

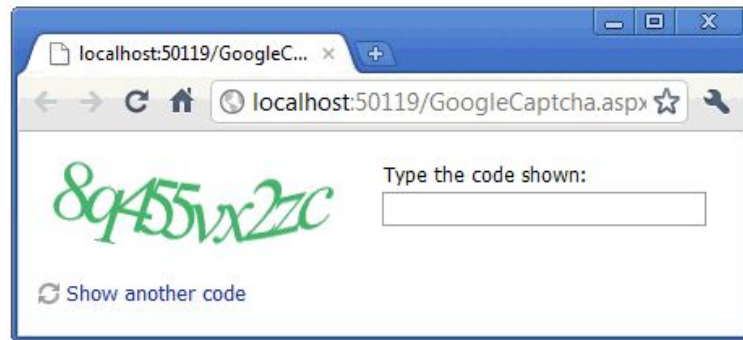
The Captcha Beater



Mauris, Nikhil, Randy & Srivatsan

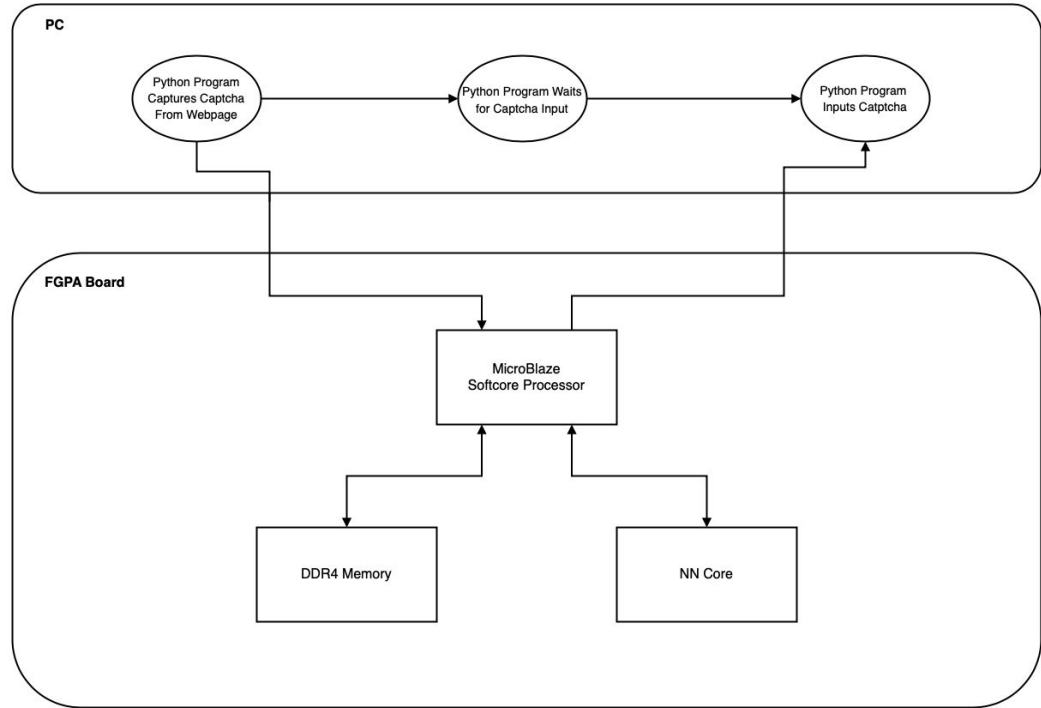
Project Overview

- We are focusing on text recognition for Captchas
- Implement a neural network on an FPGA to tackle the issue
- Potential applications:
 - Malicious captcha farms
 - Captcha solving assistant to individuals that are visually impaired
 - Converting handwriting to text
 - Finding vulnerabilities in captcha generators
 - Captcha avoidance for automated testing purposes



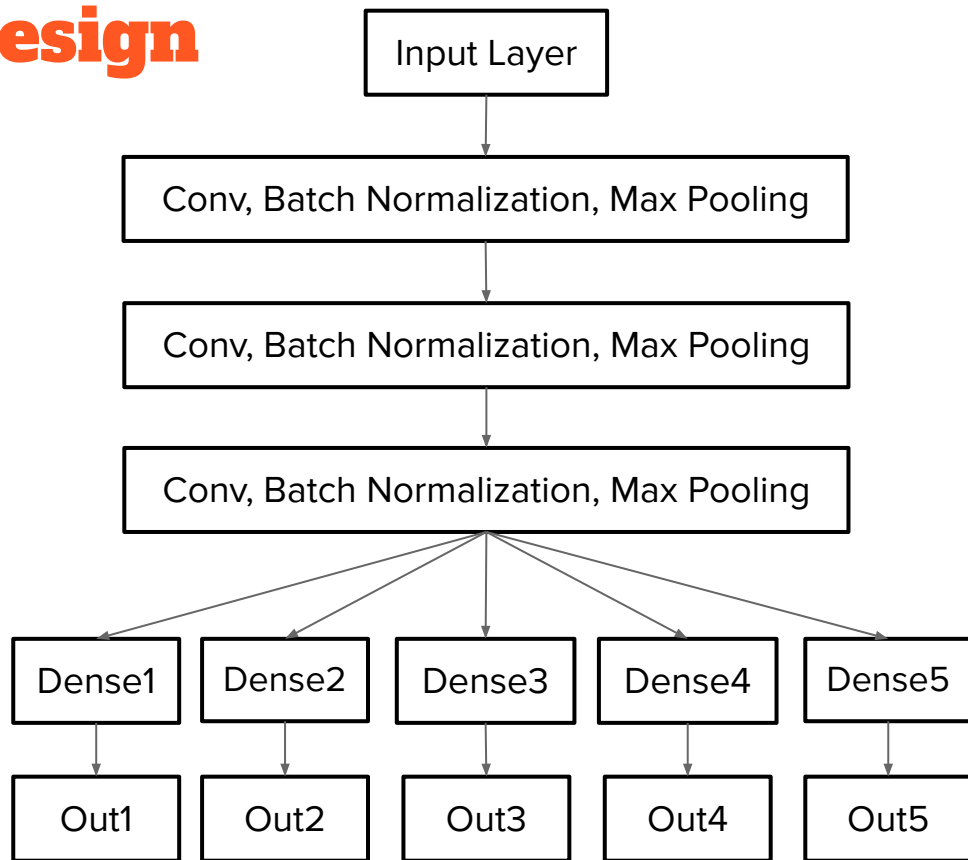
Proposed System

- Python program captures captcha and sends image to FPGA memory
- Softcore Microblaze conducts forward propagation to predict captcha
- Captcha prediction is sent back to PC and entered in
- Softcore MicroBlaze processor can be used to train neural network if time permits



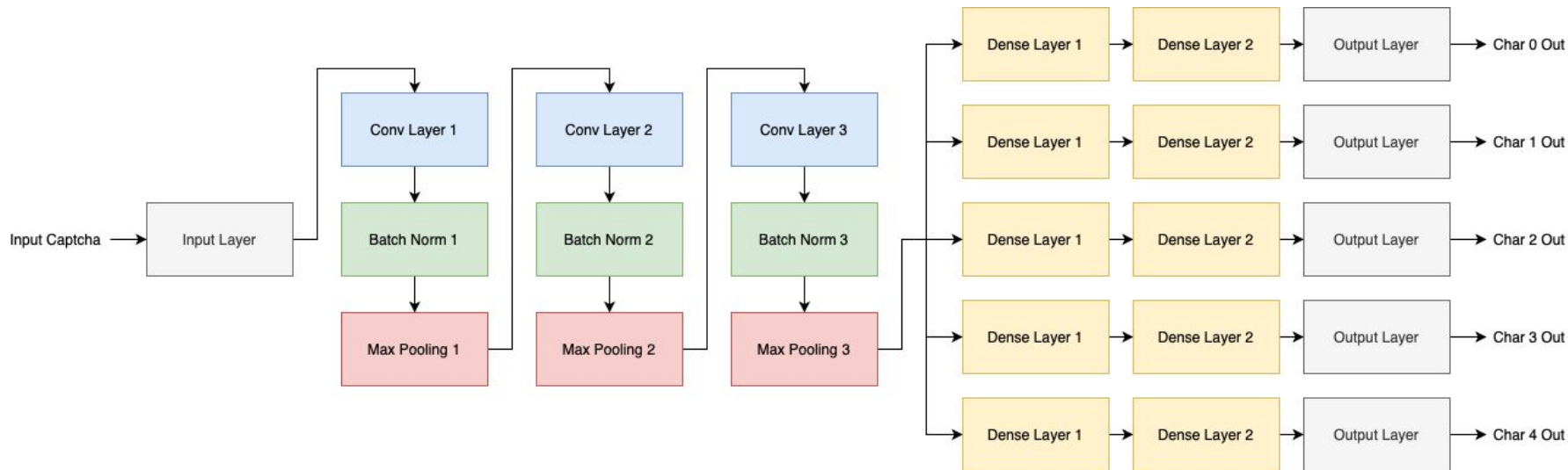
Original Neural Net Design

- Planning on using a model that can splice each character in the captcha and classify them individually
- Model will have 'n' outputs for 'n' characters of the captcha
- Subset of characters to be used (19 total valid characters for now)



Updated Neural Net Design

- Added a second dense layer to improve accuracy



What's in HLS?

- Neural Network will be implemented in HLS
 - Convolution
 - Batch Normalization
 - Max Pooling
 - Dense Layers
- What else is needed that is not HLS?
 - Softcore MicroBlaze
 - DDR4 Memory
 - Communication Components - Use FPGA IP Cores for data transfer between PCIe/Ethernet, Memory, Microprocessor & Neural Network Core

Hardware Requirements

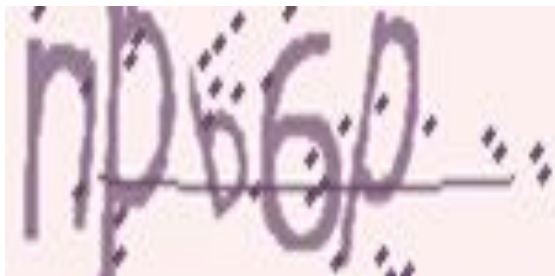
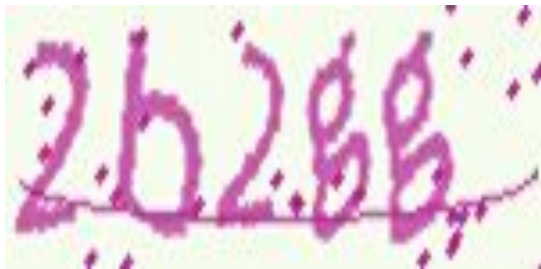
- FPGA Board - ADM-PCIE-8V3
 - Xilinx Virtex® UltraScale™ XCVU095-2 - FFVC1517
 - 2x 100G Ethernet MACs (incl. KR4 RS-FEC)
 - 2x 150G Interlaken cores
 - 4x PCI Express x8 Gen3
- PC & FPGA communication through PCIe Gen3
- Time permitting we would like to include multiple FPGAs to simulate a cloud infrastructure

Workflow & Testing

- Using GitHub for version control
- Partitioned the work so it could be parallelized
 - Neural Network layers
 - System-Level Blocks (PCIe, MicroBlaze and DDR4)
- Manually inspected outputs using minimal test best for each layer during design phase
- Fully automated test bench for more rigorous testing of 1000 samples
- Modular design for ease of integration - each module is tested for functionality
- No timezone issues faced - only one person outside of Ontario (Alberta)

Progress to Date

- Constructed custom dataset which of 5-character CAPTCHAs which (for now) uses a pool of 19 characters



Progress to Date (Neural Network)

- Constructed basic machine learning model which predicts CAPTCHAs at 80% accuracy
 - This meets our target as CAPTCHAs are designed to be solved by humans at an 80% accuracy
 - There is room for improvement with a larger dataset and more hyperparameter tuning
 - Takes about 2 hours to train the model
- Replicated the model in C
 - Extracted the weights, structure of the model such that it can be coded in C
 - Verified working through a test bench we created
 - Using floating point arithmetic
- Implemented and tested fixed point arithmetic in the C model
 - Fixed point implemented in all parts of model aside from Softmax function
 - Achieves > 80% accuracy on 1000 samples

Progress to Date (System Design)

- Created block diagram utilizing PCIe, MicroBlaze & DDR4
- Setting up Vivado SDK application to test DMA data transfer between
 - PC -> PCIe -> DDR4 through DMA driver
 - DDR4 -> MicroBlaze
 - MicroBlaze -> DDR4
 - DDR4 -> PCIe -> PC through DMA driver



Current Steps

- Looking into whether fixed point can be used in the Softmax layer
 - Might be able to remove Softmax layer entirely
- Working on increasing model accuracy further
 - Adding 75,000 images to dataset to increase the model's capabilities
- Testing software model on MicroBlaze
- Test hardware using CoSim to verify functionality and performance
- Add pragmas to optimize hardware

Next Steps

- Map weights and biases to the neural network IP Core
- Create an AXI Stream wrapper for neural network IP Core
- Integrate project - MicroBlaze, PCIe, DDR4 & Neural Network IP Core
- Test neural network on FPGA with a couple captchas
- Create mock CAPTCHA application (time permitting)
- Present and finalize report

Plan & Milestones

1. Get a software version working 
 - Finding or creating an appropriate dataset
 - Using that dataset to train a neural network
 - Achieve reasonable results/accuracy
2. Get a neural net working on an FPGA through HLS
3. Developing a method to communicate with an external PC for data transfer (Images In, Results Out) 
4. Integration
 - Send a captcha to the FPGA
 - FPGA interprets captcha and determines result
 - FPGA sends result back to PC
 - Make fixes as necessary
5. Test the application on a legitimate website or customized application
6. Complete the final report & demo

Challenges Faced

- Finding a good quality dataset that is readily available
 - Solved by creating our own dataset
- Coming up with an appropriate neural network structure
 - Solved through experimentation, research
- Understanding the way the Keras framework operates under the hood
 - Needed to understand the way weights are stored and the order in which they are stored
 - Solved through experimentation and careful manual testing
- Setting up board constraints for Vivado block diagram

Project Conclusion

- Plan is to finish end of April 30th
- Plan is to present/demo the following week
- Worst case estimate is early/middle of May