Energy modeling of inference workloads with AI accelerators at the Edge: A benchmarking study

Michalis Kasioulis, Moysis Symeonides, Giorgos Ioannou, George Pallis, Marios D. Dikaiakos
Department of Computer Science
University of Cyprus, Nicosia, Cyprus
Email: {mkasio01, msymeo03, ioannou.george, pallis, mdd}@ucy.ac.cy

***Abstract*—Analyzing and modeling the performance and energy consumption of hybrid Edge Computing systems with embedded devices and Artificial Intelligence (AI) accelerators is crucial, yet challenging due to the lack of systematic methods and tools for measuring and estimating energy consumption. We address this gap by introducing a systematic methodology and a toolset to benchmark AI accelerators and their host devices with inference workloads representing mock Convolutional Neural Network (CNN) models with varying input sizes, network sizes, layer types, and kernel sizes. The primary contributions of this work include the development of the benchmarking methodology, the creation and analysis of a comprehensive dataset comprising power benchmark results, and the development of a predictive model for estimating the energy consumption of ML workloads on the Coral TPU (Tensor Processing Unit) accelerator connected to the edge device. The dataset, generated from extensive testing on the deployed topology, is released and can be used for further studies that seek to enhance the energy efficiency and performance optimization for Edge Computing applications.**

***Index Terms*—power modeling, edge computing, AI**

I. Introduction

As the number of Internet of Things (IoT) devices increases [3], Edge Computing plays a key role for efficient data processing and management. The latter is underlined by the rapidly increasing demand for Edge Computing hardware, which accounts for more than 43% of the total Edge Computing market revenue in 2023 [5]. The increased hardware capabilities allow the deployment of AI applications, like image classification, at the edge. Towards improving inference tasks, AI accelerators have become widespread in edge deployments, enabling more advanced on-device AI processing and much faster inferences. However, the rapid adoption of such devices raises concerns about their overall energy consumption and highlights the necessity for energy-efficient solutions.

To improve AI workload efficiency and understand their energy demands, it is essential to develop new methods for accurately measuring and estimating the power consumption of AI-enabled edge devices under certain conditions, arising in the context of practical deployment of AI applications in edge environments [12]. For example, traditional methods [6] apply to computing devices for estimating the power consumption based on resource utilization metrics or performance counters but fall short in the case of AI accelerators, like Coral TPU or Intel Neural Compute Stick 2 (NCS2). These accelerators are only designed to perform inference tasks operating as black boxes, without exposing any relevant information in terms of resource utilization [9]. This requires alternative approaches for the devices' power estimation when performing inference.

A series of studies tried to evaluate and forecast the energy consumption and the performance of AI/ML models running on edge accelerators [2], [8], [9], highlighting the correlation between the model size and accelerator's performance metrics, such as latency, energy consumption, etc. These studies evaluate only the AI accelerators without taking into account their relationship with the rest of the edge device components and the effects these have on AI workload performance and power consumption. Specifically, several factors impact the utilization metrics of both the accelerator and the edge device, affecting the overall energy consumption of AI accelerator-enabled edge devices. These factors include the parameters of the model, input size, connection type (e.g., USB V2 or V3) and its configurations. The latent correlations between these parameters and the edge device utilization metrics as well as the energy consumption are yet to be fully explored.

This observation motivates this work, which aims to explore the following research questions: ***RQ1:*** *How do the different ML model parameters influence the power consumption of the overall system, and separately the edge device and the respective accelerator?*; ***RQ2:*** *How do other performance metrics like inference time are influenced by the different model's parameters?*; and ***RQ3:*** *Can we create AI models that estimate the power consumption and inference time while providing the feature importance of the selected CNN models' inference parameters?* Answering the above questions highlights the main contributions of our work, which are: (i) formulating the energy consumption of an accelerator-enabled edge device, including power modeling of both the edge device and AI accelerator; (ii) conducting systematic benchmarks following a device-agnostic methodology on a Raspberry Pi 4 with a Coral TPU accelerator and creating a comprehensive publicly available dataset[1] with utilization and performance metrics; (iii) performing exploratory analysis to reveal correlations between CNN models and power consumption and training estimation models for power and performance.

The rest of the paper is structured as follows: Sec. II presents the related work. Sec. III formulates the energy modeling on AI-enabled edge devices, and Sec. IV includes our experimental setup. Sec. V and VI present our benchmarking methods

[1] https://bit.ly/44LRV88

and exploratory analysis, respectively. Moreover, Sec. VII shows our prediction models, and VIII concludes the paper.

II. Related Work

There is a plethora of research papers that examine the energy consumption and latency implications of ML/AI workloads. For example, Trihinas et al. [12] investigate how AI workloads influence the energy consumption and carbon emission of edge devices giving guidance for future research and evaluation tools, but without the introduction of AI accelerators. Tu et al. [13] propose a novel measurement study for the energy characterization of a set of mobile phone devices with different hardware characteristics that can perform inference on the chip. In [11], the authors performed a set of benchmarks on GPU-equipped devices, for the evaluation of the TensorRT optimizations applied in terms of inference performance on a set of images. However, *these works leave unexplored the energy consumption and performance of AI accelerator-enabled edge devices, focusing solely on mobile phone chipsets, ML-inference on CPUs or powerful GPUs, or ML training tasks.*

Concentrating more on the reduction of the CNN models' computational complexity, Blott et al. [1] explore the effectiveness of various algorithmic optimization techniques, such as pruning and quantization across a variety of AI accelerators. Other systems try to predict the performance of CNN and Deep Learning (DL) models on heterogeneous GPU-enabled systems [2], [14]. For example, EDLAB, an end-to-end benchmark designed to assess the performance of edge AI/ML accelerators, proposed in [8]. Making use of a similar approach, the authors of [9] introduced PETET, a predictive modeling framework that employs ML techniques to forecast the performance and power consumption of TPU-based applications. In [10], the authors apply a systematic benchmark methodology to a set of edge devices with heterogeneous characteristics, to extract useful insights relating to the operational efficiency of those devices when performing inference on images with Neural Network (NN) models of different architectural characteristics. Although these works are more relevant to our research, *they ignore the power demands of the host device and do not examine the hidden dependencies between the edge device's resource utilization metrics, the connected AI accelerator, and the AI workload latency*.

III. Energy & Power Consumption Formulation

In this section, we formally define the energy and power consumption equations of an AI accelerator-enabled edge device. Firstly, the energy consumption, denoted as E and measured in Joules (J), represents the energy required for a computing system to complete a specified workload [7], [12]. The energy consumption is described by the following equation (1): $E = P \cdot t$, where P denotes the power consumption in Watts (W), which is the rate at which energy is used by the device, and t is the duration in seconds (s) required to perform the task. The power consumption (P) of a computing system is determined by both its idle and dynamic power profiles, namely $P = P_{dyn} + P_{idle}$ (2), where P_{idle} is the minimum level of power that the system needs to be functional, and P_{dyn} is the power demand of the system while it is performing its computational load. In the case of accelerator-enabled edge devices, the system includes two computing devices that collaborate with each other, namely, the edge node ($node$) and the accelerator (acc). Consequently, the overall power demand of an AI-enabled edge computing device can be defined by $P(system) = P(node) + P(acc)$, where $P(node)$ is the power consumption of the edge node and $P(acc)$ is the respective value for the device's AI accelerator. Considering the equation 2, the power demands of each device can be decomposed into the idle and the dynamic power consumption as defined by $P(node) = P_{dyn}(node)$ + $P_{idle}(node)$ (3), and $P(acc) = P_{dyn}(acc)$ + $P_{idle}(acc)$ (4) Specifically, $P_{dyn}(node)$ and $P_{idle}(node)$ present the dynamic and idle power consumption of the edge node, and $P_{dyn}(acc)$ and $P_{idle}(acc)$ the respective values for the accelerator.

The static idle power consumption can be easily defined, either by the manufacturer of the device or by simply measuring the power consumption of the respective device when it does not perform any computation. Next, we formulate the dynamic power modeling as $P_{dyn}(x) = P_{util}(U(x))$ (5), which is the power consumption function that calculates the power based on the utilization level of the component x, where the utilization level $U(x)$ of component x is defined as the fraction of the Used to Available Total Resources and provided by $U(x) = \frac{Total\ Resources\ Used}{Total\ Available\ Resources} * 100$ (6). In order to find the parameters of the $P_{dyn}(node)$ function for an edge device, we can perform repeatable benchmarks on the components, like the CPU, with different levels of utilization while capturing the overall power consumption of the $node$. Having both power consumption for different levels of utilization and the $P_{idle}(node)$, we can fit these measurements in regression models. For example, CPU utilization follows a linear trend with power consumption in Raspberry PI 4 as we demonstrate in Section V-B.

Unfortunately, we can not follow the same approach for an AI accelerator. Specifically, the current state-of-the-art accelerators do not allow access to their utilization metrics. Thus, we cannot use Eq. 5 to identify the relationship between the accelerator's utilization and its power consumption. Knowing that the deployed workload on a system influences its utilization metrics [9], we can infer the level of the accelerator's utilization based on the type of the deployed AI/ML model and its parameterization. So, the $P(acc)$ can be described as $P(acc) = P_{dyn}(f(workload)) + P_{idle}(acc)$, where $P_{idle}(acc)$ is constant, depending on the initial configuration of the accelerator (e.g., USB connection type), and $f(workload)$, which is a black-box function that determines the effects of different AI/ML parameterization on the accelerator's utilization, and, consequently, power consumption.

IV. Experimental Setup

In this section, we describe our experimental setup that covers a variety of Edge deployment scenarios. For example, let us consider a municipality deploying an urban traffic management

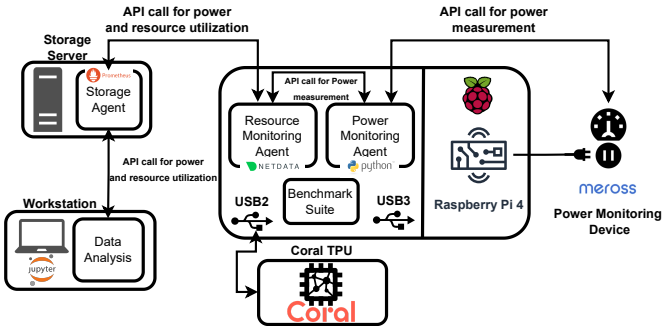

Fig. 1. Experimental Setup Overview.

system for video analysis to identify accidents. Instead of sending the stream to the Cloud, which is costly and slow, battery-powered, solar-rechargeable edge devices are placed around the city for on-site object detection and classification. In this scenario, forecasting the energy consumption and the AI performance is crucial for the selection of the respective AI model because (i) the devices solely rely on their renewable resources and batteries; and (ii) the administrators need real-time responses. Thus, a variety of CNN model architectures should be evaluated, since they provide better results than other object classification and detection techniques.

Having as a driver the previous scenario, we use a Raspberry Pi 4 Model B as the edge device under test due to its widespread usage and its ability to run complex calculations. Connected to it, via USB is the Google Coral TPU accelerator, selected for its efficiency in running TensorFlow Lite models[2]. Coral TPU can be operational on both USB V2 & V3 and also provides two different execution runtime modes, namely maximum (MAX) and standard (STD). These separate versions determine TPU's operating frequency, with MAX mode running at a frequency of up to 500 MHz, and STD mode (default) running at a reduced frequency of up to 250 MHz.

The edge node's power consumption is monitored via a Meross smart plug[3], which allows real-time power usage metrics collection via its API (Fig. 1). For resource usage monitoring, we deploy a Netdata[4] agent on the device designed to collect real-time (per second) metrics and the data were stored to a remote Prometheus[5] time-series storage server.

V. Benchmark and Analysis Methodology

A. Identification of Idle Power

Firstly, we focus on the identification of the idle power consumption of the edge device $P_{idle}(node)$ and accelerator $P_{idle}(acc)$. Thus, we leave the system running in idle mode with various configurations for 10 minutes and we capture the power consumption. The configurations include (i) the edge device without the accelerator; (ii) the connection of the TPU on different USB ports (V2/V3); and (iii) the configuration of the TPU in different modes (STD/MAX). The edge device's idle power consumption was found to be 3.1W (both mean and median values). Figure 2 depicts the box plots of power consumption for the edge device when the TPU is connected.

[2] https://www.tensorflow.org/lite [3] https://meross.com/en-gc/smart-plug/
[4] https://www.netdata.cloud [5] https://prometheus.io

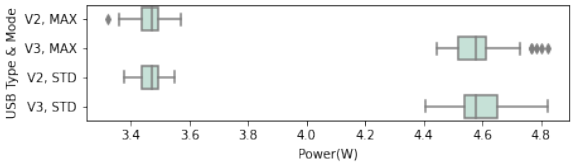

Fig. 2. Idle Power Consumption of the system with TPU connected.

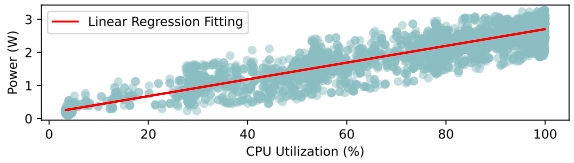

Fig. 3. CPU utilization VS power consumption, with fitting line.

Interestingly, we observe that the TPU mode seems not to influence the power consumption distribution in both connection types USB V2 and V3 and that is because frequency scaling is performed only when the TPU performs a task and not in idle mode. On the contrary, the USB type contributes to the idle power consumption of the accelerator-enabled edge device. So, USB V2 has 3.46W as the mean power consumption and 3.7W as the median, and USB V3 has 4.58W as both the median and mean. Since the mean and median differences are negligible, we use the mean idle values for the rest of our analysis.

B. Edge Device Dynamic Power Consumption

Next, we focus on finding the dynamic power consumption of the edge device. Previous studies [7] analyzing the power consumption of edge devices and more specifically the Raspberry Pi 4 device indicated that the main contributor to the overall power consumption is the CPU which is in line with the associated studies performed on high-end servers [6]. For that reason, we solely focus on CPU utilization and its contribution to the power consumption of the edge device.

While the power estimation tools, can be used to estimate the power consumption of hardware components and computing nodes [6], they are dependent on the underlying CPU architecture to work and cannot be applied to ARM-based devices. To tackle this issue, our methodology creates a new power consumption device model by following an iterative approach and stressing the device's CPU employing 1, 2, and 4 cores respectively when executing the stress command [6].The benchmark results (Fig. 3) indicate a linear relationship between the CPU utilization rate and the CPU's power calculated after subtracting the system's idle from the overall power consumption. The linear relationship is represented in Eq 7 after applying a regression analysis to the collected results.

$$P_{dyn}(rpi4) = 0.025 * U_{CPU}(rpi4) + 0.17 \text{ W} \quad (7)$$

C. Parameterization & Compilation of CNN models

At this step, we generate CNN models with different characteristics and architectural variations. Specifically, we change four main parameters:

Input size: This mostly influences the amount of computation required for the first convolutional layer, so we need to examine how this affects the overall power consumption and

[6] https://linux.die.net/man/1/stress

inference time. For this feature, we adjust the input size of the models across different resolutions of grayscale images ranging from low resolution 144p (256×144) to 4K (3840x2160) and filled with random values in the range [0,1]. In this round of experiments, the network size was formed by 5 convolutional layers with each layer composed of 10 filters and the kernel size remained fixed at 3.

Kernel size: This defines the types of patterns the network can effectively learn. Smaller kernels may focus on finer details, while larger kernels capture more global features. This characteristic intuitively influences the computational complexity, thus we create models with the kernel size being between 1 and 20. For this round of experiments, the input size remained fixed at 1280x720 and the network size was the same as in the case of variable input size.

Number of filters: The distribution of filters in the network plays an important role in its ability to recognize patterns on inputs and its accuracy during object detection and image classification. However, larger in size models require more energy due to the higher computational complexity. To capture this behavior, we created models with an increasing number of filters per layer and the number of layers, to cover a wide spectrum of CNN formats. If otherwise stated, the input size used for the trials was 96x96 and the kernel size was set at 1.

Layer type: We used 3 different types of layers that can be found in real-world CNN models, namely: (i) Convolutional layers; (ii) Convolutional layers followed by activation functions; and (iii) Depthwise Separable layers. Below we provide a short description for each layer type we used in this study.

- **Convolutional** layers are the foundation of CNN models, responsible for feature extraction from input data. Studying these blocks helps to establish a baseline for understanding how basic convolution operations affect the power needs and performance of AI accelerators.
- **Convolutional followed by activation functions**, help to assess the additional computational and power overhead introduced by applying non-linear transformations after convolution operations. In our benchmarks, we make use of a Gated Linear Unit (GLU) activation function.
- **Depthwise Separable** represents a more advanced and efficient convolution operation that decomposes a standard convolution into a depthwise convolution followed by a pointwise convolution. This reduces the computational complexity and parameter count, potentially offering significant energy savings on hardware accelerators.

For model creation, we made use of the publicly available coral-benchmark library[7] by making the necessary adjustments to the underlying code, since the benchmark was tailored to measure only the inference time between the CNN model's layers. To optimize the models' deployment, and make them compatible with TPU, we apply quantization to the models, reducing their precision to 8-bit integers. The next step is the model compilation, where we use the Edge TPU Compiler[8], which compiles a TensorFlow Lite model (.tflite file) into a format that is compatible with the TPU.

D. Extracted Datasets, Analysis & Model Training

The next stage involves the inference procedure of the already created models with 1000 random inputs based on their input size while monitoring the resource utilization and total power consumption of the deployment. We repeat this procedure for four different setups regarding the type of USB port that was used for connecting the TPU accelerator to the edge device (USB V2/V3) and the mode that the TPU was operating (STD/MAX). This allows us to explore the dynamics that USB V3 provides when it comes to the bitrate of data and at what cost, in terms of energy consumed and also the effectiveness of running the TPU on maximum mode using a higher clock speed for the cost of some additional power.

Then, we retrieve the results from the storage server based on the recorded timestamps and calculate the average resource utilization and power consumption from each run that are further used in our exploratory analysis and models' training. We create 2618 distinct points in the generated dataset consisting of 18 columns including inter alia the input features mentioned above, the average static power consumption of the system, CPU and TPU power consumptions, inference time per input, the joules consumed per input, etc.

Finally, we train multiple regression models, for power consumption and inference time as target metrics, using k-fold cross-validation and selecting the one with the least reported error using the Mean Absolute Error metric. Using PyCaret[9] ML library, we iteratively optimize AI/ML models by applying hyperparameter tuning and perform a post-training analysis on feature importance explaining the selected models' results.

VI. Exploratory Analysis Results

A. Kernel & Input Size

In Fig. 4, we deployed 3 different metrics (total average static power, inference time, and joules per input) against the increasing input size in MBytes. We can observe that power is not significantly impacted by input size and this is most probably explained by the fact that input size defines the computation needed on the first convolutional layer and for large in size networks this could be negligible. However, the energy that is spent as the input size increases is strongly connected to the time needed to process a single input. Since the power remains stable and also recalling Eq. 1, we can intuitively understand that the energy needed is defined by the linear relationship that characterizes the inference time and input size. After breaking down the overall power consumption into CPU and TPU power (Fig. 5) we can observe that the CPU power is constantly higher than that of TPU although both remain relatively stable meaning that the power needed for the CPU to handle pre-processing and communication is higher than the respective power needed by TPU to run inference for the selected CNN used in this round of benchmarks. Moreover, the power needed by the CPU is almost the same for the two

[7] https://github.com/fmfi-compbio/coral-benchmark

[8] https://coral.ai/docs/edgetpu/compiler

[9] https://pycaret.org

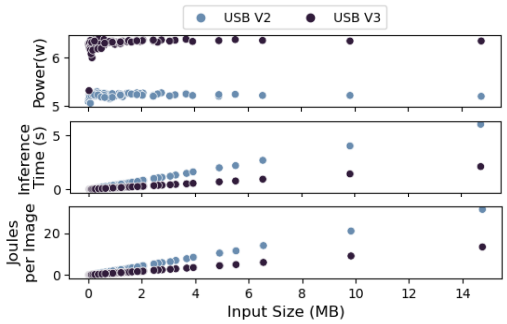

Fig. 4. Metrics vs Input Size.

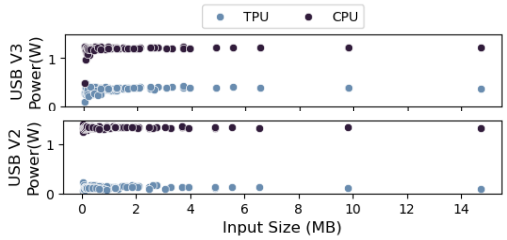

Fig. 5. CPU & TPU power vs Input Size for USB types.

USB types while in the case of USB V3, the TPU requires more power than when connected to the USB V2 port. That is due to the fact that USB V3 provides faster data transfer rates to the TPU in order to process the input faster and therefore reduces the inference time per input and energy needed as shown in Fig. 4. Therefore we conclude that USB V3 is more energy efficient as the network's input size increases.

The relevant results for the increasing kernel size are presented in Fig. 6 and 7. The same pattern as in the case of input size can be observed in this case, however with a smoother linear relationship between the kernel size and the energy consumption, which is strongly related to the inference time. After examining the relevant graphs for the two USB types, we observe that for both types, as the kernel size increases, the TPU power tends to increase while the CPU power tends to decrease at the same rate and at a faster rate for USB 3.0. A larger kernel size typically means more complex convolutions to cover a wider spectrum of the input image. This translates to more TPU overhead and also more CPU wait time until underlying convolutions are completed minimizing the CPU overhead. For USB V3, this transition is more sharp as its data rate seems to increase power consumption steeply.

B. Accelerator Connectivity

Next, we examine the effect of the USB connection port version on power consumption and inference duration. We already highlighted that the idle power consumption of USB V2 is less than V3 (Fig. 2), so here we solely focus on the power consumption during the inference period. Fig. 8 shows the distribution of the system's overall power consumption when performing inference over USB V2 and V3. The majority of points fall at 4.8 to 5.2W for USB V2 and for USB V3 power consumption is evenly distributed in the range of 5.2 to 6.2W. Interestingly, there is an increased number of points at 4.3W for USB V2, and, also, there is a significant difference in the lower and upper bounds of the two USB

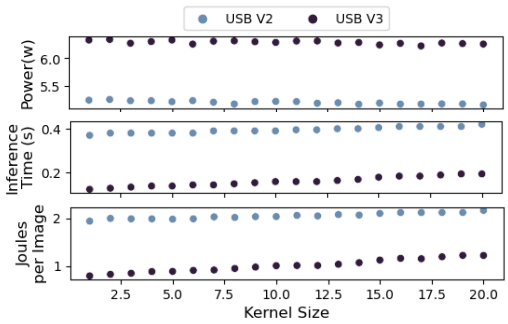

Fig. 6. Metrics vs Kernel Size.

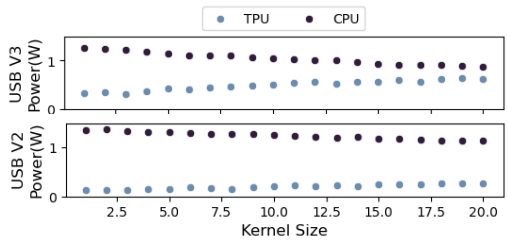

Fig. 7. CPU & TPU power vs Kernel Size for USB types.

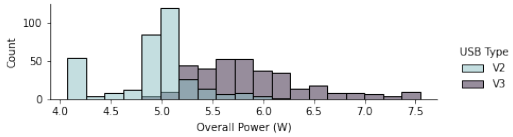

Fig. 8. Power Distribution Histograms for USB types.

Fig. 9. Mean Power Consumption for USB types.

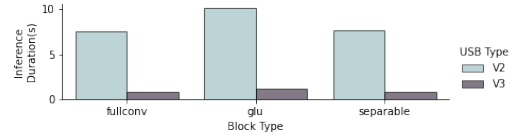

Fig. 10. Mean Inference Duration for USB types.

types. The latter indicates the difference on the data rates that characterize the two USB types, which directly affect the CPU and TPU utilization rates and the respective power consumption. The same observation was extracted for each block type of our experiments, with the mean average power consumption of different block types for both USB V2 and V3 maintaining consistent patterns (Fig. 9), without significant deviations meaning that the USB type is dominating the layers' block type in terms of the average power consumption.

Examining the average inference duration (Fig. 10), we observe significant differences between the two USB types with an increase of more than 5 seconds on average in all the cases. At the block type level, GLU (Gated Linear Unit) filters run over USB V2 show the most noticeable differences due to the computational overhead of the activation functions following each convolutional layer. The difference in inference time is understated in the case of USB V3 suggesting that for CNNs where activation functions are dominating among layers, USB V3 is suggested for reduced inference times.

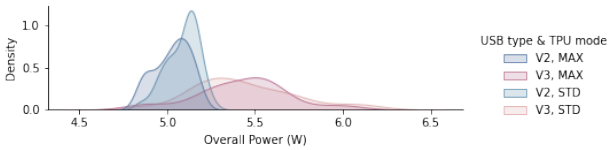

Fig. 11. Power consumption distribution of TPU modes & USB types.

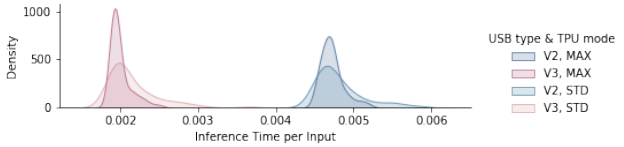

Fig. 12. Inference duration distribution of TPU modes & USB types.

C. TPU modes

To compare the two TPU modes, we made use of a small dataset part including smaller in-size CNNs since we noticed that the accelerator stops operating when its temperature increases after a specific point which is caused by larger networks with more activation functions and as noted by the manufacturer[10]. Fig. 11 and 12 present the density distributions of different configurations (USB connectivity & TPU mode) for the power consumption and inference time per input, respectively. Based on the results of the power consumption density plot, the distribution is characterized mainly by the USB type, and more specifically USB V3 shows a wider range of values with all the peak values falling within the range of 5-5.5W. This variability is expected given the TPU's capability to dynamically adjust the data rate when connected via USB V3. In the case of inference time per input, it is evident that USB V3 has superior performance with peaks at around 0.002s relative to 0.048s in the case of USB V2. Also, MAX TPU modes show narrower distributions providing more stable inference performance. In combination with USB V3, this results in optimal performance by fully utilizing the higher data rates that USB V3 offers.

D. CPU utilization & Power Consumption

We plotted the CPU utilization and power consumption of the system for different block types based on the total number of filters in the CNN in Fig. 13 and 14, respectively. The darkness of the data points represents the number of layers that form the neural network. According to the CPU plot, there is a clear trend of decreasing before flattening as the number of filters increases over USB V2. The trend in USB V3 is slightly different, where there is a slight increase after some point before flattening. This behavior indicates that as the number of filters increases, the CPU takes more time to transfer the data to the TPU for processing, resulting in lowering the CPU utilization caused by the fact that the TPU needs more time for the inference. For large CNN models after a specific point, the models cannot fit in TPU memory and therefore the data are partially transferred and processed so the CPU utilization remains stable after that point. In the case of USB V3, where the capacity of the link is larger, the CPU usage remains at a higher level as expected. Interestingly,

[10] https://coral.ai/static/files/Coral-USB-Accelerator-datasheet.pdf

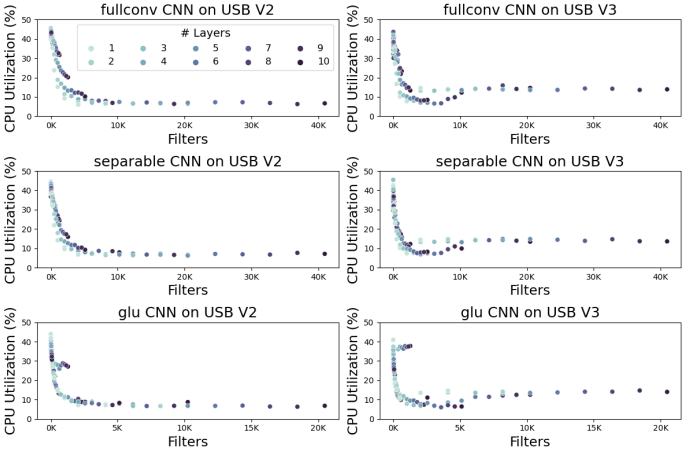

Fig. 13. CPU utilization VS total filters for different configurations.

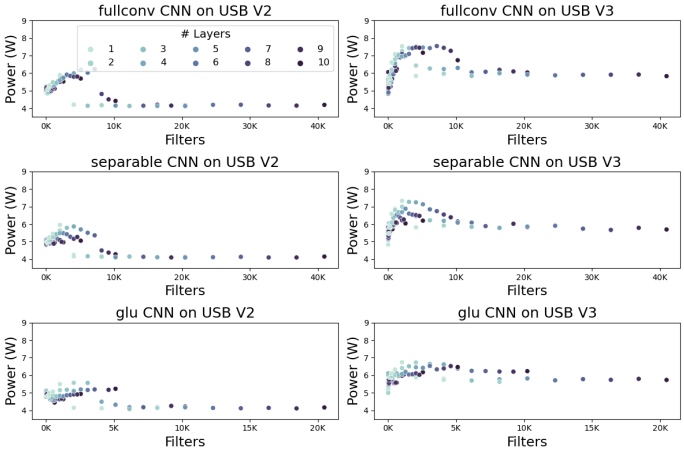

Fig. 14. Power consumption VS total filters for different configurations.

there are some data points (for less than 100K filters with large number of layers) where CPU utilization reaches lower levels. These cases highlight that these models fit in the TPU memory in contrast to the respective light blue points (same number of filters but fewer layers) that cannot fit in TPU memory due to a higher number of computational parameters in total. This is due to the complexity of convolutional filters in each layer which involve more computational parameters than in the case where the filters were distributed to a higher number of layers. This is due to the fact that a valid padding [4] is applied to the generated CNNs, meaning that the spatial dimensions of the feature maps are decreased with each layer moving forward.

The system's power consumption has the same trend in all cases, where there is an increase in the static power consumption as the CNN size increases up to a specific peak point before steadily lowering and remaining at the same levels. The increase in the power pulling of the overall system is caused by the TPU as the CNN size increases and as long as it fits in the TPU's main memory. After that point, it decreases until reaching the TPU's maximum capacity; from that point onward the static power consumption of the system remains relatively stable. Moreover, as the network size increases the TPU takes more time to process incoming data so CPU utilization remains idle for more time leading to a decrease in the overall system power. Finally, USB V3 trials remain at

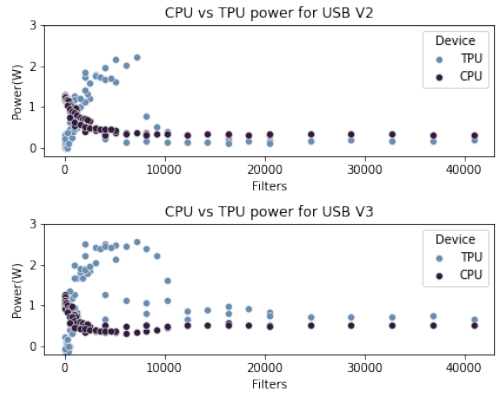

Fig. 15. Power consumption of CPU/TPU VS total filters for USB types.

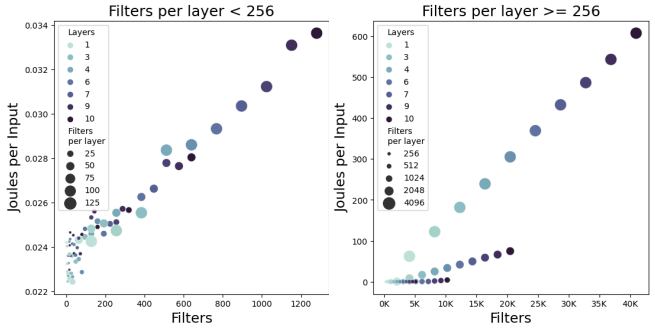

Fig. 16. Joules per input VS total filters.

higher levels of power than USB V2, which is expected due to the higher idle consumption that characterizes USB V3.

E. Power Consumption of CPU & TPU

By using the aforementioned CPU power model (Sec. V), we isolate the power that is allocated to the TPU accelerator, and we plotted the results for the respective power consumptions relevant to the number of filters in Fig. 15 for USB V2 and USB V3. In both cases, we observe that for small CNNs, the CPU power is higher than the TPU power, since the CPU needs more power to load the model and data to the TPU rather than the TPU itself to load the model and process the data. However, as the CNN size increases, the TPU suppresses the relevant CPU power but up to a specific point before starting to drop again and reaching the CPU level where it is maintained. The CPU power indicates dropped CPU usage as the CNN size increases as it takes more time for the TPU processing. After approximately 10k filters, the CPU needs to partially load the model to the accelerator for performing inference so both the CPU & TPU utilization remain stable.

F. Joules per Input

For the estimation of 'Joules per input' in each round of benchmarks, we keep track of the average static power of the system, as well as the inference time in seconds needed to complete the inference for each provided input. The energy consumed to process each input by the model under study is a function (Eq. 1) of the static power and time. The respective results in comparison with the total number of filters are presented in Fig. 16. The left plot depicts the small models (up to 1250 filters) and the right plot depicts the large models (up to 40960 filters). The size of the points shows the number of filters in each layer. As we can observe in the left plot, the relationship between the two metrics is linear for small CNN models. However, in the case of large models, the relationship is polynomial depending on the number of filters used in each layer when the number of layers is constant. When the number of layers is increasing and the number of filters in each layer is constant, the relationship is linear. Considering that the power needs remain stable for large models based on Fig. 14, we conclude that the inference time defines the relationship between energy and the CNN size.

Regression Model	MAE	MSE	RMSE	MAPE
Power Consumption Models				
Extra Trees	**0.089**	0.025	0.158	**0.016**
CatBoost	0.093	**0.024**	**0.153**	0.017
Random Forest	0.096	0.030	0.169	0.017
Light Gradient Boosting	0.104	0.030	0.17	0.018
Decision Tree	0.113	0.047	0.210	0.020
Inference Duration Models				
Extra Trees	**0.25**	**28.44**	**2.59**	**0.12**
CatBoost	0.35	32.94	2.92	2.15
Random Forest	0.57	36.79	4.23	0.31
Decision Tree	0.60	65.01	5.36	0.25
Light Gradient Boosting	1.32	52.48	6.13	11.92

TABLE I
COMPARISON OF REGRESSION MODELS

VII. Estimators & Feature Importance Analysis

A. Estimators Training & Evaluation

Next, we train multiple models for power consumption and inference duration estimation using the collected data and the provided CNN architecture characteristics. To optimize the models' accuracy, hyperparameter tuning was conducted (see Sec. V). Specifically, we evaluated 18 models with diverse hyperparameters for the generation of each model. Due to the limited paper space, Table I reports the error metrics of the best models sorted by the MAE for power consumption and inference duration estimation. In both cases, the best model is the Extra Trees model, with its MAE and MAPE (Mean Average Percentage Error) being 0.089 and 1.6% respectively for the power consumption estimator and 0.25, 12% for the inference duration estimator, followed by Catboost and Random Forest. That is because Extra Trees handles better categorical variables that exist in our generated dataset and complex data interactions by structuring feature relationships using underlying decision trees and fine-tuning those trees' depths. We use MAE to select the most accurate models because the 'inference time' output target has a decimal precision and might vary across a broad range and MAE helps to directly measure the average impact of prediction errors. In the case of the power consumption model, MAE is equally effective even if the range of values is smaller since it provides an absolute measure of the error in the same unit as the target variable, which in this case is measured in Watts. The rest of the list includes Decision Tree and Gradient Boosting models, in a different order between power and inference time models.

B. Feature Importance Analysis

Power consumption: The USB type dominates with a feature importance score of 0.51 due to the different operational

characteristics of the two USB types and more specifically the operational efficiency of USB V3 over V2, designed for higher transfer rates over the USB channel which has a significant impact on the overall system power consumption. The second most important feature is the number of filters per layer scoring 0.14. This is related to the computational intensity of convolutional layers, where a higher number of filters in each layer, increases the parallel processing within the CNNs and therefore increases the overall power consumption. The rest of the features in the plot are related the overall capacity of the network in terms of the total filters across it and the total number of layers with scores 0.12 and 0.07 respectively. *Inference time:* When it comes to the inference time per provided input, the most dominant feature is the number of filters per layer with a score of 0.37, implying that the number of computations within each layer is strongly related to the inference time. A higher number of filters per layer implies more computations per convolution operation meaning higher inference time. The influence of USB type with a 0.25 score in this case has a beneficial influence on the inference time per input, meaning that the usage of USB V3 over V2 translates to a lower processing time. The next important feature is the use of activation functions after each convolutional layer with GLU scoring 0.18, which incorporates additional computational steps and therefore more processing time is required relevant to the other two block types used in the benchmarking procedure. The total number of filters follows with 0.10 and defines the operational complexity of the network, meaning more filters across all layers also increases the inference time.

VIII. Conclusion

In this study, following a comprehensive device-agnostic benchmark methodology, we examined the power consumption and inference performance of AI accelerator-equipped edge devices, specifically the Google Coral TPU connected to a Raspberry Pi 4. By varying the CNN model's characteristics and TPU's configurations, we analyzed how these factors influence the edge system's power efficiency and computational speed. Specifically, by answering the *RQ1*, highlighted that (i) the Input Size & Kernel Size do not influence the device's overall power needs, but, in the case of Kernel Size, the power needs of TPU & CPU are changing (especially, when TPU is connected to USB V3) even if the overall power is stable (Sec. VI-A); (ii) USB type and TPU modes seem to influence both the overall power needs, as well as the separate TPU & CPU power demands (Sec. VI-B and VI-C); and (iii) interestingly, the power demand of each subsystem (TPU/CPU) follows a slightly different pattern based on USB type when we change the number and types of CNN filters (Sec. VI-E). In terms of *RQ2* we found that (i) the inference time increases linearly with Input and Kernel Size, and, similarly, influences the energy (Joules per input), with USB V2 having the worst results; (ii) USB type is the most crucial parameter in terms of performance (duration and power), while TPU MAX mode provides more stable inference latency; and (iii) CNN parameters, namely 'filters per layer' and 'number of layers', have a linear trend with energy demands for small networks, while for large networks correlation is again linear for the latter but exponential for the former (Sec. VI-F). Regarding *RQ3*, our best model (Extra Trees) exhibits the lowest error for both the power consumption and inference duration estimators. After our feature importance analysis, we conclude that the TPU's operating conditions, like the USB connection type, the CNN's number of layers, and filters per layer are important for the power consumption estimation in contrast to the input and kernel sizes that seem to influence less the system's energy dynamics. Finally, the datasets and the results generated from our effort is publicly available providing opportunities for future research, while our future directions involve the testing of real-world AI edge workloads, a tool for extracting the characteristics of CNNs and estimating their energy consumption, and the evaluation of our methods on various accelerators and edge devices.

Acknowledgement. This work is part of GreenAnalyzer that has indirectly received funding from the European Union's Horizon Europe research and innovation action programme, via the aerOS Open Call issued and executed under the aerOS project (Grant Agreement no. 101069732).

References

[1] Blott, M., Fraser, N.J., Gambardella, G., Halder, L., Kath, J., Neveu, Z., Umuroglu, Y., Vasilciuc, A., Leeser, M., Doyle, L.: Evaluation of optimized cnns on heterogeneous accelerators using a novel benchmarking approach. IEEE Transactions on Computers **70**(10), 1654–1669 (2021)

[2] Bouzidi, H., Ouarnoughi, H., Niar, S., Cadi, A.A.E.: Performance prediction for convolutional neural networks on edge gpus. In: Proceedings of the 18th ACM International Conference on Computing Frontiers. p. 54–62. CF '21, New York, NY, USA (2021)

[3] Buckland, E.: Edge computing market sizing forecast: Third release (Dec 2023), https://stlpartners.com/research/edge-computing-market-sizing-forecast-third-release/

[4] Goodfellow, I., et. al.: Deep learning. MIT press (2016)

[5] Grand View Research: Edge computing market size and trends (2024), https://www.grandviewresearch.com/industry-analysis/edge-computing-market

[6] Jay, M., Ostapenco, V., Lefèvre, L., Trystram, D., Orgerie, A.C., Fichel, B.: An experimental comparison of software-based power meters: focus on cpu and gpu. In: 2023 IEEE/ACM CCGrid. pp. 106–118 (2023)

[7] Kasioulis, M., Symeonides, M., Pallis, G., Dikaiakos, M.D.: Power estimation models for edge computing devices. In: European Conference on Parallel Processing Workshops. pp. 257–269. Springer (2023)

[8] Kong, H., Huai, S., Liu, D., Zhang, L., Chen, H., Zhu, S., Li, S., Liu, W., Rastogi, M., Subramaniam, R., Athreya, M., Lewis, M.A.: Edlab: A benchmark for edge deep learning accelerators. IEEE Design and Test **39**(3), 8–17 (2022)

[9] Ni, Y., Kim, Y., Rosing, T., Imani, M.: Online performance and power prediction for edge tpu via comprehensive characterization. In: 2022 Design, Automation and Test in Europe Conference and Exhibition (DATE). pp. 612–615 (2022)

[10] Saini, A., Shende, O.B., Pandit, M.K., Sen, R., Ananthanarayanan, G.: Bang for the buck: Evaluating the cost-effectiveness of heterogeneous edge platforms for neural network workloads. In: 2023 IEEE/ACM SEC. pp. 94–107 (2023)

[11] Shafi, O., Rai, C., Sen, R., Ananthanarayanan, G.: Demystifying tensorrt: Characterizing neural network inference engine on nvidia edge devices. In: 2021 IEEE IISWC. pp. 226–237 (2021)

[12] Trihinas, D., Thamsen, L., Beilharz, J., Symeonides, M.: Towards energy consumption and carbon footprint testing for ai-driven iot services. In: IC2E. pp. 29–35 (2022)

[13] Tu, X., Mallik, A., Chen, D., Han, K., Altintas, O., Wang, H., Xie, J.: Unveiling energy efficiency in deep learning: Measurement, prediction, and scoring across edge devices. In: IEEE/ACM SEC. pp. 80–93 (2023)

[14] Wang, C.C., Liao, Y.C., Kao, M.C., Liang, W.Y., Hung, S.H.: Perfnet: Platform-aware performance modeling for deep neural networks. In: Proceedings of the RACS. p. 90–95. RACS '20, ACM, New York, NY, USA (2020)