FPGA vs. GPU for Machine Learning

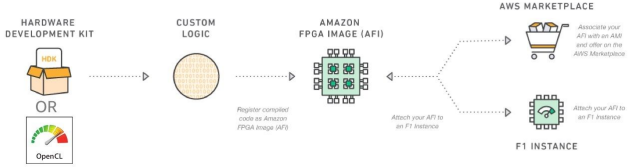
Caleb Burke, Ben Leone

Background

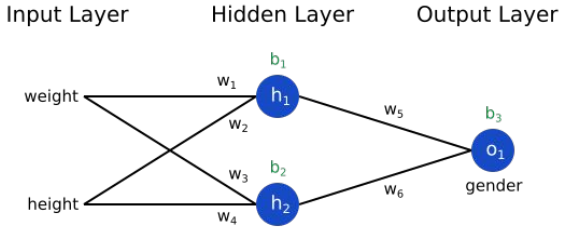
- Comparison of machine learning algorithm on FPGAs and GPUs
- FPGAs have performance benefits but can be more difficult to code or optimize
- Examined current offerings of machine learning frameworks (i.e. PyTorch, TensorFlow, EC2)

Methods

- Python to C neuron translation
- Vivado C to HDL High Level Synthesis (HLS)
- Register-transfer level (RTL) design
- AWS EC2 F1 instance FPGA image creation
- Python & C Benchmarking
- BU Shared Compute Cluster (SCC)

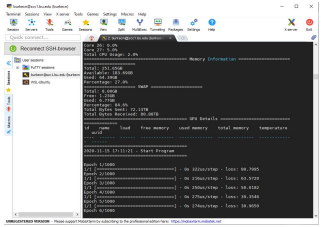


Neural Network Example

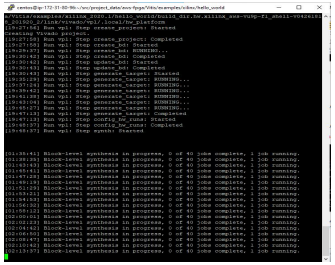


Results

Python neural net train, test benchmark local & SCC

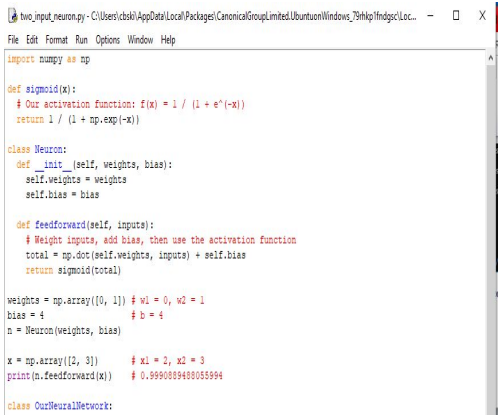


Vitis AWS Getting started C/C++ acceleration on AWS



Code

- Implemented simple ML algorithm in Python
- Implemented same algo into C for basic ML neuron
- Used several neurons to make simple network (diagram to left; code below)



```
two_input_neuron.py - C:\Users\cbski\AppData\Local\Packages\CanonicalGroupLimited.UbuntuonWindows_79rhkp1fndgsc\Loc...
File  Edit  Format  Run  Options  Window  Help
import numpy as np

def sigmoid(x):
  # Our activation function: f(x) = 1 / (1 + e^(-x))
  return 1 / (1 + np.exp(-x))

class Neuron:
  def __init__(self, weights, bias):
    self.weights = weights
    self.bias = bias

  def feedforward(self, inputs):
    # Weight inputs, add bias, then use the activation function
    total = np.dot(self.weights, inputs) + self.bias
    return sigmoid(total)

weights = np.array([0, 1]) # w1 = 0, w2 = 1
bias = 4                   # b = 4
n = Neuron(weights, bias)

x = np.array([2, 3])       # x1 = 2, x2 = 3
print(n.feedforward(x))    # 0.9990889488055994

class OurNeuralNetwork:
```

Conclusions

- FPGA and GPUs will both continue to have impacts in machine learning
- Different approach to code base (software vs HDLs) and optimization
- Ensure full understanding of various platforms and technologies before implementation