Demo: On-The-Fly Deployment of Deep Neural Networks on Heterogeneous Hardware in a Low-Cost Smart Camera

D. Velasco-Montero
Inst. Microelectrónica de Sevilla (IMSE-CNM), CSIC-Univ. Sevilla (Spain)
Tel. +34 954466666
delia@imse-cnm.csic.es

J. Fernández-Berni
Inst. Microelectrónica de Sevilla (IMSE-CNM), CSIC-Univ. Sevilla (Spain)
Tel. +34 954466666
berni@imse-cnm.csic.es

R. Carmona-Galán
Inst. Microelectrónica de Sevilla (IMSE-CNM), CSIC-Univ. Sevilla (Spain)
Tel. +34 954466666
rcarmona@imse-cnm.csic.es

Á. Rodríguez-Vázquez
Inst. Microelectrónica de Sevilla (IMSE-CNM), CSIC-Univ. Sevilla (Spain)
Tel. +34 954466666
angel@imse-cnm.csic.es

ABSTRACT
This demo showcases a low-cost smart camera where different hardware configurations can be selected to perform image recognition on deep neural networks. Both the hardware configuration and the network model can be changed any time on the fly. Up to 24 hardware-model combinations are possible, enabling dynamic reconfiguration according to prescribed application requirements.

CCS Concepts
• Computing methodologies → Artificial intelligence → Computer vision → Computer vision tasks → Scene understanding.

Keywords
Smart camera; embedded vision system; visual inference; deep neural networks; heterogeneous hardware.

1. INTRODUCTION
The Deep Learning (DL) paradigm [1] has rapidly become the reference technology within the field of computer vision. It has boosted the number of practical scenarios where visual information can be leveraged to make better decisions and actuations. In addition to achieve much higher accuracy than classical vision algorithms, Deep Neural Networks (DNN) has also brought about a unified approach for many tasks: image recognition, segmentation, optical flow estimation etc. However, taking this paradigm into smart cameras is not straightforward since DNNs, and in particular Convolutional Neural Networks (CNNs) [2], are computationally heavy and memory-hungry. They typically consume a significant percentage of the resources available in embedded systems. This is why both industry and academia are widening the spectrum of DL technological components – hardware, software libraries, development frameworks etc. – aiming at speeding up CNNs in an efficient way. Likewise, new CNN models tailored for embedded devices are reported almost on a daily basis.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.

ICDSC '18, September 3-4, 2018, Eindhoven, Netherlands
@ 2018 Copyright is held by the owner/author(s).
ACM ISBN 978-1-4503-6511-6/18/09.
https://doi.org/10.1145/3243394.3243705

Overall, in this demo we will show the performance of some of these components and models on a low-cost smart camera based on the Raspberry Pi (RPI) 3 model B [3]. We make use of its 4-core CPU and GPU to accelerate 5 different DNNs for 1000-category image recognition, namely Network in Network, GoogLeNet, RestNet-50, SqueezeNet and MobileNet. We have also implemented LeNet for digit recognition. Additionally, we can deploy all these models on a commercial USB stick including an ASIC for inference acceleration [4]. This stick can work in parallel with other units connected to the host computer. In our case, we will show the performance of two of them operating simultaneously.

2. VISITOR EXPERIENCE
The general set-up of the demo is depicted in Fig. 1. Visitors will be able to interact with the RPi-based camera. They will select the particular hardware configuration and DNN model to be executed at any time. This combination can be changed on the fly by simply pressing prescribed keys on a keyboard. In this way, a live performance comparison can be carried out. There will be objects available around belonging to the categories recognized by the models for the visitor to test the system. When presenting one of these objects to a USB camera, the corresponding category identified by the network will be displayed on a screen together with the images being captured. The visitors will also be able to visualize the power consumption of the external accelerators through USB current meters.

3. CONCLUSIONS
A smart camera capable of dynamically adapting its inference performance is experimentally showcased in this demo. We demonstrate that a low-cost vision system can load different network models and change its internal hardware set-up on the fly in order to meet specific requirements at application level in terms of accuracy, power consumption, throughput, memory consumption etc. We also prove that, despite the heavy computational load of CNNs, it is still possible to leverage the latest developments at both software and hardware level to accomplish real-time operation at reasonable energy cost.

ACKNOWLEDGMENTS
This work was supported by Spanish Government MINECO (European Region Development Fund, ERDF/FEDER) through Project TEC2015-66878-C3-1-R, by Junta de Andaluca CEICE through Project TIC 2338-2013 and by EU H2020 MSCA ACHIEVE-ITN, Grant No 765866.

REFERENCES

[1] LeCun, Y., Bengio, Y., Hinton, G.: Deep Learning. *Nature* 521(7553), 436-444 (2015)

[2] Verhelst, M., Moons, B.: Embedded Deep Neural Network Processing. *IEEE Solid- State Circuits Magazine* 9(4), 55-65 (2017).

[3] "Raspberry Pi 3 Model B." 2016 https://www.raspberrypi.org/products/raspberry-pi-3-model-b/.

[4] "Intel Movidius Neural Compute Stick." 2017 https://developer.movidius.com/

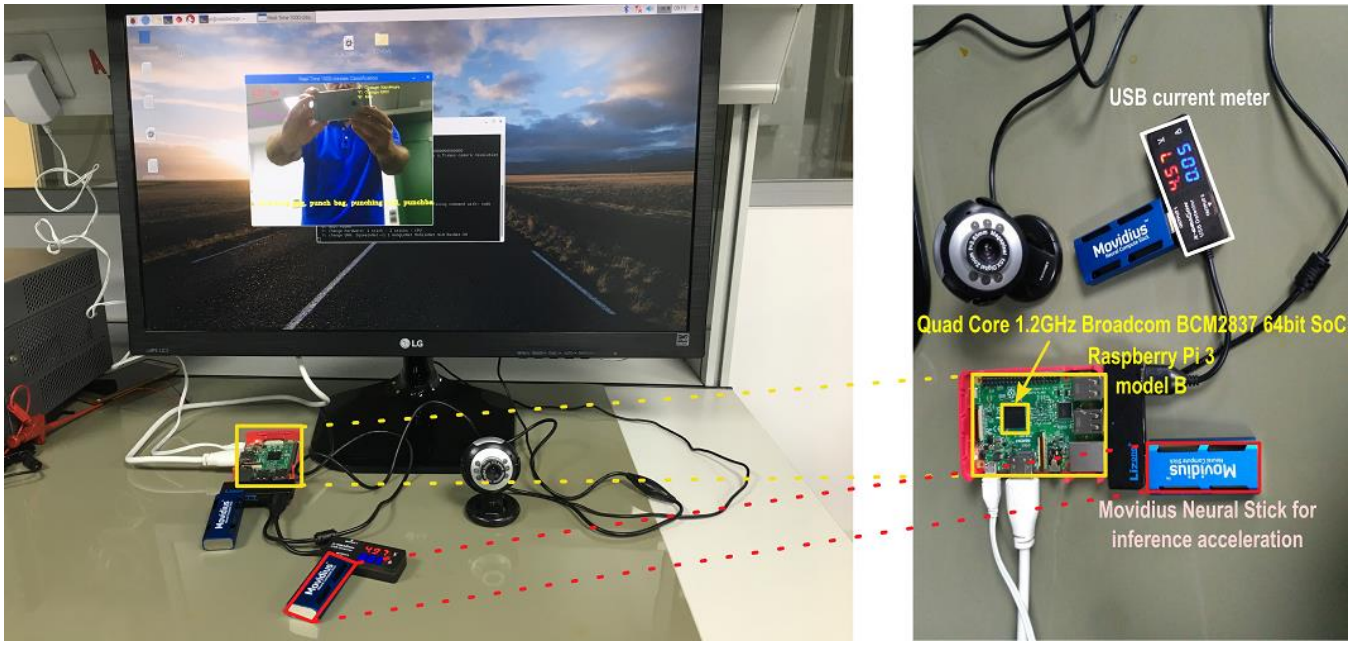

Figure 1. General set-up of the demo (left) and close-up of the hardware components (right). The CPU and GPU are part of the core chip of the Raspberry Pi model 3B [3]. Inference can also be externally accelerated through a commercial ASIC [4] connected via USB. Visitors will be able to observe its power consumption in real time. Several units of this ASIC – two in our case − can be connected simultaneously for parallelization.