CG4002: Computer Engineering Capstone Project

Hardware AI

Lecturer : Dr. Sangit Sasidhar (sangit@nus.edu.sg)
Department of Electrical and Computer Engineering

Hardware AI Requirements

- Classify the Laser Tag Game Actions
- Implement 1 neural network model on software for classification
- Implement the same neural network model on the FPGA (Use C++ High Level Synthesis (HLS)
- Train and run the model with the test data
- Evaluate the performance of the NN on the FPGA

PYNQ

AVNET
Reach Further

Ultra96
64-bit Arm architecture coupled with Xilinx programmable logic

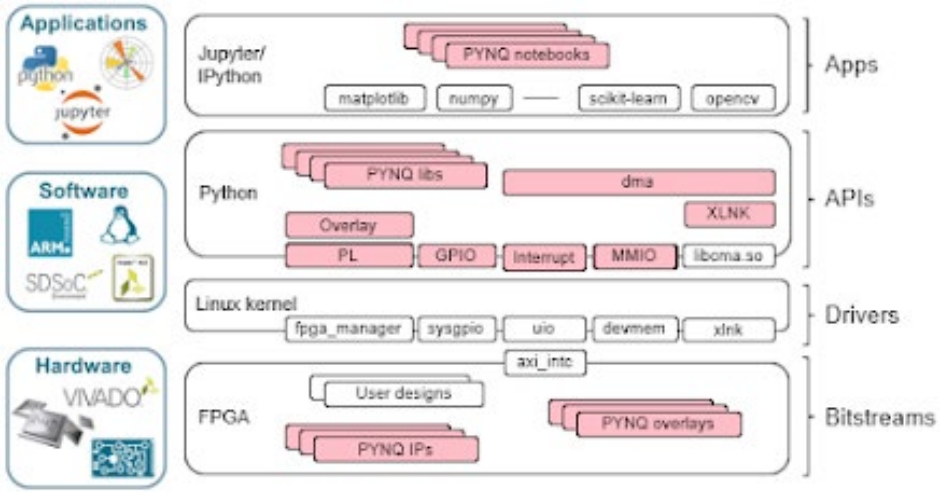
python

XILINX

/ULTRA96

Boards

Ultra96 Features

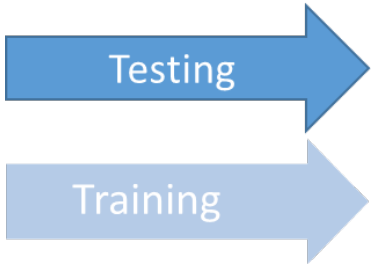


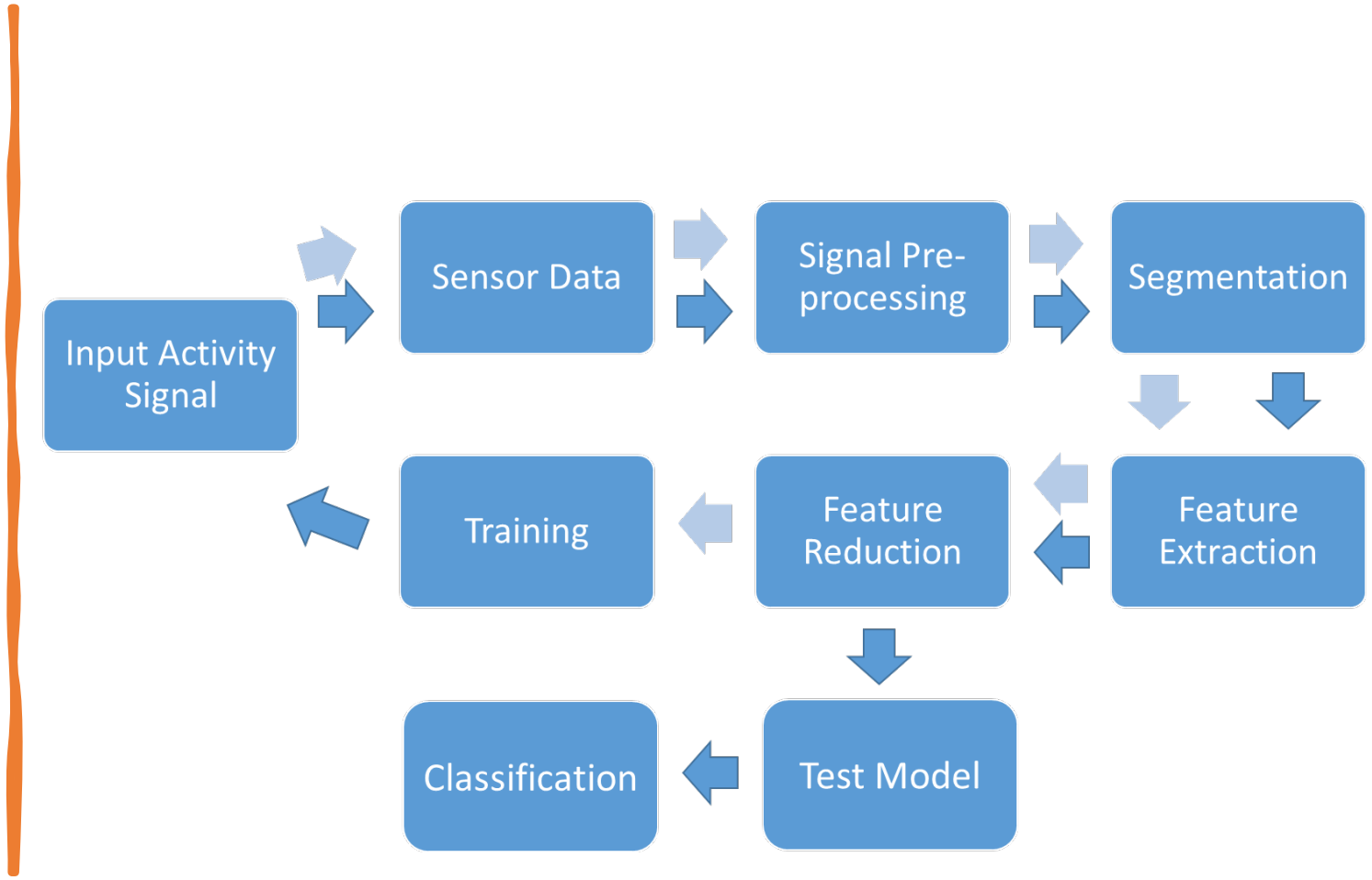
SoC	Xilinx Zynq UltraScale+ MPSoC ZU3EG A484
RAM	Micron LPDDR4 memory provides 2 GB of RAM in a 512M x 32 configuration
Storage	Delkin 16 GB microSD card + adapter
Wireless	802.11b/g/n Wi-Fi and Bluetooth 4.2 (provides both Bluetooth Classic and Low Energy (BLE))
USB	1x USB 3.0 Type Micro-B upstream port 2x USB 3.0, 1x USB 2.0 Type A downstream ports
Display	Mini DisplayPort (MiniDP or mDP)

AI / Machine Learning

- What?
  - Algorithms that improve their performance at a task through experience
  - Computers learn to solve problems
    - Voice Recognition
    - Activity Detection
    - Stock Market forecasting
    - Face Detection etc
- Essence of Learning Problems
  - Some patterns exist
  - Difficult to pin down mathematically/analytically/formally
  - Data exists/can be generated for these patterns

AI / Machine Learning





AI / Machine Learning

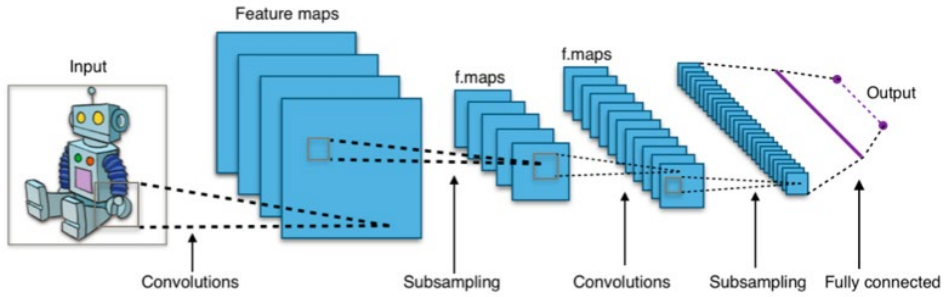
- Pre-processing
  - Smooth the signal or remove the noise
  - Normalization or scaling
- Segmentation
  - Identify start and/or end of actual activity in the given data window
  - Onset detection
- Feature Extraction
  - Transforming raw data into numerical features
  - Time-Domain- Mean, Variance, Median etc
  - Frequency Domain – Entropy, energy, Peak/median frequency, wavelets etc

AI / Machine Learning

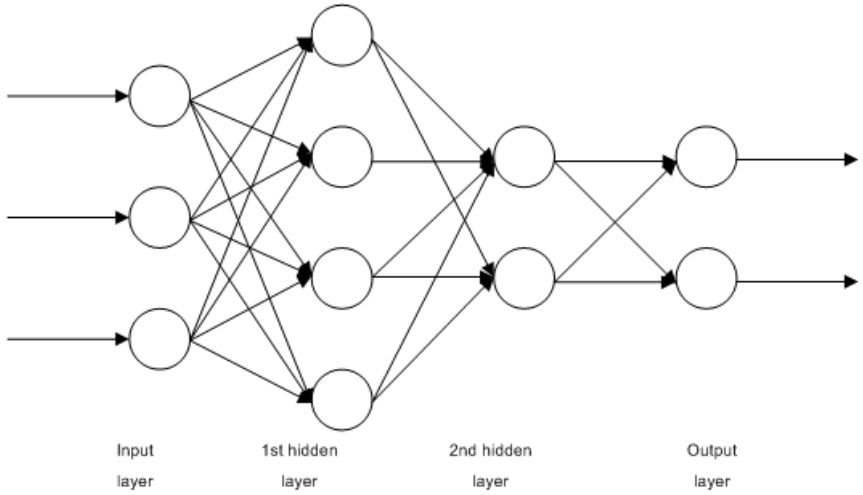
- Feature Reduction
  - Reduce the number of features
  - Reduces the classifier complexity
  - PCA, Random Forest, ICA etc
- Classification
  - Artificial Neural Networks
  - Support Vector Machines
  - Linear Classifiers
  - K-nearest neighbor (KNN)

Software Model Assessment

- Generalization performance of an ML method relates its prediction capability on independent data sets
  - Test Error – the average error that results from using a trained model to predict the response on a new observation/independent test sample
  - Training Error – average loss over the training sample
- Randomly divide the dataset into three parts
  - Training set – fit the model
  - Validation set – prediction error for model selection
  - Test set – assessment of prediction error of the chosen model
  - Typical split- 50% training, 25% validation, 25% test



Neural Networks

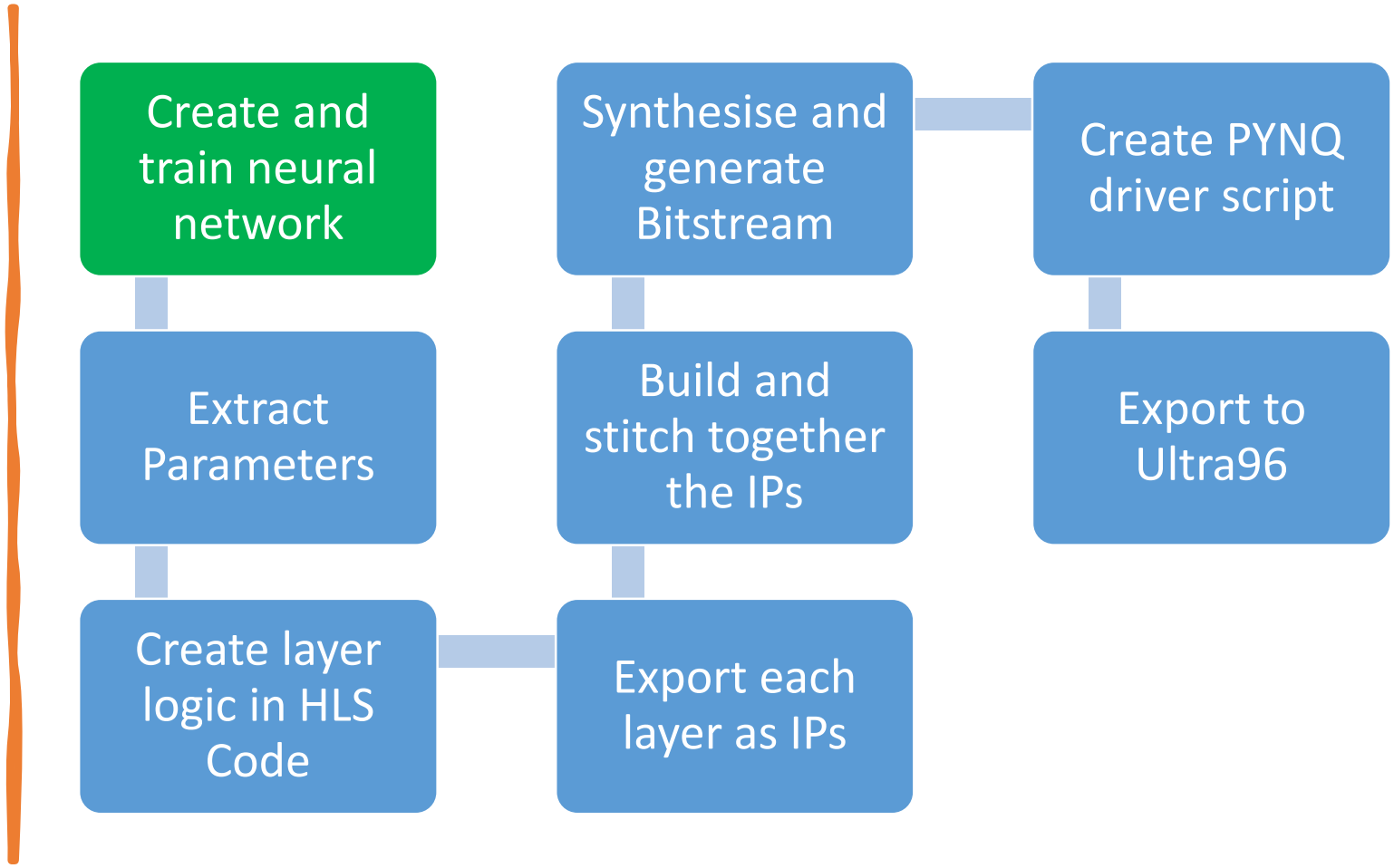


- Multiple Layers
- Each Layer consists of Multiple Neurons
- Input Layer – Features Extracted
- Hidden Layers – Do not connect to the outer world
- Output Layer – Labels/Classes of the activity

Sample Datasets

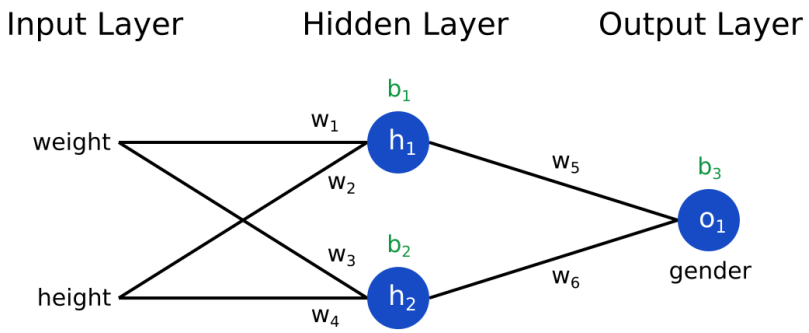
- https://archive.ics.uci.edu/ml/datasets.php
  - Filter with different criteria or search with certain keywords (e.g. sensor; action recognition)
  - Cite: Smartphone-Based Recognition of Human Activities and Postural Transitions Data Set, UC Irvine
- https://www.kaggle.com/vmalyi/run-or-walk
  - A dataset containing labelled sensor data from accelerometer and gyroscope
- Many other activity data sets available online

FPGA Implementation: Hardware FPGA Design



***Good starting Point: https://wiki.nus.edu.sg/display/ee4218/EE4218+Labs**

FPGA Implementation: Neural Network Layer



- Convolution and Linear Layer

$$Output = bias + (input \cdot weight)$$

where;

- $input = 1 \times i\ vector\ of\ input\ features$
- $weight = i \times j\ vector\ of\ weights$
- $bias = 1 \times j\ vector\ of\ biases$
- $output = 1 \times j\ vector\ of\ neuron\ outputs$

➔Matrix multiplication

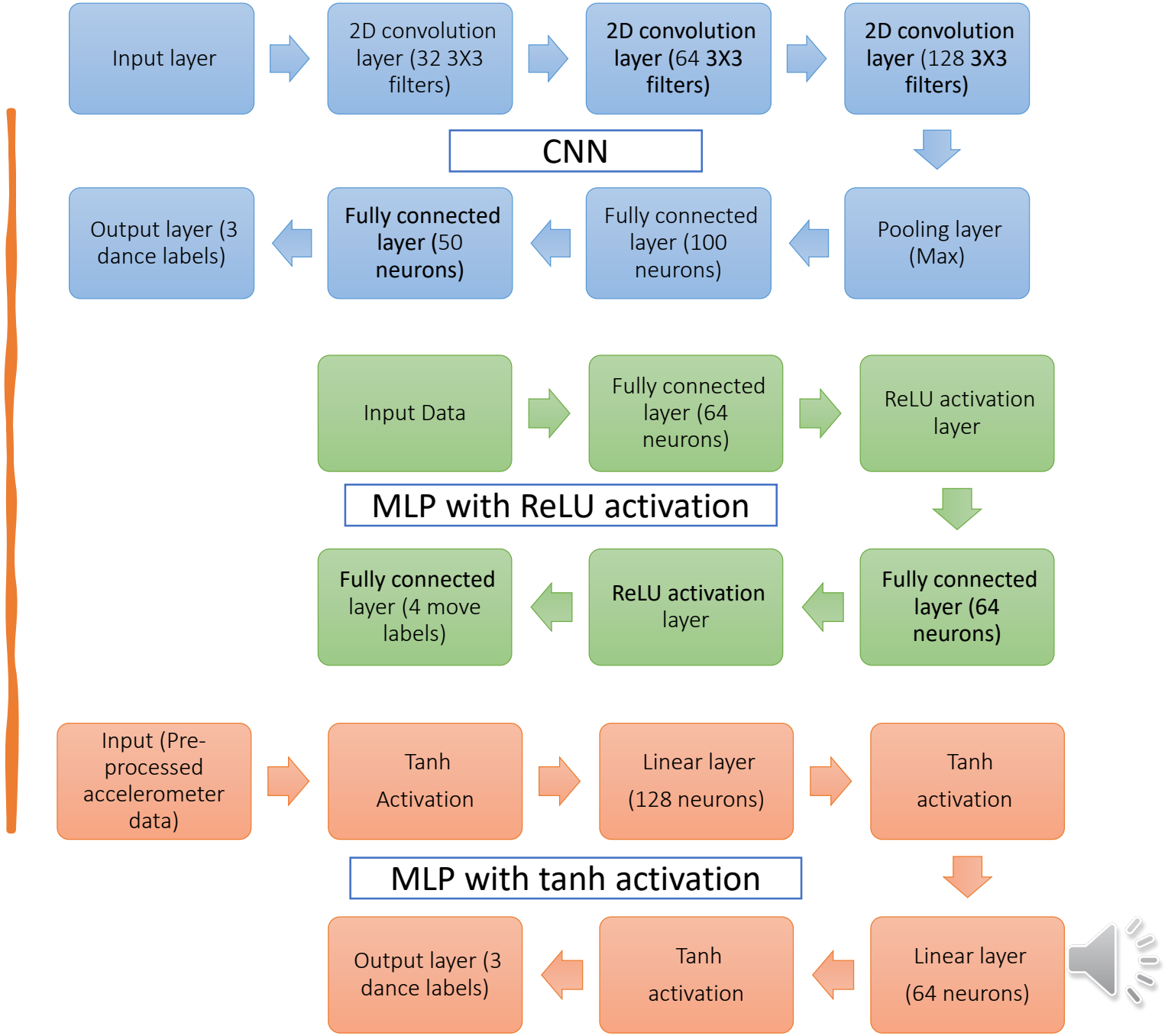
FPGA Implementation: Activation Function - ReLU

- ReLU : Rectified Linear Unit
- Simple Calculation:

$$
\begin{aligned}
&if\ input > 0:\\
&\quad return\ input\\
&else:\\
&\quad return\ 0
\end{aligned}
$$

- Can be implemented in the FPGA using Look Up Tables (LUTs)

FPGA Implementation: Sample Neural Networks



FPGA Implementation: Optimisation

- IP data streams are 32-bit floats → compiler needs to synthesise hardware for floating point operations
- Vivado HLS has “ap_fixed” data type for fixed point conversion and quantisation of data
- Fixed point Conversion: Requires checking for absolute maxima and minima of all the data
- Quantisation: reduces the precision of the data→ may affect prediction accuracy
- Precision loss is evaluated at each layer level and at the model level to find a small enough bit-width

Evaluation of Hardware AI

- Evaluation of the inference model and its comparison to the software ML implementation (Accuracy, recall and precision)
- Hardware co-simulation at the HLS stage
- Optimization of the design using parallelization and pipelining
- Hardware Utilization: BRAM, DSP, FF, LUT
- Ultra96 Power Consumption Measurement: https://github.com/Avnet/Ultra96-PYNQ/blob/master/Ultra96/notebooks/common/ultra96_pmbus.ipynb

Documentation

- A proper documentation is very useful in debugging
- Document everything in a wiki / knowledge bank (eg: NUS wiki, GIT, Microsoft Teams)
  - Include the links. Always have wiki open whenever you google
  - Save all the datasheets, libraries you used (you need to specify the library source in your code clearly)
- Will help you in final documentation. It will also serve as a learning journal
- “Oh, I had seen it somewhere, can’t recall where” issues can be minimized

Thank you

PS: Check the Hardware AI under Week 6 (Individual Full Subsystem Test) for expected deliverables