Tufts UNIVERSITY | School of Engineering

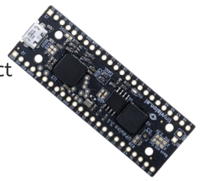
TinyML | Machine Learning on Budget FPGA

Built by: Drew Cohen, Tom Lyons, Cory Lemberg, Kinan Rabbat, Jacob Carulli

Sponsor: Steven Bell

SEE OUR WEBSITE!

PROBLEM



Design and implement a low-power "ML at the edge" system to perform visual object recognition on an ES4 level FPGA to get future students excited about machine learning on hardware. Implement machine learning algorithm in hardware description language on low power FPGA within strict timing, space, and resource constraints.

SOFTWARE DESIGN

A good image classification problem should be interesting, feasible, and measurable. We chose to classify decimal digits in the vein of MNIST classification, a well-known benchmark for machine learning using hardware aware software design.

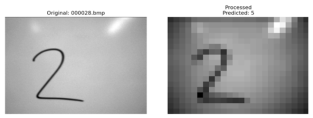
WHICH MODEL TO USE?

Goals for our ML Model:

- Stay within our memory budget (128k bits)
- Feasible for hardware implementation
  - MLP (winner): balances sound performance with feasibility of implementation in hardware
  - CNN: sliding window logic difficult in hardware

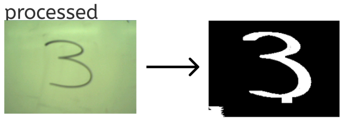
MAJOR ISSUE FACED

Lens shading on camera images.



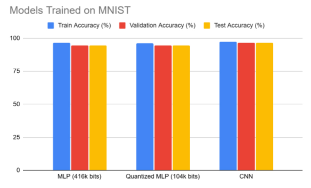
CAMERA IMAGE PROCESSING

original
processed

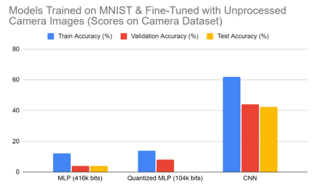


Processing images gets rid of lens shading and thickens digits

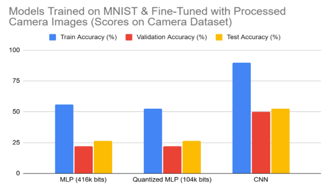
Grid Search: Tuned # of neurons in each hidden layer, input image resolution, dropout, and weight decay



All models achieve >90% accuracy on MNIST



Models suffer with unprocessed camera images

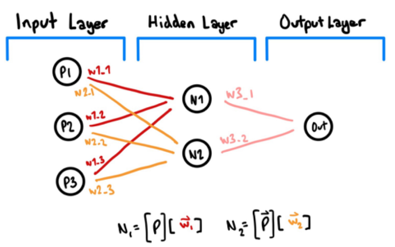


Models improve with image processing

HARDWARE CONSTRAINTS

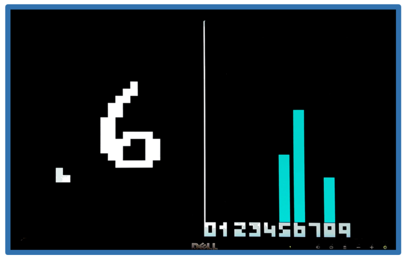
4x 256kbit Single Port SRAM	30x 4kbit DP Embedded Block Ram	8x 16x16 DSP
16 bit words with 4 bit nibbles	Customizable words size with 1 bit nibbles	Optimized Hardware for fast Multiply & Adding
1 Mbit SRAM for features	120kbit EBR for weights	Multiply Accumulate for Dot Product

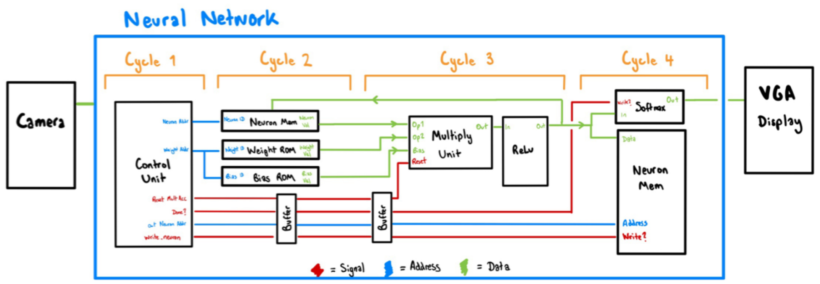
HARDWARE DESIGN

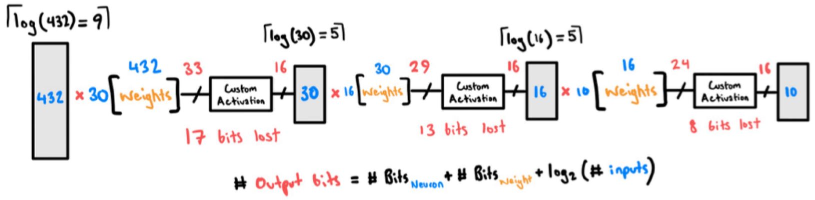


$N_1 = [\vec{P}][\vec{w}_1]$ $N_2 = [\vec{P}][\vec{w}_2]$

FINAL VGA OUTPUT

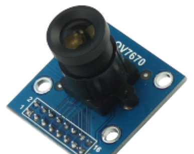






$\#\ \text{Output bits} = \#\ \text{Bits}_{\text{Neuron}} + \#\ \text{Bits}_{\text{weight}} + \log_2(\#\ \text{inputs})$

OV7670 CAMERA (~$2)



- Register configuration using custom SCCB module
- Output as QVGA 422YUV
- Compressed further to 20x15 pixels
- Streamed greyscale luminance stored in variety of bit depths in SRAM
- Image is compressed further and stored in additional SRAM for classification

OUTCOME

- Trained a model to successfully (>90%) classify digits from idealized dataset, camera and setup adjustments will improve accuracy further
- Successfully recreated MLP operations on $30 FPGA w/ pipelined memory usage
- Built out OV7670 camera support from scratch
- Created and conducted a trial lab for future ES4 students to create a smaller version of our project