Hardware-software Implementation of quantised Siamese Neural Networks for Energy Efficient Real-Time Object Tracking – a demo

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

Daniel Gacek
AGH University of Science and Technology Krakow, Poland
E-mail: danielgacek97@gmail.com

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

***Abstract*—This demo presents a hardware-software object tracking system based on a Siamese neural network. With the use of Brevitas and FINN tools, the selected neural network was quantised and then deployed on the ZCU 104 board, with the Zynq UltraScale+ MPSoC device from Xilinx. The obtained results indicate that appropriate quantisation allows to significantly reduce the size and computational complexity of the model, with a relatively small impact on tracking effectiveness.**

I. Introduction

Object tracking is one of the basic functionalities in modern vision systems. The aim is to determine the trajectory of an object, or objects, moving on the scene, usually after providing the algorithm with information about its features – e.g. by selecting a bounding box in the first frame. Tracking algorithms performance has improved significantly in recent years – mainly due to the use of deep convolutional neural networks. Nevertheless, it is still one of the most complex vision problems, especially if the system needs to run on an embedded platform with a limited energy budget.

II. SIAM-based tracking

A Siamese neural network is a concept of building a model that ”specialises” in comparing two images. This model consists of two branches with identical, in terms of architecture, networks – hence the name Siamese. For tracking, these networks can be fully convolutional (without fully connected layers) [1] so that (1) it is possible to compare images of different sizes; (2) the output from the network is a three-dimensional representation of the input. During tracking, two images are presented to the network: the template and a search window. The template is obtained during the mentioned initialisation (in the simplest variant). The second is a region of interest (ROI) in the current image where we expect to find the object. It is selected based on the previous location of the object. The feature maps obtained for the ROI and the tracked object are finally cross-correlated. As a result, a heat-map is formed, the maximum of which indicates the probable new location of the tracked object. In the proposed solution, it was decided to develop a network based on the AlexNet architecture as in [1].

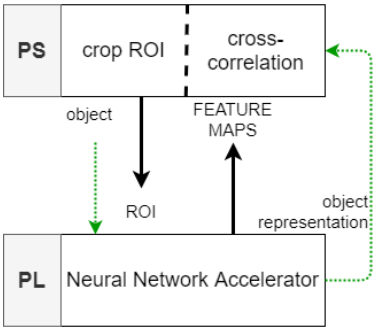

Fig. 1. Hardware-software implementation of tracking using Siamese neural networks. The green arrows indicate the initialisation stage.

III. The proposed object tracking system

An overview of the proposed hardware-software system is presented in Figure. 1. The choice of architecture was determined by the characteristics of the target computing platform – ZCU 104 with the Zynq UltraScale+ MPSoC device from Xilinx – and the capabilities of the FINN quantised neural network implementation tool.

In the current version, the input images are saved on an SD card, ultimately their source should be a camera. At first, the sequences are loaded into the memory of the processor system (PS). Next the appropriate image fragment – the selected object in case of initialisation or the search window – is sent to the programmable logic (PL) via DMA. In the PL part, the forward pass of the network is computed and the feature map for a given image returned to the PS where correlation is performed. Finally a similarity map is computed, and the new object location is determined. In the presented hardware-software system, the PYNQ framework was used. It provides an appropriate abstraction of operations enabling easy read and write access from memory and supports the FINN tool.

A. Optimisation of the SIAM network

Deep neural networks are complex algorithms, both computationally and memory-wise. Their use in embedded devices with limited resources and energy budget is challenging. Apart

TABLE I
TRACKING PERFORMANCE OF THE QUANTISED NETWORK

Exp	FL	HL	LL	AF	Pr	mIoU	mCE
X1_1	FP32	FP32	FP32	FP32	0.491	0.369	95.331
X1_2	INT8	INT16	INT8	INT32	0.474	0.361	98.812
X1_3	INT8	INT8	INT8	INT32	0.486	0.368	96.016
X1_4	INT8	INT4	INT8	INT32	0.482	0.369	98.651
X1_5	INT8	INT2	INT8	INT32	0.491	0.373	96.446
X1_6	INT8	BINARY	INT8	INT32	0.457	0.346	103.604

TABLE II
TRACKING PERFORMANCE OF THE QUANTISED NETWORK ARCHITECTURES SUPPORTED BY FINN

Exp	HL	AF	Pr	mIoU	mCE
X2_1	INT2	INT8	0.485	0.364	100.272
X2_2	INT2	INT4	0.476	0.360	93.953
X2_3	INT2	INT2	0.431	0.323	101.109

from the obvious solution of limiting the size of the model, quantisation is now often used. It assumes changing the used numeric representation from floating point to fixed point with limited precision. For quantisation, weights, activation functions and sometimes input data can be chosen.

Network quantisation can be done in two ways. The first assumes that the model is trained on single or double precision floating point numbers and only then quantised. This usually results in a loss of precision, but can significantly reduce computational and memory complexity. The second option is to train a model with quantised weights. This is a more time-consuming option (selecting the appropriate precision, sometimes individually for each layer), but it allows for better effectiveness. As an additional advantage of quantisation, the overfitting prevention should be mentioned. For the conducted experiments the Xilinx Brevitas tool was used [3].

Table I shows the tracking performance after quantisation of the Siamese network. The model was trained on data from the ILSVRC15 dataset, same as in [1]. The evaluation was performed on data from TempleColor, VOT14, and VOT16 datasets (without the OTB sequence). The subsequent rows of the table show the results of experiments with varying precision in the following layers: input (FL), hidden (HL), last (LL) and activation (AF). The effectiveness of the tracker was measured by precision (PR – location error below a preset pixel threshold), average IoU (mIoU) and average Centre Error (mCE). Experiment X1_1 is the reference (without quantisation). For other networks, the number of bits in the first and last layer remained unchanged and was equal to 8 bits. For the activation layer, the output precision was set to 32 bits. Analysing the results, only a slight decrease in the effectiveness of the algorithm can be noticed, even in the case of firm quantisation of the network weights.

B. Implementation of the SIAM network

The FINN [2] tool was used to implement the Siamese convolutional network in the programmable logic of the Zynq device. This tool, in its current version (October 2020) imposes some limitations on hardware-realisable networks: (1) network weights must be quantised to INT2 or binary, (2) the activation function can be of the following type: INT16, INT8, INT4, INT2. Table II presents the results for selected architectures supported by FINN. In addition, the first layer must be an activation layer – however, when the input is represented by non-negative values, the ReLU activation can be treated as "null", i.e. it does not affect the input data. It is also important to note that there is no need to realise the Siamese network as actually a two-branch network. After initialisation, it is only needed to store the network representation of the object (template), and then, using only one branch, process the ROI.

C. Tracking

The first step is initialisation, i.e. determining the representation of the tracked object. Using the manually selected bounding box, the object is cropped from the first frame, square-padded (if needed), and scaled to 127x127. The object in this form is presented to the network, and then its representation in the form of a feature map is stored in the memory. In subsequent frames, based on the previous location of the object, the ROI is cropped. Its basic size is set to 255x255. Nevertheless, if one wants to take into account a possible change in the size of the object, one needs to analyse several ROIs of different sizes (i.e. different scales). After computing the feature map for the ROI, the cross-correlation is computed. Then, the obtained heat-maps are analysed and the one for which the object indication was the best (the probability was the highest) is selected. Depending on the best scale value, the width and height of the new bounding box is updated.

IV. CONCLUSIONS

In this demonstration, a hardware-software tracking system using a Siamese network, implemented on the ZCU 104 board with the Zynq UltraScale+ MPSoC device was presented. The neural network was quantised, which allowed for the acceleration of calculations and the deployment in programmable logic of the used platform. A slight decrease in efficiency compared to the reference solution, with a simultaneous significant reduction in model size and energy consumption, indicates the validity and usefulness of the applied approach. During the demo we will present the tracking system running on the ZCU 104 board.

ACKNOWLEDGEMENT

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

REFERENCES

[1] Luca Bertinneto et al. *Fully-Convolutional Siamese Networks for Object Tracking*, 2016.

[2] FINN homepage, https://xilinx.github.io/finn/. Last accessed 2nd November 2020

[3] Brevitas homepage, https://xilinx.github.io/brevitas/. Last accessed 2nd November 2020