A FAST METHOD TO FINE-TUNE NEURAL NETWORKS FOR THE LEAST ENERGY CONSUMPTION ON FPGAS

Morteza Hosseini [*1], **Mohammad Ebrahimabadi** [*1], **Arnab Neelim Mazumder** [1], **Houman Homayoun**[2], **Tinoosh Mohsenin**[1]
[1] University of Maryland, Baltimore County, MD, USA
[2] University of California, Davis, CA, USA

ABSTRACT

Because of their simple hardware requirements, low bitwidth neural networks (NNs) have gained significant attention over the recent years, and have been extensively employed in electronic devices that seek efficiency and performance. Research has shown that scaled-up low bitwidth NNs can have accuracy levels on par with their full-precision counterparts. As a result, there seems to be a trade-off between quantization (q) and scaling (s) of NNs to maintain the accuracy. In this paper, we propose QS-NAS which is a systematic approach to explore the best quantization and scaling factors for a NN architecture that satisfies a targeted accuracy level and results in the least energy consumption per inference when deployed to a hardware–FPGA in this work. Compared to the literature using the same VGG-like NN with different q and s over the same datasets, our selected optimal NNs deployed to a low-cost tiny Xilinx FPGA from the ZedBoard resulted in accuracy levels higher or on par with those of the related work, while giving the least power dissipation and the highest inference/Joule.

1 INTRODUCTION

With the rapid growth of the computational ability of processors, convolutional neural networks (CNNs) have evolved to the point where they can surpass human-level accuracy in many applications such as speech recognition and computer vision (Tan & Le, 2019; Howard et al., 2017). With all the advancements in their performance, nevertheless, the energy consumption of CNNs on electronic devices is still far away from any levels comparable to their biological paradigms.

General-purpose CPUs and GPUs can efficiently test CNNs with precisions ranging from 8-bit fixed-point to 64-bit floating-point for inference (Jacob et al., 2018). Even though they can handle SIMD (Single Instruction Multiple Data) instructions such as xnor and population-count operations to implement extremely quantized NNs such as binarized neural networks (BNNs) (Hubara et al., 2016), they are not yet as versatile as FPGAs to efficiently implement CNNs with arbitrary quantization (Zhao et al., 2017). In contrast, FPGAs are commercial off-the-shelf devices that allow the implementation of any arithmetic with arbitrary precision using any desired implementation styles from serial to parallel, and in many cases, are used for prototyping and proof-of-concept development.

In light of minimizing energy consumption of CNNs on hardware, many recent efforts take quantization as a variable to minimize to the point at which accuracy is maintained (Khoram & Li, 2018; Choukroun et al., 2019). Also, many hardware-oriented methods take quantization fixed as a priori (Mirzaeian, 2019). For instance, Yang et al. (2019) adopted a 4-bit quantization in advance to start off with proposing an algorithm-hardware co-design for efficient CNN accelerators. All these effective methods, however, often disregard that a scaled-up lower-quantized (or vice-versa) CNN might enjoy higher efficiency on hardware while maintaining the same accuracy. Most recently, neural architecture search (NAS) has emerged as a methodology that relies on search strategies to manually or automatically explore efficient CNNs. Abdelfattah et al. (2020) proposed Codesign-NAS that exploits reinforcement learning based strategies to automatically navigate the search space of CNNs and hardware architectures to simultaneously improve accuracy and efficiency, which takes approximately 1000 GPU-hours to provide an efficient solution for the CIFAR-100 dataset on FPGA. *In this paper, we raise a problem as: given a choice of CNN that has a few degrees of freedom implemented on a hardware that also has a few degrees of freedom, what is the best selection of the independent variables that meets an implementation goal?* A grid search over a span of variously

[*]Equal contribution.

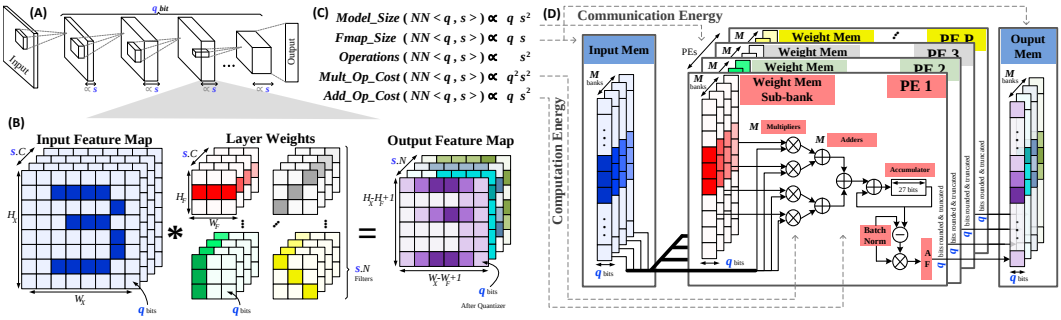

Figure 1: (A) a baseline $\mathcal{NN}\langle q, s\rangle$, (B) an arbitrary layer of $\mathcal{NN}\langle q, s\rangle$ in more detail, (C) characteristics of $\mathcal{NN}\langle q, s\rangle$, and (D) a systolic architecture that adopts $\mathcal{NN}\langle q, s\rangle$ for implementation.

quantized scaled CNNs experienced on hardware can provide options to trade off between accuracy and efficiency, but not necessarily the most optimal options. We present QS-NAS which is a fast regression-based method to explore the optimal quantization and scaling factors for a CNN architecture that results in the least energy consumption per inference and meets a targeted accuracy when deployed to a hardware.

2 Problem Statement and a Regression-Based Solution

To simplify and unify formulations, we assume a CNN is made by stacking L blocks (Fig. 1-A). Each block is composed of a 2D convolution with weights and input quantized to q bits (input/output layers excluded), followed by a batch-normalization and a quantizing ReLU activation function as instructed by Coelho Jr et al. (2020). To compensate the accuracy loss resulted by low quantization, we impose a scaling variable s to each 2D convolution layer of the CNN, that scales the number of all filters per layers, and correspondingly the number of channels per filter & fmap as illustrated in Fig. 1-B. We refer to this CNN as either $\mathcal{NN}\langle q, s\rangle$ or as q-bit NN-s. As denoted in Fig. 1-C, the sizes of the model and fmap of $\mathcal{NN}\langle q, s\rangle$ are proportional to qs^2 and qs respectively that will govern the sizes of weight memory and fmap memory in hardware. The computation is also proportional to s^2. Since $q \leq 8$ in our work, we observe that look-up table (LUT)-based MAC components are more energy-efficient than FPGA's built-in digital signal processor (DSP) units, and note that the power and utilization costs of LUT-based combinational multipliers and adders in FPGAs are proportional to q^2 and q respectively (Sun et al., 2008). Given an adequate choice of FPGA, we consider a commonly-practiced systolic architecture, as depicted in Fig. 1-D, consisting of a 2D grid of $P \times M$ multiply-accumulate (MAC) units, and define an optimization problem as:

$$\begin{aligned} \min_{q\,,\,s} \quad & Energy(\ \mathcal{HW} \mid \mathcal{NN}\langle q, s\rangle\ ) \\ s.t. \quad & \mathcal{NN}\langle q, s\rangle = \bigodot_{i=1\ldots L} \mathcal{F}_i^{\langle q,\, s\,\hat{N}_i,\, \hat{H}_{F_i},\, \hat{W}_{F_i},\, s\,\hat{C}_i\rangle}(X^{\langle q,\, \hat{H}_{X_i},\, \hat{W}_{X_i},\, s\,\hat{C}_i\rangle}) \\ & Accuracy(\mathcal{NN}) \geq \text{target\_accuracy} \end{aligned} \tag{1}$$

that reads: minimize the energy per inference of CNN deployed to hardware with respect to q and s, provided that a target_accuracy is satisfied. The solution to this problem lies in understanding the behaviour of both the functions $Energy$ and the $Accuracy$. Based on the evidence and experiments we use regression-based approximators to predict the behaviour of the two functions.

It is well-studied that scaling up CNNs increases their accuracy (Tan & Le, 2019; Howard et al., 2017), and that the accuracy of quantized models degrades to their least when the quantization is lowered to BNNs (Zhao et al., 2017; Umuroglu et al., 2017). Clearly, the accuracy does not exceed a certain level and will saturate to a level no more than 100% for large q and s. We postulate that a rational polynomial as formulated in Eqn. (2)-II can approximate the accuracy of $\mathcal{NN}\langle q, s\rangle$, because it satisfies all the mentioned characteristics. We also model the energy and power of the hardware with meticulous consideration upon full recognition of its components and their toggling rate during the run-time. For the hardware, the majority of energy consumption for a workload $\mathcal{NN}\langle q, s\rangle$ are attributed to computation of multiplications ($\propto q^2s^2$) and additions ($\propto qs^2$), and communication for the model weight ($\propto qs^2$) and the fmap ($\propto qs$) parameters. Taking a static power into account, if the execution time is constant for all implementations of $\mathcal{NN}\langle q, s\rangle$, the energy and power become

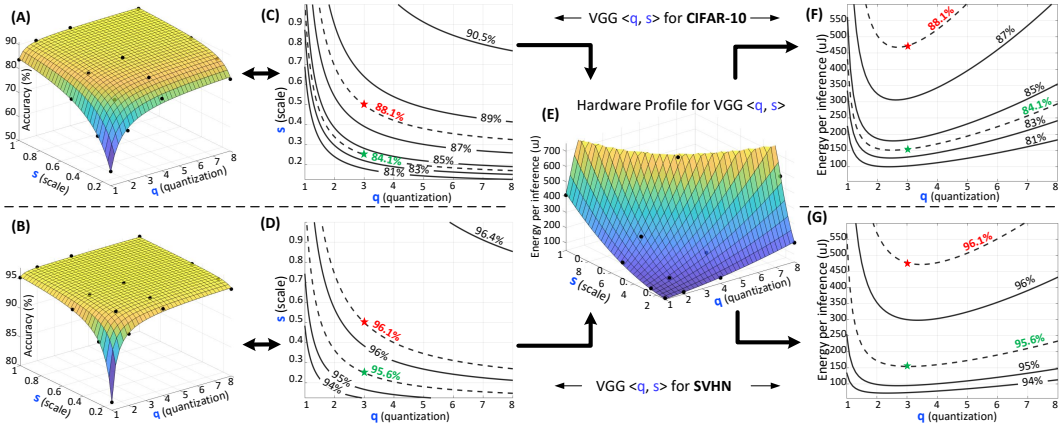

Figure 2: (A)&(B) regression on the accuracy of 16 cross-product experiments (black dots) of VGG$\langle q, s\rangle$, (C)&(D) accuracy contours extracted from the regression, (E) regression on J/inference of VGG$\langle q, s\rangle$ deployed and experienced (black dots) on FPGA, and (F)&(G) Concluding the minimum J/inference of VGG$\langle q, s\rangle$ on FPGA w.r.t. q and accuracy for CIFAR-10 & SVHN respectively.

proportional and will correspond to Eqn. (2)-I, which will be further elaborated in Section 4. Thus:

$$\begin{cases} Energy(\mathcal{HW}|\mathcal{NN}_{\langle q,s\rangle}) & = \hat{E}_3.q^2.s^2 + \hat{E}_2.q.s^2 + \hat{E}_1.q.s + \hat{E}_0 \\ Accuracy(\mathcal{NN}_{\langle q,s\rangle}) & = \dfrac{\hat{A}_6.q.s \;\; + \hat{A}_5.s \;\;\; + \hat{A}_4.q \;\; + \hat{A}_3}{q.s \;\; + \hat{A}_2.s \;\;\; + \hat{A}_1.q \;\; + \hat{A}_0} \end{cases} \tag{2}$$

where $\hat{E}_i$ and $\hat{A}_i$ are constants determined after regression. To minimize the $Energy$, we plug s as a function of q and $Accuracy$ in function $Energy$ and solve $\frac{\partial Energy}{\partial q}|_{Accuracy=target\_accuracy} = 0$.

3 Training VGG $\langle q, s\rangle$ on CIFAR-10 and SVHN

We perform our experiments on two datasets: CIFAR-10 and SVHN. We selected a VGG-like architecture (Simonyan et al., 2014) that is extensively used in the works of Hubara et al. (2016); Zhao et al. (2017); Umuroglu et al. (2017); Alemdar et al. (2017); Prost-Boucle et al. (2017) using various scale and quantization with the following architecture and with the q-bit layers implicitly included:

$$(in)-2\times(64sC_{3\times3})-MP_{2\times2}-2\times(128sC_{3\times3})-MP_{2\times2}\;-2\times(256sC_{3\times3})-MP_{2\times2}-2\times(512sFC)-10SVM-(out). \tag{3}$$

This VGG has approximately $2 \times 149s^2$ million q-bit multiply or add operations and a model size $3.7qs^2$ Mb. The largest fmap size is contributed by its second layer which has a size $64qs$ Kb. We down-scaled this VGG from 1 to 1/8th, and up-scaled the quantization from 1, which is a BNN, to 8 bits. We used QKeras (Coelho Jr et al., 2020) libraries to train each of the 16 cross-product experiment in 120 epochs. We used a least-square-error method to fit the rational polynomial to our 16-point experiments. Figs. 2-(A) and 2-(B) show the fitted surfs for the 16 experiments (black dots). For the CIFAR-10 and the SVHN, the root mean square error (RMSE) is 0.74% and 0.18% respectively. We evaluate this regression on two new testing data, i.e. VGG$\langle q{=}3, s{=}1/4\rangle$ and VGG$\langle q{=}3, s{=}1/2\rangle$, that predicts 84.1% / 95.7% and 88.1% / 96.1%, whilst actual values are 84.2% / 95.9% and 88.7% / 96.2% for CIFAR-10 / SVHN datasets respectively. On a 2D plot, Figs. 2-(C) & 2-(D) demonstrate how the accuracy of the VGG for the two datasets behave with respect to q and s.

4 Scalable Hardware to Implement VGG $\langle q, s\rangle$

Similar to works of Zhao et al. (2017); Umuroglu et al. (2017); Alemdar et al. (2017); Prost-Boucle et al. (2017), we design a hardware accelerator for FPGAs, as partially depicted in Fig. 1-D and described in Verilog HDL, that relies on FPGA's resources and BRAMs that store the weight and intermediate fmap of VGG$\langle q, s\rangle$ during run-time. All our hardware configurations were implemented on a ZedBoard evaluation platform which is equipped with the Xilinx FPGA XC7Z020-CLG484 that incorporates 4.9Mb (=140×36Kb) on-chip BRAMs. Also, all power and latency analyses were measured using the Xilinx Vivado Design Suite. The design comprises two main blocks: 1) an array of P processing engines (PEs) that each incorporates M multipliers/adders and a weight memory sub-bank that stores a partition of the CNN weights, and 2) an input and an output memory that swap turn per process termination of every CNN's layer and store the temporary fmap data. Both

Table 1: Comparison with FPGA implementations using differently quantized scaled VGG-like architectures on CIFAR-10 and SVHN. P_{chip} and P_{wall} are the powers of FPGA and the board.

Dataset	Authors	Platform	Price	Workload (NN-64s)	Quant. (q bit)	Accuracy (%)	Clock (MHz)	Throughput (FPS)	P_{chip} (W)	P_{wall} (W)	FPS/P_{chip} (Inference/J)	FPS/P_{wall} (Inference/J)
CIFAR-10	**This work**	ZedBoard	**low**	NN-16	3	84.16±(0.15)	143	2,982	**0.42**	-	**7,158**	-
	This work	ZedBoard	**low**	NN-32	3	88.74±(0.18)	143	2,982	**1.23**	-	2,418	-
	Umuroglu et al. (2017)	ZC706	high	NN-64	1	80.10	200	21,900	3.60	11.7	6,080	1,870
	Prost-Boucle et al. (2017)	VC709	high	NN-64	2	86.71	250	27,043	6.80	-	3,976	-
	Prost-Boucle et al. (2017)	VC709	high	NN-128	2	89.39	250	13,526	13.64	-	992	-
	Zhao et al. (2017)	ZedBoard	low	NN-128	1	88.68	143	168	-	4.7	-	35.8
	Alemdar et al. (2017)	VC709	high	NN-128	2	87.89	200	1,695	9.58	-	178	-
SVHN	**This work**	ZedBoard	**low**	NN-16	3	95.86±(0.10)	143	2,982	**0.42**	**-**	**7,158**	-
	This work	ZedBoard	**low**	NN-32	3	96.21±(0.08)	143	2,982	**1.23**	**-**	2,418	-
	Umuroglu et al. (2017)	ZC706	high	NN-64	1	96.40	200	21,900	3.60	11.7	6,080	1,870
	Prost-Boucle et al. (2017)	VC709	high	NN-64	2	97.60	250	27,043	7.08	-	3,820	-
	Alemdar et al. (2017)	VC709	high	NN-64	2	97.27	200	3,390	4.80	-	709	-
	Prost-Boucle et al. (2017)	VC709	high	NN-128	2	97.70	250	13,526	13.70	-	987	-

the partitioned CNN weights and the fmap data are packed along their channels for every VGG's layer and are stored and folded in one or multiple entries in their designated BRAMs in hardware. Each PE concurrently computes one output channel from a layer of the CNN at a time, while internally processes its task using an input-channel tiling scheme where the packed fmap values directly accessed from the input memory, and the packed weight values from the weight sub-banks are element-wise multiplied and accumulated. Using a roofline model, the most energy-efficient configuration of the hardware for our workload VGG $\langle q, s\rangle$ is given by P=$64s$ and M=64 that correspond to the VGG's second layer. With P= M= $64s$ and the $Model_Size = 3.7qs^2$Mb, the width and depth of the total weight memory is $4096qs^2$ and $3.7qs^2/64s/64s/q$ Mega (=872) entries respectively. Also, given the largest $Fmap\_Size = 64qs$ Kb of our VGG, the width and depth of the instantiated input/output memory units in hardware are $64qs$ and $64qs/64s/q$ Kilo (=1024) entries. Taking 1024 for the depth of all memory sub-banks in our hardware configurations, the total design requires approximately $[4096qs^2/36]$ and $2 \times [64qs/36]$ of 36Kb BRAMs for the weight memory and input/output memory allocation that justifies employing the tiny FPGA from the ZedBoard for the span of q and s in this work. The peak performance of the hardware is $2.P.M.freq$ that, given P= M= $64s$, is approximately $1.2s^2$ GOPS at clock rate 143 MHz. Considering the $2 \times 149s^2$ million operations of our VGG $\langle q, s\rangle$, the execution time per inference and the throughput of all our implementations at clock rate 143 MHz remain approximately constant and equal to 0.34 mS and 2,982 FPS respectively. Thus, with scaling the VGG$\langle q, s\rangle$, the efficiently-configured $\mathcal{HW}$ scales correspondingly such that the width of weight and fmap memory units, and consequently their power consumption, scales by qs^2 and qs respectively. The number of multipliers and adders scales by $PM \propto s^2$, and their power consumption scale by q^2s^2 and qs^2 respectively. Thus, with the constant execution time and considering a static power, the Eqn. (2)-I is once again concluded from an implementation perspective.

The 16 cross-product experiments of the VGG $\langle q, s\rangle$ were implemented on the FPGA and the power, delay, and energy per inference were measured for each deployment. Fig. 2-(E) shows the energy measurements of 12 out of the 16 experiments (black dots) on our selected tiny FPGA, and a surf in accordance to Eqn. (2)-I that fits the experiments using a least-square-error method with an RMSE= 9.7uJ. For evaluation, we tested the $Energy(\mathcal{HW}_{\langle P=16,\, M=16\rangle}|\mathcal{NN}_{\langle q=3,\, s=1/4\rangle})$ and the $Energy(\mathcal{HW}_{\langle P=32,\, M=32\rangle}|\mathcal{NN}_{\langle q=3,\, s=1/2\rangle})$ configurations, for which the polynomial predicts 154 uJ and 477 uJ, whereas their actual measurements are 140 uJ and 414 uJ respectively. Having determined the two unknown functions, the system of Eqn. (2) is thus established and different optimal VGG architectures for different accuracy levels can be pursued. Figs. 2-(F) and 2-(G) plot the function $Energy$ w.r.t. to q and $Accuracy$, demonstrating convex curves that reveal different quantization levels result in the least energy consumption given different accuracy levels for the two datasets. We look for delicate pairs of q and s that result in moderately high (VGG$\langle q=3,\, s=1/2\rangle$) and low (VGG$\langle q=3,\, s=1/4\rangle$) accuracy levels, such that q is a natural number near the minima of the convex curves, and s is a power of 2 (for hardware-friendly implementation), and the selection of both of which do not over-utilize the FPGA resources.

5 Comparison to the Related Work

Compared to the literature, as summarized in Table 1, using the same VGG-like CNN architecture, our selected optimal CNN configurations, VGG$\langle q=3,\, s=1/2\rangle$ and VGG$\langle q=3,\, s=1/4\rangle$, deployed to the tiny low-cost low-power Xilinx FPGA from the ZedBoard results in accuracy levels higher or on par with those of the related work, while giving the least power dissipation and the highest inference/J.

6 CONCLUSION

We proposed QS-NAS: a regression-based approach to explore the optimal quantization (q) and scaling (s) of a NN for the least energy consumption on FPGAs. Having designed a scalable hardware, we empirically approximate both accuracy and energy per inference of $\mathcal{NN}_{\langle q,\, s\rangle}$ with polynomials. Given the two approximators, we obtain a pair of $\langle q,\, s\rangle$ that minimizes the energy on hardware for a targeted accuracy. Our methodology is fast, simple, and scalable and has no commitment to a specific quantization, but rather takes quantization (in tandem with scaling) as a variable to optimize.

REFERENCES

Mohamed S Abdelfattah et al. Codesign-nas: Automatic fpga/cnn codesign using neural architecture search. In *The 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2020.

Hande Alemdar et al. Ternary neural networks for resource-efficient ai applications. In *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017.

Yoni Choukroun, Eli Kravchik, Fan Yang, and Pavel Kisilev. Low-bit quantization of neural networks for efficient inference. In *ICCV Workshops*, pp. 3009–3018, 2019.

Claudionor N Coelho Jr et al. Ultra low-latency, low-area inference accelerators using heterogeneous deep quantization with qkeras and hls4ml. *arXiv preprint arXiv:2006.10159*, 2020.

Andrew G Howard et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Itay Hubara et al. Binarized neural networks. In *Advances in neural information processing systems*, 2016.

Benoit Jacob et al. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

Soroosh Khoram and Jing Li. Adaptive quantization of neural networks. In *International Conference on Learning Representations*, 2018.

Ali Mirzaeian. A technical report on accelerator design for deep neural networks. Technical report, 2019.

Adrien Prost-Boucle, Alban Bourge, Frédéric Pétrot, Hande Alemdar, Nicholas Caldwell, and Vincent Leroy. Scalable high-performance architecture for convolutional ternary neural networks on fpga. In *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, pp. 1–7. IEEE, 2017.

Karen Simonyan et al. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Junqing Sun et al. High-performance mixed-precision linear solver for fpgas. *IEEE Transactions on Computers*, 2008.

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

Yaman Umuroglu et al. Finn: A framework for fast, scalable binarized neural network inference. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2017.

Yifan Yang, Qijing Huang, Bichen Wu, Tianjun Zhang, Liang Ma, Giulio Gambardella, Michaela Blott, Luciano Lavagno, Kees Vissers, John Wawrzynek, et al. Synetgy: Algorithm-hardware co-design for convnet accelerators on embedded fpgas. In *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 23–32, 2019.

Ritchie Zhao et al. Accelerating binarized convolutional neural networks with software-programmable fpgas. In *Proceedings of the 2017 ACM/SIGDA International Symposium on FPGAs*, 2017.