Unveiling Energy Efficiency in Deep Learning: Measurement, Prediction, and Scoring across Edge Devices

Xiaolong Tu[1] Anik Mallik[2] Dawei Chen[3] Kyungtae Han[3] Onur Altintas[3] Haoxin Wang[1] Jiang Xie[2]
[1]Georgia State University [2]The University of North Carolina at Charlotte [3]Toyota InfoTech Labs
{xtu1,haoxinwang}@gsu.edu,{amallik,linda.xie}@uncc.edu,{dawei.chen1,kt.han,onur.altintas}@toyota.com

Abstract

Today, deep learning optimization is primarily driven by research focused on achieving high inference accuracy and reducing latency. However, the energy efficiency aspect is often overlooked, possibly due to a lack of sustainability mindset in the field and the absence of a holistic energy dataset. In this paper, we conduct a threefold study, including energy measurement, prediction, and efficiency scoring, with an objective to foster transparency in power and energy consumption within deep learning across various edge devices. Firstly, we present a detailed, first-of-its-kind measurement study that uncovers the energy consumption characteristics of on-device deep learning. This study results in the creation of three extensive energy datasets for edge devices, covering a wide range of kernels, state-of-the-art DNN models, and popular AI applications. Secondly, we design and implement the first kernel-level energy predictors for edge devices based on our kernel-level energy dataset. Evaluation results demonstrate the ability of our predictors to provide consistent and accurate energy estimations on unseen DNN models. Lastly, we introduce two scoring metrics, PCS and IECS, developed to convert complex power and energy consumption data of an edge device into an easily understandable manner for edge device end-users. We hope our work can help shift the mindset of both end-users and the research community towards sustainability in edge computing, a principle that drives our research. Find data, code, and more up-to-date information at https://amai-gsu.github.io/DeepEn2023.

CCS Concepts: • **Computer systems organization** → **Embedded and cyber-physical systems**; • **Computing methodologies** → **Machine learning**.

SEC '23, December 06–09, 2023, Wilmington, DE

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Keywords: Edge AI, Deep Neural Network, Energy Consumption

ACM Reference Format:
Xiaolong Tu[1] Anik Mallik[2] Dawei Chen[3] Kyungtae Han[3] Onur Altintas[3] Haoxin Wang[1] Jiang Xie[2]. 2023. Unveiling Energy Efficiency in Deep Learning: Measurement, Prediction, and Scoring across Edge Devices. In *Proceedings of The Eighth ACM/IEEE Symposium on Edge Computing (SEC '23).* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Recently, there has been heavy investment in implementing various AI applications on mobile and edge devices, for instance, (1) *vision-based* AI applications, such as image classification [1–3], face recognition [4, 5], object detection and tracking [6–8], image super-resolution [9–11], segmentation [12], pose estimation [13], and gesture recognition [14]; (2) *natural language processing* (NLP) based applications, such as smart reply [15], question answering [16], language translation [17, 18], and sentiment analysis [19, 20]; and (3) *voice-based* applications, such as virtual-assistant [21], speech recognition [22], and sound classification [23].

Despite the remarkable advances in edge device capabilities such as functionality, computation power, and storage capacity, the limited energy capacity has been the major bottleneck in promoting advanced edge AI applications. On one hand, edge AI applications, particularly those that involve intensive computing resources such as deep learning algorithms, tend to consume a significant amount of energy [24, 25]. On the other hand, mobile and edge devices are typically powered solely by embedded batteries, so their energy capacity is significantly constrained by form factor requirements, safety considerations, manufacturing costs, and concerns on the environmental impact of the battery technology used. As a result, heavy battery usage of an application often results in low ratings or subpar user experience. A survey [26] finds that about 55% of users surveyed would give a negative review to a mobile application that consumes a lot of battery, indicating that energy consumption is a crucial aspect of the user experience that cannot be overlooked. These observations raise intuitive questions: *How can we identify the energy bottlenecks and optimize the energy efficiency of on-device deep learning for diverse edge devices? What are the primary factors that have a large impact on the*

energy consumption of deep neural network (DNN) executions, the core of on-device deep learning? Where is the energy spent inside a DNN execution? Answering these questions, however, is challenging, due to the lack of holistic understanding of the intricacies of power and energy consumption in DNN executions on edge devices. First and foremost, *we cannot optimize what cannot be measured.* The energy efficiency of an edge device is more than its AI hardware capability in isolation. Instead, it is coupled with the on-device deep learning software stack, whose net performance is shrouded beneath the DNN models and end-to-end processing pipeline of diverse edge AI applications. Second, *we cannot optimize what is under-appreciated or neglected in the design.* Most existing research and development in deep learning primarily aim to reduce inference latency and enhance accuracy, often neglecting to consider the impact on energy efficiency. As a result, it becomes crucial to strike a balance between improving energy efficiency and enhancing performance in on-device deep learning for modern edge devices.

In this paper, we study the problem of accurate energy measurement, prediction, and understandable scoring of on-device deep learning, and make three concrete contributions towards enabling *transparency of power and energy consumption inside on-device deep learning across diverse edge devices*.

First, we conduct the first detailed measurement study to accurately quantify the energy consumed by on-device deep learning across diverse modern edge devices. Our measurement study covers three dimensions, including the power and energy consumption of kernels, state-of-the-art (SOTA) DNN models, and widely-used edge AI applications. Our measurements reveal multiple key observations, which remain consistent across eight different measured edge devices. Overall, we measure and collect fine-grained power traces and accurate energy consumption data for (1) 16 types of kernels with 1,847 unique configurations, (2) nine SOTA DNN models with 50 variants each, and (3) six widely-used edge AI applications on eight commercial edge devices executed with mobile CPU and GPU. These measurements result in creation of three large-scale power and energy datasets, including the kernel-level, model-level, and application-level datasets for on-device deep learning on edge devices.

Second, based on our kernel-level energy dataset and the observations gained in the measurement study, we design and implement kernel-level energy predictors on both mobile CPU and GPU. To the best of our knowledge, this is the first energy predictor for on-device deep learning on commercial edge devices (e.g., modern smartphones), which can provide consistently accurate energy estimation on unseen DNN models. This offers an effective approach to extend our measurements and observations derived from a limited DNN model space to new DNN models, which enhances the extensibility of our measurement study.

Lastly, beyond valuing research that aims at improving the energy efficiency of on-device deep learning, it is crucial that our measurement study are accessible to a wide audience, such as end-users with non-technical backgrounds. For instance, presenting an energy efficiency score, ranging from 0 to 100, should be more straightforward and easier to understand than telling end-users that their device will consume 120.090 mJ per inference to run MobileNetv1 with CPUs. To this end, we develop two scoring metrics: *power consumption score (PCS)* and *inference energy consumption score (IECS)*. These two scoring metrics help to distill the power and energy efficiency of an edge device in an intuitive and understandable way. We present a complete scoring results for eight edge devices benchmarked by leveraging our application-level dataset.

2 Background and Challenges

2.1 Background

DNN models are the core of on-device deep learning and consume a major portion of both computational and energy resources on mobile and edge devices. A DNN model consists of a sequence of primitive operations, such as convolution2D (`conv`), depthwise convolution2D (`dwconv`), activations, pooling, and fully-connected (`fc`), which are organized into layers, allowing the network to learn complex patterns from input data. To enhance the computational efficiency of the DNN inference (i.e., to reduce inference latency and avoid redundant memory access), kernel fusion (or operator fusion) is a key optimization and has been incorporated in SOTA DNN execution frameworks, such as TVM [27], TFLite [28], and MNN [29]. For instance, three individual operations, `conv`, batch normalization (`bn`), and rectified linear unit (`relu`) can be fused into one composite operation, `conv+bn+relu`[1], to achieve inference acceleration on edge devices. This means the entire sequence can be processed as a single step, which reduces memory access (since intermediate results don't need to be written to and read from memory) and kernel launch overhead. Hence, given its crucial role in runtime optimization, a kernel is typically considered as the fundamental unit for scheduling and execution in deep learning frameworks, particularly on edge devices [30].

2.2 Challenges

C1: Accuracy. In order to optimize the energy efficiency of DNN executions on resource-constrained edge devices, it is crucial to gain a deep understanding of the energy consumption characteristics associated with various DNN models across different edge hardware platforms, such as mobile CPUs and GPUs. Consequently, the importance of conducting accurate measurement studies on real devices is becoming increasingly paramount. However, measuring accurate energy consumption on a real edge device is non-trivial. The challenges arise from two main observations: (1) existing energy profiling methods for mobile and edge devices, which

[1]In this paper, `+` represents kernel fusion.

Table 1. MobileNetv1 energy consumption.

	CPU		GPU	
	Energy	Error	Energy	Error
Built-in	132.420mJ	10.3%	19.254mJ	30.64%
Ground-truth	120.090mJ	-	27.760mJ	-

Table 2. MobileNetv1 individual kernel energy consumption.

Kernels	CPU		
	Built-in (mJ)	Ground-truth (mJ)	Error
conv+relu	3.914	3.984	1.76%
dwconv+relu	5.578	4.814	15.8%
conv+relu	8.020	7.739	3.62%
dwconv+relu	8.682	8.193	5.96%
conv+relu	2.649	2.422	9.36%
dwconv+relu	5.211	4.428	17.6%
conv+relu	1.225	0.930	31.8%
dwconv+relu	1.541	1.285	20.0%
conv+relu	2.030	1.643	23.5%
dwconv+relu	7.824	6.549	19.5%
conv+relu	3.450	2.933	17.6%
dwconv+relu	0.174	0.149	16.8%
conv+relu	1.179	0.972	21.3%
dwconv+relu	2.879	2.448	17.6%
conv+relu	12.394	11.324	9.45%
dwconv+relu	0.524	0.466	12.4%
conv+relu	14.112	12.976	8.76%
dwconv+relu	0.906	0.771	17.5%
conv+relu	12.065	11.095	8.74%
dwconv+relu	1.108	0.944	17.3%
conv+relu	14.446	13.327	8.39%
dwconv+relu	0.409	0.367	11.5%
conv+relu	12.240	11.357	7.77%
dwconv+relu	0.299	0.267	11.7%
conv+relu	4.349	4.019	8.20%
dwconv+relu	0.110	0.093	17.6%
conv+relu	4.353	3.902	11.6%
global-pool	0.071	0.062	14.4%
fully connected	0.664	0.636	4.42%

error $\leq$ 5% 5% < error $\leq$ 10%
10% < error $\leq$ 20% error > 20%

rely on built-in current sensors, cannot capture power consumption at high time granularity (i.e., less than 100 ms); and (2) the growing level of integration in the electronic circuits of edge devices presents challenges when attempting to connect them with an external power monitor.

First, most SOTA DNN models can achieve inference latencies of 10 to 200 ms when executed on mobile CPUs. These latencies can be significantly reduced to a range of 1 to 50 ms when executed on mobile GPUs [30]. On the other hand, a DNN model usually consists of tens or hundreds of kernels that run sequentially on the edge device [30–32], each potentially having an execution time of less than a millisecond. Therefore, to accurately capture the instantaneous power variations within a DNN inference, which includes the precise power consumption of individual kernels, an ideal power sampling rate should be less than 1 ms. However, we have observed that existing edge devices, such as smartphones, typically have built-in current sensors (e.g., fuel gauge) with a time-granularity of approximately 100 ms to 1 second. This restricts the sampling rate at which the sensors can measure the power drawn by the device to $1-10$ times per second. This indicates that the existing built-in current sensors cannot fully capture the fine-grained, kernel-level power variations within a DNN inference on the edge device, resulting in inaccurate measurements.

We have conducted a measurement study on a real device, Huawei P40 Lite, to investigate the extent of this discrepancy compared to the ground-truth power and energy consumption[2]. As shown in Tables 1 and 2, measurements dependent on the device's built-in current sensor produce large errors in both the overall DNN model (10.3% – 30.64%) and individual kernel (1.76% – 31.8%) energy consumption[3]. Moreover, as we show in Section 4, using the energy dataset created by a built-in current sensor to train an energy predictor results in consistently poor prediction accuracy. In addition, Fig. 1 demonstrates that the built-in current sensor also fails to capture the characteristics of power variations among kernels within a DNN inference. For instance, there is usually a sudden power rise at the start of `conv` executions, and a sudden power drop in most `dwconvs`.

Consequently, these observations indicate that existing energy profiling solutions for mobile and edge devices that heavily rely on the built-in current sensor may fail to offer accurate power measurements for DNN executions (e.g., power profiler [34], reading virtual file `current_now` *from* `/sys/class/power_supply/battery/` *[35], and reading battery level drops from* `ACTION_BATTERY_CHANGED` *[36, 37]).*

Second, one of the common methods to measure accurate and fine-grained power consumption for mobile and edge devices in the research community is to connect the device to an external power monitor with a high sampling rate [38–41]. However, we find that connecting newer commercial devices, especially smartphones released after 2017, to an external power monitor requires significant effort due to the increasing level of integration of their electronic circuits. Fig. 2 compares the battery connector in an older Samsung smartphone, the Galaxy S5, released in 2014, with that of a newer Samsung model, the Galaxy S20, released in 2020. The battery connector in a smartphone is used to connect a battery to its integrated circuit board. Older smartphones,

[2]In this paper, the ground-truth power and energy consumption is measured by connecting the real device to the Monsoon power monitor [33].
[3]The energy consumption is calculated by multiplying the measured power consumption by the model/kernel inference latency. To ensure that the energy consumption errors are primarily caused by the power measurement inaccuracy, we use the ground-truth latency when calculating the energy consumption in the built-in current sensor measurements.

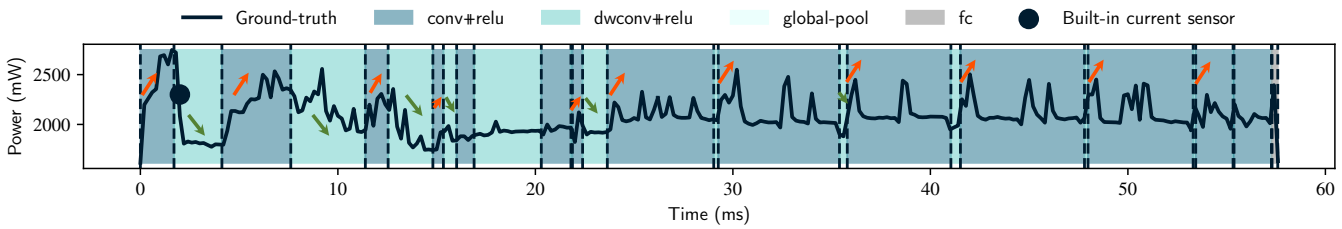
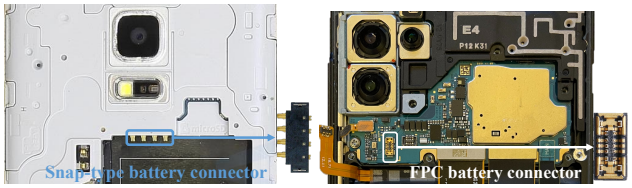


Figure 1. Comparison of time-granularity between the device's built-in current sensor and external power monitor. Tested mobile device: Huawei P40 Lite



(a) Samsung Galaxy S5 with a snap-type battery connector (2014)

(b) Samsung Galaxy S20 with an FPC battery connector (2020)

Figure 2. Comparison between an older snap-type battery connector and a modern FPC connector.

including the Galaxy S5, use a specific type of battery connector known as a "snap-type connector". Featuring four metal prongs, as shown in Fig. 2(a), the snap-type connector allows for easy identification of the positive and negative terminals and enables connection to an external power monitor. However, advanced smartphones such as the Galaxy S20 use a proprietary, tiny, and delicate Flexible Printed Circuit (FPC) battery connector, as shown in Fig. 2(b). The FPC connector's small size and delicate construction make it challenging to work with, requiring specialized tools and expertise to connect it to an external power monitor that offers higher accuracy. This might be one of the main reasons that recent research papers typically rely on the built-in current sensor for measuring coarse-grained power consumption on mobile and edge devices [35, 36, 42].

Consequently, although external power monitors with high sampling rates show promising accuracy in measurement, the challenges associated with connecting newer commercial devices to such external monitors can be a significant barrier.

C2: Extensibility. In recent years, we have witnessed a significant surge in the development of DNNs, particularly those specifically designed to address the increasing demand for mobile and edge devices. This has led to the invention of several milestone Convolutional Neural Network (CNN) models, including, but not limited to, AlexNet, DenseNet, GoogleNet, and MobileNet. Moreover, the advent of Neural Architecture Search (NAS) has accelerated advancements in the design and optimization of novel CNN models by automating the design process and facilitating customization. While measuring the energy consumption of DNN inferences on real devices is highly desirable for various tasks, such as serving as a ground-truth dataset for training energy predictors for on-device deep learning, it is practically infeasible and excessively time-consuming to measure all DNN models individually. For example, we spend approximately 2.1 days to measure 200 models on a single device, while ProxylessNAS [31] explores nearly 0.3 million models in a single round of search. This predicament leads to a critical challenge: how can we ensure the observations and measurements derived from a limited DNN model space can be extensible to new (unseen) DNN models?

Consequently, the huge and expansive model-design space significantly challenges the extensibility of energy measurements on real mobile and edge devices.

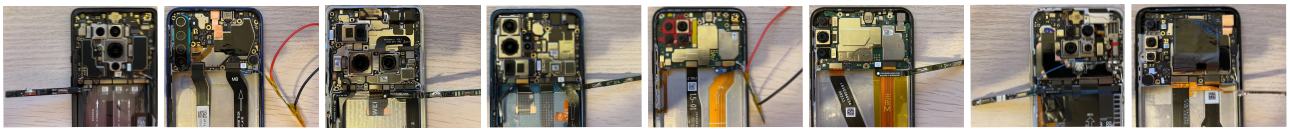
C3: Understandability. In addition to valuing research aimed at reducing the energy consumption of DNN executions, it is essential that our measurement study is accessible to a wide audience, such as end-users with non-technical backgrounds. As we presented in Section 1, end-users consider the energy efficiency of their devices as one of the most critical factors. Results that are easy to understand can help end-users make informed purchasing decisions. For instance, presenting an energy efficiency score, ranging from 0 to 100, could be more straightforward and easier to understand than simply telling the end-user that the device will consume 120.090 mJ per inference to run MobileNetv1 with CPUs. Consequently, end-users can compare different devices and choose the one that best suits their needs. On the other hand, for the research community, an easily adoptable measurement method or energy dataset can accelerate progress in developing energy-efficient DNN models, designing energy predictors, or searching for DNN models with energy/power constraints within a vast model-design space. Currently, due to a lack of sustainability mindset, the optimization of DNNs is primarily driven by research focused on achieving high inference accuracy and minimizing latency.

We hope our work can help shift the mindset of both end-users and the research community towards sustainability, a principle that drives our research.

Table 3. Specifications of Measured Edge Devices and Chipsets

Model		OnePlus 8 Pro	Xiaomi Redmi Note8	Huawei Mate40 Pro	Huawei P40 Pro	Huawei P40 Lite	Huawei P40 Lite E	Xiaomi Redmi K30 Ultra	Motorola One Macro
SoC		SD 865	SD 665	Kirin 9000	Kirin 990 5G	Kirin 810	Kirin 710F	Dimensity1000+	Helio P70
Vendor		Qualcomm	Qualcomm	HiSilicon	HiSilicon	HiSilicon	HiSilicon	MediaTek	MediaTek
CPU	M	A77+A55	A73+A53	A77+A55	A76+A55	A76+A55	A73+A53	A77+A55	A73+A53
	C1	4+4	4+4	4+4	4+4	2+6	4+4	4+4	4+4
	F1	2.84 GHz	2.0 GHz	3.13 GHz	2.86 GHz	2.27 GHz	2.2 GHz	2.6 GHz	2.1 GHz
GPU		Adreno 650	Adreno 610	Mali G78	Mali G76	Mali G52	Mali G51	Mali G77	Mali G72
Dedicated AI accelerator		Hexagon698 DSP	Hexagon686 DSP	Ascend Lite+Tiny NPU Da Vinci 2.0	Lite+Tiny NPU Da Vinci	D100Lite NPU Da Vinci	None	MediaTek3.0 APU	MediaTek APU
OS (Android)		10	10	10	10	10	10	10	9
NNAPI support		Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Battery	C2	4510 mAh	4500 mAh	4400 mAh	4200 mAh	4200 mAh	4000 mAh	4500 mAh	4000 mAh
	R	No	No	No	No	No	No	No	No
Class		Flag	Mid-range	Flag	Flag	Mid-range	Mid-range	Flag	Mid-range

* SD: Snapdragon, M: Microarchitecture, C1: CPU Cores, F1: Maximum Frequency, S: Display Size, C2: Battery Capacity, and R: If battery is removable.



(a) OnePlus 8 Pro (b) Redmi Note8 (c) Mate40 Pro (d) P40 Pro (e) P40 Lite (f) P40 Lite E (g) K30 Ultra (h) One Macro

Figure 3. Measured devices with segregated BMS chips.

3 Energy Measurement and Dataset

We conduct a measurement study and create three energy datasets: kernel-, model-, and application-level datasets. Overall, we collect fine-grained power traces and accurate energy consumption data for (1) 16 types of kernels with 1,847 unique configurations, (2) nine SOTA DNN models with 50 variants each, and (3) six widely-used edge AI applications on eight commercial edge devices.

3.1 Energy Measurement

We develop a reproducible energy measurement methodology, which facilitates the collection of accurate and fine-grained power consumption of kernels, DNN models, and end-to-end edge AI applications on modern edge devices.

Proposed solution for C1: accuracy. As discussed in Section 2, although external power monitors demonstrate promising accuracy and time granularity for tracing power variations within a DNN execution, establishing a physical connection between a modern edge device with an FPC battery connector and such a monitor is nontrivial. To address this challenge, we first use a mechanic mobile device DC power cable [43] that is designed to fit multiple device models, including those with FPC connectors, to connect the tested devices to an external power monitor. This method requires little effort on the part of the benchmarking researchers. However, we find that the tested devices cannot boot due to the lack of proprietary battery management system (BMS) chips. BMS is an electronic system that manages and monitors the performance and safety of a device battery, and is typically attached to the battery in modern edge devices. The device OS must communicate with the proprietary BMS to check the status and safety of the battery before allowing the phone to power on. Hence, the device cannot boot if its battery is disconnected or an unauthorized battery is connected. We have studied multiple alternatives to address this issue, and we find that the most effective method is to segregate the BMS chip from the device battery without tearing it down, and use it as a bridge to connect the device to the external power monitor. This method strikes a good balance between the effort required and reproducibility. We have validated this method on eight different modern smartphones, as illustrated in Fig. 3. All of the tested devices are able to power on with full functionality using this method.

We develop a detailed documentation to provide step-by-step instructions on how to implement this method on other modern edge devices, which will help the community to reproduce our measurements and apply this technique to their own research or practical applications.

Rules for measurement. Since the power consumption of mobile and edge devices can be easily influenced by the environment, such as heat dissipation and background activities, it is crucial to create specific rules for measurement. These rules can bolster the consistency and reliability of power measurements across diverse devices and testing conditions. By controlling and accounting for environmental factors, we can mitigate their influence on our power data collection, and thus gain a more accurate understanding of the inherent power and energy consumption characteristics of DNN executions. To this end, we establish a set of rules for power measurements. Through our observation, these rules effectively ensure consistency and reproducibility[4].

- Disable adaptive brightness and set the display to the lowest brightness level.
- Turn off WiFi, Bluetooth, cellular network, and Near-Field Communication (NFC) interfaces to minimize the interference on the accuracy of power measurements.
- Shut down and disable any background applications and services to minimize the interference on the accuracy of measurements.
- Conduct measurements with a room-temperature between 20 and 25°C.
- Maintain an air gap with proper ventilation to regulate the temperature of the smartphone and prevent runtime thermal throttling.
- Configure the screen refresh rate to 60 Hz.
- Configure the camera sample rate to 15 frames per second, if the executed edge AI applications require the use of the device camera.
- Set up a 2-minute cooldown interval between individual tests to allow the device to cooldown.

Devices and tools. We select eight modern edge devices with eight distinct mobile SoCs that include at least one high-end and one mid-range SoC from leading chipset vendors, such as Qualcomm, HiSilicon, and MediaTek. Their specifications are summarized in Table 3. The selected mobile SoCs can serve as representative examples of advanced and widely used mobile AI silicons in the past two years. Unless stated, all power consumption data are measured by the Monsoon power monitor with a 5000 Hz sampling rate. Note that other power monitors with a sampling rate under a millisecond are also applicable. The latency of DNN inferences, including both model-level and kernel-level latencies, is measured by the TFLite `benchmark tool` [44].

[4]Although understanding how the power consumption of DNN executions may vary with noisy background activities is important (since it is close to practical use cases), it is equally crucial to isolate and understand the intrinsic power characteristics of the DNNs, independent of these variations. This is one of the primary goals of our measurement study in this paper.

3.2 Energy Dataset

Kernel-level. As we introduced in Section 2, kernels constitute the fundamental units of execution in deep learning frameworks, with their types and configuration parameters significantly influencing the energy consumption during DNN executions. Table 4 illustrates that `conv+bn+relu` kernels typically consume more energy than other kernel types. Furthermore, the configuration for each kernel type varies. For `conv+bn+relu` and `dwconv+bn+relu` kernels, the primary configurations includes input height and width (HW)[5], input channel number (C_{in}), output channel number (C_{out}), kernel size (KS), and stride (S). Table 5 presents a comparison of the energy consumption between two `conv+bn+relu` kernels with different configurations, both run on a mobile CPU. One kernel configuration consumes a considerable 125.232mJ of energy, whereas the other expends a mere 0.064mJ. As a result, examining the impact of kernel configurations on energy consumption lays the foundation for a comprehensive understanding of energy consumption during DNN executions on edge devices.

To this end, we present our kernel-level energy dataset collected from real edge devices. To build the dataset, as presented in Table 4, we initially generate a large number of kernels with a variety of types (16 types for CPU and 10 types for GPU) featuring a range of configurations in the `tflite` format (e.g., 1032 `conv+bn+relu` and 349 `dwconv+bn+relu` kernels). The number of sampled configurations for each kernel type hinges on two main factors: its configuration dimension and its impact on the overall energy consumption during DNN executions (e.g., we observe that the `conv+bn+relu` kernel accounts for more than 70% of the total energy consumption in most SOTA CNN models on edge devices). These kernel configurations are randomly sampled in accordance with the sampling strategy proposed in [30]. Then, we measure the average power consumption and inference latency for each generated kernel running on individual edge devices. Each power and latency value is the average of at least 100 inference runs. We conduct these measurements independently on both CPUs and GPUs. As shown in Table 4, our kernel-level energy dataset spans a broad spectrum with different levels of energy consumption.

In Fig. 4, we seek to investigate how the five configurations (i.e., HW, C_{in}, C_{out}, KS, and S) impact the energy consumption of `conv+bn+relu`. In each evaluation, we vary a single configuration (e.g., HW) while maintaining the other four constants. The results reveal that the relationship between the energy consumption and the configurations is non-linear. As illustrated in Fig. 4(a), the energy consumption demonstrates a progressive increase with the growth

[5]In CNN models, input height usually is equal to input width.

Table 4. Measured kernels per device in our kernel-level dataset.

Kernels	Energy Consumption (mJ)		# Measured kernels		Avg. FLOPs (M)	Configurations
	CPU min - max	GPU min - max	CPU	GPU		
conv+bn+relu	0.002 - 1200.083	0.002 - 120.152	1032	1032	250.137	$(HW, C_{in}, C_{out}, KS, S)$
dwconv+bn+relu	0.022 - 222.609	0.016 - 0.658	349	349	28.364	(HW, C_{in}, KS, S)
bn+relu	0.002 - 161.334	0.001 - 14.594	100	100	4.710	(HW, C_{in})
relu	0.001 - 141.029	0.003 - 6.86	46	46	7.983	(HW, C_{in})
avgpool	0.066 - 7.711	0.034 - 1.142	28	28	0.670	(HW, C_{in}, KS, S)
maxpool	0.054 - 7.779	0.032 - 1.214	28	28	0.521	(HW, C_{in}, KS, S)
fc	0.038 - 94.639	-	24	-	14.744	(C_{in}, C_{out})
concat	0.001 - 42.826	0.066 - 3.428	142	142	0	$(HW, C_{in1}, C_{in2}, C_{in3}, C_{in4})$
others	0.001 - 132.861	0.003 - 10.163	98	72	-	(HW, C_{in})

Table 5. Energy consumption of conv+bn+relu kernels with different configurations on mobile CPU.

	$(HW, C_{in}, C_{out}, KS, S)$	
	(112, 64, 128, 3, 1)	(28, 22, 22, 1, 1)
Energy (mJ)	125.232	0.064

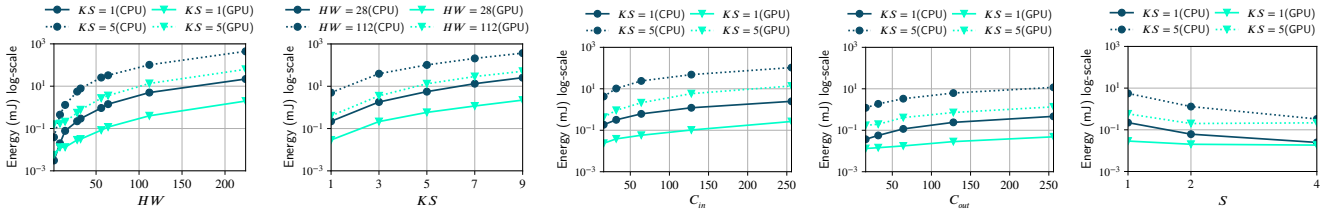
of HW. For instance, when running on the mobile CPU, the energy consumption of conv+bn+relu increases by approximately 1.85× (0.077mJ to 0.22mJ), 3.2× (0.22mJ to 0.93mJ), 4.37× (from 0.93mJ to 5.0mJ), 3.36× (5.0mJ to 21.81mJ), as HW doubles from 14 to 28, 28 to 56, 56 to 112, and 112 to 224, respectively. While operating on the mobile GPU, the energy consumption of the conv+bn+relu exhibits a similar trend but at a different rate. In this case, its energy consumption increases by roughly 1.21× (0.013mJ to 0.029mJ), 1.89× (0.029mJ to 0.083mJ), 3.79× (0.083mJ to 0.399mJ), 3.98× (0.399mJ to 1.988mJ) when HW doubles from 14 to 28, 28 to 56, 56 to 112, and 112 to 224, respectively. Moreover, we find that KS has the most significant impact on the energy consumption of conv+bn+relu. This is because the majority of energy consumption of kernel conv+bn+relu is attributed to convolutional layer. Within the convolutional layer, KS has the most significant impact due to its quadratic relationship with computational cost, while other parameters have a linear relationship. Specifically, when doubling each of the configuration, KS (from 3 to 5), HW (from 14 to 28), C_{in} (from 128 to 256), and C_{out} (from 128 to 256), the corresponding increases in energy consumption are approximately 2.08×, 1.85×, 1.05×, and 1.18× respectively. This finding demonstrates the disproportionate influence of KS on energy consumption relative to the other parameters.

Insights: the above observations underscore the importance of adaptive configuration selection in enhancing the energy efficiency of DNNs on edge devices. Given that our kernel-level dataset covers a wide range of configurations, each associated with an energy consumption label, it can serve as a valuable resource for guiding the selection of optimal configurations, searching for energy-efficient kernel configurations that meet specific energy constraints, and training kernel-level energy predictors (present in Section 4).

Fig. 5 presents a comparison of the energy consumption between running conv+bn+relu with identical configurations on an edge device's CPU and GPU. Interestingly, we find that using the mobile GPU for executing the conv+bn+relu kernel does not always result in energy savings when compared to running the same kernel on a mobile CPU, especially when the HW and KS parameters are on the lower side. For instance, when testing on the Huawei P40 Pro with the kernel configurations of $HW = 1$, $KS = 1$, $C_{in} = 480$, and $C_{out} = 20$, we find that the energy consumed by the GPU exceeds that of the CPU by more than a factor of 6.6. While the magnitude of this difference may vary across different edge devices, the overall pattern of increased energy consumption on the GPU under these conditions appears to be consistent. Typically, GPUs are more energy-efficient than CPUs as they exhibit lower inference latency, especially for large kernels that require high computational power. For small kernels, however, the inference latency on GPUs and CPUs does not show a significant difference, and GPUs might be less energy-efficient than CPUs. This is because the power consumption of GPUs is usually higher than that of CPUs, attributed to their greater I/O bandwidth and multiple cores designed for parallel computing [45].

Insights: This observation is crucial for designing effective kernel execution scheduling strategies on edge devices. Rather than only considering the type of kernel, the specific configuration of the kernel should also be taken into account when deciding where to execute it (e.g., on the mobile CPU or GPU).

In addition, our kernel-level dataset includes fine-grained power traces for each individual kernel, referred to as *power slices* in this paper. These collected power slices provide valuable insights for analyzing intra-kernel power variations. One of the primary observations in power slices is that the intra-kernel power variation exhibits a "high-initial, flat-later" pattern, illustrated in Fig. 6, when the kernel is



(a) $C_{in}/C_{out}/S$=20/120/1 (b) $C_{in}/C_{out}/S$=20/120/1 (c) $HW/C_{out}/S$=28/120/1 (d) $HW/C_{in}/S$=28/20/1 (e) $HW/C_{in}/C_{out}$=28/20/120

Figure 4. Energy consumption of `conv+bn+relu` vs. kernel configurations.

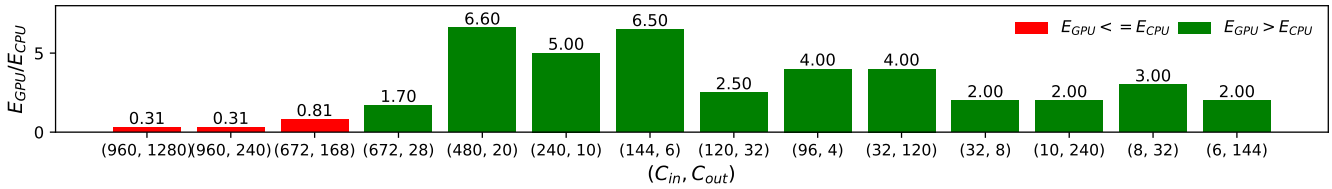


Figure 5. Comparison of energy consumption of `conv+bn+relu` with identical configurations on mobile CPU and GPU ($HW = 1, KS = 1, S = 1$, measured device: Huwei P40 Pro). Using the mobile GPU to execute the kernel does not always save the device's energy compared to using the mobile CPU.

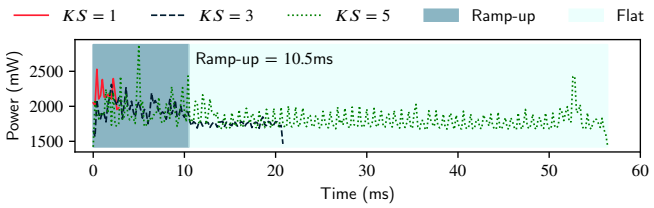
executed on a mobile CPU and the execution time exceeds a certain threshold. Fig. 6(a) reveals an initial power surge at the beginning of kernel execution on Huawei P40 Pro, equipped with a Kirin 990 5G chipset. This ramp-up phase continues for approximately 10.5ms. Following the initial ramp-up, the power consumption settles into a more consistent, flatter profile that persists until the end of the kernel's execution. We conduct validations across varying kernel configurations, with varying execution time, as well as on various edge devices to ascertain the consistency of this observation. We find the same pattern on the measured devices, as demonstrated in Figs. 6(b) and 6(c). Interestingly, devices powered by chipsets from the same vendor (e.g., Kirin 990 5G and Kirin 810) exhibit a nearly identical ramp-up time (10.5ms and 10.2ms), while the Snapdragon 855's[6] ramp-up time is around 6.2ms. The "high-initial, flat-later" pattern primarily arises due to power management techniques implemented in modern processors on edge devices. For instance, the Dynamic Voltage and Frequency Scaling (DVFS) technique can dynamically adjust a processor's voltage and frequency during runtime, based on computational demands. At the beginning of a computationally intensive kernel execution, DVFS may increase the frequency to ensure the task's timely completion. It then lowers the frequency once the task becomes more manageable, resulting in a relatively flat power consumption profile. The variation in ramp-up times among different chipsets and vendors can be attributed to the unique DVFS strategies they employ.

[6]This device is not listed in Table 3.

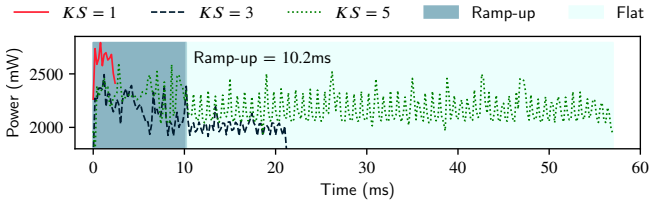
Table 6. Measured DNN models per device in our model-level dataset.

Models	Energy consumption (mJ) CPU min - max	GPU min - max	Avg. FLOPs (M)
AlexNets	36.97 - 355.58	7.69 - 91.80	815
DenseNets	231.93 - 488.87	66.21 - 133.58	1760
GoogleNets	145.03 - 262.45	52.66 - 90.04	1535
MobileNetv1s	53.59 - 136.79	17.36 - 42.44	519
MobileNetv2s	30.85 - 175.07	8.81 - 48.35	419
ProxylessNASs	58.34 - 162.11	17.70 - 49.29	526
ResNet18s	251.52 - 1432.67	64.19 - 391.97	3888
ShuffleNetv2s	25.26 - 81.41	-	319
SqueezeNets	92.55 - 388.16	34.55 - 134.65	1486

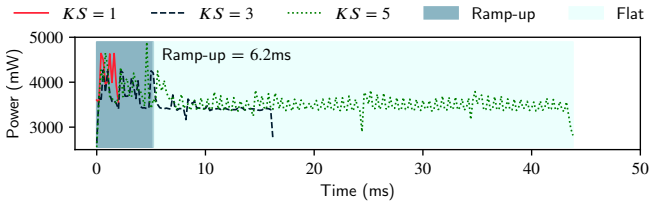
Insights: The ramp-up time can negatively impact power and energy efficiency on edge devices, particularly when executing kernels with relatively small configurations (where their execution time are less than the ramp-up time). As illustrated in Fig. 6(b), the ramp-up phase causes the `conv+bn+relu` *kernel with* $HW = 112$, $C_{in} = 20$, $C_{out} = 120$, $S = 1$ *to consume* 25.3% *and* 19.6% *more power than kernels with larger KSs, specifically 3 and 5. Nevertheless, kernels with smaller configurations are often preferred for implementation on edge devices to save computational resources. This highlights the critical importance of optimizing the ramp-up phase for edge devices. For instance, if the aforementioned kernel with* $KS = 1$ *can be executed directly in the flat phase, it can result in a reduction of energy consumption by* 23.1%.



(a) HiSilicon Kirin 990 5G (4 × CortexA76 + 4 × CortexA55 CPUs)



(b) HiSilicon Kirin 810 (2 × CortexA76 + 6 × CortexA55 CPUs)



(c) Qualcomm Snapdragon 855 (4 × CortexA76 + 4 × CortexA55 CPUs)

Figure 6. Measured fine-grained power slices for `conv+bn+relu` with $HW = 112, C_{in} = 20, C_{out} = 120, S = 1$. The intra-kernel power variation exhibits a "high-initial, flat-later" pattern.

Model-level. We introduce our model-level energy dataset, which collects nine SOTA DNN models. These models represent a mix of both manually-designed and NAS-derived models, each with distinct kernel types and configurations. For each model, we generate 50 variants for conducting power and energy measurements by re-sampling the C_{out} and KS for each layer. Specifically, we randomly sample the new output channel number from a range of 20% to 180% of the original C_{out}, while the KS is sampled from the set of values: $\{1, 3, 5, 7, 9\}$. Table 6 summarizes the details of the measured DNN models in the dataset. In general, running these models on mobile GPUs results in an energy consumption reduction of approximately 49% to 79%, compared to the execution on mobile CPUs.

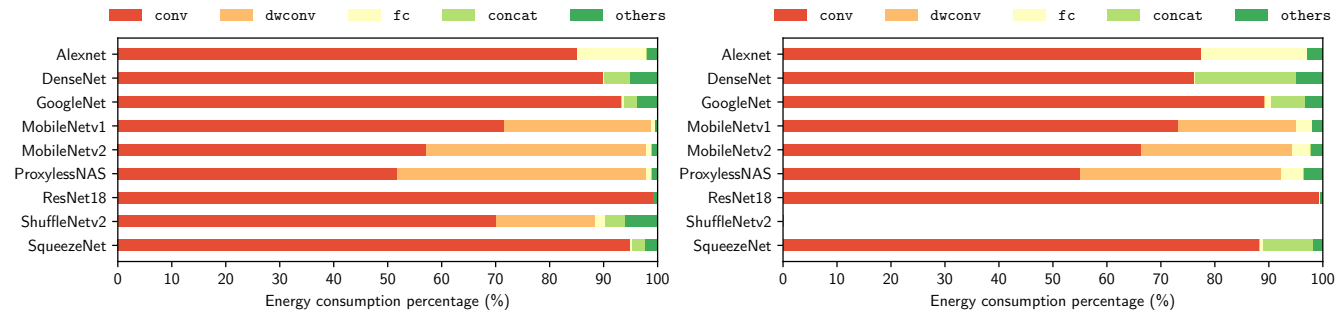
Fig. 7 presents the energy consumption breakdown of individual models by kernel types. The four kernel types that consume the most energy are `conv+bn+relu`, `dwconv+bn+relu`, `fc`, and `concat`. They account for 79.27%, 14.79%, 2.03%, and 1.5% of the total model energy consumption on mobile CPUs, respectively. On mobile GPUs, these kernels represent 78.17%, 10.91%, 4.01%, and 4.28% of the total model energy consumption. Furthermore, in most models, `conv+bn+relu` and `dwconv+bn+relu` account for the main energy percentages. On average, `conv+bn+relu` and `dwconv+bn+relu` take 93.97% and 87.74% of the total model energy consumption on the mobile CPU and GPU, respectively.

Table 7. Kernel configurations of two AlexNets.

Kernels	Configurations	
	AlexNet 1 (Fig. 8(a))	AlexNet 2 (Fig 8(b))
`conv+relu 1`	(224, 3, 89, 5, 4)	(224, 3, 70, 7, 4)
`maxpool 1`	(224, 89, 3, 2)	(55, 70, 3, 2)
`conv+relu 2`	(28, 89, 153, 7, 1)	(28, 70, 115, 7, 1)
`maxpool 2`	(28, 153, 3, 2)	(28, 115, 3, 2)
`conv+relu 3`	(13, 153, 460, 5, 1)	(13, 115, 345, 5, 1)
`conv+relu 4`	(13, 460, 230, 1, 1)	(13, 345, 128, 5, 1)
`conv+relu 5`	(13, 230, 204, 7, 1)	(13, 128, 307, 3, 1)
`maxpool 3`	(13, 204, 3, 2)	(13, 307, 3, 2)
`global-pool 1`	(1, 204)	(1, 307)
`fc 1`	(204, 3686)	(307, 3686)
`fc 2`	(3686, 6144)	(3686, 6963)
`fc 3`	(3686, 1000)	(3686, 1000)
Total energy (mJ)	242.888	151.414

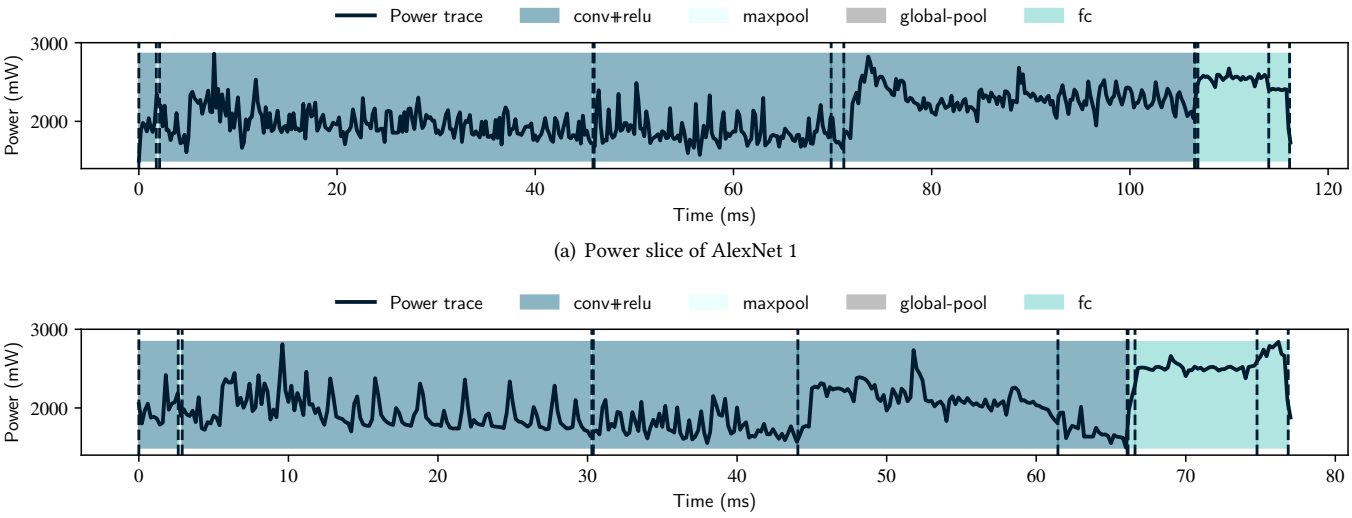
In addition, similar to the kernel-level dataset, our model-level dataset collects fine-grained power slices for all the measured DNN models. For instance, Fig. 8 illustrates the measured power slices of two AlexNets with distinct kernel configurations, whose specific configurations are detailed in Table 7. These model-level power slices offer (1) a holistic view of the precise power variations associated with each kernel within the DNN model, (2) the temporal and sequential aspects of kernel executions, and (3) a visual approach to easily identify the power and energy bottlenecks within a specific DNN model.

Applications-level. While the kernel- and model-level datasets can be beneficial for researchers and developers in understanding, modelling, and optimizing power and energy efficiency of DNN executions, end-users generally have a greater interest in the energy consumption of those frequently used AI applications on their devices. This is because the application's energy efficiency directly affects device's battery life, which is critical to the user experience. To this end, we create an application-level dataset, which uncovers the end-to-end energy consumption of six popular edge AI applications, covering three main categories: vision-based (object detection, image classification, super resolution, and image segmentation), NLP-based (natural language question answering), and voice-based applications (speech recognition). As shown in Table 8, we measure the power and energy consumption of each application with multiple reference DNN models that operate under four distinct computational settings, including CPU with a single thread, CPU with four threads, GPU delegate, and the NNAPI delegate. The dataset can serve as a resource for exploring the energy consumption distribution throughout the end-to-end processing pipeline of an edge AI application. For example, we can use the dataset



(a) Mobile CPU (b) Mobile GPU

Figure 7. DNN model energy consumption percentage breakdown. The top four most energy-consuming kernel types are `conv+bn+relu` (conv), `dwconv+bn+relu` (dwconv), `fc`, and `concat`.



(a) Power slice of AlexNet 1

(b) Power slice of AlexNet 2

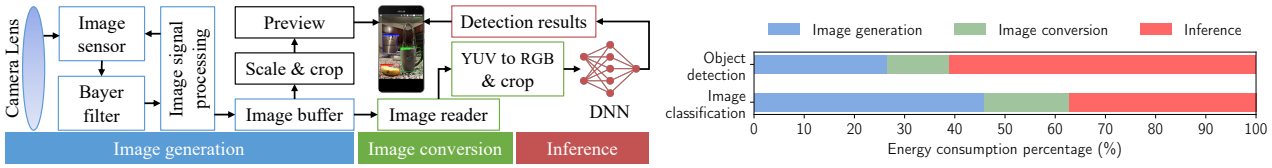
Figure 8. The model-level fine-grained power slices provided by our dataset can offer (1) a holistic view of the precise power variations associated with each kernel within the DNN model, (2) the temporal and sequential aspects of kernel executions, and (3) a visual approach to easily identify the power and energy bottlenecks within a specific DNN model.

to examine the energy consumed in generating image frames, converting these frames from YUV to RGB, and conducting DNN inference within an object detection application. Fig. 9 depicts the energy consumption breakdown based on the processing phases in the object detection. It demonstrates that our application-level dataset can provide interpretable observations for comprehending who is the primary energy consumer in the end-to-end edge AI application. Additionally, the application-level dataset provides essential inputs for our edge device scoring system (present in Section 5). Due to the page limit, we will not present additional measurement results in this paper.

Time cost. Finally, in Table 9, we report the time cost associated with performing measurements and creating our datasets. On a single edge device, we spend 23.1, 4.7, and 1.5 days, respectively, on (1) measuring the power and energy consumption of all the generated kernels, DNN models, and edge AI applications, and (2) creating the corresponding power and energy datasets. We will open-source our datasets and code for other researchers and developers. Collectively, we anticipate that the community will collaborate to create a larger scale energy dataset for a variety of edge devices.

4 Energy Prediction

In this section, we present our proposed solution to address **C2: extensibility**. To extend the applicability of our measurement study to a wider variety of DNN models, including those not present in our dataset, we design and implement a kernel-level energy predictor which can accurately predict the energy consumption of new DNN models on edge



(a) End-to-end processing pipeline for object detection and image classification [46]

(b) Energy consumption percentage breakdown

Figure 9. End-to-end energy consumption breakdown for object detection and image classification (application-level dataset).

Table 8. Measured edge AI applications per device in our application-level dataset.

Category	Application	No.	Reference DNN models	Delegate				Model size (MB)
				CPU1	CPU4	GPU	NNAPI	
Vision-based	Image detection	DNN1	MobileNetv2, FP32, 300 × 300 pixels	✓	✓		✓	24.2
		DNN2	MobileNetv2, INT8, 300 × 300 pixels	✓	✓		✓	6.9
		DNN3	MobileNetv2, FP32, 640 × 640 pixels	✓	✓		✓	12.3
		DNN4	MobileNetv2, INT8, 640 × 640 pixels	✓	✓		✓	4.5
	Image classification	DNN5	EfficientNet, FP32, 224 × 224 pixels	✓	✓	✓	✓	18.6
		DNN6	EfficientNet, INT8, 224 × 224 pixels	✓	✓		✓	5.4
		DNN7	MobileNetv1, FP32, 224 × 224 pixels	✓	✓	✓	✓	4.3
		DNN8	MobileNetv1, INT8, 224 × 224 pixels	✓	✓		✓	16.9
	Super resolution	DNN9	ESRGAN [47], FP32, 50 × 50 pixels	✓		✓		5
	Image segmentation	DNN10	DeepLabv3 [12], FP32, 257 × 257 pixels		✓			2.8
NLP-based	Natural language question answering	DNN11	MobileBERT [48], FP32	✓	✓		✓	100.7
Voice-based	Speech recognition	DNN12	Conv-Actions-Frozen [49], FP32	✓	✓		✓	3.8

Table 9. Time cost of measurements per edge device.

	Kernels	Models	Applications
Measure time per device	23.1 days	4.7 days	1.5 days

devices. The predictors are trained using our kernel-level dataset and evaluated by our model-level dataset.

4.1 Design and Implementation

Our designed kernel-level energy prediction method is inspired by nn-meter [30] which proposed a kernel-based latency predictor for DNN models. However, nn-meter does not support energy prediction. We propose using a rationale akin to that of nn-meter for the design of our kernel-level energy predictor, especially given that kernels run sequentially on current edge devices. The key contributions of our proposed energy predictor include: (1) being the first energy predictor for modern edge devices, achieving an accuracy of 86.2% (making it the most accurate energy predictor for edge devices to date) for unseen DNNs (i.e., those with unfamiliar kernel configurations); and (2) being the first kernel-level energy predictor for DNN executions on modern edge devices. Notably, most existing research primarily uses FLOPs to estimate the energy consumption of DNN executions, resulting in generally low prediction accuracy for unseen DNN models. The core of our kernel-level energy prediction method is that we build and train a predictor for each type of kernel (e.g., conv+bn+relu) using the kernel-level energy dataset presented in Section 3. The total energy consumption of a DNN model is then predicted by summing the estimated energy consumption of all kernels within that DNN model. Our energy predictors are implemented using the random forests regression, a machine learning algorithm known for its robustness, handling of high dimensional spaces, and its capability to model complex non-linear relationships.

4.2 Performance Evaluation

Comparison baselines. We implement two baselines to compare the energy prediction accuracy: (1) FLOPs-based predictor: recent work has leveraged FLOPs to estimate the energy consumption of DNN inference [50]. We train FLOPs predictors using linear regression. Given the FLOPs of a DNN model, the predictor can estimate its inference energy consumption. (2) BIC-based predictor: to demonstrate the critical role that our fine-grained kernel-level dataset plays in accurately predicting energy consumption, we also train energy predictors using the power data sampled by the edge device's built-in current (BIC) sensor. To ensure a fair comparison (i.e, to confirm that any prediction errors in energy consumption are largely due to the inaccuracy of the built-in current sensor's power measurement), we take three steps: (1) using the ground-truth latency when calculating the energy consumption in the BIC training dataset, and (2) training the predictor with the same amount of data, covering the same number of kernels and identical configurations, and using the random forests regression.

Metrics. The prediction performance is evaluated through the root mean square error (RMSE), root mean square percentage error (RMSPE), ±10%, and ±15% accuracy. The latter two metrics represent the percentage of models whose predicted energy consumption lies within the specified error bounds relative to actual measured energy consumption. In this paper, ±15% accuracy is the default metric. Smaller RMSE/RMSPE and larger ±10%/±15% indicate better prediction performance.

Comparison results on unseen DNN models. We select AlexNets, GoogleNets, MobileNetv1s, MobileNetv2s, and ShuffleNetv2s for the comparison study. As the FLOPs-based predictor requires training with model-level data (i.e., the FLOPs of DNN models), we adopt a leave-one-out cross-validation approach. We set aside one model (e.g., 50 models of GoogleNets) as the test set, and use the remaining four models (e.g., 50 models each of AlexNets, MobileNetv1s, MobileNetv2s, and ShuffleNetv2s) as the training set to train the predictor. Our kernel-level predictor and the BIC-based predictor do not require model-level data for training.

The comparison results are depicted in Fig. 10. Our kernel-level energy predictor consistently outperforms the other two baselines, delivering the highest prediction accuracy. Those baselines fail to achieve comparable levels of prediction accuracy on unseen DNN models. Specifically, our predictor achieves an average prediction accuracy of 86.2%, significantly higher than FLOPs, 31.3%, and BIC, 12.7%. The poor prediction accuracy of BIC, particularly on mobile GPU, demonstrates the indispensability of a fine-grained power and energy dataset when training a reliable energy predictor for edge devices. The significant drop in prediction performance of BIC on the mobile GPU is due to the fact that DNNs typically achieve much shorter execution time on the GPU compared to the CPU. This shorter execution time on the GPU necessitates a higher power sampling rate. Moreover, the performance gap between our kernel-level predictor and the FLOPs-based predictor reflects the gain derived through considering the runtime optimization of edge devices, such as kernel fusion. Table 10 presents the prediction results evaluated across all nine DNN models in our model-level dataset. In addition, we calculate the kernel configuration overlaps between the training (kernel-level dataset) and the evaluation (model-level detaset) datasets. Results show that our energy predictors have only seen 1.1% (CPU) and 1.8% (GPU) of the configurations in the evaluation dataset, which further attests the effectiveness of our kernel-level energy predictors on unseen models.

Discussion. Our kernel-level energy predictor exhibits slightly lower prediction accuracy compared to the latency predictor developed in nn-meter [30]. This might primarily be due to the fact that (1) nn-meter manually sets CPU frequency of the measured device to a fixed value (2.42GHz) when profiling the latency for building the training dataset and evaluating the prediction accuracy. This creates a more controlled environment for latency measurement and prediction. However, to ensure practicality, our kernel-level energy predictor does not establish a fixed CPU frequency during energy measurement and prediction. This results in greater variability and potential uncertainty in the energy prediction, yet it more accurately reflects real-world usage scenarios where the CPU frequency is typically dynamic. (2) The scale of our energy training dataset is less extensive than that of the latency training dataset in nn-meter[7], as collecting fine-grained power data is significantly more time-consuming than profiling latency data, particularly on modern edge devices. Hence, we anticipate the community will collectively collaborate to further enhance the scale of our datasets.

5 Scoring System

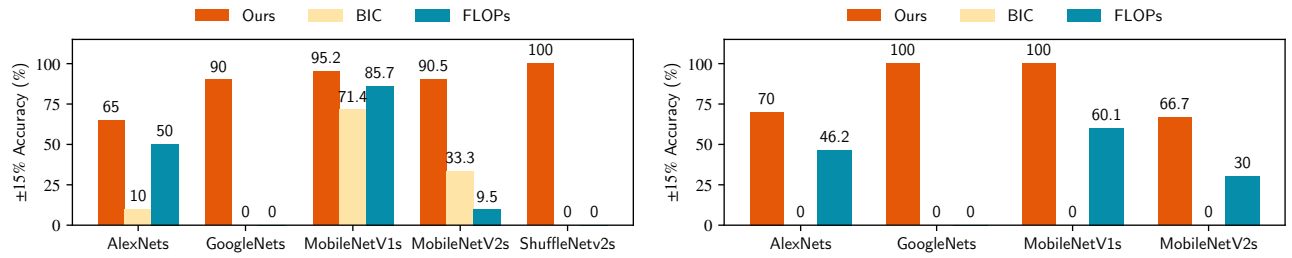
In this section, we introduce our method to tackle challenge **C3: understandability**. We develop a scoring system for diverse edge devices by leveraging our application-level dataset. To ensure that the energy efficiency assessment result is accessible to a broad audience, in particular, edge device end-users with non-technical backgrounds, we develop two scoring metrics, namely *power consumption score (PCS)* and *inference energy consumption score (IECS)*. These two scoring metrics help to distill the power and energy efficiency of a device in an intuitive and understandable way.

PCS. The PCS is designed to capture the aggregated power efficiency (PE) for running all six edge AI applications with 12 reference DNN models using CPU, GPU, and NNAPI delegates. It is calculated as $PCS = \frac{\sum_{i=1}^{n} PE_i}{n}$, where n is the total number of reference DNN models and $PE = (1 - \frac{APC}{TDP}) \times 100$. APC denotes the average power consumption for inferences. Thermal design power (TDP), measured in watts, represents the maximum power an edge device is designed to consume under normal operating conditions. The ratio $\frac{APC}{TDP}$ provides an indication of how efficiently a device is using its power budget, with a lower ratio indicating better PE.

IECS. The IECS is designed to assess edge device energy efficiency, and calculated as the sum of inference energy consumption (IEC) for all six edge AI applications under CPU, GPU, and NNAPI delegates. IEC is defined as the number of inferences per unit of energy, where it factors in the trade-off between PE and inference latency. An edge device with a higher IECS is considered more energy-efficient.

Results. Fig. 11 compares our proposed PCS with the AI inference score developed by AI Benchmark [42] across diverse edge devices. Note that the AI inference score does not take into account power and energy efficiency. The figure illustrates a tradeoff between AI performance, power consumption, and its selling price, where a larger ball in the figure represents a higher selling price for the device. An edge device that exhibits superior power efficiency (higher

[7]For example, 1032 (our energy dataset) vs. 15824 (the latency dataset in nn-meter) `conv+bn+relu` configurations.



(a) Mobile CPU (b) Mobile GPU

Figure 10. Comparison of energy prediction performance. Our predictors trained by the kernel-level dataset achieves the highest accuracy on unseen DNNs.

Table 10. Energy prediction results on mobile CPU and GPU.

Model variants	Mobile CPU				Mobile GPU			
	RMSE (mJ)	RMSPE (%)	±10% (Acc.)	±15% (Acc.)	RMSE (mJ)	RMSPE (%)	±10% (Acc.)	±15% (Acc.)
AlexNets	32.2	12.9	60.0%	65.0%	9.4	15.5	40.0%	70.0%
DenseNets	30.8	7.1	70.0%	100%	16.5	19.6	10.0%	35.0%
GoogleNets	25.1	11.9	20.0%	90.0%	3.8	5.5	95.0%	100%
MobileNetv1s	7.8	8.7	80.9%	95.2%	1.7	6.7	80.9%	100%
MobileNetv2s	7.8	8.3	76.2%	90.5%	3.1	11.5	47.6%	66.7%
ProxylessNASs	13.3	11.7	47.6%	71.4%	2.5	8.2	76.2%	95.2%
ResNet18s	44.6	6.1	95.2%	100%	30.5	13.1	38.1%	71.0%
ShuffleNetv2s	3.2	5.8	100%	100%	-	-	-	-
SqueezeNets	19.6	10.4	57.1%	90.5%	7.9	10.0	61.9%	85.7%

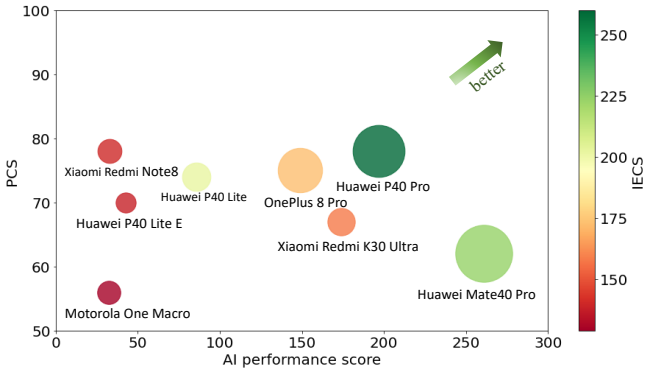


Figure 11. Comparison of the proposed PCS and AI inference score [42]. It presents a tradeoff between AI performance, power consumption, and the selling price. The larger the ball, the higher the selling price of the device.

PCS) and AI inference performance (higher AI performance score) is positioned towards the top right corner of the figure.

We find that scoring metrics significantly influence benchmarking results for edge devices. For instance, although the Huawei Mate40 Pro achieves the highest AI performance score, it holds the second worst PCS. Conversely, the Xiaomi Redmi Note8 attains the highest PCS while having the second lowest AI performance score. These observations highlight the need for the development of IECS that balances power efficiency with AI inference performance. In Fig. 11, the color of each ball indicates the IECS of each edge device. The Huawei P40 Pro presents the best equilibrium between AI performance and power efficiency, as indicated by its IECS and its position in the figure. We also provide the complete IECS results in Table 11.

6 Discussion

Limitations. Our current measurements and datasets are on modern smartphones equipped with mobile CPUs and GPUs. While they cover a broad spectrum of edge hardware, they might not be comprehensive. To further increase the heterogeneity, we plan to extend our energy datasets by including other modern edge devices, such as Jetson Nano, Coral TPU, and Raspberry Pi 4.

The proposed kernel-level energy predictor is built offline and will not be updated dynamically during DNN executions. Naturally, the prediction accuracy could be further improved by factoring in more environmental complexities, such as the available computing and memory resources on an edge device. We will leave this as an area for our future work.

Automated measurement. Table 9 illustrates that the majority of the time cost comes from energy profiling. Developing an automated measurement and profiling method can enhance the time efficiency for collecting a large-scale and more comprehensive dataset that includes a variety of edge devices and kernel configurations. The kernel-level energy predictor could also benefit, as prediction accuracy may

Table 11. IECS results based on application-level energy dataset

Device Model			OnePlus 8 Pro	Xiaomi Redmi Note8	Huawei Mate40 Pro	Huawei P40 Pro	Huawei P40 Lite	Huawei P40 Lite E	Xiaomi Redmi K30 Ultra	Motorola One Macro
SoC			Snapdragon 865	Snapdragon 665	Kirin 9000	Kirin 990 5G	Kirin 810	Kirin 710F	Dimensity 1000+	Helio P70
DNN1	CPU1	ECI	0.3209	0.4421	0.2421	0.1715	0.2649	0.5132	0.4640	0.4481
	CPU4	ECI	0.2858	0.4656	0.2439	0.1854	0.2790	0.5418	0.4735	0.4441
	NNAPI	ECI	0.3130	0.4844	0.2221	0.2030	0.2419	0.4148	0.4421	0.4237
DNN2	CPU1	ECI	0.2228	0.2592	0.1228	0.1069	0.1469	0.2452	0.2158	0.2270
	CPU4	ECI	0.2034	/	0.1186	0.1021	0.1497	0.2375	0.1880	0.2334
	NNAPI	ECI	0.2149	0.0265	0.1128	0.0933	0.1361	0.1623	0.3032	0.2443
DNN3	CPU1	ECI	1.3727	2.3948	1.2222	1.2684	1.3527	3.0454	1.6245	2.4968
	CPU4	ECI	1.3868	2.4183	1.1794	1.3390	1.3869	3.1238	1.6299	2.4786
	NNAPI	ECI	1.3905	2.4250	1.1156	1.2899	1.3407	2.7439	1.7859	2.4988
DNN4	CPU1	ECI	0.6314	1.2265	0.5194	0.5117	0.6088	1.4706	0.6435	1.2757
	CPU4	ECI	0.6150	1.2402	0.5437	0.5424	0.5692	1.5131	0.7760	1.2519
	NNAPI	ECI	0.6120	1.2311	0.5082	0.5174	0.5942	1.3755	0.7759	1.3334
DNN5	CPU1	ECI	0.1792	0.2009	0.1055	0.1103	0.1301	0.2058	0.2261	0.2123
	CPU4	ECI	0.1778	0.2041	0.1249	0.1154	0.1419	0.1875	0.2252	0.2144
	GPU	ECI	0.0979	0.1432	0.1118	0.0964	0.0954	0.1714	0.1615	0.1504
	NNAPI	ECI	0.1238	0.2295	0.2350	0.3672	0.3287	0.1916	0.4128	0.2351
DNN6	CPU1	ECI	0.1923	0.1678	0.1015	0.0797	0.1079	0.1675	0.1928	0.1657
	CPU4	ECI	0.1870	0.1619	0.1191	0.0806	0.1061	0.1403	0.1962	0.1457
	NNAPI	ECI	0.9329	0.2756	1.3280	1.6405	1.3641	0.1267	0.0819	2.9408
DNN7	CPU1	ECI	0.3309	0.2139	0.1475	0.1261	0.1449	0.2247	0.2803	0.2302
	CPU4	ECI	0.1854	0.2348	0.1448	0.1228	0.1414	0.2429	0.2572	0.2348
	GPU	ECI	0.0803	0.1326	0.1461	0.0816	0.0943	0.1536	0.1586	0.1650
	NNAPI	ECI	0.1277	0.2623	0.2247	0.1652	0.1758	0.2246	0.1631	0.2644
DNN8	CPU1	ECI	0.1633	0.1635	0.0956	0.0814	0.0887	0.1248	0.1796	0.1943
	CPU4	ECI	0.1555	0.1811	0.0884	0.0764	0.0887	0.1250	0.1743	0.1614
	NNAPI	ECI	0.0711	0.0678	0.0457	0.0466	0.0772	0.0996	0.0758	0.0789
DNN9	CPU1	ECI	1.6846	3.0001	1.4124	1.5211	1.5064	3.9297	2.0102	3.1099
	GPU	ECI	0.1369	0.5716	0.9453	0.1502	0.3426	0.4584	0.2993	0.4053
DNN10	CPU4	ECI	0.3411	0.3481	0.3943	0.3819	0.3451	0.7609	0.6042	0.7436
DNN11	CPU1	ECI	1.5330	3.1282	1.9101	1.8226	1.8624	3.7850	2.2376	3.2543
	CPU4	ECI	2.1198	2.8539	1.8117	1.8889	0.3534	3.9754	2.0826	2.9359
	NNAPI	ECI	4.5117	/	2.0306	0.4334	0.4605	2.0196	2.4897	7.0580
DNN12	CPU1	ECI	0.3367	0.3038	0.1094	0.1681	0.1209	0.3491	0.6353	0.4105
	CPU4	ECI	0.3029	0.2610	0.2489	0.1530	0.1220	0.3222	0.4117	0.3606
	NNAPI	ECI	0.3241	0.2798	0.2264	0.1695	0.0861	0.2988	0.3785	0.3535
IECS			173	140	224	260	202	138	158	129

* ECI (Joule): energy consumption per inference denotes the amount of energy required to execute a single inference on a mobile AI device; IECS: inference energy consumption score.

Ranking: 1 2 3 4 5 6 7 8

improve with more training data. Furthermore, automated profiling could help minimize human influence, leading to more accurate measurements.

Energy prediction for concurrent executions. Our energy predictor is premised on the fact that kernels currently run sequentially on edge devices. In the future, DNN inference may run concurrently on multi-core chipsets. Kernels processed in parallel might consume less energy than when processed sequentially, but more than individual kernels. The energy prediction performance for concurrent execution might be lower than for sequential execution, as concurrent operations introduce greater uncertainties in energy consumption. This aspect requires further experimentation.

7 Related Work

Energy measurement for edge devices. A number of research works have proposed different methodologies and developed frameworks for measuring the energy consumption in mobile and edge devices. The Green Miner proposed

in [39] can physically measure the energy consumption of mobile devices such as Android phones and automate the testing of applications. The GfxDoctor developed in [51] can systematically diagnose energy inefficiencies in app graphics at the app source-code level. However, none of these works have studied fine-grained energy measurement of DNNs on modern edge devices.

Edge AI benchmark. A few recent studies developed mobile AI benchmarks that measure the performance of on-device training and inference. For example, AI Benchmark [42, 52] is arguably the first benchmark suite for mobile devices, which primarily focuses on Android smartphones and measures only the latency. MLPerf Mobile [53, 54] presents the first industry-standard open-source benchmark for performance and accuracy evaluation of mobile AI devices. Additionally, AIoTBench [55] comprises a wider range of model architectures and AI frameworks, with a focus on assessing the inference capabilities of mobile and embedded devices. However, none of these edge AI benchmarks focused on energy efficiency of on-device learning and extensive energy dataset creation for modern edge devices.

8 Conclusion

We conduct energy consumption measurement studies for on-device deep learning. We have created extensive energy datasets at the kernel-level, model-level, and application-level to facilitate research aimed at improving the energy efficiency of deep learning on diverse edge devices. Building upon our energy datasets, we have developed kernel-level predictors that can accurately estimate the energy consumption of DNN executions on edge devices. Furthermore, we have implemented two scoring metrics to enhance the understandability of our energy measurement results. These contributions provide valuable resources and tools for advancing energy-efficient deep learning on edge devices.

References

[1] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[3] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016.

[4] Haoxiang Li, Zhe Lin, Xiaohui Shen, Jonathan Brandt, and Gang Hua. A convolutional neural network cascade for face detection. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5325–5334, 2015.

[5] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823, 2015.

[6] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[7] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7310–7311, 2017.

[8] Y. Wu, J. Lim, and M. Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848, 2015.

[9] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):295–307, 2015.

[10] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4681–4690, 2017.

[11] Radu Timofte, Shuhang Gu, Jiqing Wu, and Luc Van Gool. Ntire 2018 challenge on single image super-resolution: Methods and results. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 852–863, 2018.

[12] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proc. European Conference on Computer Vision (ECCV)*, pages 801–818, 2018.

[13] George Papandreou, Tyler Zhu, Liang-Chieh Chen, Spyros Gidaris, Jonathan Tompson, and Kevin Murphy. Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model. In *Proc. European Conference on Computer Vision (ECCV)*, pages 269–286, 2018.

[14] Francisco Javier Ordóñez and Daniel Roggen. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16(1):115, 2016.

[15] Iulian V Serban, Chinnadhurai Sankar, Mathieu Germain, Saizheng Zhang, Zhouhan Lin, Sandeep Subramanian, Taesup Kim, Michael Pieper, Sarath Chandar, Nan Rosemary Ke, et al. A deep reinforcement learning chatbot. *arXiv preprint arXiv:1709.02349*, 2017.

[16] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[17] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[18] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215*, 2014.

[19] Aliaksei Severyn and Alessandro Moschitti. Twitter sentiment analysis with deep convolutional neural networks. In *Proc. the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 959–962, 2015.

[20] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, 2013.

[21] Veton Kepuska and Gamal Bohouta. Next-generation of virtual personal assistants (microsoft cortana, apple siri, amazon alexa and google home). In *Proc. IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 99–103, 2018.

[22] Chung-Cheng Chiu, Tara N Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J Weiss, Kanishka Rao, Ekaterina Gonina, et al. State-of-the-art speech recognition with sequence-to-sequence models. In *Proc. 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4774–4778, 2018.

[23] Haomin Zhang, Ian McLoughlin, and Yan Song. Robust sound event recognition using convolutional neural networks. In *Proc. 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 559–563, 2015.

[24] Haoxin Wang, BaekGyu Kim, Jiang Xie, and Zhu Han. Leaf + aio: Edge-assisted energy-aware object detection for mobile augmented reality. *IEEE Transactions on Mobile Computing*, 22(10):5933–5948, 2023.

[25] Haoxin Wang and Jiang Xie. User preference based energy-aware mobile AR system with edge computing. In *Proc. IEEE Conference on Computer Communications (INFOCOM)*, pages 1379–1388, 2020.

[26] Users Reveal Top Frustrations That Lead to Bad Mobile App Reviews. https://finance.yahoo.com/news/apigee-survey-users-reveal-top-120200656. Accessed on March 2023.

[27] Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Haichen Shen, Meghan Cowan, Leyuan Wang, Yuwei Hu, Luis Ceze, et al. {TVM}: An automated {End-to-End} optimizing compiler for deep learning. In *Proc. 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 578–594, 2018.

[28] 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 *Proc. 12th USENIX symposium on operating systems design and implementation (OSDI)*, pages 265–283, 2016.

[29] Xiaotang Jiang, Huan Wang, Yiliu Chen, Ziqi Wu, Lichuan Wang, Bin Zou, Yafeng Yang, Zongyang Cui, Yu Cai, Tianhang Yu, et al. MNN: A universal and efficient inference engine. In *Proc. Machine Learning and Systems (MLSys)*, pages 1–13, 2020.

[30] Li Lyna Zhang, Shihao Han, Jianyu Wei, Ningxin Zheng, Ting Cao, Yuqing Yang, and Yunxin Liu. NN-meter: Towards accurate latency prediction of deep-learning model inference on diverse edge devices. In *Proc. the 19th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 81–93, 2021.

[31] Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct neural architecture search on target task and hardware. In *Proc. International Conference on Learning Representations (ICLR)*, 2019.

[32] Xiaoliang Dai, Peizhao Zhang, Bichen Wu, Hongxu Yin, Fei Sun, Yanghan Wang, Marat Dukhan, Yunqing Hu, Yiming Wu, Yangqing Jia, et al. Chamnet: Towards efficient network design through platform-aware model adaptation. In *Proc. the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11398–11407, 2019.

[33] Monsoon Power Monitor. https://www.msoon.com/specifications. Accessed on March 2023.

[34] Abhilash Jindal and Y Charlie Hu. Experience: developing a usable battery drain testing and diagnostic tool for the mobile industry. In *Proc. the 27th Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 804–815, 2021.

[35] Kittipat Apicharttrisorn, Xukan Ran, Jiasi Chen, Srikanth V Krishnamurthy, and Amit K Roy-Chowdhury. Frugal following: Power thrifty object detection and tracking for mobile augmented reality. In *Proc. the 17th Conference on Embedded Networked Sensor Systems (SenSys)*, pages 96–109, 2019.

[36] Xukan Ran, Haolianz Chen, Xiaodan Zhu, Zhenming Liu, and Jiasi Chen. Deepdecision: A mobile deep learning framework for edge video analytics. In *Proc. IEEE Conference on Computer Communications (INFOCOM)*, pages 1421–1429, 2018.

[37] Xiaomeng Chen, Abhilash Jindal, Ning Ding, Yu Charlie Hu, Maruti Gupta, and Rath Vannithamby. Smartphone background activities in the wild: Origin, energy drain, and optimization. In *Proc. the 21st Annual International Conference on Mobile Computing and Networking*, pages 40–52, 2015.

[38] Abhinav Pathak, Y Charlie Hu, and Ming Zhang. Where is the energy spent inside my app? fine grained energy accounting on smartphones with Eprof. In *Proc. the 7th ACM European Conference on Computer Systems (EuroSys)*, pages 29–42, 2012.

[39] Abram Hindle, Alex Wilson, Kent Rasmussen, E Jed Barlow, Joshua Charles Campbell, and Stephen Romansky. Greenminer: A hardware based mining software repositories software energy consumption framework. In *Proc. the 11th ACM Working Conference on Mining Software Repositories*, pages 12–21, 2014.

[40] Haoxin Wang, BaekGyu Kim, Jiang Xie, and Zhu Han. Energy drain of the object detection processing pipeline for mobile devices: Analysis and implications. *IEEE Transactions on Green Communications and Networking*, 5(1):41–60, 2021.

[41] Andrea McIntosh, Safwat Hassan, and Abram Hindle. What can android mobile app developers do about the energy consumption of machine learning? *Empirical Software Engineering*, 24:562–601, 2019.

[42] Andrey Ignatov, Radu Timofte, William Chou, Ke Wang, Max Wu, Tim Hartley, and Luc Van Gool. AI benchmark: Running deep neural networks on android smartphones. In *Proc. the European Conference on Computer Vision (ECCV) Workshops*, 2018.

[43] Mechanic Mobile Device DC Power Cable. https://www.amazon.com/Mechanic-Supply-Mobile-Repair-Control/dp/B089F1PM1F. Accessed on March 2023.

[44] TensorFlow Performance Measurement. https://www.tensorflow.org/lite/performance/measurement. Accessed on March 2023.

[45] Da Li, Xinbo Chen, Michela Becchi, and Ziliang Zong. Evaluating the energy efficiency of deep convolutional neural networks on cpus and gpus. In *2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom)*, pages 477–484, 2016.

[46] TensorFlow Lite Object Detection. https://www.tensorflow.org/lite/examples/object_detection/overview. Accessed on March 2023.

[47] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *Proc. European Conference on Computer Vision (ECCV) Workshops*, 2018.

[48] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[49] Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

[50] Radosvet Desislavov, Fernando Martínez-Plumed, and José Hernández-Orallo. Trends in AI inference energy consumption: Beyond the performance-vs-parameter laws of deep learning. *Sustainable Computing: Informatics and Systems*, 38:100857, 2023.

[51] Ning Ding and Y Charlie Hu. GfxDoctor: A holistic graphics energy profiler for mobile devices. In *Proc. the Twelfth European Conference on Computer Systems*, pages 359–373, 2017.

[52] Andrey Ignatov, Radu Timofte, Andrei Kulik, Seungsoo Yang, Ke Wang, Felix Baum, Max Wu, Lirong Xu, and Luc Van Gool. AI benchmark: All about deep learning on smartphones in 2019. In *Proc. 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 3617–3635, 2019.

[53] Vijay Janapa Reddi, Christine Cheng, David Kanter, Peter Mattson, Guenther Schmuelling, Carole-Jean Wu, Brian Anderson, Maximilien Breughe, Mark Charlebois, William Chou, et al. Mlperf inference benchmark. In *Proc. ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, pages 446–459, 2020.

[54] Vijay Janapa Reddi, David Kanter, Peter Mattson, Jared Duke, Thai Nguyen, Ramesh Chukka, Ken Shiring, Koan-Sin Tan, Mark Charlebois,

William Chou, et al. Mlperf mobile inference benchmark: An industry-standard open-source machine learning benchmark for on-device AI. In *Proc. Machine Learning and Systems (MLSys)*, volume 4, pages 352–369, 2022.

[55] Chunjie Luo, Xiwen He, Jianfeng Zhan, Lei Wang, Wanling Gao, and Jiahui Dai. Comparison and benchmarking of AI models and frameworks on mobile devices. *arXiv preprint arXiv:2005.05085*, 2020.