From Tensors to FPGAs: Accelerating Deep Learning

Hardik Sharma[†] Jongse Park[†§] Balavinayagam Samynathan[§] Behnam Robatmili[§]
Shahrzad Mirkhani[§] Hadi Esmaeilzadeh[‡§]

[†]Georgia Institute of Technology [§]Bigstream, Inc. [‡]University of California, San Diego

hsharma@gatech.edu {jongse, bala, behnam, shahrzad}@bigstream.co hadi@end.ucsd.edu

ABSTRACT

Deep Neural Networks (DNNs) are compute-intensive learning models with growing applicability in a wide range of domains, such as vision, robotics, video analytics, speech recognition, natural language processing, targeted advertising, and web search. With diminishing benefits from technology scaling, the research community is increasingly turning to specialized accelerators for DNNs. Even though ASICs provide significant gains in performance and efficiency for DNNs, they may not cope with the ever-evolving DNN models. Furthermore, ASICs and customized cores come at the price of high non-recurring engineering costs over long design periods. FPGAs are an attractive choice for DNNs since they represent an intermediate point between the efficiency of ASICs and the programmability of general purpose processors, and are becoming available across different market segments. However, obtaining both performance and energy efficiency with FPGAs is a laborious task even for expert hardware designers. Furthermore, the large memory footprint of DNNs, coupled with the FPGAs' limited on-chip storage makes DNN acceleration using FPGAs more challenging.

This work tackles these challenges by devising MLWEAVER (Figure 1), a framework that *automatically generates* a synthesizable accelerator for a given (DNN, FPGA) pair from a high-level specification in Tensorflow [1]. To achieve large benefits while preserving automation, MLWEAVER generates accelerators using hand-optimized design templates. First, MLWEAVER translates a given high-level DNN specification to a novel ISA that represents a macro dataflow graph of the DNN. The MLWEAVER compiler is equipped with our optimization algorithm that tiles, schedules, and batches DNN operations to maximize data reuse and best utilize target FPGA's memory and other resources. The final result is a custom synthesizable accelerator that best matches the needs of the DNN while providing high performance and efficiency gains for the target FPGA.

We use MLWEAVER to generate accelerators for eight different deep networks targeted for three different FPGAs, Xilinx Zynq, Altera Stratix V, and Altera Arria 10. We rigorously compare the generated accelerators to multicore CPUs (ARM A15 and Xeon E3) and many-core GPUs (Tegra K1, GTX 650Ti, and Tesla K40). Table 1 reports the results. These

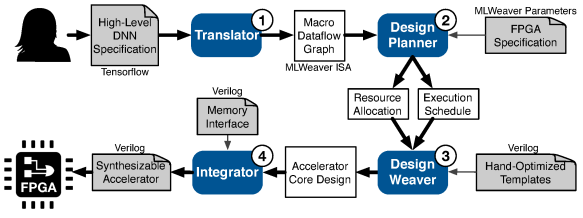

Figure 1: Overview of MLWEAVER which takes a high-level specification of the DNN and the target FPGA to generate the accelerator design as synthesizable Verilog along with the accelerator execution schedule and the layout of the DNN model in the memory.

Table 1: Speedup and Performance-per-Watt comparison of MLWeaver generated accelerators. Each cell represents the benefits of the FPGA in row-heading relative to the platform in column-heading.

FPGA	ARM A15	Xeon E3	Tegra K1	GTX 650Ti	Tesla K40
Speedup Comparison					
Zynq	4.7×	0.59×	0.52×	0.15×	0.03×
Stratix V	22.39×	2.81×	2.43×	0.7×	0.15×
Arria 10	47.26×	5.94×	5.08×	1.48×	0.33×
Performance-per-Watt comparison					
Zynq	11.5×	16.6×	1.7×	3.2×	1.6×
Stratix V	5.5×	7.9×	0.8×	1.5×	0.8×
Arria 10	9.6×	13.9×	4.8×	2.7×	1.3×

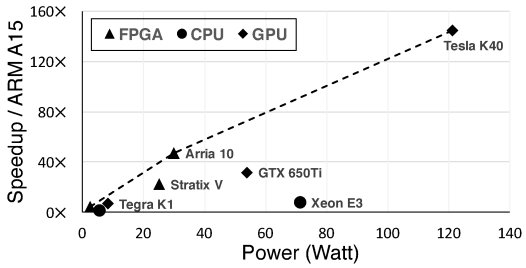

Figure 2: MLWEAVER generated accelerators for Zynq and Arria 10 lie on the Pareto frontier (the dashed line). These results suggest that MLWEAVER makes FPGAs a compelling alternative when the power budget is limited while for high power setting, GPUs are more effective.

results show that MLWEAVER generated accelerators exceed CPUs in performance and in two of three cases (Zynq and Arria 10) deliver higher Performance-per-Watt than GPUs. To achieve these benefits, the programmer only defines the topology and layers of the DNN (< 300 lines of code) without dealing with hardware design or optimization. The relatively low programmer effort is particularly significant since the source code for our templates is over 10,000 lines of code and is optimized hardware by experts over the course of one year. As Figure 2 illustrates, the MLWEAVER generated accelerators for Zynq and Arria 10 lie on the Pareto frontier. The Tesla GPU represents the high-power high-performance Pareto optimal point. The results suggest that when power budget is limited, MLWEAVER enables FPGAs to operate as a platform of choice for deep networks.

To show the effectiveness of our solution, we will perform a demo that the MLWEAVER-generated accelerator executes a DNN-based real-time object detection algorithm (YOLO [2]). We will obtain the input video stream from camera attached to a flying drone. After the conference, we will open-source the software and hardware code for broader community engagement.

1. REFERENCES

[1] Martín Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous distributed systems. *arXiv:1603.04467 [cs]*, 2016.

[2] Joseph Redmon et al. You only look once: Unified, real-time object detection. *arXiv:1506.02640*, 2015.