QADAM: Quantization-Aware DNN Accelerator Modeling for Pareto-Optimality

Ahmet Inci[1], Siri Garudanagiri Virupaksha[1], Aman Jain[1], Venkata Vivek Thallam[1], Ruizhou Ding[1], Diana Marculescu[1,2]
Carnegie Mellon University[1], The University of Texas at Austin[2]
{ainci, sgarudan, amanj, vthallam, rding, dianam}@andrew.cmu.edu

***Abstract*—As the machine learning and systems communities strive to achieve higher energy-efficiency through custom deep neural network (DNN) accelerators, varied bit precision, and quantization levels, there is a need for a design space exploration framework that incorporates quantization-aware processing elements into the accelerator design space while having accurate and fast power, performance, and area models. In this work, we present *QADAM*, a highly parameterized quantization-aware power, performance, and area modeling framework for DNN accelerators. Our framework can facilitate the future research on design space exploration of DNN accelerators for various design choices such as bit precision, processing element type, scratchpad sizes of processing elements, global buffer size, device bandwidth, number of total processing elements in the the design, and DNN workloads. Our results show that different bit precisions and processing element types lead to significant differences in terms of performance per area and energy. Specifically, our proposed lightweight processing elements achieve on par accuracy results and up to $5.7\times$ more performance per area and energy improvement when compared to the INT16 based implementation.**

I. Introduction

Deep neural networks (DNNs) have achieved remarkable accomplishments across various applications ranging from image recognition [25], object detection [26], to natural language processing [5]. However, the increasing model size and computational cost of these models become a challenging task for on-device machine learning (ML) endeavours due to the stringent performance per area and energy constraints of the edge devices. To this end, while machine learning practitioners focus on model compression techniques [3], [6], [9], computer architects investigate hardware architectures to overcome the energy-efficiency problem and improve the overall system performance [12]–[16].

As computing community hits the limits on consistent performance scaling for traditional architectures, there has been a rising interest on enabling on-device machine learning through custom DNN accelerators. As we deeply care about performance per area and energy-efficiency from a hardware point of view, tailored DNN accelerators have shown significant improvements when compared to CPUs and GPUs [2], [7], [17], [19]. To better understand the trade-offs of various architectural design choices and DNN workloads, there is a need for a design space exploration framework that can rapidly iterate over various designs and generate power, performance, and area (PPA) results. To this end, in this work we present

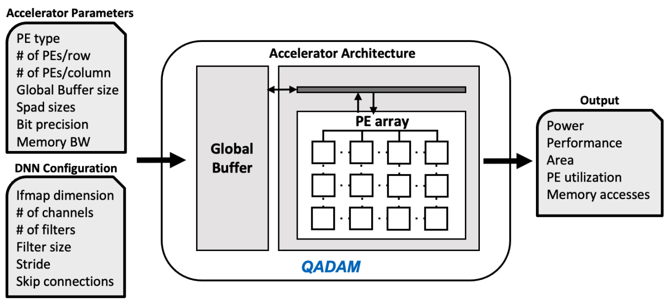


Fig. 1. Schematic depicting *QADAM* framework, with accelerator parameters and DNN configuration as inputs. The framework takes in accelerator parameters and layer-wise DNN configurations and generates power, performance, area results, and statistics on hardware utilization and memory accesses.

QADAM, a quantization-aware power, performance, and area modeling framework for DNN accelerators.

This work makes the following contributions:

- We present QADAM, a quantization-aware power, performance, and area modeling framework for DNN accelerators. Our framework can enable future research on design space exploration of DNN accelerators for various design choices such as bit precision, processing element types, scratchpad sizes of processing elements, global buffer size, device bandwidth, number of total processing elements in the design, and DNN workloads.
- Our framework provides power, performance, and area results not just for a single hardware design point but for a range of different hardware designs as opposed to prior art [1], [20]. Thus, it can be used to analyze trade-offs of various architectural design choices and DNN workloads at the same time to achieve Pareto-optimal design points in terms of accuracy and hardware-efficiency metrics such as performance per area and energy.

The rest of the paper is organized as follows. In Section II, we present a literature review on power and runtime models for CNNs and design space exploration frameworks for hardware accelerators. In Section III, we describe the architectural details of the *QADAM* framework and the details of our methodology for power, performance, and area modeling of DNN accelerators. In Section IV, we show experimental results demonstrating the efficiency of *QADAM*'s PPA models and the efficacy of lightweight processing elements to conventional

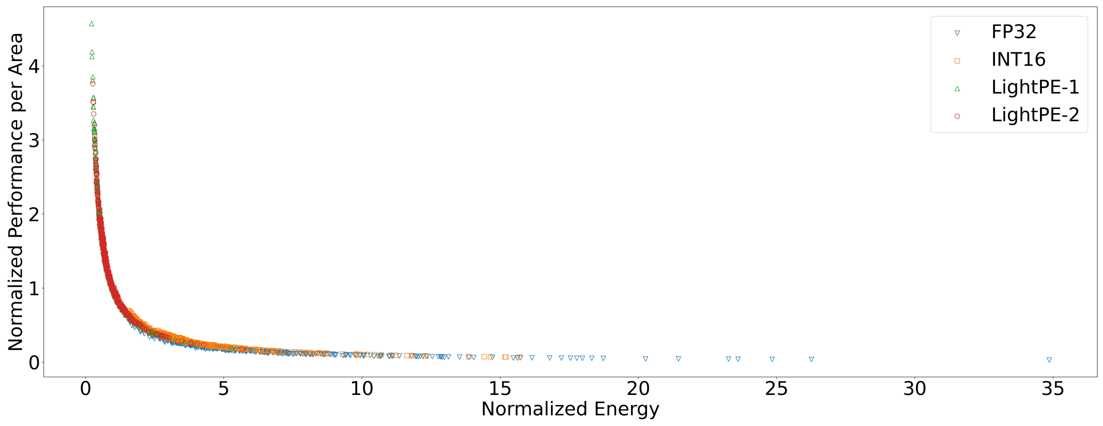


Fig. 2. Different PE types and bit precision lead to significant differences in performance per area and energy. Therefore, there is a need for a design space exploration framework that incorporates quantization-aware processing elements and rapidly iterate over various designs.

designs in terms of performance per area and energy through a suite of case studies. Finally, Section V concludes the paper by summarizing the results.

II. Related Work

Prior art has proposed runtime and energy models for DNN workloads [1], [20]. However, these models have been implemented specifically for GPU platforms and thus they create an important limitation for a design space exploration of hardware architectures and potentially hardware and machine learning model co-design opportunities [8], [27], [28]. On the other hand, systems community has proposed tools and simulation methodologies for accelerator design. For example, SCALE-Sim [21] is a cycle accurate, systolic-array based DNN accelerator simulator. Similarly, Aladdin [22] is a pre-RTL power and performance accelerator simulator. Although these tools help to perform preliminary analysis on the design space for accelerators in different aspects, they do not incorporate specialized quantization-aware processing elements and they do not generate RTL output of the chosen design based on the input hardware configuration which is an important impediment for enabling deployment of DNNs onto edge devices, as the actual deployment of the hardware design takes significant amount of engineering effort.

III. Methodology

In this section, we first explain the implementation details and architectural components of our *QADAM* framework, as depicted in Figure 1. Next, we detail the lightweight processing elements (LightPE) that we implemented in our framework to provide a specialized processing element (PE) type for quantized DNN models. Finally, we explain our power, performance, and area modeling and design space exploration methodology.

A. *QADAM Framework*

To enable comprehensive design space exploration for DNN accelerators for on-device machine learning, we implemented *QADAM*, a highly parameterized spatial-array based DNN accelerator framework in RTL. Our framework enables hardware designers and machine learning practitioners to rapidly iterate over various accelerator designs and DNN configurations and better understand trade-offs of different architectural components of the design for dizzying requirements of deploying machine learning models to edge devices. Moreover, hardware designers can also use the automatically generated RTL code to follow the design synthesis flow.

As depicted in Figure 1, *QADAM* framework is based on spatial-array based accelerators and utilizes row stationary dataflow which has been demonstrated to optimize the data movement in the storage hierarchy [2]. *QADAM* features a set of processing elements organized as a 2D array and a global buffer that stores input feature maps, filters, and activations. The number of PEs in each dimension can be tuned for different power, performance, and area requirements. In each PE, there are input feature map, filter, and partial sum scratchpads and a multiply-accumulate (MAC) unit which can be chosen between a conventional MAC unit and a shift-add unit based on the desired bit precision. Each of these architectural components can be tuned in a flexible and automated manner to perform a comprehensive design space exploration for on-device edge accelerators.

B. *Lightweight Processing Elements (LightPE)*

To enrich the design space of hardware accelerators and achieve a better Pareto-frontier in terms of performance per area and energy-efficiency perspectives, we include LightPE implementations in our framework. LightPEs utilize 8 bits for activations and 4 bits and 8 bits for weights for LightPE-1 and LightPE-2 designs, respectively. As 4 bit and 8 bit quantization techniques for on-device machine learning became prevalent in various computing platforms, we provide these specialized quantization-aware PE types in our *QADAM* framework to help hardware designers to enrich their design space and find better Pareto-frontiers.

Besides their low-precision benefits such as reducing the storage requirements, LightPEs also replace the multiplications with more energy and area-efficient one shift or a limited number of shifts and add operations [6]. Therefore, they also

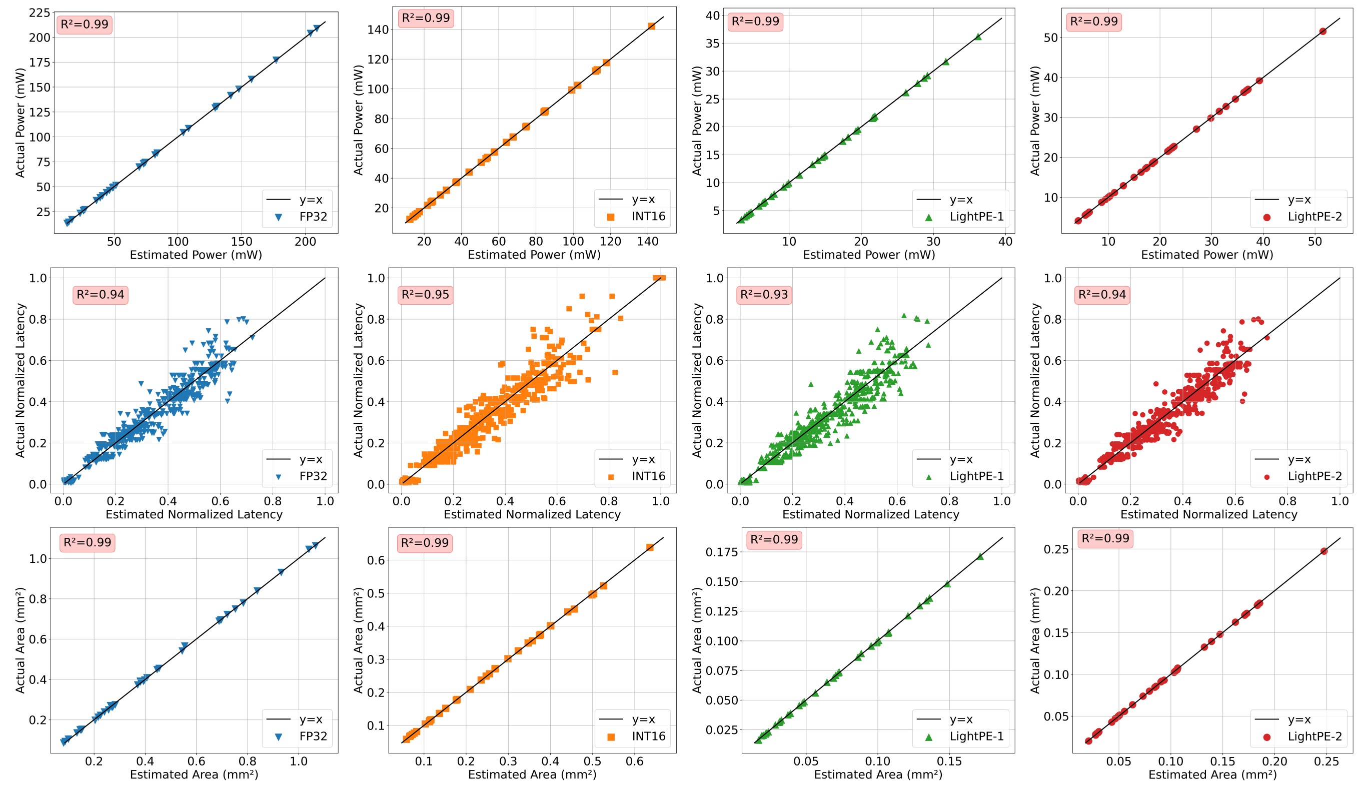


Fig. 3. Power (top chart), performance (middle chart), and area (bottom chart) estimation results for various processing element types such as FP32, INT16, LightPE-1, and LightPE-2. Each data point corresponds to a different hardware configuration that can be achieved by using the corresponding processing element type. As it can be seen, the proposed polynomial model agrees closely with the actual values extracted from the synthesis tools.

achieve significant power and area gains when compared to full-precision 32 bit floating point (FP32) and 16 bit integer (INT16) based designs with only slight accuracy degradation [6]. As a result, LightPEs provide an enriched design space for hardware designers and machine learning practitioners to analyze various trade-offs between accuracy and performance per area and energy. To this end, Figure 2 shows that different PE types and precision lead to significant differences in terms of performance per area and energy. These results also reinforce the need for a design space exploration framework that incorporates quantization-aware hardware.

C. Power, Performance, and Area Modeling

To build our quantization-aware power, performance, and area models, we use various hardware and DNN configurations. Specifically, to cover this comprehensive design space of hardware accelerators, we run experiments by varying global buffer size, number of PEs per row and column in the 2D PE array, bit precision, and PE type (FP32, INT16, LightPE-1, and LightPE-2). Within each PE, we also vary individual scratchpad sizes for input feature map, filter, and partial sum.

We use Synopsys Design Compiler and the open-source FreePDK45 which is a commonly used process design kit [24] to synthesize our designs to obtain power, area, and initial timing results. We use Synopsys VCS RTL simulator to perform functional verification and collect timing information for various DNN configurations such as VGG-16 [23], ResNet-20, ResNet-34, ResNet-50, and ResNet-56 [10] that are implemented in our testbenches. After collecting power, area, and timing results, we use polynomial regression models and model selection techniques based on *k*-fold cross validation [18] to tune the model parameters and fit the model.

IV. Results

In this section, we present power, performance, and area modeling results for each processing element type and perform a design space exploration on various DNN models such as VGG-16 [23], ResNet-20, ResNet-34, ResNet-50, and ResNet-56 [10] on CIFAR-10, CIFAR-100, and ImageNet datasets to iterate through our framework to demonstrate the flexibility of *QADAM* for future studies.

As detailed in Section III, *QADAM* framework provides power, performance, and area models that significantly speed up the design space exploration. Figure 3 shows the actual and estimated power, performance, and area results for each processing element type such as FP32, INT16, LightPE-1, and LightPE-2. Each data point in Figure 3 corresponds to a different hardware accelerator configuration in the comprehensive design space. As shown by the results, *QADAM*'s PPA models achieve high correlation to the actual PPA values. Figure 3 also shows that the FP32 implementation has the highest area and power cost whereas LightPEs have the lowest area and power results when one processing element is considered. This shows the hardware-efficiency of LightPEs when compared to conventional PE implementations.

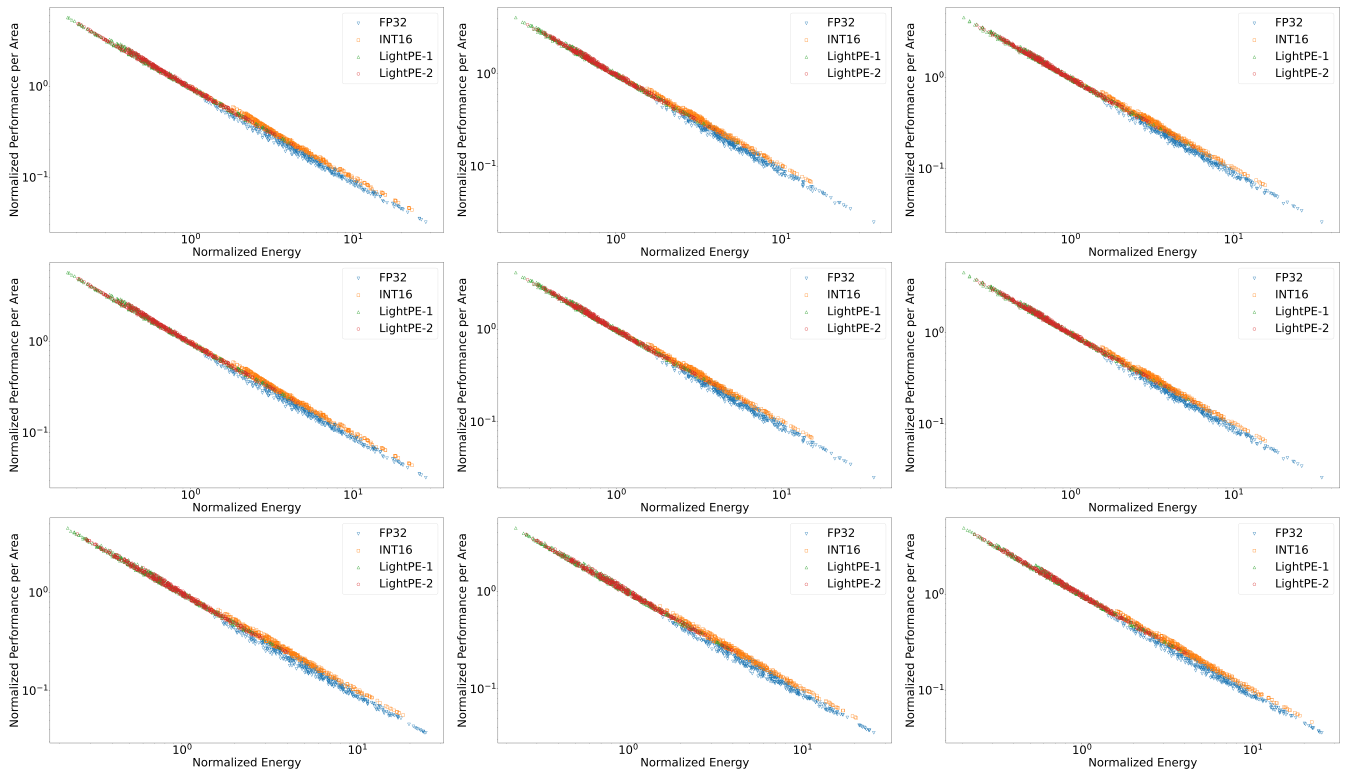


Fig. 4. Normalized performance per area vs. normalized energy results with respect to the INT16 hardware configuration with the highest performance per area for VGG-16 (left), ResNet-20 (middle), and ResNet-56 (right) for CIFAR-10 (top row), CIFAR-100 (middle row), and ImageNet (bottom row) design spaces. As it can be seen, LightPEs consistently outperform conventional INT16 and FP32 based designs in various models and datasets, thereby showing the benefits of using lower precision generalize across a variety of models.

A. Design Space Exploration Results

To show the hardware-efficiency of LightPEs to conventional PE types, we perform design space exploration on VGG-16 [23], ResNet-20, and ResNet-56 models on CIFAR-10/CIFAR-100 and VGG-16, ResNet-34, and ResNet-50 [10] models on ImageNet datasets as shown in Figure 4. We show the normalized performance per area and normalized energy results for each PE type with respect to the baseline INT16 based implementation with the highest performance per area for the given design space.

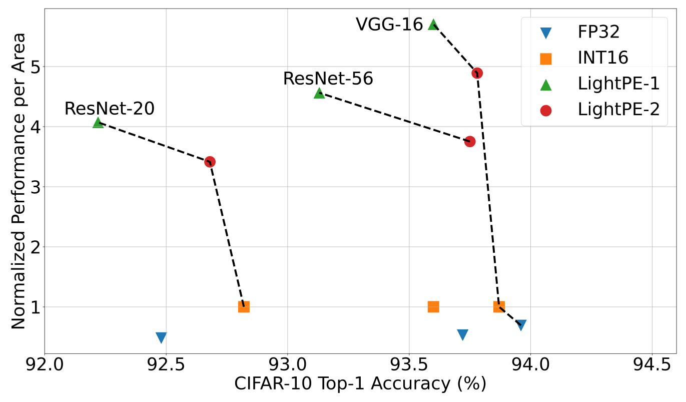
Figure 4 shows that LightPE implementations consistently outperform INT16 and FP32 implementations in both aspects, which proves their efficacy in terms of hardware-efficiency. Specifically, LightPE-1 and LightPE-2 achieve $4.8\times$ and $4.1\times$ more performance per area and $4.7\times$ and $4\times$ less energy on average across all workloads and datasets when compared to the best INT16 hardware configuration, respectively. On the other hand, INT16 baseline implementation achieves $1.8\times$ more performance per area and $1.5\times$ less energy on average when compared to the best FP32 configuration.

These conclusions hold for all the models and the datasets considered in this work such as VGG-16, ResNet-20, ResNet-34, ResNet-50, and ResNet-56 thereby showing that the benefits of using lower precision generalize across a variety of models. We conclude that different bit precisions and PE types can lead to significantly different performance per area and energy results which are two critical metrics for machine learning and systems community strives to improve upon.

B. Pareto-Optimality for Accuracy and Performance per Area

To show the accuracy and performance per area trade-off for different processing element types, we perform a Pareto-front analysis by training VGG-16, ResNet-20, and ResNet-56 models for CIFAR-10 and CIFAR-100 datasets. For both datasets, we perform five trials for each DNN model and processing element type and plot the mean top-1 accuracy results. The training recipe for both CIFAR-10/CIFAR-100 datasets follows prior art [4], [11] which uses stochastic gradient descent with nesterov momentum, weight decay 0.0005, batch size 128, 0.1 initial learning rate with decrease by $5\times$ at epochs 60, 120, and 160, and train for 200 epochs in total. We note that this training recipe is tuned for full-precision models. Therefore, the accuracy results for LightPE variants might be higher with proper hyperparameter tuning.

Figure 5 shows the normalized performance per area and accuracy results for FP32, INT16, LightPE-1, and LightPE-2. Performance per area results are normalized with respect to the best INT16 configuration for each DNN model. We plot the hardware configurations with the highest performance per area results for each processing element type. Next, we perform a Pareto-front analysis among different processing element types and show the Pareto-frontier with a dashed line for each DNN



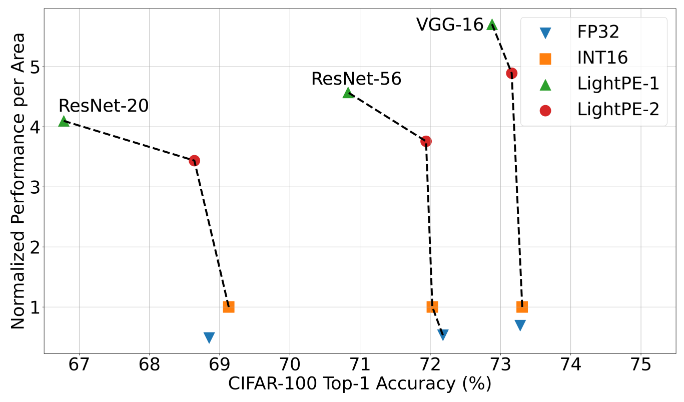
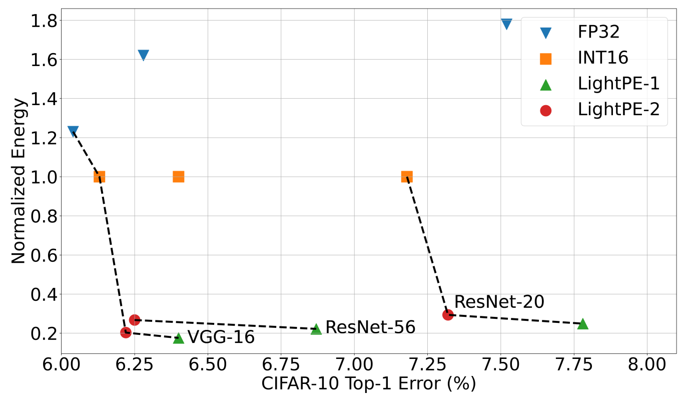


Fig. 5. Normalized performance per area and top-1 accuracy results for various processing element types such as FP32, INT16, LightPE-1, and LightPE-2 for CIFAR-10 (left chart) and CIFAR-100 (right chart). Each data point corresponds to the hardware configuration with the highest performance per area for the corresponding processing element type. Pareto-front is shown with a dashed line for each DNN model. LightPEs are consistently on Pareto-front for various DNN models.



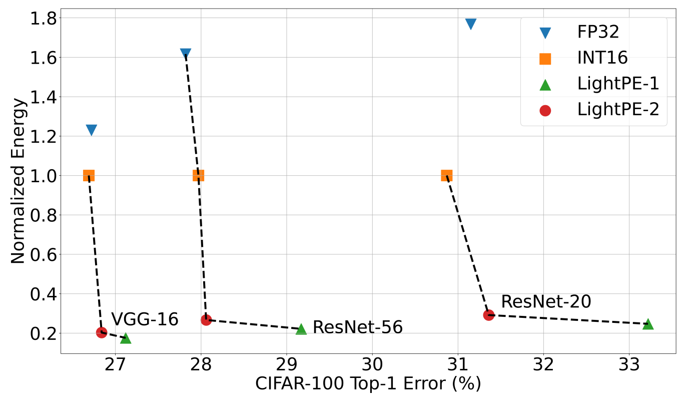


Fig. 6. Normalized energy and top-1 error results for various processing element types such as FP32, INT16, LightPE-1, and LightPE-2 for CIFAR-10 (left chart) and CIFAR-100 (right chart). Each data point corresponds to the hardware configuration with the lowest energy for the corresponding processing element type. Pareto-front is shown with a dashed line for each DNN model. LightPEs are consistently on Pareto-front for various DNN models.

model. We show that LightPEs are consistently on Pareto-front for various DNN models and datasets, whereas FP32 and INT16 based designs are occasionally dominated by LightPE variants. We show that LightPE-1 and LightPE-2 achieve on par accuracy results with FP32 and INT16 while achieving up to $5.7\times$ and $4.9\times$ more performance per area when compared to INT16 configuration, respectively.

C. *Pareto-Optimality for Accuracy and Energy*

We also perform a Pareto-front analysis for accuracy and energy results. We follow the same training methodology explained in Section IV-B. Figure 6 shows the normalized energy and accuracy results for FP32, INT16, LightPE-1, and LightPE-2 based designs. Energy results are normalized with respect to the best INT16 configuration for each DNN model. We show that LightPEs are systematically on Pareto-front for various DNN models and datasets. Specifically, LightPE-1 and LightPE-2 achieve $4.7\times$ and $4\times$ less energy on average across different workloads and datasets when compared to INT16 configuration, respectively. We also show that as model complexity increases, the accuracy gap between LightPEs and FP32 and INT16 based designs decreases. Thus, we conclude that our proposed LightPEs have promising results for training larger models with negligible accuracy loss while achieving significant performance per area and energy improvements.

V. Conclusion

In this work, we present *QADAM*, a quantization-aware highly parameterized power, performance, and area modeling framework for DNN accelerators. Our framework can foster the future research on design space exploration of DNN accelerators for various design choices such as bit precision, processing element type, scratchpad size of processing elements, global buffer size, device bandwidth, number of total processing elements in the the design, and DNN workloads. Our results show that different bit precisions and processing element types lead to significant differences in terms of performance per area and energy. Specifically, LightPE-1 and LightPE-2 achieve $4.8\times$ and $4.1\times$ more performance per area and $4.7\times$ and $4\times$ energy improvement on average when compared to the best INT16 hardware configuration, respectively. We also show that our proposed LightPEs consistently achieve Pareto-optimal results in terms of accuracy and performance per area and energy. Therefore, design space exploration of quantization-aware DNN accelerators merits a meticulous analysis that take these factors into account.

References

[1] E. Cai, D.-C. Juan, D. Stamoulis, and D. Marculescu, "Neuralpower: Predict and deploy energy-efficient convolutional neural networks," *arXiv preprint arXiv:1710.05420*, 2017.

[2] Y. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," in *ISCA*. Piscataway, NJ, USA: IEEE Press, 2016.

[3] T.-W. Chin, R. Ding, C. Zhang, and D. Marculescu, "Towards efficient model compression via learned global ranking," in *CVPR*, June 2020.

[4] T.-W. Chin, R. Ding, C. Zhang, and D. Marculescu, "Towards efficient model compression via learned global ranking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT*, 2019.

[6] R. Ding, Z. Liu, R. D. S. Blanton, and D. Marculescu, "Lightening the load with highly accurate storage- and energy-efficient lightnns," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 11, no. 3, Dec. 2018.

[7] M. Gao, J. Pu, X. Yang, M. Horowitz, and C. Kozyrakis, "Tetris: Scalable and efficient neural network acceleration with 3d memory," *SIGARCH Computer Architecture News*, vol. 45, no. 1, pp. 751–764, 2017.

[8] S. Gupta and B. Akin, "Accelerator-aware neural network design using automl," 2020.

[9] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," 2016.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[11] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang, "Soft filter pruning for accelerating deep convolutional neural networks," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, ser. IJCAI'18. AAAI Press, 2018, p. 2234–2240.

[12] A. Inci, E. Bolotin, Y. Fu, G. Dalal, S. Mannor, D. Nellans, and D. Marculescu, "The architectural implications of distributed reinforcement learning on cpu-gpu systems," *arXiv preprint arXiv:2012.04210*, 2020.

[13] A. Inci, M. M. Isgenc, and D. Marculescu, "DeepNVM++: cross-layer modeling and optimization framework of non-volatile memories for deep learning," *arXiv preprint arXiv:2012.04559*, 2020.

[14] A. Inci, M. M. Isgenc, and D. Marculescu, "Cross-layer design space exploration of nvm-based caches for deep learning," *NVMW*, 2021.

[15] A. Inci, S. G. Virupaksha, A. Jain, V. V. Thallam, R. Ding, and D. Marculescu, "Qappa: Quantization-aware power, performance, and area modeling of dnn accelerators," *2nd On-Device Intelligence Workshop, MLSys*, 2021.

[16] A. F. Inci, M. M. Isgenc, and D. Marculescu, "Deepnvm: A framework for modeling and analysis of non-volatile memory technologies for deep learning applications," ser. DATE '20, 2020.

[17] N. Jouppi, C. Young, N. Patil, D. A. Patterson, and et al., "In-datacenter performance analysis of a tensor processing unit," *ISCA*, 2017.

[18] F. Mosteller and J. W. Tukey, "Data analysis, including statistics," in *Handbook of Social Psychology, Vol. 2*, G. Lindzey and E. Aronson, Eds. Addison-Wesley, 1968.

[19] A. Parashar, M. Rhu, A. Mukkara, A. Puglielli, R. Venkatesan, B. Khailany, J. Emer, S. W. Keckler, and W. Dally, "Scnn: An accelerator for compressed-sparse convolutional neural networks," *ISCA*, 2017.

[20] H. Qi, E. R. Sparks, and A. Talwalkar, "Paleo: A performance model for deep neural networks," in *Proceedings of the International Conference on Learning Representations*, 2017.

[21] A. Samajdar, Y. Zhu, P. Whatmough, M. Mattina, and T. Krishna, "Scale-sim: Systolic cnn accelerator simulator," *arXiv preprint arXiv:1811.02883*, 2018.

[22] Y. Shao, B. Reagen, G.-Y. Wei, and D. Brooks, "Aladdin: A pre-rtl, power-performance accelerator simulator enabling large design space exploration of customized architectures," *ISCA*, 2014.

[23] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[24] J. E. Stine, I. Castellanos, M. Wood, J. Henson, F. Love, W. R. Davis, P. D. Franzon, M. Bucher, S. Basavarajaiah, J. Oh, and R. Jenkal, "Freepdk: An open-source variation-aware design kit," in *MSE'07*, 2007.

[25] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 2019, pp. 6105–6114.

[26] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10 778–10 787, 2020.

[27] L. Yang, Z. Yan, M. Li, H. Kwon, L. Lai, T. Krishna, V. Chandra, W. Jiang, and Y. Shi, "Co-exploration of neural architectures and heterogeneous asic accelerator designs targeting multiple tasks," in *DAC*, 2020, pp. 1–6.

[28] Y. Zhou, X. Dong, B. Akin, M. Tan, D. Peng, T. Meng, A. Yazdanbakhsh, D. Huang, R. Narayanaswami, and J. Laudon, "Rethinking co-design of neural architectures and hardware accelerators," 2021.