops, Oct 2019.

Diana Wofk, Fangchang Ma, Tien-Ju Yang, Sertac Karaman, and Vivienne Sze. Fastdepth: Fast monocular depth estimation on embedded systems. In 2019 International Conference on Robotics and Automation (ICRA), pp. 6101–6108. IEEE, 2019.

Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 10734–10742, 2019.

Yannan N. Wu, Joel S. Emer, and Vivienne Sze. Accelergy: An Architecture-Level Energy Estimation Methodology for Accelerator Designs. In IEEE/ACM International Conference On Computer Aided Design (ICCAD), 2019.

Qingcheng Xiao, Yun Liang, Liqiang Lu, Shengen Yan, and Yu-Wing Tai. Exploring heterogeneous algorithms for accelerating deep convolutional neural networks on fpgas. In Proceedings of the 54th Annual Design Automation Conference 2017, DAC '17, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450349277. doi: 10.1145/3061639.3062244. URL https://doi.org/10.1145/3061639.3062244.

Xilinx Inc. Vivado High-Level Synthesis, a. https://https://www.xilinx.com/products/design-tools/vivado/integration/esl-design.html, accessed 2019-09-16.

Xilinx Inc. Xilinx zynq-7000 soc zc706 evaluation kit. https://www.xilinx.com/products/boards-and-kits/ek-z7-zc706-g.html, b. (Accessed on 09/30/2020).

Yunyang Xiong, Hanxiao Liu, Suyog Gupta, Berkin Akin, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Vikas Singh, and Bo Chen. Mobiledets: Searching for object detection architectures for mobile accelerators. arXiv preprint arXiv:2004.14525, 2020.

Antoine Yang, Pedro M. Esperança, and Fabio M. Carlucci. Nas evaluation is frustratingly hard. In International Conference on Learning Representations, 2020. URL https://openreview.net/forum?id=HygrdpVKvr.

Xuan Yang, Jing Pu, Blaine Burton Rister, Nikhil Bhagdikar, Stephen Richardson, Shahar Kvatinsky, Jonathan Ragan-Kelley, Ardavan Pedram, and Mark Horowitz. A systematic approach to blocking convolutional neural networks, 2016.

Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. Nas-bench-101: Towards reproducible neural architecture search. In International Conference on Machine Learning, pp. 7105–7114, 2019.

Shan You, Tao Huang, Mingmin Yang, Fei Wang, Chen Qian, and Changshui Zhang. Greedynas: Towards fast one-shot nas with greedy supernet. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1999–2008, 2020.

Chen Zhang, Peng Li, Guangyu Sun, Yijin Guan, Bingjun Xiao, and Jason Cong. Optimizing fpga-based accelerator design for deep convolutional neural networks. In Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, FPGA '15, pp. 161–170, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450333153. doi: 10.1145/2684746.2689060. URL https://doi.org/10.1145/2684746.2689060.

Jianhao Zhang, Yingwei Pan, Ting Yao, He Zhao, and Tao Mei. dabnn: A super fast inference framework for binary neural networks on arm devices. In Proceedings of the 27th ACM International Conference on Multimedia, pp. 2272–2275, 2019.

Xiaofan Zhang, Junsong Wang, Chao Zhu, Yonghua Lin, Jinjun Xiong, Wen-mei Hwu, and Deming Chen. Dnnbuilder: An automated tool for building high-performance dnn hardware accelerators for fpgas. In Proceedings of the International Conference on Computer-Aided Design, ICCAD '18, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450359504. doi: 10.1145/3240765.3240801. URL https://doi.org/10.1145/3240765.3240801.

Cheah Wai Zhao, Jayanand Jegatheesan, and Son Chee Loon. Exploring iot application using raspberry pi. International Journal of Computer Networks and Applications, 2(1):27–34, 2015.

Y. Zhao, C. Li, Y. Wang, P. Xu, Y. Zhang, and Y. Lin. Dnn-chip predictor: An analytical performance predictor for dnn accelerators with various dataflows and hardware architectures. In ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1593–1597, 2020.

A MORE VISUALIZATION ON MEASURED HARDWARE-COST FOR THE FBNET SEARCH SPACE

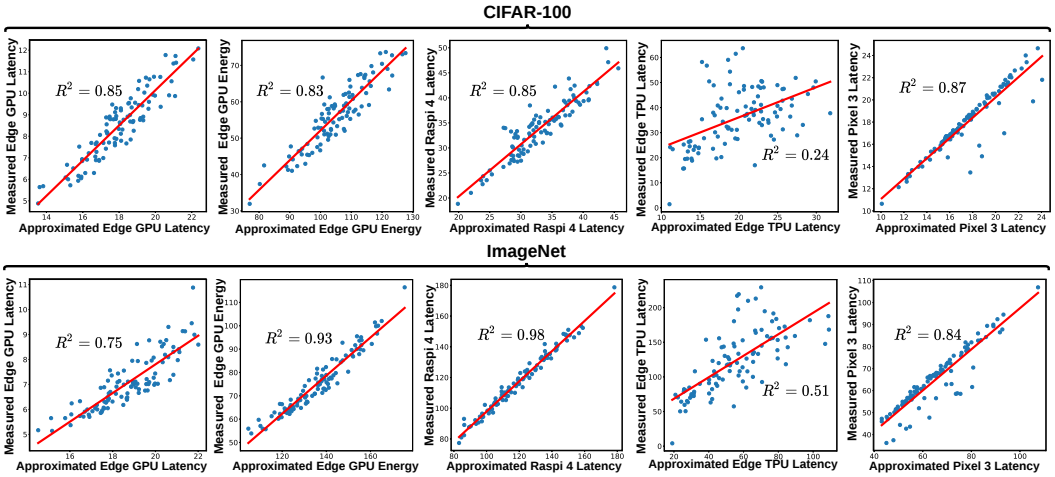

Figure 6: Comparison between the **approximated** and **measured** hardware-cost on CIFAR-100 (Top) and ImageNet (Bottom), where the red line indicates the fitting line for all the measured data, and R^2 represents the square of the Pearson Correlation Coefficient (Benesty et al., 2009).

Fig. 6 shows a comparison between the approximated and measured hardware-cost of randomly sampled 100 architectures when being executed on commercial edge devices using the ImageNet and CIFAR-100 datasets, which verifies that our approximation of summing up the performance of the unique blocks is a simple yet quite accurate for providing the hardware-cost for networks in the FBNet space, consistent with our observation in Table 2.

B COMPARING THE PERDITION OF EYERISS USING ACCELERGY+TIMELOOP AND DNN-CHIP-PREDICTOR

Both Accelergy (Wu et al., 2019)+Timeloop (Parashar et al., 2019) and DNN-Chip Predictor (Zhao et al., 2020) are able to simulate the latency and energy cost of Eyeriss (Chen et al., 2016), a SOTA ASIC design, when giving the network architectures. From Table 6, they nearly give the same estimation of the latency and energy cost (i.e. the mean of differences is 6.096%, the standard deviation of the differences is 0.779%, the Pearson correlation coefficient is 0.9998, and the Kendall Rank correlation coefficient is 0.9633, in term of the average performance) when benchmarked in 3 datasets of NAS-Bench-201. Therefore, we use the average value of their predictions as the estimated Eyeriss latency and energy in our proposed **HW-NAS-Bench**.

Table 6: The differences of estimation given by Accelergy (Wu et al., 2019)+Timeloop (Parashar et al., 2019) and DNN-Chip Predictor (Zhao et al., 2020) in 3 datasets of NAS-Bench-201

Datasets	Hardware-cost	Mean of Differences	Standard Deviation of Differences	Pearson Correlation Coefficient TPU	Kendall Rank Correlation Coefficient
CIFAR-10	Latency	1.648%	0.642%	0.9999	0.9888
	Energy	10.96%	1.035%	0.9997	0.9374
CIFAR-100	Latency	1.572%	0.611%	1.0000	0.9888
	Energy	10.93%	1.029%	0.9997	0.9374
ImageNet16-120	Latency	1.338%	0.520%	0.9999	0.9888
	Energy	10.13%	0.840%	0.9998	0.9388
Average Performance		6.096%	0.779%	0.9998	0.9633

C MINOR MODIFICATIONS ON FBNET SEARCH SPACE WHEN BENCHMARKING ON CIFAR-100

We demonstrate our modification on FBNet search space when benchmarking on CIFAR-100 (i.e., the setting in Section 5) via comparing the marco-architecture before and after such modification in Table 7.

Table 7: Left: the marco-architecture of the search space proposed in the FBNet (Wu et al., 2019) for ImageNet classification. Right: our modified search space to fit the input image size in CIFAR-100 dataset. In the tables, "TBS" means the layer type needs to be searched, Stride denotes the stride of the first block in the stage. The modified parameters are emphasized as bold characters.

Input Shape	Block	Filter#	Block#	Stride
$224^2 \times 3$	3×3 conv	16	1	2
$112^2 \times 16$	TBS	16	1	1
$112^2 \times 16$	TBS	24	4	2
$56^2 \times 24$	TBS	32	4	2
$28^2 \times 32$	TBS	64	4	2
$14^2 \times 64$	TBS	112	4	1
$14^2 \times 112$	TBS	184	4	2
$7^2 \times 184$	TBS	352	1	1
$7^2 \times 352$	1×1 conv	1984	1	1
$7^2 \times 1984$	7×7 avgpool	-	1	1
1504	fc	1000	1	-

Input Shape	Block	Filter#	Block#	Stride
$\mathbf{32}^2 \times 3$	3×3 conv	16	1	**1**
$\mathbf{32}^2 \times 16$	TBS	16	1	1
$\mathbf{32}^2 \times 16$	TBS	24	4	**1**
$\mathbf{32}^2 \times 24$	TBS	32	4	2
$\mathbf{16}^2 \times 32$	TBS	64	4	2
$\mathbf{8}^2 \times 64$	TBS	112	4	1
$\mathbf{8}^2 \times 112$	TBS	184	4	2
$\mathbf{4}^2 \times 184$	TBS	352	1	1
$\mathbf{4}^2 \times 352$	1×1 conv	1504	1	1
$\mathbf{4}^2 \times 1504$	$\mathbf{4} \times \mathbf{4}$ avgpool	-	1	1
1504	fc	100	1	-

D DETAILS IN THE PIPELINE TO COLLECT HARDWARE-COST

D.1 COLLECT PERFORMANCE ON EDGE GPU:

NVIDIA Edge GPU Jetson TX2 (Edge GPU) (NVIDIA Inc., a) is a commonly used commercial edge device, consisting of a quad-core Arm Cortex-A57, a dual-core NVIDIA Denver2, a 256-core Pascal GPU, and a 8GB 128-bit LPDDR4, for various deep learning applications including classification (Li et al., 2020), segmentation (Siam et al., 2018), and depth estimation (Wofk et al., 2019), targeting IoT and self-driving environments. Although widely-used TensorFlow (Abadi et al., 2016) and PyTorch (Paszke et al., 2019) can be used in Edge GPU directly, to achieve faster inference, TensorRT (NVIDIA Inc., b), a C++ library for high-performance inference on NVIDIA GPUs, is more commonly used as the runtime environment in Edge GPU when only benchmarking inference (Wang et al., 2019; NVIDIA Inc., c).

Therefore, we pre-set the Edge GPU in the max-N mode to make full use of the resource on it following (Wofk et al., 2019). When plugging Edge GPU into the hardware-cost collection pipeline, we first compile the PyTorch implementations of network architectures in both NAS-Bench-201 and FBNet spaces to TensorRT format models. In this way, we can benefit from the optimized inference implementation within TensorRF runtime environment. And then we can benchmark the architectures in Edge GPU to further measure the energy and latency using the sysfs (Patrick Mochel and Mike Murphy.) of the embedded INA3221 (Texas Instruments Inc.) power rails monitor.

D.2 COLLECT PERFORMANCE ON RASPI 4:

Raspberry Pi 4 (Raspi 4) (Raspberry Pi Limited.) is the latest Raspberry Pi device, which is a popular hardware platform for general purpose IoT applications (Zhao et al., 2015; Basu et al., 2020) and is able to support deep learning applications with specifically framework designs (Google LLC., f; Zhang et al., 2019; Geiger & Team, 2020). We choose the Raspi 4 type with a Broadcom BCM2711 SoC and a 4GB LPDDR4 (Raspberry Pi Limited.). Similar as Edge GPU, Raspi 4 can run architectures in the TensorFlow (Abadi et al., 2016), PyTorch (Paszke et al., 2019) or TensorFlow Lite (Google LLC., f) runtime environments. We utilize TensorFlow Lite (Google LLC., f) which can further boost the efficiency.

To collect hardware-cost operating on Respi 4, an official TensorFlow Lite interpreter is pre-configured in the Raspi 4, following the settings in (Google LLC., 2020). We benchmark the possible architectures in HW-NAS-Bench on Raspi 4 after compiling them to TensorFlow Lite (Abadi et al., 2016) format to measure the resulting latency.

D.3 COLLECT PERFORMANCE ON EDGE TPU:

Edge TPU (Google LLC., a) is a series of dedicated ASIC accelerators developed by Google for targeting AI inference at the edge, which can be used for classification, pose estimation, and segmentation (Xiong et al., 2020; Google LLC., b) with extremely high efficiency (e.g., $2.32\times$ more efficient than a single SOTA desktop GPU, GTX 2080 Ti, in terms of number of fixed-point operations per watt (Google LLC., d)). In our proposed collection pipeline, we choose the Dev Board (Google LLC., a) which provides the most functions among all products.

To collect hardware-cost in Edge TPU, all the architectures to be benchmarked will be converted to TensorFlow Lite (Google LLC., f) format first from Keras (Chollet et al., 2015) implementation. After that, an in-house compiler (Google LLC., c) will convert the TensorFlow Lite models into a more compressed format. This pipeline uses the least converting tools to make sure as many operations as possible are supported compared with others (e.g., from PyTorch-ONNX (Bai et al., 2020) implementation). Only latency is collected for Edge TPU since it lacks accurate embedded power rails monitor. We do not consider FBNet' search space for Edge TPU and more details are in Appendix A.

D.4 COLLECT PERFORMANCE ON PIXEL 3:

Pixel 3 (Google LLC., e) is one of the latest Pixel mobile phones that are widely used as the target platform by recent NAS works (Xiong et al., 2020; Howard et al., 2019; Tan et al., 2019) and machine learning framework benchmark (Google LLC., f). In our implementation, the Pixel 3 is pre-configured to use big cores following the setting in (Xiong et al., 2020; Tan et al., 2019). Similar to the case of Raspi 4, we first convert the possible architectures in the search spaces of our proposed HW-NAS-Bench into TensorFlow Lite format and then use the official benchmark binary files to measure the latency for each architecture.

D.5 COLLECT PERFORMANCE ON ASIC-EYERISS:

For hardware-cost data colection in ASIC, we consider Eyeriss (ASIC-Eyeriss) a SOTA ASIC accelerator (Chen et al., 2016). The Eyeriss chip features 168 processing elements (PEs) which are connected through a configurable dedicated on-chip network into a 2D array. A 128KB SRAM is shared by all PEs and further divided into multiple banks, each of which can be assigned to fit the input feature maps or partial sums. Thanks to these configurable hardware settings, we can deploy the optimal algorithm-to-hardware mappings on Eyeriss to minimize the energy or latency by exploiting data reuse for different layers.

In order to find the optimal mappings and evaluate the performance metrics on Eyeriss, we adopt SOTA performance simulators for DNN accelerators (1) Accelergy (Wu et al., 2019)+Timeloop (Parashar et al., 2019) and (2) DNN-Chip Predictor (Zhao et al., 2020). Both of the simulators can characterize the Eyeriss' architecture, do mapping exploration, and predict the energy cost and latency metrics. Given the Eyeriss accelerator and layer information (e.g, layer type, feature map size, kernel size) in NAS-Bench-201 and FBNet, Accelergy+Timeloop reports the en-

ergy cost and latency characterization through an integrated mapper that finds the optimal mapping for such layer when being executed in Eyeriss. The inputs to DNN-Chip Predictor are the same as Accelergy+Timeloop, except that we can set the optimization metric as energy/latency/energy-delay product. DNN-Chip Predictor schedules mapping for the optimization metric and generates the estimated hardware-cost. We report the average prediction from the two as the estimated hardware-cost of Eyeriss, and more details can be found in Appendix B.

D.6 COLLECT PERFORMANCE ON FPGA:

FPGA is a widely adopted AI acceleration platform which can offer higher flexibility in terms of hardware for accelerating AI applications. For collecting hardware-cost data in FPGA, we construct a SOTA chunk based pipeline structure (Zhang et al., 2018; Shen et al., 2017) as our FPGA implementation. By configuring multiple sub-accelerators (chunks) and assigning different layers to different sub-accelerators(chunks), we can balance the throughput and hardware resource consumption. To further free up our implantation's potential to reach the performance frontier across different architectures, we additionally configure hardware settings such as the number of PEs, interconnection method of PEs, and tiling/scheduling of the operations, which are commonly adopted by FPGA accelerators (Chen et al., 2017; Zhang et al., 2015; Yang et al., 2016). We then compile all the architectures using standard Vivado HLS toolflow (Xilinx Inc., a) and obtain the bottleneck latency, the maximum latency across all sub-accelerators (chunks) of the architectures on a Xilinx ZC706 development board with Zynq XC7045 SoC (Xilinx Inc., b).

To verify our implementation, we compare our implementation's performance with SOTA FPGA accelerators (Zhang et al., 2018; Xiao et al., 2017) given the same architecture and dataset as shown in Table 8. We can see that our implementation has SOTA performance and thus provides insightful and trusted hardware-cost estimation for the HW-NAS-Bench.

Table 8: Our implemented FPGA accelerators for HW-NAS-Bench vs. SOTA FPGA accelerators on VGG16@ImageNet using Zynq XC70Z45 as device.

	(Zhang et al., 2018)	(Xiao et al., 2017)	**Our Implementation**
Resource Utilization	680/900 DSP	824/900 DSP	723/900 DSP
Performance (GOP/s)	262	230	**291**