Online Performance and Power Prediction for Edge TPU via Comprehensive Characterization

Yang Ni[1], Yeseong Kim[3]⋆, Tajana Rosing[2] and Mohsen Imani[1]⋆
[1]University of California Irvine, [2]University of California San Diego, [3]Daegu Gyeongbuk Institute of Science and Technology
⋆Corresponding authors: yeseongkim@dgist.ac.kr; m.imani@uci.edu

***Abstract*—In this paper, we characterize and model the performance and power consumption of Edge TPU, which efficiently accelerates deep learning (DL) inference in a low-power environment. Systolic array, as a high throughput computation architecture, its usage in the edge excites our interest in its performance and power pattern. We perform an extensive study for various neural network settings and sizes using more than 10,000 DL models. Through comprehensive exploration, we profile which factors highly influence the inference time and power to run DL Models. We show our key remarks for the relation between the performance/power and DL model complexity to enable hardware-aware optimization and design decisions. For example, our measurement shows that energy/performance is not linearly-proportional to the number of MAC operations. In fact, as the computation and DL model size increase, the performance follows a stepped pattern. Hence, the accurate estimate should consider other features of DL models such as on-chip/off-chip memory usages. Based on the characterization, we propose a modeling framework, called `PETET`, which perform online predictions for the performance and power of Edge TPU. The proposed method automatically identifies the relationship of the performance, power, and memory usages to the DL model settings based on machine learning techniques.**

I. Introduction

Deep learning (DL) has recently brought great opportunities to various fields, such as the internet of things, speech recognition, computer vision, and natural language processing, etc. The edge computing environments, e.g., mobile and smart hubs, also deploy an increasing number of DL models for better user experience that further expand DL applications [1], [2]. The inference performance, i.e., the on-device response time of DL models, and power costs are keys to successful deployment. However, they are prone to have poor performance due to the limited resources of the edge environments where a powerful GPU is usually unavailable.

Earlier researchers have tried to address this issue by designing accelerators based on field-programmable gate array (FPGA) [3] [4] and custom application-specific integrated circuits (ASIC) [5], [6]. They are built to efficiently utilize the limited power and to provide real-time processing capability. To deploy those accelerators effectively in practice, we need accurate estimates of their performance and power. For example, we need to determine where to allocate DL tasks to meet the real-time, power resource, and accuracy requirements. Also, there are substantial combinations of DL hyperparameters during the DL model developments. It makes the optimization and design decisions difficult, e.g., whether low-power, resource-limited edge devices are capable of running models with sufficient performance.

A simple assumption used in the engineering field for the estimates is that the inference performance can be characterized by the arithmetic computation speed of the hardware, i.e., how many floating-point operations (FLOPs) or multiply-accumulate operations (MACs) can be processed in unit time. For example, [7] and [5]roughly characterizes the relative inference performance using the FLOP metric. However, these methods do not guarantee accurate estimates since there is no simple linear relationship between the computation costs and performance/power consumption. Recent work [8], [9] showed that multiple factors affect the performance of different DL models running on GPU, e.g., inter-component communication time and memory I/O time. We can anticipate that standardized accelerators such as Google Edge TPU and Intel Movidius would have similar performance/power characteristics related to the computation and memory usage. Increasing number of applications will rely on these hardware platforms in the future, and the characterization and modeling for accurate performance estimation of the DL models are still open research topics.

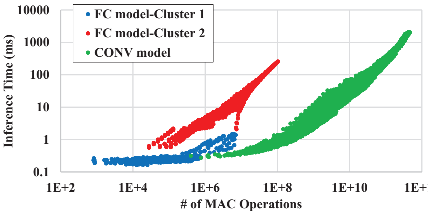


Fig. 1: Performance of FC networks and CONV models

In this paper, we focus on the characterization and modeling of the performance and power consumption for a commercial DL accelerator, Google Edge TPU. Edge TPU runs DL models created by TensorFlow (TF), which has the most users in the world [10]. We first show a detailed characterization of the performance and power for the Edge TPU device. Instead of using a few popular DL models, we perform extensive exploration of more than 10,000 DL models composed of either convolution layers or fully-connected layers to achieve a general and comprehensive characterization. We measure the inference time and power consumption of models under test. Our power/performance measurements present a non-linear relationship to computation cost. We present which factors have high impacts on performance and power, e.g., the number of MACs, network settings, and on-chip/off-chip memory usage, with several key remarks to guide analysis of the Edge TPU resource usages.

Based on the characterization results, we propose a modeling framework, called `PETET` (Performance/power Estimation Technique for Edge TPU), which predicts the execution time and power for different DL models from their model descriptions without performing the model compilation. The proposed framework automatically learns memory, performance, and power profiles based on machine learning (ML) to effectively utilize our extensive measurements used for the characterization. Our evaluation shows that `PETET` can accurately predict performance, power, and energy consumption online with errors of less than 10% for both fully-connected and convolutional networks. The proposed modeling framework also provides accurate estimates for state-of-the-art DL models, MobileNet [11], and VGG [12], only with 6.51% average error rate.

II. TPU Performance/Power Characterization

To perform a comprehensive characterization for the performance and power of Edge TPU, we first generated a large number of DL models using TensorFlow. The maximum and the minimum number of MAC operations are 4.7 $\times 10^{11}$ and 192, covering the range of MAC and parameter sizes used in most of the popular DL models. We measure the execution time to process one sample data along with the power consumption using MakerHawk TC66C USB multimeter. In total, we have 10,158 measurement results for different DL models, 7,560 and 2,598 for fully-connected (FC) and convolutional (CONV) neural networks, respectively. We then characterize the inference time and power consumption changes to discover critical factors affecting the resource usages.

A significant part of the inference process happens inside the matrix multiply unit of the Edge TPU. Therefore, it is reasonable for us to expect a close relationship between the number of MAC operations and inference time. However, as shown in Figure 1, we found that the number of MACs has a non-linear relationship with the inference

978-3-9819263-6-1/DATE22/©2022 EDAA

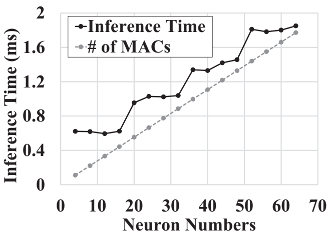


Fig. 2: MACs and performance of FC models with different neuron numbers

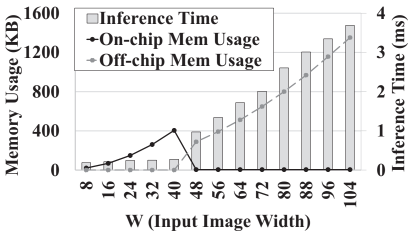


Fig. 3: Memory and performance of FC models with different input sizes

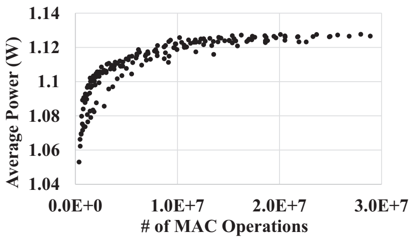


Fig. 4: Average power consumption for different FC models

performance. For example, we observe that the inference time of FC models forms at least two major clusters, one of which has a significantly smaller slope than the other one. For the CONV models, we find that a certain number of MACs may correspond to a large range of inference time. The results also show that different layer types lead to distinct patterns in the performance curve, presenting a huge difference in the inference time. For example, when both CONV and FC models have 1×10^8 MAC operations, CONV model inference is significantly slower than the FC model.

We thereby characterize models composed of FC and CONV layers separately. Next, we discuss our key remarks about how parameters besides the number of MAC operations, e.g., the model settings and the communication cost, influence the performance of the DNN models. We denote the number of layers by L, the number of neurons by N, the input size by S, and the filter depth of the convolution layers by F. Since we create models assuming they take three-channel square image data, the image width, W, determines the input size by $S = 3 \cdot W \cdot W$.

A. Characterization of Fully-Connected Network Models

Remark 1. Step-wise relationship between the performance and the number of MACs. The first finding to discuss is that the inference performance shows a non-linear, step-wise relationship with the model computation size, i.e., the number of MAC operations. For example, Figure 2 shows the inference time when changing the number of neurons (N) in each of FC model layers while using $L = 1$ and $W = 48$, so that the number of MACs increases linearly. The results clearly show that the inference time forms a series of steps. The inference time shows nearly no change within each step, whereas it undergoes a significant jump across two adjacent stages. Our reasoning behind this observation is that the number of MACs only represents the computation cost of the inference process; the communication cost also exists because of the slower link between the Edge TPU and the host machine. To better explain the non-linear behavior, we, in turn, analyze another possible factor, i.e., memory usage.

Remark 2. A significant impact of memory usage on the inference performance. Before discussing the influence of TPU memory usage, we describe the memory structure of the Edge TPU runtime environment. The Edge TPU uses two types of memory, on-chip memory and off-chip memory. The on-chip memory of the Edge TPU refers to a roughly 8 MB of SRAM located inside the Edge TPU. The off-chip memory refers to the host machine memory usage, mainly used when the on-chip memory is full. When the off-chip memory is used, the Edge TPU needs to retrieve that data from the host machine during inference. Thus, we infer that the utilization of the off-chip memory will have a significant impact on the performance.

Figure 3 shows how the performance changes over different input sizes, $S = 3 \cdot W \cdot W$ where $N = 20$ and $L = 3$. The results present a clear jump that separates the data points into two parts. In particular, for models with smaller sizes, i.e., W being less than 48, the inference time is relatively short and does not change significantly. In contrast, it increases for the larger input sizes. Most importantly, we observed that the performance change closely follows the off-chip memory usage. It implies that the off-chip memory usage has relatively high impacts on the performance. Figure 3 also shows that both on-chip and off-chip memory usage have a non-linear relationship with the input size. We observe various conditions besides the input size, in which the off-chip is prioritized over the on-chip memory usage. Here, we infer that the input size of 48 serves as a threshold for memory location selection.

To conclude the discussion, the memory usage has a clear relationship with the performance of FC models, especially when utilizing the off-chip memory. We thereby characterize the FC models as memory-bounded due to the communication between the Edge TPU and off-chip memory. The key challenge for accurate performance estimates is how to understand the memory allocation behavior of the closed-source TPU compiler, which translates the TFlite file and DL model settings. We will discuss our detailed estimation method in Section III.

Remark 3. Log-shaped relationship between power consumption and the number of MACs for FC networks. When we record the inference time, we also measure the power consumption of the inference process. We focus on the average power level due to the natural instability of instant power and the inevitable measurement error. Figure 4 shows the relationship between the average power level and the number of MACs for various FC models. In the experiment, we observed a *log*-shaped relationship, which shows that the power level increases gradually at the beginning and converges with the increase in the number of MACs. We reason that the utilization of the processing units saturates as the number of MACs continues to increase. Also, this figure indicates the possibility to model the power consumption of FC network inference.

B. Characterization of Convolutional Neural Network Models

Remark 4. Multiple parameters in model settings influence the performance. We observed that the CONV models utilize the off-chip memory less frequently. It is because the number of weights required is relatively smaller than the FC models. Thus, we need to understand the performance relationship with multiple factors besides the off-chip memory usage. We first examine how the inference time changes for different filter depths ranging from 32 to 320 when using $L = 5$ and $W = 48$, obtaining the results depicted in Figure 5. We observe that both on-chip memory usage and performance have a similar non-linear step-wise behavior. It implies that the on-chip memory usage significantly influences the inference time. We also changed the number of layers where each layer in the model is the same, using $F = 320$ and $W = 176$. In this case, the number of MAC operations changes linearly. Figure 6 summarizes the results for this experiment. It shows that the execution time linearly increases until the off-chip memory is activated, and once then, the slope is changed due to the extra communication overhead.

To conclude, the CONV models are neither fully memory-bounded nor compute-bounded; the performance is at least influenced by three different factors, i.e., the two types of memory usages and the number of MACs, in which no factor always dominates the others.

Remark 5. Non-linear relationship between the average power during inference and the number of MACs for CONV models. Figure 7 shows the power measurement results for three groups of data points, which represent DL models of different settings with the changes of the filter depth. We observe that the power consumption also converges as the number of MAC operation increases in a similar fashion to the FC model cases. However, there are variations among

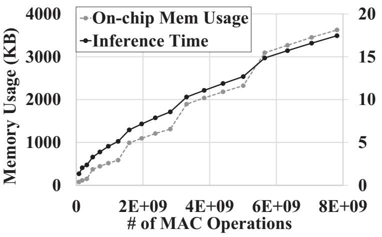


Fig. 5: Performance and on-chip memory usage with different filter depths

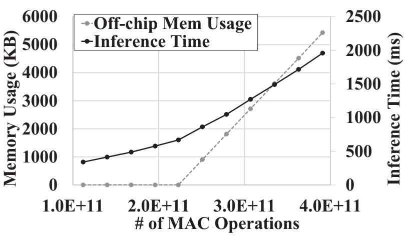


Fig. 6: Performance of CONV models with different input sizes

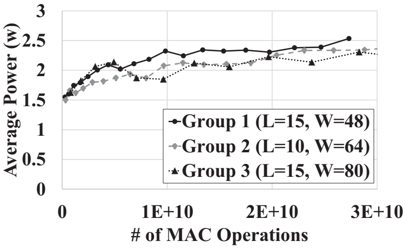


Fig. 7: Average power consumption for different CONV models

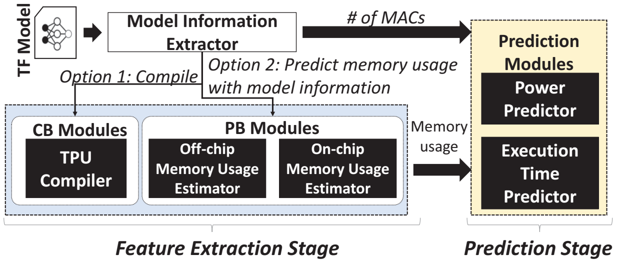


Fig. 8: Overview of PETET Modeling

the three groups due to the multiple underlying factors discussed in the performance analysis, which influence memory usage and performance characteristics.

III. Modeling Methodology

A. PETET Framework

Based on our characterization results, we build a modeling framework, called PETET, which predicts the performance and power of DL models on Edge TPU. Figure 8 shows the overview of the proposed PETET framework. We take into account both the memory usage and computational workloads (i.e., the number of MAC operations) to build the prediction model. For a given TF model, the PETET framework processes two stages: *feature extraction stage* and *prediction stage*.

During the feature extraction stage, the model information extractor module parses the TF model to obtain the DL model information such as the model input data size, the number of parameters (weight and bias values) and the number of MAC operations. This module can be implemented using profiler APIs of the TensorFlow library. Once parsing the model information, we need the on-chip/off-chip memory usage information to accurately predict performance and power. We obtain the memory usage information in either of two different ways, called *CB* (Compiler-Based) and *PB* (Prediction-Based) modes. In the CB mode, PETET directly runs the TPU compiler module and extracts the memory usage with the compiled TFlite model. However, it requires the complete compilation environment, which is undesired for many prediction practices, and consumes the extra compile time to get the predicted results. Thus, PETET provides another option, called the *PB* mode, which utilizes ML algorithms to estimate the memory usage using the extracted model information. After obtaining the number of MAC operations and the on-chip/off-chip memory usage, the prediction modules during the prediction stage compute the performance and power using them.

B. Prediction Model Learning

Memory Usage Estimator Modeling In the offline, PETET first learns the relationship between the on-chip/off-chip memory usage and the extracted model information, to build the learning models used in the *PB* modules. As discussed in Section II, the memory usage is highly related to the performance/power; there is no simple linear relationship to the memory usage. Thus, we should model the problem with multi-dimensional function to consider the multiple factors, e.g., the model input size and the number of parameters. We address this problem using ML algorithms to automatically learn the performance and power of DL models from our extensive measurements. In particular, we utilize the random forest (RF) algorithm [13], which learns patterns based on multiple decision trees with randomized sample/feature selections. Our key intuition behind this is that RF algorithm is capable of representing non-linear relationships between input features and output values while prioritizing key features that can be extracted from the TF model description. Besides, the RF algorithm utilizes the decision tree for its base estimator, which is suitable to automatically identify various thresholds and conditions for memory allocations between the on-chip and off-chip that the closed-source TPU compiler determines.

Let $\mathbf{v}$ be the input features obtained using the model information extractor module. We train two RF models for each of on-chip and off-chip memory, i.e., $mem_{offchip} = f_{offchip}(\mathbf{v})$ and $mem_{onchip} = f_{onchip}(\mathbf{v})$, where $mem_{offchip}$ and mem_{onchip} are the off-chip/on-chip memory usage in bytes and $f_{offchip}$ and f_{onchip} are the functions that the RF algorithm learns. As an output of the RF algorithm, we can estimate the impurity importance metric [13] for each feature. In our experiments, the RF algorithm selects the following features as its top-5 features: network type (FC or CONV), the number of layers, the input size, the number of MAC operations, and the number of parameters. It shows that the RF algorithm successfully selects the important features that highly affect the memory usage and performance/power discussed in Section II.

Performance/Power Predictor Modeling We then performed learning of the performance and power models for the prediction modules of PETET. As described in Section II, since both the memory usage and the amount of computational workloads are the two significant factors behind the performance and power, we formulate the problem as: $f(mem_{offchip}, mem_{onchip}, \#\_of\_MACs)$. We train the target function for each of the execution time and power level using the RF algorithm to learn the discussed non-linear relationships. Note that we use the measured memory usage as the feature values during the training, whereas during the prediction (i.e., inference), it can be replaced with the outputs of the memory usage estimator functions, $f_{offchip}(\mathbf{v})$ and $f_{onchip}(\mathbf{v})$.

IV. Modeling Technique Evaluation

A. Experimental Setup

We implemented the PETET framework using scikit-learn library for statistical analysis and ML. We utilize the same 10,158 measurements shown in Section II to learn the RF models of PETET. We randomly sample half of the total measurements for training the RF models, and we use the rest half to test the prediction quality. We learn 20 decision trees with a depth of 6 for each RF model by default. We report the mean absolute percentage error (MAPE) to evaluate the prediction quality of the proposed technique.

Overhead Table I shows the running time overhead of PETET prediction evaluated on Intel i5-5250U CPU. As discussed in Section III-A, PETET predicts in two different modes, called *CB* and *PB*. Since the CB mode compiles the TF model to obtain the memory usage

TABLE I: Running time overhead.

Compiler-Based (CB)		*Prediction-Based (PB)*
Largest (11M)	Smallest (193)	Average
74.6 sec	62.7 ms	12.7 ms

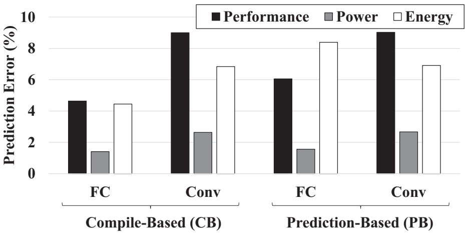


Fig. 9: Prediction error of PETET

information, the running time is dependent on the compile time. For example, for the DL model with the largest parameters, the compile takes 74.5 seconds, taking most of the overhead. In contrast, we observe that the PB mode is light-weight and there is no large difference depending on the model size in the running time.

B. Prediction Accuracy

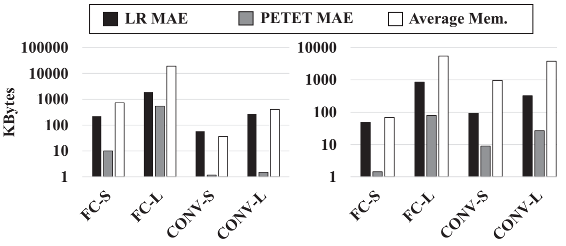
Figure 9 summarizes the prediction error of PETET for performance, power, and energy estimates. The results show that the proposed PETET framework accurately estimates the three metrics. For example, using the PB mode, it predicts the execution time, power, and energy with MAPEs of 9.02% (6.06%), 2.67% (1.56%), and 6.91% (8.38%), respectively, for different convolutional (fully-connected) networks. We also observed a negligible difference in the prediction quality (3.94% error) between the PB and CB modes.

To better understand how much accurately PETET estimates the memory usage in the PB mode using the RF algorithm, we compare the mean absolute error (MAE) of the off-chip/on-chip memory usage estimators with linear regression (LR)-based estimations. Here, we split the measurements into two groups depending on the execution times of the DL models for each of fully-connected and convolutional networks, creating four datasets, called FC-S (short), FC-L (long), CONV-S, and CONV-L. In general, the longer execution time represents the larger memory usage. Figure 10 shows the MAEs of the LR and PETET approaches along with the average memory usage for each dataset. It shows that the RF-based PETET significantly outperforms the LR-based approaches, meaning that the characterization of the non-linear relationship among parameters is essential in the memory usage prediction. For example, for the CONV-S case, the LR creates a high degree of errors, 57 KBytes, where the average memory usage of this dataset is 36.4 KBytes. The proposed technique predicts the memory usage only with 1.2 KBytes error.

To summarize, with the PB mode, PETET can accurately predict the performance and power of various DL models online, without actual compilation, unlike the CB mode. As discussed in Section IV-A, it is also light-weight, and thus, we expect that the PB mode would be an ideal choice in practice, e.g., estimating performance for hyperparameter searches.

C. Cross-Validation with State-of-the-Art Models

To understand how PETET performs the prediction for practical DL models, we also cross-validated the accuracy for the state-of-the-art (SoA) image recognition networks: MobileNet V1/V2 [11] and VGG16/VGG19 [12]. Since the SoA models support different input sizes and other parameters, e.g., α values for MobileNet, we tested various configurations for each model. Table II shows that PETET accurately predicts the performance and energy of the popular models. The average error for the energy estimates is 6.51% for all the models.



(a) Off-chip Memory
(b) On-chip Memory

Fig. 10: Summary of memory usage estimation errors

TABLE II: Estimation for SoA Models. Msmt. and Pred. stand for measurement and prediction, respectively.

Name	α	Input Size	Performance Msmt.	Performance Pred.	Energy Msmt.	Energy Pred.	MAPE
MobileNet V1	1	224	3.82ms	3.92ms	5.8mJ	5.32mJ	8.34%
	1	160	2.47ms	2.35ms	3.59mJ	3.47mJ	3.43%
	0.5	224	2.26ms	2.05ms	3.10mJ	2.97mJ	4.42%
	0.5	160	1.52ms	1.71ms	1.96mJ	2.22mJ	12.69%
MobileNet V2	1	224	4.17ms	4.68ms	5.94mJ	6.30mJ	5.98%
	1	160	2.82ms	3.24ms	3.86mJ	4.00mJ	5.03%
	0.5	224	3.79ms	4.29ms	5.00mJ	5.30mJ	7.54%
	0.5	160	2.48ms	2.82ms	3.15mJ	3.41mJ	8.15%
VGG16	N/A	224	34.39ms	34.75ms	61.32mJ	66.75mJ	8.85%
	N/A	192	29.28ms	25.12ms	52.10mJ	48.69mJ	6.55%
	N/A	128	23.00ms	21.86ms	34.26mJ	33.16mJ	3.22%
VGG19	N/A	224	34.39ms	34.94ms	63.26mJ	67.28mJ	6.34%
	N/A	192	29.3ms	30.33ms	48.87mJ	50.89mJ	4.12%
	N/A	128	22.97ms	26.57ms	34.23mJ	36.44mJ	6.44%
						Avg.	**6.51%**

V. Conclusion

In this paper, we characterize the performance and power of Google Edge TPU. Throughout extensive exploration, we show multiple findings for the cutting-edge accelerator, including that the number of MAC operations is not linearly proportional to performance/power, and the memory usage should be considered to obtain accurate estimates. Based on the analysis, we propose an ML-based modeling framework, that accurately estimates the power, performance, and energy with less than 10% errors.

Acknowledgments

This work was supported in part by National Science Foundation (NSF) #2127780, #2112167, #2052809, #2003279 and Semiconductor Research Corporation (SRC) Task #2988.001, SRC CRISP, and Department of the Navy, Office of Naval Research, grants #N00014-21-1-2225 and #N00014-22-1-2067, and a generous gift from Cisco. Yeseong Kim was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (Ministry of Science and ICT) (NRF-2018R1A5A1060031).

References

[1] Z. Zhou *et al.*, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738–1762, 2019.

[2] Z. Zou *et al.*, "Scalable edge-based hyperdimensional learning system with brain-like neural adaptation," in *SC*, 2021, pp. 1–15.

[3] Y. Ma *et al.*, "An automatic RTL compiler for high-throughput FPGA implementation of diverse deep convolutional neural networks," in *FPL*. IEEE, 2017, pp. 1–8.

[4] ——, "Optimizing the convolution operation to accelerate deep neural networks on FPGA," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 26, no. 7, pp. 1354–1367, 2018.

[5] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *ISCA*. ACM, 2017, pp. 1–12.

[6] M. Imani *et al.*, "Revisiting hyperdimensional learning for fpga and low-power architectures," in *HPCA*. IEEE, 2021, pp. 221–234.

[7] J. Turner *et al.*, "Characterising across-stack optimisations for deep convolutional neural networks," in *IISWC*. IEEE Computer Society, 2018, pp. 101–110.

[8] H. Qi *et al.*, "Paleo: A performance model for deep neural networks," in *ICLR*, 2017.

[9] E. Cai *et al.*, "Neuralpower: Predict and deploy energy-efficient convolutional neural networks," *arXiv preprint arXiv:1710.05420*, 2017.

[10] K. Dinghofer and F. Hartung, "Analysis of criteria for the selection of machine learning frameworks," in *ICNC*. IEEE, 2020, pp. 373–377.

[11] M. Sandler *et al.*, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *CVPR*, 2018, pp. 4510–4520.

[12] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[13] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.