Department of Electrical and Electronic Engineering

Real-time object classification with FPGA-based convolutional neural networks

Roy Miles, Supervisor: Dr Jose Nunez-Yanez
Department of Electrical and Electronic Engineering, University of Bristol, 22/03/18

1. Background

- Convolutional neural networks (CNNs) receive an image, which is transformed through a series of hidden layers, until it reaches the final fully-connected layer where the class score and certainty is evaluated.
- The kernels used in the convolutional layers hierarchically extract more abstract features from the image, these are then combined to classify the object in the image.
- CNNs are **universal function approximators**, and can be trained to evaluate any classification mapping.
- Convolutional neural networks are extensively used in **computer vision and natural language processing**. With the advent of technologies such as driver-less car, the need for **high-performance low-latency classification** is becoming more prevalent.
- A typical **convolutional neural network** uses a floating point model, which incurs significant redundancy.
- **Binary neural networks** (BNNs) exploit this redundancy by converting the input activations, weights and output activations to binary format (see Fig 1).
- Vector multiplication and summations are reduced to simple **low latency logic operations** such as XORs and pop-counts, which are easily ported onto the FPGA fabric.

2. Objectives

1. Port the binary neural network implementation (for the PYNQ framework) onto the **Zedboard.**
2. Evaluate the performance benefit of using the hardware implementation in various test cases.
3. Extract **regions of interest** from a real-time video stream for classification.
4. Assess classification performance under occlusion and noise.
5. Identify any **performance bottlenecks** and address them individually.
6. Improve the frame rate by accelerating the computationally intensive tasks on the FPGA.

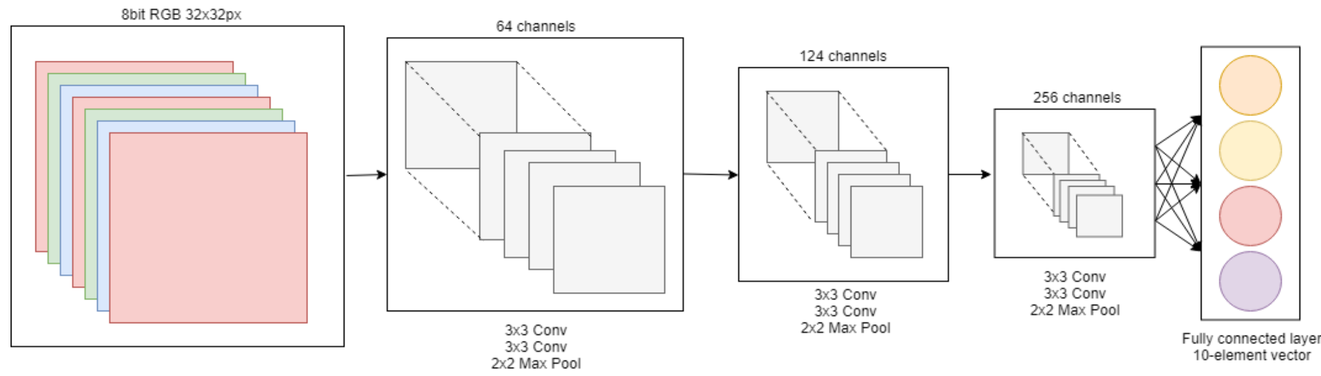

Fig 1. Block design of the FINN CNV binary neural network.

3. Research

The methods considered for extracting the regions of interest were:

- **Motion segmentation** using the morphological close transformation.
- **Colour segmentation** using k-means clustering.
- Dividing the frame up into a grid of regions which are each classified with a certainty threshold.
- The BNN implementation uses overlapping computation and communication to **maximise throughput**.
- To make full use of this design, the **pipeline must be filled for each frame**.
- The original PYNQ design reads and writes each images for classification to file.
- Lots of file operations can affect the overall FPS.

By fully utilising the pipeline of the BNN, and avoiding unnecessary I/O operations per frame, the **overall frame classification latency can be improved by 4x**.

The BNN performs well against occlusion and rotation of objects, however **performs poorly on the applications of additive gaussian noise**.

Fig 2. Segmenting the image using k-means clustering.

- **Colour segmentation** performs poorly on some example frames, as can be seen in figure 2.
- This is because the regions of interest, under the CIFAR data set, consist of a range of colours which will be fragmented during the clustering.

4. Results

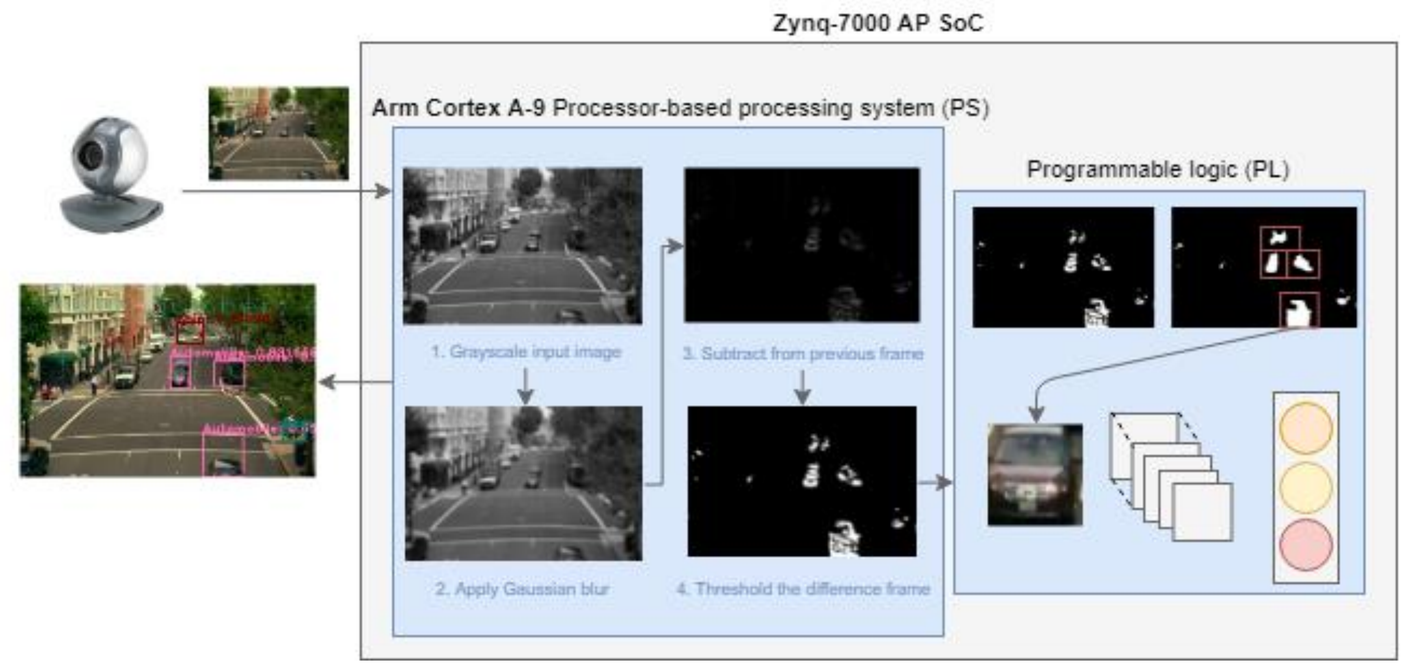

Fig 3. Full pipeline from frame capture to classification.

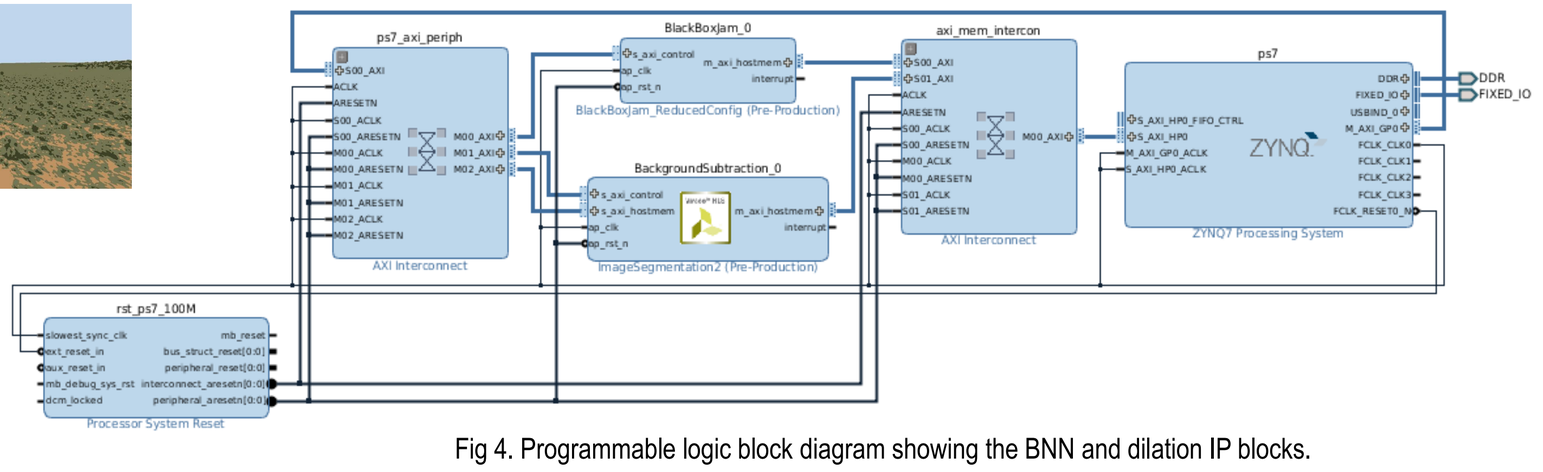

Fig 4. Programmable logic block diagram showing the BNN and dilation IP blocks.

5. Conclusion and future research

The results demonstrate that an FPGA-based convolutional neural network can achieve **high performance on concrete video stream applications.**

By reducing the physical size of the BNN design, the FPGA can be used to **accelerate other computationally intensive tasks**.

Implementation	BRAM_18K(%)	FF(%)	LUT(%)	Latency(us/image)	Throughput(images/s)
Software BNN	-	-	-	780515	1.28
Hardware BNN	96	20	67	328	3050
Reduced hardware	36	18	62	4622	216

Table 1. Resource utilisation and performance comparison between hardware and software BNN implementations

The BNN design uses a **heterogeneous architecture**, which allocates the FPGA resources based on FPS requirement. By reducing the number of SIMD lanes and processing elements, the **resource utilisation of the BNN can be reduced at a trade-off for throughput** (see table 1). The other software bottlenecks can then be accelerated alongside the BNN to improve the performance of the overall design.

A tree-based k-means clustering algorithm is demonstrated alongside the BNN, as well as an implementation of the dilation algorithm, both of which were significant latency bottlenecks for extracted the regions of interest for each frame. The dilation implementation relied on extensive use of **parallelism and a dataflow architecture** to achieve very low latency, but was not enough to overcome the offloaded host memory access overhead.

University of BRISTOL

References

[1] https://arxiv.org/pdf/1612.07119.pdf - "FINN: A Framework for Fast, Scalable Binarized Neural Network Inference"

[2] https://github.com/Xilinx/BNN-PYNQ - "BNN implementation for the PYNQ framework"