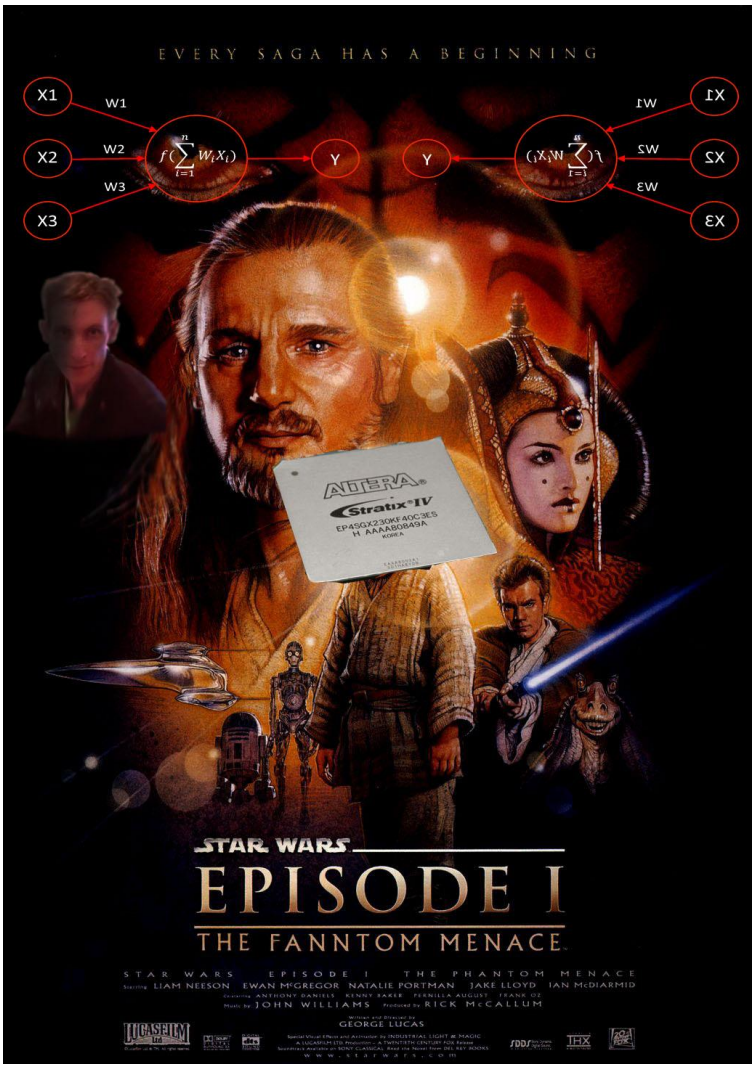
P20 - FANN(tom Menace)

FPGA ACCELERATED NEURAL NETWORK

Team Leader: Luke Schwartzkopff (SCHLUK004) | Chief Researcher: Kiuran Naidoo (NDXKIUR001) | Toolsmith & Repository Manager: Othniel Konan (KNNOTH001)



Project Description

The project aims to use FPGAs to accelerate a neural network. The output of each node within a neural network is as a result of the sum of the results of the previous layer with weights applied to each of these results and optionally (but usually and usefully) then applying some function (often sigmoid) to the result [1]. This clearly is a massively parallel operation with similar operations used throughout - something with which FPGAs could potentially excel at.

Operations using reasonably complex neural nets - especially those used for image processing which usually happen to be convolutional [2], are computationally expensive. Because of this, they often fail to be useful in realtime applications. In my personal experience, when trying to run number plate recognition on a RPI 2B (900MHz Cortex-A7 quadcore + 1GB RAM), operations would take in excess of 6 seconds.

It is clear that an FPGA implementation could help alleviate this problem - especially in embedded applications where powerful desktop-level processors are not available and operations must generally run in near-realtime in order to be useful. In this respect, it could also potentially be a huge cost saving measure such that a powerful processor need not be included in the embedded system. There is evidence to suggest that this sort of FPGA acceleration could be effective [3]. For the scope of this project, simple realtime image recognition will be detected.

Prototype Specification

- A PC-based application (probably ready-made software) will be used to train a convolutional neural network using the backpropagation and gradient descent algorithms [1].
- A custom-made PC application will be used to convert the neuron weighting information and network structure into an HDL.
- This HDL can then be used with its appropriate synthesiser to program an FPGA.

- In order for the FPGA to have access to an image, an image will be placed into its memory for testing purposes or if time restraints do not allow for another method.
- Ideally, the FPGA will be connected to a camera via a RPI. The RPI will handle the camera drivers and convert the images to a nice serial bitstream easily readable for the FPGA periodically. An appropriate colour depth and image resolution will be chosen. [4]
- The neural network will likely implement a reasonably simple image recognition operation i.e ("Is this an apple or not an apple?").
- The output can be displayed on LEDs or an LCD screen.

Criteria for acceptable solution

- Neural network trained correctly (can be tested standalone on PC).
- Custom PC-application produces syntactically correct and sensible HDL.
- The HDL is also *correct* and the FPGA works to recognise an image under ideal circumstances while reading from its internal memory.
- Achieve reasonable speedup over a golden measure implemented on a Raspberry Pi or similar system.
- (hopefully!) interface with the camera correctly and update the image being recognised.
- (hopefully!) recognise images fast enough to seem realtime.

Bibliography

[1] "Artificial Neural Network". *En.wikipedia.org*. N.p., 2017. Web. 23 May 2017.

[2] "Cs231n Convolutional Neural Networks For Visual Recognition". *Cs231n.github.io*. N.p., 2017. Web. 23 May 2017.

[3] "FPGA Acceleration Of Convolutional Neural Networks - Nallatech". *Nallatech*. N.p., 2017. Web. 23 May 2017.

[4]"Camera Module - Raspberry Pi". *Raspberry Pi*. N.p., 2017. Web. 23 May 2017.