Trained CNN and Inference Acceleration on FPGA for MNIST Dataset Recognition.

Connor James Espenshade (cje2136)
Stephen Ogunmwonyi (iso2107)
Bradley Jocelyn (bcj2124)
Aymen Norain (aan2161)

Convolutional Neural Networks have become an essential part of modern pattern recognition and computer vision. However, the significant computational complexity of CNNs can lead to significant energy consumption issues and output latency issues especially in an embedded systems platform. Field Programmable Gate Arrays (FPGAs) have emerged as a potential avenue for accelerating the workload due to the inherent parallelism in the operation of a CNN. FPGAs have shown significant success and performance improvements in the types of matrix multiplication required for CNN inference.

FPGA Implementation:

FPGAs allow for custom hardware implementations that can be tailored to mimic specific aspects of the CNN i.e convolution, pooling and activation functions. Their reconfigurability will enable rapid testing and FPGAs can offer extremely low latency processing for real-time applications. By also taking advantage of pipelining, we can increase throughput and take advantage of the parallel processing capabilities. One factor for us to be cognizant of is the limited resources on an FPGA. This is why we're focusing on accelerating inference on the FPGA on a trained model. The aim of this project is training a model and then accelerating inference on an FPGA. We will be testing this on the MNIST dataset

MNIST Dataset:

The dataset we will be training for is the MNIST data set of handwritten digits. It is a common benchmarking test for image classification algorithms in machine learning. The dataset consists of handwritten digit images (0 to 9). It contains

70,000 greyscale images of digits each being 28x 28 pixels in size and the dataset is divided into 60,000 training images and 10,000 testing images. The low pixel resolution and monochrome nature of the images allows even simple models to achieve accurate results

Algorithm:

We will plan on training a custom CNN to recognize MNIST through TensorFlow/Keras, and implement the generated layers and weights in Verilog hardware. There are an abundance of tutorials for this process, one is available here https://www.kaggle.com/code/amyjang/tensorflow-mnist-cnn-tutorial. Another thought would be to use Resnet-18 or some established CNN, but these are often larger and more complex than the MNIST dataset requires, requiring more unnecessary Verilog functions and more opportunities for something to go awry.

Hardware/Software Interface

As for the interface between hardware and software, we will have a dedicated interface to manage the transfer of data between the neural network/engine and/or a camera potentially. A control protocol will initiate, monitor, and synchronize inference operations, with a data bus to be implemented to support efficient, low-latency communication.

Major Tasks:

1) CNN model development and training
2) RTL Hardware implementation
3) Verification of RTLcomponents
4) Interface between CNN and FPGA

Future Opportunities to Expand

- Camera to detect the number written by the user in real time
- EMNIST, expanding to the dataset which adds alphabetic characters