XNOR CNNs in FPGA: Real-time Detection and Classification of Traffic Signs in 4K – a demo

Dominika Przewlocka
AGH University of Science and Technology Krakow, Poland
E-mail: dprze@agh.edu.pl

Marcin Kowalczyk
AGH University of Science and Technology Krakow, Poland
E-mail: kowalczyk@agh.edu.pl

Tomasz Kryjak, *Senior Member IEEE*
AGH University of Science and Technology Krakow, Poland
E-mail: tomasz.kryjak@agh.edu.pl

I. Introduction

Traffic sign detection is required in many applications, such as advanced driver assistance and safety systems, autonomous vehicles or traffic signs maintenance. In this demo we present a two stage traffic sign recognition system using Binary Convolutional Neural Networks implemented on the ZCU 104 development board with Zynq UltraScale+ MPSoC (Multi-Processor System on Chip) device from Xilinx. The use of a heterogeneous computing platform allowed to obtain real-time classification and energy efficiency.

II. System overview

Detection: In the first place, the RGB video stream is converted to the HSV colour space. Then red and blue pixels are segmented using fixed thresholds. The binary image is subjected to several median filtrations and morphological closing operations. For this demo it was assumed that only one fixed size traffic sign is present in the image, because we focused on demonstrating the performance of the XNOR CNN. However, a complete version of the system would require connected component labelling and analysis. The obtained bounding box parameters are used in a simplified scaling module, where the input signs are resized to 32×32 resolution, which corresponds to the network's input.

Classification: Candidates for traffic signs are classified by a XNOR-type neural network with architecture is presented in Table I. The network is characterized by binary weights and activations. The input image is represented by real values, which means that the first convolutional layer is not fully binary. After successful classification a proper bounding box is imposed on the output video stream.

III. Hardware implementation of XNOR CNN

Convolutional Block: The convolutional block consists of layers: convolutional, maxpooling, batch normalization and binary activation. The block's input is stored in multiple BRAMs (18Kb), from which each one represents one channel (in case of the first layer) or one feature map (in case of subsequent layers). The filter weights are stored in on-board registers. The input data is sent in a serial-parallel manner, i.e. subsequent pixels from a context are read from memory and aggregated in a one-dimensional register. The filter receives contexts in a specific position (with a specific central pixel) along all input channels. In this way, the weighted sum along the channels is calculated. All filters work in parallel. Data is read from the memory in such a way, as to create in turn a "maxpooling context". After maxpooling, one pixel (one value) is obtained, which then is subjected to successively: batch normalization and binary activation. The results are written in parallel from all filters, to different, appropriate BRAM instances.

TABLE I: Architecture of the proposed neural network

Layer	Input size	Output size	Kernel size
Conv-1	$32 \times 32 \times 3$	$28 \times 28 \times 64$	$5 \times 5 \times 64$
Max-1	$28 \times 28 \times 64$	$14 \times 14 \times 64$	2×2
Conv-2	$14 \times 14 \times 64$	$10 \times 10 \times 128$	$5 \times 5 \times 128$
Max-2	$10 \times 10 \times 128$	$5 \times 5 \times 128$	2×2
FC-1	$5 \times 5 \times 128$	512	$1 \times 1 \times 512$
FC-2	512	43	$1 \times 1 \times 43$

A. Dense Block

Implementation of a fully connected layer assumes the use of a larger BRAM memory due to the significantly higher number of weights. The input pixels are read in a serial way from the features map of the previous layer. Before receiving the data, a set of weights corresponding to the connection for all neurons is loaded from the BRAM memory. The input is XNORed with each weight (i.e. the equivalent of multiplication in networks with full precision is performed) and then the results are sent to the respective accumulators representing the neurons. The weighted sum of all the layer's neurons is calculated in parallel. Then data (vector of all neurons) is serialized, bias is added and values are normalized, if needed. In the case of last layer, the winner neuron is thresholded in order to eliminate uncertain classifications.

IV. Summary

This work is a base for further research on accelerating BCNNs in FPGA and embedding them into real-time vision systems. The key feature of the presented system, in context of design reuse, is the ability to use the developed modules to implement binary convolutional neural network of nearly custom architecture, for any purpose (not only traffic signs' classification).

The work presented in this paper was supported by the National Science Center project no. 2016/23/D/ST6/01389