FPGA CNN Object Detection Accelerator

Hardware-Accelerated 3-Layer Convolutional Neural Network on Zynq-7020

Tejas Dahake
Bachelor of Technology
Indian Institute of Information Technology Dharwad

Contents

1 Abstract

This project implements a custom 3-layer Convolutional Neural Network (CNN) accelerator on a Xilinx Zynq-7020 FPGA (PYNQ-Z2 board) for real-time 6-class object detection at **24 FPS**.

The entire CNN compute pipeline — convolution, ReLU activation, and max pooling — executes in FPGA hardware, achieving approximately **5× speedup** over optimized ARM C code running the identical model.

Metric	Value
FPGA inference	6.8 ms per 128×128 image
ARM C inference	248 ms per image
FPGA/ARM speedup	∼5×
Classes	airplane, cat, zebra, bus, bicycle, donut
Accuracy	56.1% (FPGA) / 54.4% (ARM)
FPGA clock	50 MHz

2 System Architecture Overview

The system follows a hardware/software co-design approach on the Zynq SoC.

2.1 Processing System (ARM Cortex-A9)

- USB Webcam capture using OpenCV
- Preprocessing: crop → grayscale → resize to 128×128
- AXI DMA for weight and image transfer
- AXI-Lite feature readout
- Spatial bin pooling + softmax classifier
- Bounding box visualization and MJPEG streaming

2.2 Programmable Logic (FPGA Fabric)

- Weight BRAM (23,184 × 8-bit)
- 8-state Layer FSM controller
- Line buffer (3-row circular cache)
- Sliding window (3×3 extraction)
- 16 parallel convolution cores
- 16 accumulator BRAMs
- 16 ReLU units

- 16 streaming max-pooling engines
- 112 feature BRAMs (all on-chip)

3 Hardware Architecture (FPGA)

3.1 Module Hierarchy

```
cnn_acc_top.v
 weight_bram.v
 feature_bram.v × 112
 line_buffer.v
 sliding_window.v
 conv_core.v × 16
 accumulator.v × 16
 ReLU.v × 16
 max_pooling_engine.v × 16
 layer_fsm.v

lyr3_cnn_axi_slave_lite.v
 lyr3_cnn_axi.v
```

3.2 Line Buffer

Provides 3 simultaneous rows for streaming 3×3 convolution. Implemented using two circular shift-register arrays. Supports a `clear` signal between layers.

3.3 Sliding Window

Extracts a 3×3 window from the line buffer. Two pipeline stages. Outputs 9 pixels and `window_valid`.

Latency: 2 clock cycles.

3.4 Convolution Core

3-stage pipelined architecture:

1. 9 parallel signed multipliers
2. 3 row sums
3. Final 20-bit signed accumulation

Throughput: 1 convolution per clock (after fill). 16 cores operate in parallel.

3.5 Accumulator

4096 × 24-bit BRAM with read-modify-write.

Modes:

- Accumulate mode
- Overwrite mode (first tile)

Enables input channel tiling.

3.6 ReLU

Pure combinational logic:

- Negative → 0
- Overflow → 255
- Else → pass lower 8 bits

Right shift applied before activation.

3.7 Max Pooling

Streaming 2×2 stride-2 downsampling. Uses coordinate parity logic (even/odd x,y). Produces output every 4 input pixels.

3.8 Layer FSM

8-state controller:

```
IDLE → LOAD_WEIGHTS → PROCESS_PASS → CHECK_CHANNEL
                                          ↓
        DONE ← CHECK_LAYER ← DRAIN_FINISH ← DRAIN_ACCUM
```

Handles weight loading, tiling, accumulation, drain, and layer transitions.

4 Feature BRAM Layout

Channels	Layer	Entries	Spatial Size
0–15	Layer 0	4096	64×64
16–47	Layer 1	1024	32×32
48–111	Layer 2	256	16×16

All feature maps stored entirely on-chip.

5 AXI Interface

Register	Offset	Function
Control	0x00	Start / Reset
Status	0x04	Busy / Done
Shift L0	0x08	ReLU shift layer 0
Shift L1	0x0C	ReLU shift layer 1
Shift L2	0x10	ReLU shift layer 2
Weight data	0x14	Weight write port
Weight addr	0x18	Weight address
Weight enable	0x1C	Weight write enable
Output channel	0x20	Feature channel select
Output address	0x24	Feature address
Output data	0x28	Feature read data

6 Software Architecture (ARM)

6.1 Training Pipeline

- COCO dataset (6 classes)
- Data augmentation
- 3-layer CNN with Quantization-Aware Training
- Exported weights.bin (23,184 bytes)

6.2 Classifier

$64 \times 16{\times}16$ features $\rightarrow$ $4{\times}4$ spatial bin pooling $\rightarrow$ 1024 features $\rightarrow$ $W(61024)x{+}b \rightarrow$ Softmax

Class Activation Map used for bounding box extraction.

7 Physical Design (Vivado Results)

7.1 Synthesis & Implementation

Stage	Status
Synthesis	synth_design Complete
Implementation	write_bitstream Complete
Failed Routes	0
Methodology	1 non-critical warning

7.2 Resource Utilization (Zynq-7020)

Resource	Used	Available	Utilization
LUT	25,901	53,200	48.7%
Flip-Flops	27,166	106,400	25.5%
BRAM	125	140	89.3%
DSP	0	220	0%

Observation: All multipliers are LUT-based (no DSP slices used). BRAM is the limiting resource.

7.3 Timing

Metric	Value
WNS	+1.490 ns
TNS	0.000 ns
WHS	+0.018 ns
THS	0.000 ns
Clock	50 MHz

All timing constraints met with positive slack.

7.4 Power

Metric	Value
Total On-Chip Power	1.786 W

8 Results

8.1 Performance Comparison

Mode	Max FPS	Avg. Inference	Speedup
FPGA	24	6.8 ms	1×
ARM C	3.7	248 ms	36× slower
ARM numpy	0.3	3300 ms	485× slower

8.2 Classification Accuracy

Platform	Overall	airplane	cat	zebra	bus	bicycle
FPGA	56.1%	73.2%	45.0%	71.8%	49.0%	43.0%
ARM	54.4%	73.2%	41.0%	77.6%	52.0%	35.0%

9 Key Design Decisions

1. 8-bit quantization
2. 16-core parallelism
3. Tiled accumulation
4. All on-chip BRAM
5. Spatial bin pooling
6. Separate classifier weights
7. Streaming max pool

10 Lessons Learned

- Fixed-point mismatches affect classifier behavior
- AXI-Lite readout is performance bottleneck
- QAT is essential

11 Future Work

- DMA-based feature readout (30+ FPS)
- Larger / deeper CNN
- HDMI input
- Dynamic batching