Fast prototyping of Quantized neural networks on an FPGA edge computing device with Brevitas and FINN

Devansh Chawda
SDU Mechatronics(CIM)
University of Southern Denmark(SDU)
Sønderborg, Denmark
decha20@student.sdu.dk

Benaoumeur Senouci
Centre for Industrial Mechanics(CIM)
University of Southern Denmark(SDU)
Sønderborg, Denmark
senouci@sdu.dk

2024 Fifteenth International Conference on Ubiquitous and Future Networks (ICUFN) | 979-8-3503-8529-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICUFN61752.2024.10625618

***Abstract*—In this paper, we propose a solution for fast prototyping of Deep learning neural network models on edge computing devices like FPGA for researchers with limited knowledge of high level languages like VHDL. We use Xilinx' Brevitas tool for Quantization and FINN framework for deployment/inference on Pynq-Z2 board. The paper will also share presently available methods for FPGA prototyping and how tools like Brevitas and FINN can be used for more efficient inference of DNN on small scale edge computers like FPGA by levaraging their 1. Quantization Aware Training(QAT) and Post Training Quantization(PTQ) 2. Streamlining networks and transformations 3. Dataflow partitioning of the NN model using FINN compiler 4. DMA, FIFO and IP generation for HW build and 5. Inference on FPGA using PYNQ python Driver. The weights and activations of a custom model were quantised from floating points to 8, 4 and 2 bit for which an accuracy drop of 0.1%, 0.8% and 7.6% was observed respectively.**

***Index Terms*—FPGA, Neural Network, DNN, CNN, Brevitas, FINN, Quantization, Pynq**

I. Introduction

In recent years, there has been a lot of development in the field of Artificial Intelligence and Machine Learning. With this the size of machine learning models especially Deep Neural Networks and their need for Computation power has also increased significantly. For embedded systems, when these models are deployed on edge devices for inference, they require powerful GPUs that consume a lot of energy for real time results. Especially in Automotive industry where energy consumption and real time response are some of the most important factors. As a result, recently we have seen a trend of Electric Vehicles manufacturers switching from GPUs to hardware devices like ASICS and FPGA.

The lack of expertise in high level languages like VHDL has restricted a lot of development for FPGAs. The lack of research for small scale FPGA devices with low memory has been another issue since high performance FPGA devices are very expensive for fast prototyping. Research has shown a significant reduction in power consumption upto 70 percent when using FPGA instead of GPUs. [1] Recent research in quantization has shown a very low drop in accuracy after quantizing floating points to 8 bits, 4 bits and even 2 bits. [2] But still Neural Networks development on FPGA depends highly on availability of open source IP cores from Xilinx.

We propose a solution for a fast, efficient and easy prototyping of neural networks on edge FPGA devices with FINN. [3] The FINN framework is an experimental Xilinx project to explore deep neural network inference on FPGA. FINN compiler maps QNNs to dataflow-style FPGA architectures. It uses pre-quantized neural network architecture from Brevitas(Library for Neural Network Quantization) and converts it to a dataflow stype architecture from which each layer is then converted to an IP core which are then connected into a FIFO architecture using DMA and then deployed on a FPGA device.

II. Related works

The advantages of running deep neural network inference on FPGA has been widely researched. But most of the research focuses on designing custom IPs for a particular model or application. Some tools like Vitis AI have been developed to help with large models on high-scale FPGA devices like Zynq UltraScale and Alveo [4]. Advances in the field of quantization has led to a new interest in developing large deep learning models for FPGA [5]. Especially 8 bit and 4 bit quantization has shown some promising results. Brevitas is a pytorch library for development of quantized neural networks(QNN) by Xilinx. It has it's own quantized operations/layers for Quantization-Aware training(QAT) and supports post training quantization(PTQ) [6]. Brevitas is a precursor to an experimental project by AMD Research and Advanced Development(RAD) called FINN. Under this project, they have developed FINN framework which converts a QNN into a dataflow-stye architecture for customised networks. And FINN compiler which helps to compile this dataflow-style architecture into hardware language for FPGA deployment. It is not a common solution for all DNNs but it relies on co-design and design space exploration for efficient hardware acceleration.

There has been a some research on this tool and its effectiveness. While some researchers like [7] focused on creating

a custom binary neural network for detecting masks during COVID-19 which requires a very careful customisation and design exploration to maintain accuracy, other researchers like [8], [9] and [10] have focused on developing quantized versions of common state-of-the-art models like YOLO, LeNet5 and AlexNet for FPGA. Majority of the research is focused on developing models for high-end FPGA platforms or convert pre-existing models.

FINN framework was mainly designed to help fast development of custom neural networks for FPGA and its deployment to edge computing devices which are typically low-end FPGAs with resource constraints. Hence, in this paper we have developed a custom neural network with quantization upto 2 bits and deploy it on an entry level Pynq-Z2 platform.

III. Methodology

A custom model was developed for the task of emotion recognition and trained using widely available dataset, i.e. FER2013 containing 35000 images of 7 different classes. The full dataset is divided into 7 classes of different human emotions like happy, sad, angry, etc. Eah image in the dataset is in grayscale and has a size of 48x48 pixels.

A. Architecture

The custom emotion detection model is based on a convolution neural network (CNN) architecture tailored for processing facial images. The model begins with a series of convolution layers, each followed by batch normalization and rectified linear unit (ReLU) activation functions to extract hierarchical features from the input images. Max pooling and dropout layers are strategically inserted to downsample the feature maps and to prevent overfitting. The feature maps are then flattened and passed through two fully connected layers. The first fully connected layer consists of 256 neurons, followed by batch normalization and ReLU activation. The final layer is a fully connected layer with seven neurons, corresponding to the seven emotion classes present in the dataset.

Incorporation of Brevitas for quantization-aware training (QAT) is integral to working with FINN framework and FPGA. This technique enables us to train the model with quantized weights and activation, effectively simulating the effects of hardware quantization during inference on FPGA platforms. By using the model with quantization constraints, we ensure compatibility with the resource-efficient demands of FPGA deployment, without compromising inference accuracy.

Weights and activations were quantized from floating point to different bit-widths, namely 8bits, 4bits and 2bits. The model is trained using the Adam optimizer with categorical cross-entropy loss, aiming to minimize the discrepancy between predicted and ground truth emotion labels. Training parameters, including batch size, learning rate, and number of epochs, are carefully selected to optimize model performance. The full architecture of the model used in the final test is as shown below.

TABLE I
Emotion Detection Model Architecture

Layer Type	Output Shape	Details
QuantConv2d	[1, 32, 46, 46]	32 filters, 3x3 kernel, INT quantization
BatchNorm2d	[1, 32, 46, 46]	Batch normalization
QuantReLU	[1, 32, 46, 46]	INT quantization
QuantConv2d	[1, 64, 44, 44]	64 filters, 3x3 kernel, INT quantization
BatchNorm2d	[1, 64, 44, 44]	Batch normalization
QuantReLU	[1, 64, 44, 44]	INT quantization
MaxPool2d	[1, 64, 22, 22]	Max pooling with 2x2 kernel
Dropout	[1, 64, 22, 22]	Dropout with 25% probability
QuantConv2d	[1, 128, 20, 20]	128 filters, 3x3 kernel, INT quantization
BatchNorm2d	[1, 128, 20, 20]	Batch normalization
QuantReLU	[1, 128, 20, 20]	INT quantization
QuantConv2d	[1, 128, 18, 18]	128 filters, 3x3 kernel, INT quantization
BatchNorm2d	[1, 128, 18, 18]	Batch normalization
QuantReLU	[1, 128, 18, 18]	INT quantization
MaxPool2d	[1, 128, 9, 9]	Max pooling with 2x2 kernel
QuantConv2d	[1, 256, 7, 7]	256 filters, 3x3 kernel, INT quantization
BatchNorm2d	[1, 256, 7, 7]	Batch normalization
QuantReLU	[1, 256, 7, 7]	INT quantization
QuantConv2d	[1, 256, 5, 5]	256 filters, 3x3 kernel, INT quantization
BatchNorm2d	[1, 256, 5, 5]	Batch normalization
QuantReLU	[1, 256, 5, 5]	INT quantization
MaxPool2d	[1, 256, 2, 2]	Max pooling with 2x2 kernel
Dropout	[1, 256, 2, 2]	Dropout with 25% probability
Flatten	[1, 1024]	Flatten output
QuantLinear	[1, 256]	256 output neurons, INT quantization
BatchNorm1d	[1, 256]	Batch normalization
QuantReLU	[1, 256]	INT quantization
Dropout	[1, 256]	Dropout with 50% probability
QuantLinear	[1, 7]	7 output neurons, INT quantization
LogSoftmax	[1, 7]	Log Softmax activation

B. Development Workflow

The development process starts with the development of a NN model, carefully designed for the specific requirements of the target application. The model architecture is designed and trained on pytorch with the help of Brevitas for QAT. Utilizing tools provided by Brevitas, the precision of the model is systematically reduced to 8, 4 and 2 bit-width representations, ensuring compatibility with FINN to achieve the desired performance metrics.

Following quantization, network surgery is applied if required. Then the model is transformed into a dataflow graph representation using FINN's ModelWrapper() function. This graph captures the computational flow of the model, expressing it as a structured network of interconnected nodes, thereby preparing it for deployment onto FPGA hardware.

The compiled model is subsequently synthesized for deployment on FPGA hardware. High-Level Synthesis (HLS) tools, including Vivado HLS or Vitis HLS, are utilized to convert the dataflow graph into synthesizable hardware description language (HDL) code. This HDL code represents the model's computations in a format used for FPGA synthesis. A folding method is applied by adjusting PE and SIMD values to control parallelism.

FPGA synthesis, a critical stage in the deployment pipeline, translates the HDL code into a FPGA bitstream, the configuration file necessary for programming the FPGA fabric. This process, performed using Vivado and Vitis, ensures that

the synthesized hardware design is compatible with the target FPGA platform.

With the FPGA bitstream generated, it is deployed onto the target PYNQ board, configuring the FPGA with the synthesized hardware design. A custom driver is developed for communication between the host CPU and the FPGA accelerator.

Inference is performed on the PYNQ Z2 board using Python scripts or Jupyter Notebooks, leveraging the custom driver to interface with the FPGA-accelerated model. Input data is processed by the FPGA accelerator, with inference results communicated back to the processor for further analysis. The full end-to-end design flow can be seen in Figure 1.

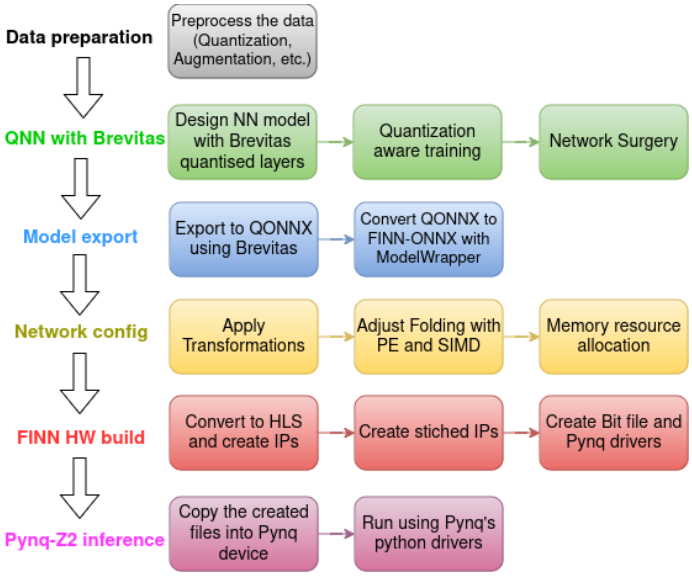

Fig. 1. Design Workflow

IV. Results

The non Quantized neural network was trained on intel i5 11th gen processesor for 10 epochs. Average runtime per epoch was around 20 minutes. After 10 epochs a test accuracy of 64% was calculated. Applying QAT, on average it took 5 minutes per epoch to train. Thereby 75% reduction in training time was observed per epoch. The neural network model was then quantised at different bit-widths and synthesized for testing the inference on FPGA. No visible change in training time was observed for quantization at different bit-widths.

TABLE II
Accuracy at different levels of quantization

Weights bit-width	Activation bit-width	Accuracy
32-bit	32-bit	64.69%
8-bit	8-bit	64.55%
4-bit	4-bit	63.81%
2-bit	2-bit	57.09%

The test accuracy for the 8bit quantised model remained same at 64%. But for 4bit weight and activation, the accuracy dropped to 63.81%. And lastly with weight and activation bit-width set to 2 bits, the accuracy dropped to a whopping 57.09% showing over 7% drop from unquantised as well as quantised at higher bit-width Neural Networks. The observed accuracy of different NNs is represented in the table 2.

V. Conclusion

In this paper, a custom neural network was developed for testing the efficiency of a neural network quantization library called Brevitas with FINN, a deep neural network inference tool for FPGA. The custom DNN was trained on FER2013 dataset for emotion recognition and results were compared various bit-width quantization of weights and activation. Using Quantization-aware Training(QAT), the training time was reduced by 75% from non-quantised model. Meanwhile the accuracy of the model remained same for 8bit and 4bit quantisation. A significant loss of accuracy was only observed while testing quantization at 2 bits. In future, this work will be used to develop a DNN which uses 3D data from a LIDAR camera to perform object detection and make driving decisions using reinforcement learning on an edge FPGA device.

References

[1] M. Qasaimeh et al, "Benchmarking vision kernels and neural network inference accelerators on embedded platforms," Journal of Systems Architecture, vol. 113, pp. 101896, 2021.

[2] T. Liang et al, "Pruning and quantization for deep neural network acceleration: A survey," Neurocomputing (Amsterdam), vol. 461, pp. 370-403, 2021.

[3] M. Blott et al, "FINN- R: An End-to-End Deep-Learning Framework for Fast Exploration of Quantized Neural Networks," ACM Transactions on Reconfigurable Technology and Systems, vol. 11, (3), pp. 1-23, 2018.

[4] Luke Kljucaric and Alan D. George. 2023. Deep Learning Inferencing with High-performance Hardware Accelerators. ACM Trans. Intell. Syst. Technol. 14, 4, Article 68 (August 2023), 25 pages.

[5] Anouar Nechi, Lukas Groth, Saleh Mulhem, Farhad Merchant, Rainer Buchty, and Mladen Berekovic. 2023. FPGA-based Deep Learning Inference Accelerators: Where Are We Standing? ACM Trans. Reconfigurable Technol. Syst. 16, 4, Article 60 (December 2023), 32 pages.

[6] Alessandro Pappalardo, "Xilinx/brevitas: v0.10.2". Zenodo, Feb. 19, 2024. doi: 10.5281/zenodo.10680356.

[7] N. Fasfous et al, "BinaryCoP: Binary neural network-based COVID-19 face-mask wear and positioning predictor on edge devices," in 2021, . DOI: 10.1109/IPDPSW52791.2021.00024.

[8] B. Günay, S. B. Okcu and H. Ḃilge, "LPYOLO: Low precision YOLO for face detection on FPGA," Cornell University Library, arXiv.org, Ithaca, 2022. DOI: 10.48550/arxiv.2207.10482.

[9] J. C. Njuguna, A. T. Çelebi and A. Çelebi, "Implementation and Optimization of LeNet-5 Model for Handwritten Digits Recognition on FPGAs using Brevitas and FINN," 2023 Innovations in Intelligent Systems and Applications Conference (ASYU), Sivas, Turkiye, 2023, pp. 1-5, doi: 10.1109/ASYU58738.2023.10296630.

[10] E. Rex and X. Wang, "Efficient Quantization and Hardware Implementation of AlexNet on Resource-limited FPGAs," 2023 IEEE 14th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON), New York, NY, USA, 2023, pp. 0399-0406, doi: 10.1109/UEMCON59035.2023.10316010.