On-Sensor Binarized Fully Convolutional Neural Network for Localisation and Coarse Segmentation

Yanan Liu[1,2], Yao Lu[2]
[1]School of Microelectronics, Shanghai University, Shanghai, China, yanan.liu@ieee.org
[2]Visual Information Laboratory, University of Bristol, Bristol, UK, yl1220@bristol.ac.uk

Abstract

Current neural networks are compatible with high-performance GPU/CPUs. However, implementing neural networks on emerging embedded sensor for inference is challenging due to sensor's unique hardware architecture and stringent computing resources. With this in mind, this work presents new methods to implement fully convolutional neural networks (FCNs) on Pixel Processor Array (PPA) sensors with many techniques to fully use the limited resources on sensor. Specifically, we, for the first time, design and train binarized FCN for both binary weights and activations using batchnorm, group convolution, and learnable threshold for binarization, producing networks small enough to be embedded on the focal plane of the PPA, with limited local memory resources, and using parallel elementary add/subtract, shifting, and bit operations only. We demonstrate the first implementation of an FCN on a PPA device, performing three convolution layers entirely in the pixel-level processors. We use this architecture to demonstrate inference generating heat maps for object segmentation and localisation at over 280 FPS using the SCAMP-5 PPA vision chip.

1. Introduction

Fully convolutional neural networks (FCN) have been used across many modern computer vision tasks such as object detection [10], classification [14] and segmentation [18, 25]. However, the deployment of deep FCN usually relies on powerful GPU/CPUs which are typically not present in emerging embedded edge devices, where cost and energy considerations dictate stringent limits on storage and computing resources. Despite this, there is an ever increasing demand for artificial intelligence on such edge devices. One promising approach to the edge computing hardware is represented by Pixel Processor Arrays (PPA). Unlike conventional vision systems, which consist of separate sensing and computing hardware, PPA devices are emerging vision architectures, integrating sensing, storage, and computing on a single silicon chip (Fig.1) [4, 13]. Such integration optimises data movements in the system, promising high performance and low-power consumption, but requires careful algorithm implementation, in order to efficiently utilise the hardware resources available in an on-sensor computing device.

Networks with binary weights and activations are difficult to train and usually suffer from performance drop when compared to their floating-point equivalent. Use of batch normalisation has been proposed to avoid gradient explosion and train binarized neural networks successfully [29]. In this work we introduce batch normalisation into binarized FCNs to improve the training efficiency and the performance of inference on PPA arrays. We implement a purely binary convolutional network containing both binarized weights and activations. The use of binary activations alleviates accumulative errors introduced by approximate computations used to perform image convolutions upon PPA hardware devices [11]. This error mitigation allows our approach to perform deeper networks than previous work [2, 3, 20] wholly upon the focal plane without encountering an increasing loss of accuracy that would occur otherwise. Furthermore, it is noted that the implementation of the batch normalisation, the sign activation function, and learnable activation threshold for binarized activations is equivalent to adding a bias matrix to the layer activations, which significantly simplifies the inference process on sensor. With this binarized FCN, the inference calculation process can be implemented entirely with efficient add/subtract, threshold, and shifting operations for all layers. This scheme specifically benefits PPA computing devices such as the one shown in Fig. 1 because it matches the simple instruction set of these devices, and reduces the impact of calculation errors caused by noise accumulation when using analogue registers (AREG) for storage and calculations, especially for activations. For experiments, we train binarized FCN and deploy it on the PPA for object localisation and coarse segmentation.

On-Sensor Computing and the SCAMP-5d Vision Sys-

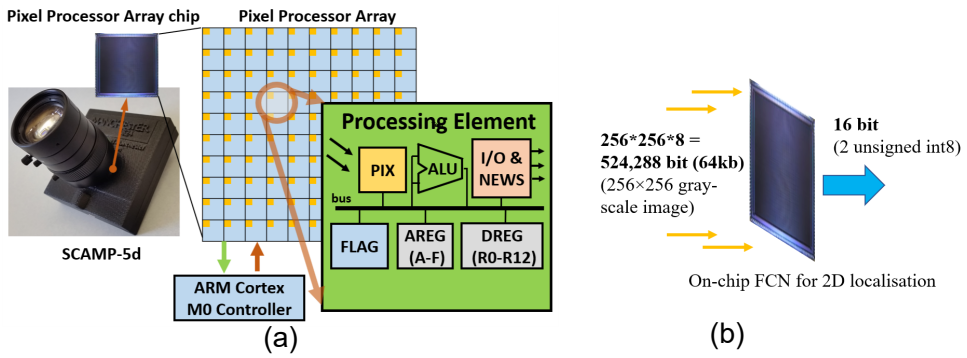

Figure 1. (a) Hardware architecture of the Pixel Processor Array (PPA) used in this work. The SCAMP-5 camera is based on a PPA vision chip with 256×256 Processing Elements (PE), which are simple programmable processor cores where parallel image processing is conducted by directly operating on analogue signals (e.g. electric current from photodetector PIX, which is proportional to the light intensity) within *analogue registers* (AREG) and bit operations within *digital registers* (DREG). (b) The ultra-compressed sensing capabilities with ×32,000 data reduction and a low operation power on PPA using the proposed FCN for 2D localisation.

tem: The concept of on-sensor computing originates from emerging novel circuit designs that enable direct signal processing on the sensing chip [32]. The SCAMP vision sensor [6] used in this work is based on a PPA concept implemented using mixed-signal analog/digital datapath (Fig. 1), other devices such as SONY IMX500[1] or Aistorm Mantis[2] integrate sensing and computing resources in a single device using alternative strategies. Our work takes advantages of the SCAMP analog/digital PPA to efficiently implement a binarized FCN. As shown in Fig. 1, each sensing element converts light into analogue signals that are immediately processed on the focal plane. Unlike the hardware architecture of standard computer vision systems, the PPA does not involve Analogue-Digital-Conversion (ADC) after sensing. Instead it directly operates on analogue electric currents, in physical proximity to the image sensor pixel circuits, accelerating the signal processing and avoiding the bottleneck of ADC and data transmission process to external processor units. However, such analogue processing introduces errors into the computation and data is prone to noise and temporal decay [11]. Fig. 1b shows the data extraction capabilities of the PPA based on the proposed FCN, effectively reducing images of thousands of pixels to specific contextual information within a tiny number of bits.

Contributions: The main contributions of this work are: 1) We propose, train, and demonstrate the use of a purely binarized network (both binary weights and activations) specifically for PPAs. This approach of binary activations addresses the accumulation of analogue computing errors and value saturation after each layer, thus enabling deeper networks while maintaining performance. 2) We present the first implementation of an FCN architecture for PPAs. New methods for group convolutional layers [30] and hundreds of convolutional filter weights storage upon the focal plane of the PPA are proposed. 3) Unlike earlier work, we apply batch normalisation during training and utilise this to learn bias parameters to be easily applied during inference on the PPA device. 4) We provide the first demonstration of object localisation and coarse segmentation tasks on a PPA, with previous works being only concerned with classification tasks.

2. Related Work

The SCAMP vision sensor has been demonstrated in several applications in the field of robotics [16, 17, 21, 27] and computer vision [1, 5, 26]. Recent work for PPAs has concentrated on CNNs and demonstrated on classification tasks [2, 3, 20]. However we found no previous PPA work on FCNs which are important for further tasks like localisation and segmentation. The research on CNN implementation and inference within PPA was pioneered by Bose et al. [2] where a CNN with a single convolutional layer was implemented upon the PPA array and a fully-connected layer upon its controller chip. Their work performs 16-bit image convolution operations using 4×4 DREG 'Super Pixel' blocks and demonstrates live digit classification based on MNIST dataset at speed of around 200 frames per second (FPS). To fully take advantage of PPA's parallel computing characteristics and further improve the CNN inference efficiency, Bose et al. [3] proposed the idea of in-pixel weight storage, where the network's weights are directly stored within the registers of the PPA's processing elements. This enabled both parallel computation of multiple convolutions, and implementation of a fully connected layer upon the PPA array resulting in a ×22 faster CNN inference (4464 FPS) on the same digit recognition task. Based on these two works, Liu et al. [20] further proposed a high-speed lightweight neural network using BinaryConnect [8] with a new method for computing convolutions upon the PPA, allowing for varying convolutional stride. Their work

[1] https://developer.sony.com/develop/imx500/
[2] https://aistorm.ai/mantis-2/

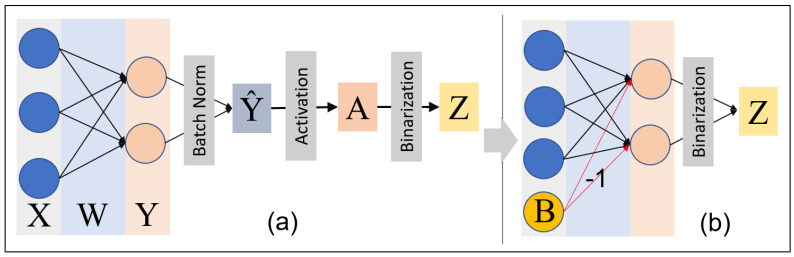

Figure 2. (a) The binarized CNN forward propagation with batch norm and learnable activation function. (b) The simplified inference process on the PPA device by transforming batch norm and activation function into a 'bias' $\boldsymbol{B}$ to be subtracted from $\boldsymbol{Y}$. The inference process is significantly simplified, with only addition/subtraction and sign operations required.

demonstrated four different classification tasks with frame rates ranging from 2,000 to 17,500 FPS with different stride setups. Based on their network, a direct servo control using real-time CNN inference results [23] and a simulated robot tracking from a drone [22] with on-sensor CNN computing results are presented. Moreover, AnalogNet [31] implements convolution with value approximation for multiplication on the focal plane using 3 kernels and fully-connected layer on the micro-controller based on the MNIST. Different from above-mentioned works, this paper advances state-of-the-art in neural network methods on PPA. We propose a new FCN network architecture suitable for PPA implementation, including three convolutional layers with binary weights and binary activations. The network firstly utilises group convolution and is trained using batch normalisation, and enables new segmentation-related applications.

3. Method

Neural network architectures for PPAs must be carefully designed taking into account model size, architecture, and the feasibility of exploiting the PPA's parallel computation and on-sensor storage. This is essential due to the limited on-sensor resources compared to standard computer hardware which may have access to powerful GPU/CPUs. This section attempts to find a balance between the network performance and its implementation on the PPA.

3.1. CNN with Binary Weights and Activations

The neural network training in our work is based on Binarized CNN [9], with binary weights and neuron activations, which can be stored and processed with bit-wise operations. Compared to BinaryConnect, Binarized CNN reduces the intermediate memory storage for activations and replaces most arithmetic operations with bit-wise operations. Such fully binarized networks are thus highly suitable for PPAs due to their small memory footprint. For the forward propagation, we take different strategies to binarise the weights and activations to simplify the binarization process. All the weights are binarized with a deterministic function Eq.1

$$w_b = sign(w_r) = \begin{cases} +1 & w_r > 0, \\ -1 & otherwise \end{cases} \tag{1}$$

$$a_b = sign(a_r - \alpha) = \begin{cases} +1 & a_r > \alpha, \\ -1 & otherwise \end{cases} \tag{2}$$

where w_r is floating-point weights and w_b is the binarized weights. In terms of activations, we train channel-wise learnable thresholds α to binarise the activations to obtain more informative binary feature maps, inspired by work [24]. Additional coefficients, introduced by channel-wise thresholds, have low impact on the implementation efficiency on the PPA. In Eq.2, α is the trainable thresholds for binarization of each channel, a_r is the real-valued activations and a_b is the binarized activations. During the training process, using standard backpropagation and stochastic gradient descent, the gradients are calculated with the floating-point weights. The weights and activations are only binarized during forward pass. In our work, the Binarized CNN is trained on a PC machine and the CNN inference process is implemented on the SCAMP-5d vision system.

The training process for batch norm parameters can be seen from [19] Algorithm 1, in which ϵ is used to avoid a zero denominator and the main scaling and shifting parameters γ and β for batch norm are learned during the training process. Then the batch norm can be applied to manipulate activations [19]. In Fig. 2, for a single layer of Binarized CNN during forward propagation process:

$$\begin{aligned} Y &= \sum_{i=0}^{m} w_i x_i, \hat{Y} = \gamma \frac{Y - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta \\ &= \frac{\gamma}{\sqrt{\sigma^2 + \epsilon}} (Y - (\mu - \frac{\sqrt{\sigma^2 + \epsilon}}{\gamma} \beta)) \end{aligned} \tag{3}$$

Considering activation function $tanh$ does not change the sign of inputs, we have:

$$\begin{aligned} Z = sign(A) &= sign(tanh(\hat{Y} - \alpha)) = sign(\hat{Y} - \alpha) \\ &= sign(Y - (\mu - \frac{\sqrt{\sigma^2 + \epsilon}}{\gamma} \beta) - \alpha) \end{aligned} \tag{4}$$

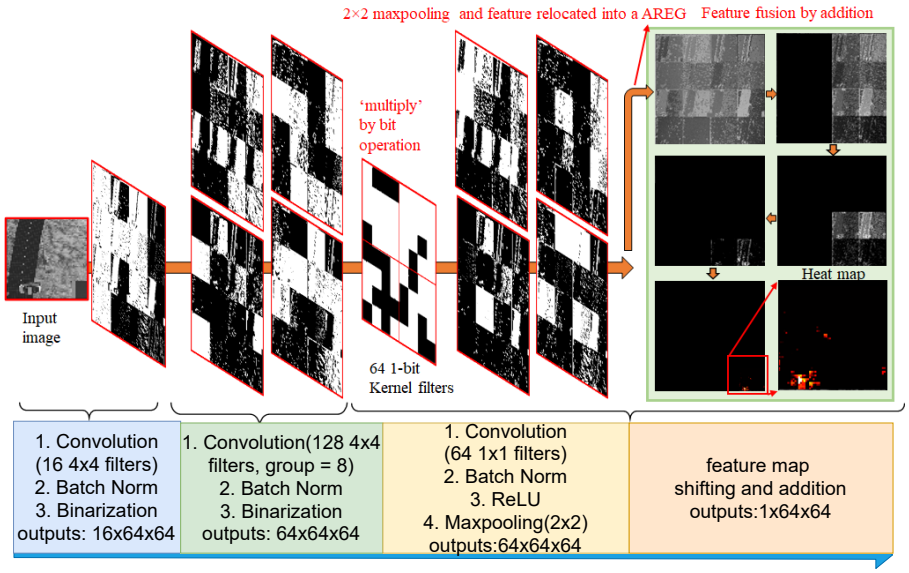

Figure 3. An overview of an on-sensor FCN architecture and inference process using a PPA for heat map generation. A three-layer FCN architecture is used in our work. The first convolutional layer can be seen from Fig. 6 in detail. In the second convolutional layer, 128 convolutional kernel filters are applied upon the 16 input binary feature maps from the first layer, generating 64 feature maps with a convolution group setup of eight. The fusion of intermediate extracted features is implemented by addition within each group. The third layer uses binary filters with a size of $64 \times 1 \times 1$, hence the final feature maps can be obtained by 'multiplication' with bit operation based on DREG. The final heat map is generated by combining these input 64 feature maps by shifting and addition operations.

$$Z = sign(Y - B) \quad (5) \qquad B = \mu + \alpha - \frac{\sqrt{\sigma^2 + \epsilon}}{\gamma}\beta \quad (6)$$

In Equation 6, $\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2, \mu = \frac{1}{n}\sum_{i=1}^{n} x_i$, where β, γ, and α are all trainable parameters that can be obtained directly after training. Thus the 'bias' $\boldsymbol{B}$ can be calculated used these parameters offline, before implementing it on the PPA. During the inference process, the batch norm and activation reduces to a bias term, as shown in Equation 5 $\boldsymbol{B}$ on $\boldsymbol{Y}$. Hence, the inference process on the PPA can be simplified as shown in Fig. 2b.

3.2. FCN architecture on sensor

FCN is a CNN architecture providing pixel-level classification, targeting image segmentation [25]. This paper firstly proposes a 3-conv layer FCN that can be implemented on a PPA sensor. In this paper, FCN is used for heat map generation by adding one convolutional layer with 128 filters and replacing the final fully-connected layer with a convolutional layer of kernel size 1×1. Fig. 3 shows the overall FCN architecture with configurations for each layer. Fig. 6 illustrates the first convolution layer, generating 16 binary feature maps. The second layer adopts group image convolution [7] of 8 on the input 16 feature maps to make a trade off between convolution computation complexity and network performance, where each of 64 outputs is generated by adding two intermediate feature maps (Fig. 9). These 64 binary feature maps from the second convolutional layer are stored in 4 DREG. The third layer then generates the final heat map representing the prediction probability distribution, taking these 64 binary feature maps and combining them within an AREG. Each binary feature map being multiplied by an associated weight of -1/1.

3.3. FCN Implementation on the PPA

This section gives the implementation detail of the binarized FCN on the PPA sensor hardware.

First Layer: 16 binary filters are replicated to fill a DREG (Fig. 7) for parallel convolution purpose [3, 20]. Fig. 5 shows the image convolution process on the PPA. The image convolution on the PPA can be decomposed as 'multiplications', shifting and addition and the convolution result is obtained by performing shifting and addition process for 16 times with a stride = 1. Then the pre-calculated bias B is plotted into 4×4 grids on a AREG and is subtracted from the feature map (seen in Fig. 6). Then the output binary image is obtained by binarizing feature map after subtracting B. In this layer, $tanh$ is used as the activation function. When implementing inference on sensor, the $tanh$ activation function is transformed into binarisation with a sign function, with offset computed from batch norm parameters as can be seen from Eq. 4. This layer shares some similarity with work [20] including the image resize, replication, and image convolution but with extra activation binarization process.

Second Layer: Group convolution is an advantageous approach for embedded devices due to the reduced number of parameters generated by group computing. It is, however,

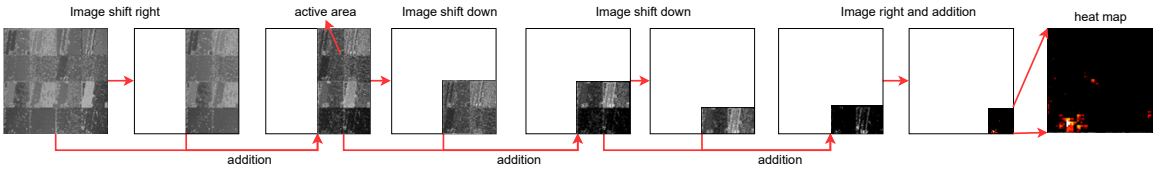

Figure 4. Feature map shifting and addition process: the final heat map is generated by adding the feature maps from the third layer.

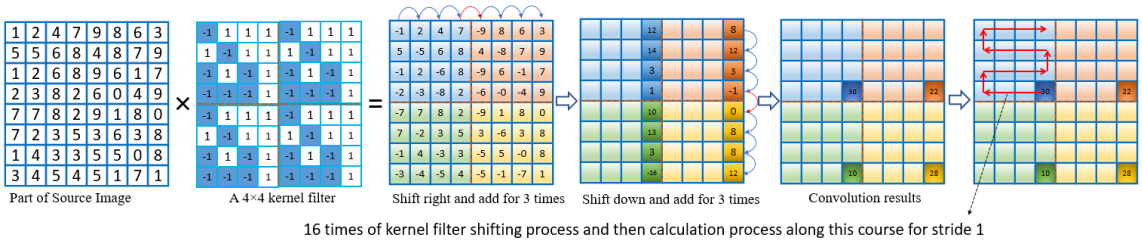

Figure 5. Image convolution on PPA with sign inversion, bit-shifting, and addition.

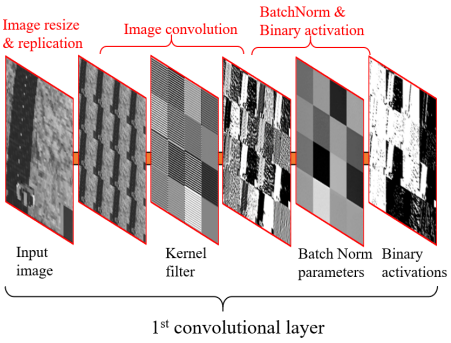

Figure 6. First convolutional layer of the FCN.

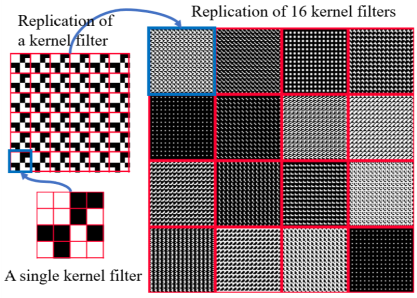

Figure 7. The layout of 16 binarized convolutional kernels in a DREG for the 1st layer.

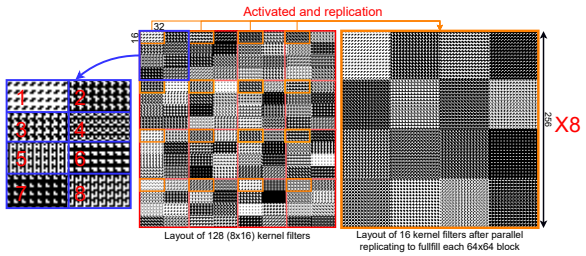

Figure 8. Left: The layout of 8 filers in a block of size 64×64. Each 4×4 filter is replicated and stored within a 16×32 PE block. Middle: The layout of 128 kernel filters stored in one DREG. Before performing a convolution operation (8 times in total) in the 2nd layer, each set of kernels (yellow blocks are for one set, for instance) are replicated to fully fill each 64×64 block (right), covering the whole 256×256 PE array with 16 replicated filters.

difficult to implement in parallel on the unique hardware architecture of the PPA. The key to the second layer is to implement the group convolution with 16 feature maps as inputs and 64 as outputs. By dividing them into eight groups, thus there are 128 binary filters need to be stored on sensor. The layout of filters directly affects the inference efficiency. We design a storage structure for filters in first (Fig. 7) and second layer. Fig. 8 illustrates the layout of these filters within one DREG. As can be seen, each time to perform a convolution, the corresponding kernel filters are activated in parallel, shifted, and replicated to fill each 64×64 block in 256×256 PEs. This filter storage structure can also extend to store more filters following the similar way to fill all PEs with 16 filters. In Fig. 9, to implement second convolution layer with 8 groups, the input 16 binary feature maps are first transformed by switching position of adjacent maps. This is followed by convolution with associated 32 filters for these 32 feature maps. Then 16 gray-scale feature maps are obtained by adding each two of 32 maps. By performing convolution for another 96 filters, 64 gray-scale feature maps can be derived. The bias matrix is subtracted and then after binarization, 64 binary feature maps are generated.

Third Layer: In this layer, as shown in Fig. 3, 64 1-bit filters are plotted on a DREG, followed by 'multiplication' with the 1-bit feature maps from the previous layer. After 1×1 convolution, these 64 feature maps in 4 DREG is relocated to 1 AREG after 2×2 maxpooling. The summation of these extracted feature can be obtained by shifting and adding into one 64×64 heat map (shown in Fig. 4). Unlike in the previous two layers, the activation function for this layer is ReLU to generate a gray-scale feature map as the

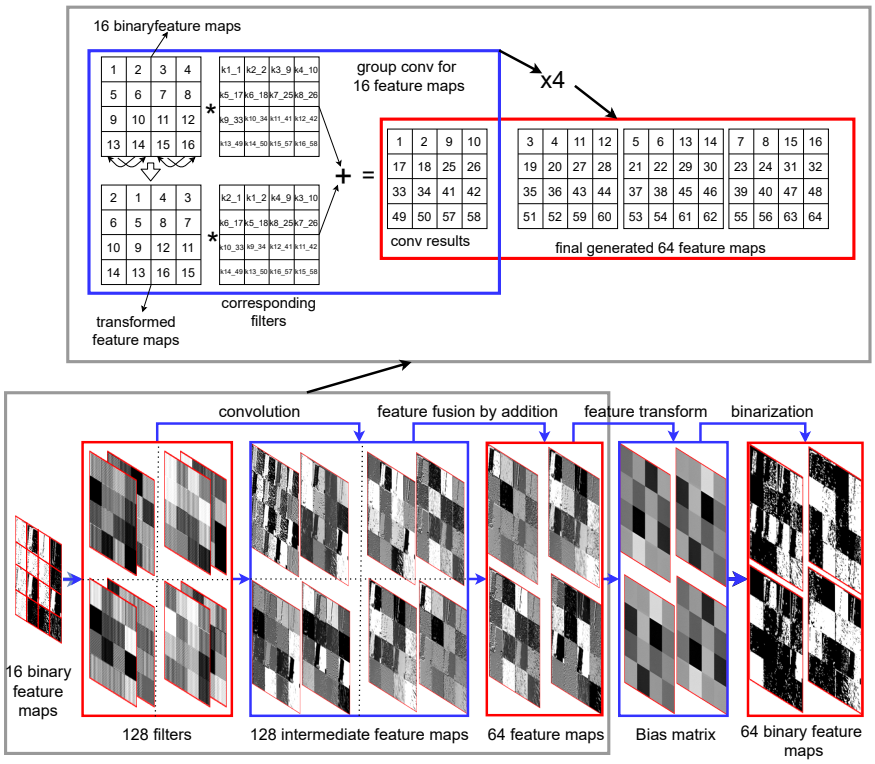

Figure 9. Group convolution on the PPA for the 2nd layer. Top: schematic diagram of group convolution. Numbers within grids represent the index of feature maps and filter layout. Bottom: group convolution on sensor. 128 convolutional kernel filters are applied on 16 input binary feature maps (far left), generating 64 feature maps by adding each two of 128 intermediate maps. Then these 64 maps are subtracted by bias matrix, followed by a binarization. Finally, 64 binary feature maps are obtained for the next layer. More details can be seen from the supplementary pdf.

final prediction result of the network.

4. SCAMP-5 Inference, Experiments, and Evaluation

This section demonstrates the application of the proposed network architecture to coarse segmentation and object 2D localisation from a bird's eye view. We implement the FCN algorithm on the SCAMP vision system hardware (Fig. 10). We set up a realistic environment in Webots [28] robot simulator (Fig. 11) for data collection and the validation of FCN deployment on sensor. Training, testing and validation datasets are collected by repeatedly taking images from a flying drone equipped with a simulated "SCAMP" and then validation images are sent to the PPA hardware for inference. Binarized FCN is trained offline based on these datasets with the method proposed in Section 3. The whole neural network for both coarse segmentation and localisation is performed on sensor.

4.1. Coarse Segmentation

Fig. 11 shows the samples of collected datasets and their annotations for segmentation of road and grass. To validate the performance of the proposed network on different tasks,

Task	IoU
Road segmentation on simulation (Computer)	74.0%
Road segmentation on sensor (PPA)	69.3%
Grass segmentation on simulation (Computer)	76.6%
Grass segmentation on sensor (PPA)	72.9%

Table 1. Intersection over Union (IoU) performance comparison between simulation on computer and execution on-sensor for coarse segmentation.

a road and grass coarse segmentation is explored in this section. As shown in Fig. 11, we directly use the road/grass shape as the ground truth for coarse segmentation. Notice that the trees and grass areas often share similar gray-scale levels with the road, making coarse segmentation unfeasible by simply using binary thresholding. Tab. 1 shows the Intersection over Union (IoU) performance comparison between FCN inference on simulation and on sensor. Specifically, IoU is measured here by counting the number of intersected pixels over the number of united pixels of the predictions and groundtruth. In addition, Fig. 13 compares the FCN training process for segmentation task between using and

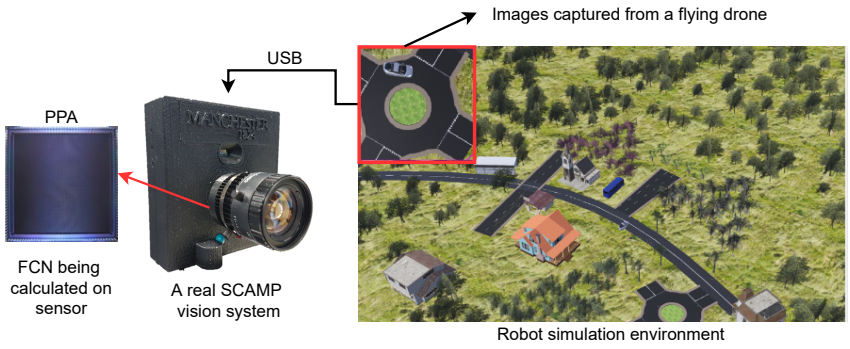

Figure 10. Experimental setups using a robot simulator and a real SCAMP vision system where the neural networks are fully computed on the PPA.

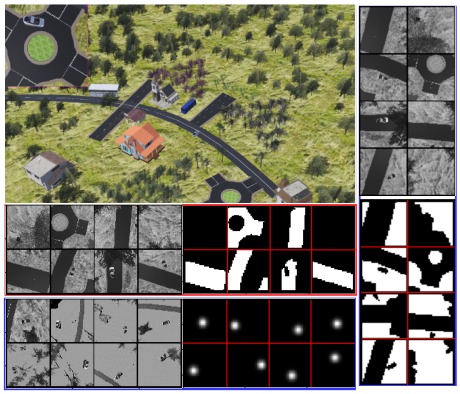

Figure 11. The data collection environment where a vehicle is moving around and images are generated from a bird's eye view of a drone. Top left: A robot simulator for environment setup and collection of training data. Middle left: The collected images from the drone's camera are converted into gray-scale images for the PPA and the segmentation annotations of the road. Right: The training and annotated datasets for grass segmentation. Bottom left: this work uses the Gaussian distribution to represent the vehicle position within an image.

not using batch norm, which shows the binarized FCN dose not converge without batch norm, justifying its use here. Some of the results can be seen from Fig. 12. Tab. 1 compares the experimental results on sensor and its counterpart baseline on computer with identical neural networks and validation images.

4.2. Object Detection

We also implemented an object detection task, based on the heat map. As for the object localisation, rather than using the vehicle segmentation image as the ground truth for training, we use Gaussian position distribution (Fig. 11) as the ground truth since the probability distribution is adequate to represent the object 2D localisation. For the validation, a distance threshold is set from 0 to 63 to count the number of predictions with a distance to the groundtruth that falls into this threshold. A zero distance means a perfect prediction. The final localisation is obtained by the

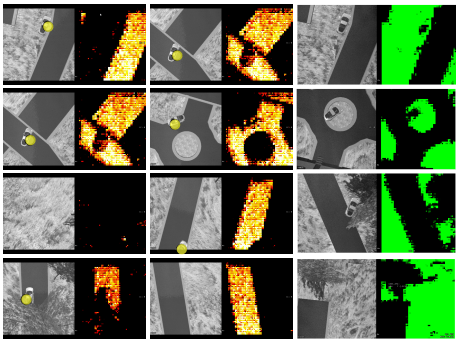

Figure 12. FCN inference results on-sensor. Left column is the input gray-scale image on sensor with yellow dots indicating the FCN inference localisation prediction and right column is the inference results for coarse segmentation on sensor. The density and distribution of colourful points (right) represent the possibility of position of the road (bright yellow) and grass (green) segmentation. The experimental performance for localisation (accuracy) and segmentation (IoU) on the PPA can be seen from Fig. 14. and Tab. 1. Experimental video: https://vimeo.com/636976542

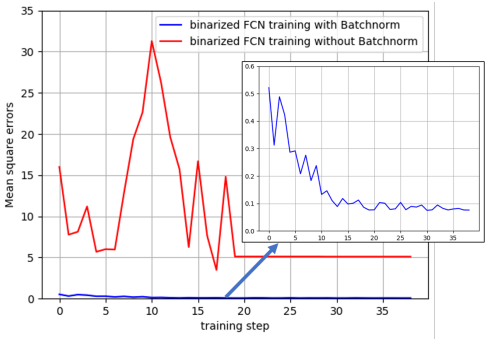

Figure 13. Training process comparison for grass segmentation with and without batch norm.

weighted sum of all the possible positions. After the test, within a distance of 10 pixels, the vehicle localisation accuracy for simulation and SCAMP is around 88% and 83% respectively (Fig. 14). Tab. 2 shows the FCN performance in terms of time, power consumption and model size.

Processing Steps	Time Cost (μs)
Image replication	112
1st Image convolution	184
1st Batch norm and activation	235
kernel filter activation and replication	212
2nd Group convolution	184×4 =736
2×2 maxpooling	35×4 = 140
2nd Batch norm and activation	235×4 = 940
Third convolutional layer	966
Total time cost	3525 (283 FPS)
number of weights	2,578
power consumption	≈ 1.5W
model size	≈ 0.31 KB

Table 2. Computation time, performance and weights for heat map generation with the binarized FCN on sensor.

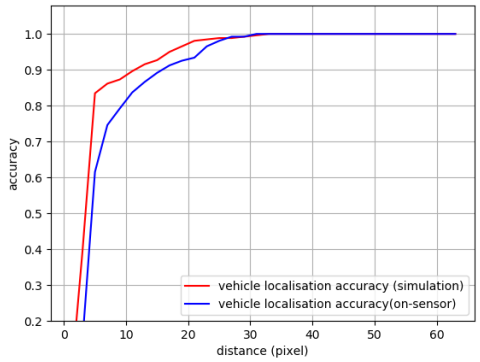

Figure 14. Performance comparison between simulation and on sensor for localisation task.

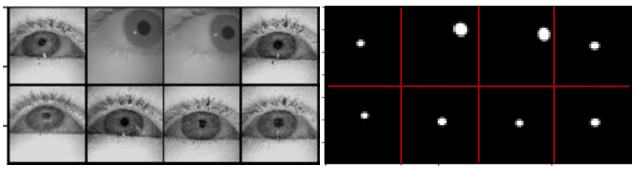

Figure 15. Selection of pupil images (Left) and their annotations (Right) from TEyeD datasets.

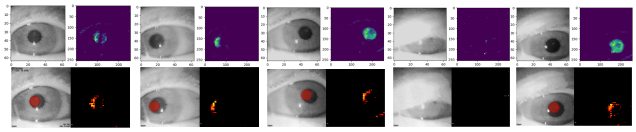

Figure 16. Selection of inference results comparison between FCN with off-sensor simulation (top) and on-sensor PPA (bottom). The red dot at the bottom left of each frame is the final position prediction calculated from the heat map on the right. Experimental video: https://vimeo.com/636978456

4.3. Pupil detection

Considering the light weight and low-power consumption of the PPA chip and the increasing popularity of virtual and augmented reality (VR/AR) based on the eye movement, it is promising to mount the PPA chip to a wearable device such as glasses in the near future, hence we explore the pupil detection with the proposed binarized FCN based on the public dataset of eye images: TEyeD [15]. Fig. 15 shows some of the training images and their annotations and Fig. 16 shows results comparison between SCAMP and simulation inference, where the accuracy curve is plotted according to the Euclidean distance between simulation/scamp inference results and the groundtruth. Within a distance of 10 pixels, the localisation accuracy for simulation and scamp is around 88% and 83% respectively. More experimental videos can be shown under the requirement of reviewers in an anonymous manner.

Notice that there is around 5% - 6% performance gap for the experiment on sensor compared to the simulation. This is due to noise in the convolution operation performed on AREG because of the inherent non-idealities of analogue computation [12] and some random bit-flipping errors observed in DREG when performing massively parallel shifting and replications. Mitigation of these issues requires further software or hardware solutions. In this work, we tried to find a balance between network complexity and viability for deployment upon the available PPA prototype hardware. Pixel-wise accurate segmentation, with a quality equal to one that can be obtained using a CPU/GPUs hardware, using embedded low-power SCAMP-5d vision system, is still a challenging task with current hardware and neural network architecture.

5. Conclusion and Future Work

On-sensor computing is important for embedded and low-lag, low-power vision systems. Due to their compactness and computational advantages, binary FCNs are increasingly appealing. In this paper, we propose and implement a new method that demonstrates carrying out an inference with a binarized FCN on an on-sensor computing device. In contrast to previous works that have mainly focused on classification with a fully-connected layer, we, for the first time, exploit the FCN architecture design, implementation method, and inference on sensor. We validate, using a real pixel processor array (PPA) hardware, on the visual competences of region segmentation and target object localisation with a latency of 3.5 milliseconds for each inference. With the development of future generation of on-sensor devices in terms of image resolution, manufacturing techniques, and local memory capacity, we believe our proposed binarized FCN can be extended with extra layers for more challenging applications.

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. 2

[2] Laurie Bose, Jianing Chen, Stephen J. Carey, Piotr Dudek, and Walterio Mayol-Cuevas. A camera that cnns: Towards embedded neural networks on pixel processor arrays. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 1, 2

[3] Laurie Bose, Piotr Dudek, Jianing Chen, Stephen J Carey, and Walterio W Mayol-Cuevas. Fully embedding fast convolutional networks on pixel processor arrays. In *European Conference on Computer Vision*, pages 488–503. Springer, 2020. 1, 2, 4

[4] Stephen J Carey, Alexey Lopich, David RW Barr, Bin Wang, and Piotr Dudek. A 100,000 fps vision sensor with embedded 535gops/w 256× 256 simd processor array. In *2013 Symposium on VLSI Circuits*, pages C182–C183. IEEE, 2013. 1

[5] Jianing Chen, Stephen J Carey, and Piotr Dudek. Feature extraction using a portable vision system. In *IEEE/RSJ Int. Conf. Intell. Robots Syst., Workshop Vis.-based Agile Auton. Navigation UAVs*, 2017. 2

[6] 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*, pages 1–2, 2018. 2

[7] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017. 4

[8] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pages 3123–3131, 2015. 2

[9] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016. 3

[10] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. *arXiv preprint arXiv:1605.06409*, 2016. 1

[11] Piotr Dudek. Accuracy and efficiency of grey-level image filtering on vlsi cellular processor arrays. In *Proc. CNNA*, pages 123–128, 2004. 1, 2

[12] Piotr Dudek and Peter J Hicks. An simd array of analogue microprocessors for early vision. In *Proc. Conf. Postgraduate Research in Electronics, Photonics and Related Fields (PREP'99)*, pages 359–362, 1999. 8

[13] Piotr Dudek and Peter J Hicks. A general-purpose processor-per-pixel analog simd vision chip. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 52(1):13–20, 2005. 1

[14] Gang Fu, Changjun Liu, Rong Zhou, Tao Sun, and Qijian Zhang. Classification for high resolution remote sensing imagery using a fully convolutional network. *Remote Sensing*, 9(5):498, 2017. 1

[15] Wolfgang Fuhl, Gjergji Kasneci, and Enkelejda Kasneci. Teyed: Over 20 million real-world eye images with pupil, eyelid, and iris 2d and 3d segmentations, 2d and 3d landmarks, 3d eyeball, gaze vector, and eye movement types. *arXiv preprint arXiv:2102.02115*, 2021. 8

[16] Colin Greatwood, Laurie Bose, Thomas Richardson, Walterio Mayol-Cuevas, Jianing Chen, Stephen J Carey, and Piotr Dudek. Tracking control of a uav with a parallel visual processor. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4248–4254. IEEE, 2017. 2

[17] Colin Greatwood, Laurie Bose, Thomas Richardson, Walterio Mayol-Cuevas, Jianing Chen, Stephen J Carey, and Piotr Dudek. Perspective correcting visual odometry for agile mavs using a pixel processor array. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 987–994. IEEE, 2018. 2

[18] Vladimir Iglovikov, Selim Seferbekov, Alexander Buslaev, and Alexey Shvets. Ternausnetv2: Fully convolutional network for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 233–237, 2018. 1

[19] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 3

[20] Yanan Liu, Laurie Bose, Jianing Chen, Stephen J Carey, Piotr Dudek, and Walterio Mayol-Cuevas. High-speed light-weight cnn inference via strided convolutions on a pixel processor array. In *The 31st British Machine Vision Conference (BMVC 2020)*, 2020. 1, 2, 4

[21] Yanan Liu, Laurie Bose, Colin Greatwood, Jianing Chen, Rui Fan, Thomas Richardson, Stephen J Carey, Piotr Dudek, and Walterio Mayol-Cuevas. Agile reactive navigation for a non-holonomic mobile robot using a pixel processor array. *IET image processing*, 2021. 2

[22] Yanan Liu, Jianing Chen, Laurie Bose, Piotr Dudek, and Walterio Mayol-Cuevas. Bringing a robot simulator to the scamp vision system. In *2021 IEEE International Conference on Robotics and Automation (ICRA) workshop: On and Near-sensor Vision Processing, from Photons to Applications*. IEEE, 2021. 3

[23] Yanan Liu, Jianing Chen, Laurie Bose, Piotr Dudek, and Walterio Mayol-Cuevas. Direct servo control from in-sensor cnn inference with a pixel processor array. In *2021 IEEE International Conference on Robotics and Automation (ICRA) workshop: On and Near-sensor Vision Processing, from Photons to Applications*. IEEE, 2021. 3

[24] Zechun Liu, Zhiqiang Shen, Marios Savvides, and Kwang-Ting Cheng. Reactnet: Towards precise binary neural network with generalized activation functions. In *European Conference on Computer Vision*, pages 143–159. Springer, 2020. 3

[25] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 1, 4

[26] Julien NP Martel, Lorenz Mueller, Stephen J Carey, Piotr Dudek, and Gordon Wetzstein. Neural sensors: Learning pixel exposures for hdr imaging and video compressive sensing with programmable sensors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 2

[27] Alexander McConville, Laurie Bose, Robert Clarke, Walterio Mayol-Cuevas, Jianing Chen, Colin Greatwood, Stephen Carey, Piotr Dudek, and Tom Richardson. Visual odometry using pixel processor arrays for unmanned aerial systems in gps denied environments. *Frontiers in Robotics and AI*, 7, 2020. 2

[28] Olivier Michel. Cyberbotics ltd. webots™: professional mobile robot simulation. *International Journal of Advanced Robotic Systems*, 1(1):5, 2004. 6

[29] Eyyüb Sari, Mouloud Belbahri, and Vahid Partovi Nia. How does batch normalization help binary training. *arXiv preprint arXiv:1909.09139*, 2019. 1

[30] Xijun Wang, Meina Kan, Shiguang Shan, and Xilin Chen. Fully learnable group convolution for acceleration of deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[31] Matthew Z Wong, Benoit Guillard, Riku Murai, Sajad Saeedi, and Paul HJ Kelly. Analognet: Convolutional neural network inference on analog focal plane sensor processors. *arXiv preprint arXiv:2006.01765*, 2020. 3

[32] Feichi Zhou and Yang Chai. Near-sensor and in-sensor computing. *Nature Electronics*, 3(11):664–671, 2020. 2