PointPillars x FPGA

Hardware-software system for 3D object detection in LiDAR point clouds based on deep neural network

Joanna Stanisz & Konrad Lis

Embedded Vision Group, Computer Vision Laboratory,

AGH University of Science and Technology, Krakow, Poland

stanisz@agh.edu.pl, kolis@agh.edu.pl

Abstract: We would like to present a hardware-software system for 3D object detection in LiDAR point clouds based on a deep neural network. The PointPillars network was used in the research, as it is a reasonable compromise between detection accuracy and the calculation complexity. To be able to run the network on an embedded device, we had to optimize the network. The Brevitas / PyTorch tools were used for network quantization and the FINN tool for hardware-software implementation in the reprogrammable Zynq UltraScale+ MPSoC device. The obtained results show that even a significant limitation of the calculation precision results in a relatively small decrease in the detection accuracy and allows the solution to be implemented on an embedded platform. The current version of the system does not run in real-time – one point cloud processing takes c.a. 30s.

The work presented in this poster was supported by the AGH University of Science and Technology project no. 16.16.120.773.

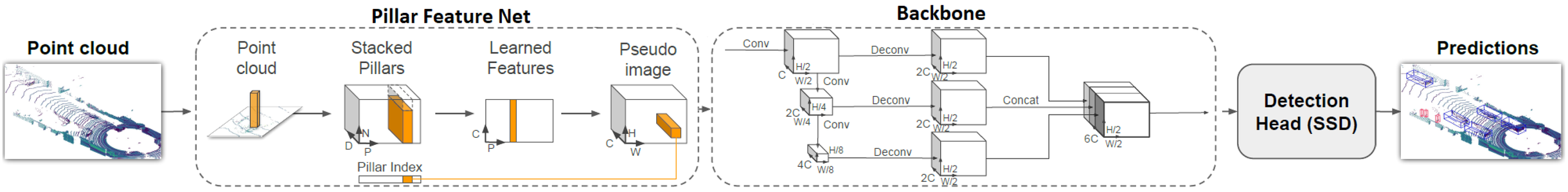

Figure 1: An overview of the PointPillars network structure [1].

The PointPillars network

The input to the PointPillars [1] algorithm is a point cloud from a LiDAR sensor. The results are oriented cuboids denoting the detected objects. A "pillar" is a 3D cell created by dividing the point cloud in the XY plane. The network structure is shown in Figure 1.

The first part, Pillar Feature Net (PFN), converts the point cloud into a sparse "pseudo-image". The number of pillars is limited, as well as the number of points in each pillar. The second part of the network – *Backbone (2D CNN)* – processes the "pseudo-image" and extracts high-level features. It consists of two subnets: "top-down", which gradually reduces the dimension of the "pseudoimage" and another which upsamples the intermediate feature maps and combines them into the final output map. The last part of the network is the *Detection Head (SSD)*, whose task is to detect and regress the 3D cuboids surrounding the objects. The objects are detected on a 2D grid using the Single-Shot Detector (SSD) network. After inference, overlapping objects are merged using the Non-Maximum-Suppression (NMS) algorithm.

The proposed car detection system

Optimisation of the PP network

There are three common methods of optimising a given deep network: reducing the number of layers, quantisation, and pruning (i.e. removing connections with insignificant weights). To check how the PointPillars network optimisation affects the detection precision (AP value) and the network size, we carried out several experiments. As a reference, we used a network with all parameters in the 32-bit floating-point (FP32) representation. We have divided the quantisation experiments into several stages. You can find their descriptions and results in our previous work [2]. Changing the calculation precision from FP32 to INT2 resulted in almost a 16-fold reduction in size at a cost of approx. 11% AP (average precision) loss.

HW/SW implementation

After quantisation, we used the Xilinx's FINN framework for hardware-software implementation [3]. The whole LiDAR data processing system was divided between programmable logic (PL), processing system (PS) and a PC computer (Figure 2).

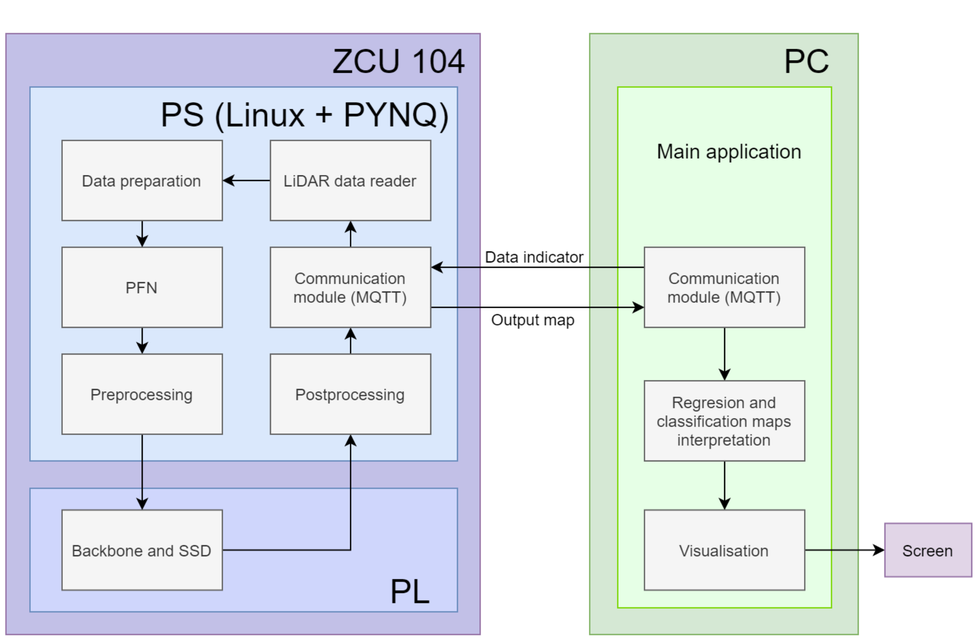

Figure 2: An overview of the proposed HW/SW detection system.

As the FINN tool is still under development and does not support all operations, we needed to simplify PointPillars and divide it as follows: implement the PFN part on the PS and the rest on the PL. In addition, we had to remove the deconvolutional layers and minimise the number of other layers in general. As the target platform, the ZCU 104 development board with Zynq UltraScale+ MPSoC was chosen. The ZCU 104 processing system runs the Linux+PYNQ environment. The PC communicates with the processing system using the MQTT (Message Queuing Telemetry Transport) protocol.

The workflow of the system is described below. Firstly, the PC sends a number of a point cloud to the PS. Then, the PS reads the point cloud from the SD card, voxelises it, and extends the feature vector for each point. For every pillar, its points are processed by the PFN, which outputs a feature vector for the whole pillar. Then, all those vectors are put into the tensor corresponding to the point cloud pillar mesh. Afterwards, the tensor is preprocessed, packed to some known format, and sent to PL using DMA. The PL computes the Backbone + Detection Head network output and sends it back to PS. The PS runs some postprocessing on the received tensor and sends the output maps to the PC. The computer splits the received tensor into classification and regression maps. Then it interprets the data and visualises it on the screen.

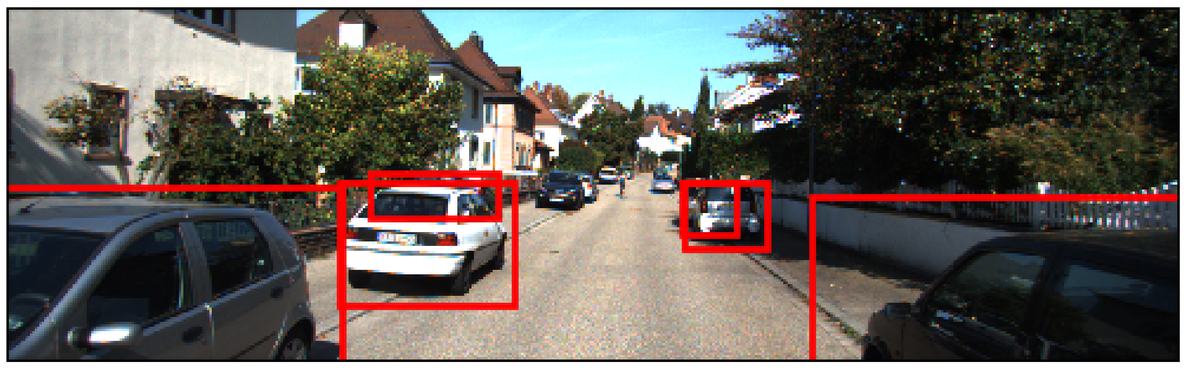

Figure 3: Detected cars marked with bounding boxes projected on image.

Figure 3 presents the camera image corresponding to the considered point cloud with rectangles surrounding the detected cars. Rectangles are 3D LiDAR data bounding boxes detected by the network and projected onto the image. The final, reduced version of PointPillars network has an AP drop of 14% to 19% (depending on the test set) in reference to the original network. However, the network size was simultaneously reduced 55 times. The peak power consumption on the ZCU 104 board is equal to 12.22W. Note that if a PC with a high performance GPU is used, the power consumption would be several dozen times larger. In the current version the average point cloud processing time is about 20 seconds: the PL part takes almost 2 seconds, the PS part c.a. 18 seconds.

Summary & Forthcoming Research

The modifications proposed to the PointPillars network architecture allowed to implement the majority of computations in the PL part of the Zynq UltraScale+ MPSoC device and a 55x size reduction of the model. However, an up to 19% drop of Average Precision, regarding the original network, was measured. The inference time of one sample is currently also far from real time, as a typical LiDAR sends a new point cloud every 0.1 second. However, we are constantly working on further optimisations. For example, the PS part can be rewritten in C++ as Python code is often not time efficient. Moving it to the PL may also be considered. Ultimately, we would like to create a demonstrator cooperating with a LiDAR sensor. Then, we plan to conduct experiments with the network for other object categories, i.e. pedestrians or cyclists, as well as the Waymo and NuScenes sets. Furthermore, we would like to use data fusion for LiDAR, camera, and radar sensors.

References

[1] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12689–12697, June 2019.

[2] J. Stanisz, K. Lis, Y. Kryjak, and M. Gorgon. Optimisation of the pointpillars network for 3d object detection in point clouds, 2020.

[3] Y. Umuroglu, J. Petri-Koenig, A. Rigoni, and H. Borras. Finn repository. https://github.com/Xilinx/finn, Last access 29 June 2020.