NullaNet Tiny: Ultra-low-latency DNN Inference Through Fixed-function Combinational Logic

Mahdi Nazemi*, Arash Fayyazi*, Amirhossein Esmaili, Atharva Khare, Soheil Nazar Shahsavani, and Massoud Pedram
School of Electrical and Computer Engineering
University of Southern California, Los Angeles, California 90089-0001
Email: {mnazemi, fayyazi, esmailid, aakhare, nazarsha, pedram}@usc.edu

EXTENDED ABSTRACT

While there is a large body of research on efficient processing of deep neural networks (DNNs) [1]–[31], ultra-low-latency realization of these models for applications with stringent, sub-microsecond latency requirements continues to be an unresolved, challenging problem. Field-programmable gate array (FPGA)-based DNN accelerators are gaining traction as a serious contender to replace graphics processing unit/fixed-point processing unit-based platforms considering their performance, flexibility, and energy efficiency. NullaNet (2018) [32], LUTNet (2019) [33], and LogicNets (2020) [34] are among accelerators specifically designed to benefit from FPGAs’ capabilities.

This paper presents NullaNet Tiny, an across-the-stack design and optimization framework for constructing resource and energy-efficient, ultra-low-latency FPGA-based neural network accelerators. The key idea is to replace expensive operations required to compute various filter/neuron functions in a DNN with Boolean logic expressions that are mapped to the native look-up tables (LUTs) of the FPGA device (examples of such operations are multiply-and-accumulate and batch normalization).

Fig. 1 depicts different steps of NullaNet Tiny’s design and optimization flow. The training module is responsible for both quantization-aware training (QAT) and fanin-constrained pruning (FCP) while the logic minimization module performs two-level and multi-level logic minimization in addition to retiming. These modules’ ultimate goal is to convert each filter/neuron into a highly optimized LUT-based implementation by enumerating all its possible input combinations, recording encountered output values, and optimizing truth tables reflecting those input-output combinations.

QAT refers to the quantization of activations to binary, bipolar, or multi-bit values during neural network training. One of the critical differences between NullaNet Tiny and prior work is that it employs different activation functions for different layers to yield higher accuracy. For example, if the inputs to a DNN assume both negative and positive values, it uses an activation function such as the sign function to capture the range of inputs better. On the other hand, if a set of values can only assume non-negative numbers, it relies on different activation functions for different layers to yield higher accuracy. For example, if the inputs to a DNN assume both negative and positive values, it uses an activation function such as the sign function to capture the range of inputs better. On the other hand, if a set of values can only assume non-negative numbers, it relies on different activation functions for different layers to yield higher accuracy.

Finally, functions of different filters/neurons are represented using truth tables which are then fed to the logic minimization module. This paper employs the ESPRESSO-II logic minimizer [36] for two-level logic minimization and Xilinx Vivado for multi-level logic minimization and retiming.

Our experimental evaluations summarized in Table I demonstrate the superior performance of NullaNet Tiny in terms of accuracy, inference latency, and resource utilization compared to prior art FPGA-based DNN accelerators on the jet substructure classification (JSC) [37] task (the three architectures are based on those described in LogicNets [34]). At about the same level of classification accuracy, compared to Xilinx’s LogicNets [34], our design achieves 2.36× lower latency and 24.42× lower LUT utilization (please note that not all reported clock frequencies are realizable on the target Xilinx VU9P FPGA). Similarly, compared to Google’s optimized design [38], our design has a 9.25× lower latency.

TABLE I

| Arch. | Accuracy (%) | LUTs (Dec.) | FFs (Dec.) | fmax (Inc.) | Remarks |
|-------|--------------|-------------|------------|-------------|---------|
| JSC-S | 69.65% (+1.85) | 39 (5.50×)  | 75 (3.30×) | 2.079 MHz (1.30×) | JSC-L 73.35% (+1.55) | 11,752 (3.20×) | 565 (1.40×) | 436 MHz (1.02×) |
| JSC-M | 72.22% (+1.73) | 1,553 (9.30×) | 151 (2.90×) | 841 MHz (1.40×) | JSC-S 69.65% (+1.85) | 39 (5.50×) | 75 (3.30×) | 2.079 MHz (1.30×) |
| JSC-L | 73.35% (+1.55) | 11,752 (3.20×) | 565 (1.40×) | 436 MHz (1.02×) | JSC-M 72.22% (+1.73) | 1,553 (9.30×) | 151 (2.90×) | 841 MHz (1.40×) |

*M. Nazemi and A. Fayyazi contributed equally to this work.
