Live Demonstration: Digit Recognition On Pixel Processor Arrays

Laurie Bose[*], Jianing Chen[†], Stephen J. Carey[†], Piotr Dudek[†] and Walterio Mayol-Cuevas[*]

Abstract

In this demo, we will showcase recent work on implementing convolutional neural networks directly on pixel processor arrays (PPA). As CNNs demonstrate enhanced performance across tasks from classification to image synthesis, it becomes essential to find the most adequate ways to realize them, especially for embedded, real-time and reactive tasks in areas across Computer Vision and Robotics. The PPA concept is one architecture that pairs sensing and massively parallel processing at the focal plane level and allow mid to high level tasks to be run wholly embedded within them. They allow operation at high framerates and low energy consumption ($\leq$ 2W), and without the need for external signal interpretation or processing. In this demo we will showcase our recent work on the implementation of CNNs on the SCAMP5 architecture [2] as a step towards true end-to-end operation on flexibly programmable PPA hardware. In particular, we will showcase handwritten number classification from image capture to classification wholly embedded on the PPA.

1. Pixel Processor Arrays for CNNs

As vision systems continue to evolve, important challenges arise from the need of low energy operation to fast processing speeds, to crucially, finding the adequate pairing of hardware and algorithms. A pixel processor array consists of an array of sensing *and* processing elements where each element is capable of both light capture and processing. Elements are arranged in a way that they are connected to their neighbours and computation results and data can be shared. Each element contains memory registers which can be digital or analog, and status and flag registers allowing for conditional execution of instructions on a per pixel basis.

When an image is captured, it can be stored on any of its registers and as many image copies either binary or grayscale can be produced and manipulated according to the number and type of register. Importantly, the nature of the PPA allows for massive parallel computation such as with bit-wise instructions e.g. NOR, NOT, addition, substraction and divsion by 2. These instructions can be selectively applied to specific registers as instructed by a controller. Each instruction received by the array is simultaneously executed on every active pixel in a SIMD execution scheme, enabling a range of applications to be performed entirely upon the PPA. A recent example being the estimation of visual odometry entirely on a PPA which receives images as input and generates ego-motion values as output [1].

[*]Faculty of Engineering, Computer Science, University of Bristol, Bristol, England

[†]School of Electrical and Electronic Engineering, The University of Manchester, Manchester, England

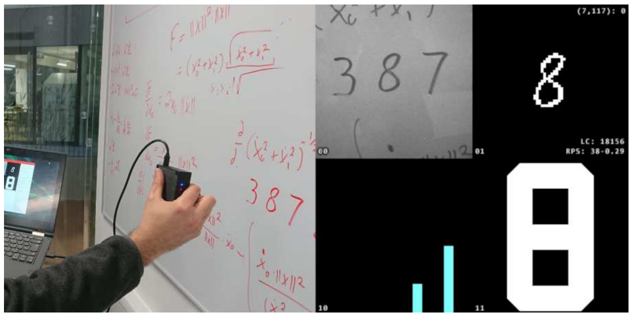

Figure 1. Digit prediction with a CNN from hand-drawn input (left) using inference of a MNIST trained CNN to produce the correct output (bottom-right corner). This demo uses the SCAMP5, a sensor-processor that enables true end-to-end capture *and* processing with a CNN on its massively parallel pixel-processor array.

We here argue that the nature of the PPA is of particular interest to parallel-hungry algorithms such as CNNs which strive for end-to-end operation, a competence also facilitated by the PPA since visual tasks such as classification and detection can be accomplished entirely within it as we will show in this demo. The SCAMP5 architecture is an example of a PPA with 13 binary and 7 analog registers which allow storing and manipulating binary and greyscale images of 256x256 resolution respectively. Our methods for CNN embedding are implemented on the SCAMP5 but they are generic enough to be used in other PPAs.

2. Digit recognition on a PPA

Our approach divides the PPA's digital register memory into 4x4 blocks of processing elements. This sacrifices image resolution for additional storage and bit depth per 4x4 pixel block. We also develop strategies to learn ternary weights (+1,0,-1) for faster computation on the PPA. We

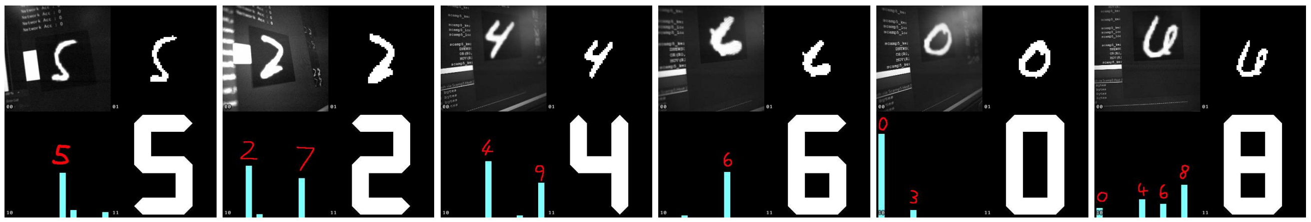
Figure 2. Examples of using the SCAMP5 MNIST trained digit recognition on displayed MNIST digits. Each composite frame contains four panels: top-left is the image captured by the sensor, top-right is the extracted, re-scaled digit image that is fed to the neural network, (bottom-left) is the classification result i.e. activations of ten output neurons, bottom right is the abstract visualization of the classification result. Despite the viewing angle distorting the extracted digits the network still achieves correct classification except in highly questionable digits such as that on the right.

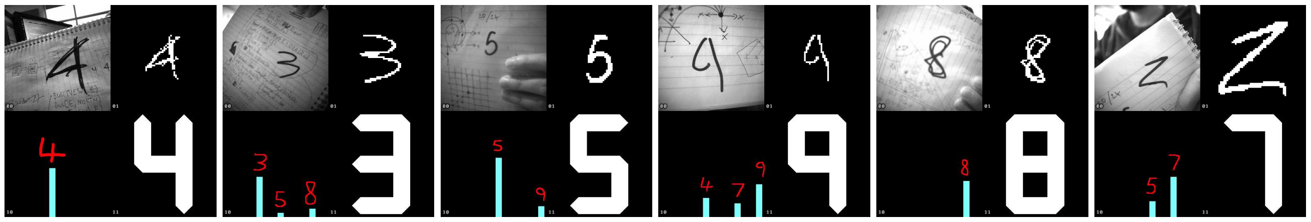
Figure 3. Similar to figure 2, examples of using the SCAMP5 MNIST trained digit recognition **on newly** hand drawn digits. Note a failure case in the right most example where an extremely angular 2 is mis-classified as a 7. The bar plot indicate the activations of the final layer neurons associated with the digits 0-9.

implement parallel methods of performing image addition, subtraction and bit-shifting on block-wise images. Using all these components we formulate how to perform ternary weight convolutions upon images, store the results of multiple such convolutions, perform max-pooling, and readout the resulting images to the PPA's internal micro-controller. We have trained ternary weight CNN networks for different tasks for the SCAMP5 PPA, demonstrating a first step towards embedding neural network processing capability directly onto the focal plane of a sensor.

Our novel scheme for ternary weight CNNs allows for faster and more adequate implementation on the SCAMP5[2], and is extendable to the larger family of PPA systems, where the output of the pixel array consists of sparse data resulting from max-pooled convolution results.

For our demonstration of handwritten digit recognition, a final fully connected layer is performed on the PPA's micro-controller and the system outputs the neuron activations of this final layer as shown in Figure 1.

For inference, we take each captured analog image and convert it into a digital format for the PPA array, lowering the spatial resolution but retaining a high bit-depth. We then perform image convolutions in this format using our ternary weight kernels, whose weights correspond to image addition and subtraction operations. The resulting images are then stored together within a full resolution analog image which then undergoes parallel max-pooling. All the above steps are performed upon the PPA's pixel array, constituting the majority of the computation for inference. Sparse readout of the array is then used to transfer specific max-pooled data to an attached micro-controller upon which a final fully connected layer is conducted.

Our results on networks trained for digit recognition on MNIST achieve comparable results between networks trained in simulation and networks executed on the real PPA. There is a gap however between the two of about 1.2%, attributed to the noise characteristics and difference in testing conditions which is subject to further scrutiny for future work.

We believe that the pairing of PPA devices with CNNs will enable a range of applications for embedding on interactive systems with the ability to learn and truly perform end-to-end tasks.

References

[1] Laurie Bose, Jianing Chen, Stephen J Carey, Piotr Dudek, and Walterio Mayol-Cuevas. Visual odometry for pixel processor arrays. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4604–4612, 2017. 1

[2] Jianing Chen, Stephen J Carey, and Piotr Dudek. Scamp5d vision system and development framework. In *Proceedings of the 12th International Conference on Distributed Smart Cameras*, page 23. ACM, 2018. 1, 2