FPGA Accelerated Deep Learning Radio Modulation Classification Using MATLAB System Objects & PYNQ

Andrew Maclellan, Lewis McLaughlin, Louise Crockett, Robert W. Stewart
Motivation

Impact of Deep Learning:

- Object recognition
- Face recognition
- Motion estimation
- Image segmentation
- ...
Motivation

Impact of Deep Learning:

- **Computer Vision**
  - Object recognition
  - Face recognition
  - Motion estimation
  - Image segmentation
  - ...

- **Natural Language Processing**
  - Language translation
  - Speech recognition
  - Text classification
  - Language modelling
  - ...

- **Deep Learning**
  - Spectrum sensing
  - Cognitive radio
  - Channel estimation
  - Modulation classification
  - LTE and 5G NR
  - ...

- **Communications**
Our Aim

Develop a workflow for training, quantising, simulating and implementing CNNs for communications on Zynq

Figure 1. Credit Mathworks - Zynq SDR Support from Communications Toolbox
Application - Automatic Modulation Classification

- Application for **Spectrum Sensing**
- Paper outlining this CNN structure by T. O’shea 2016
- Apply already proven structure and transfer it to hardware.
- Reduced modulation schemes to 2 for implementations simplicity

<table>
<thead>
<tr>
<th>Layer #</th>
<th>Layer type</th>
<th>Neurons</th>
<th>Activations</th>
<th>MACs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Input</td>
<td>2*128</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>Conv</td>
<td>64<em>1</em>3</td>
<td>ReLU</td>
<td>48384</td>
</tr>
<tr>
<td>3</td>
<td>Conv</td>
<td>16<em>2</em>3</td>
<td>ReLU</td>
<td>761856</td>
</tr>
<tr>
<td>4</td>
<td>Dense</td>
<td>128</td>
<td>ReLU</td>
<td>253952</td>
</tr>
<tr>
<td>5</td>
<td>Dense</td>
<td>2</td>
<td>Softmax</td>
<td>256</td>
</tr>
<tr>
<td>6</td>
<td>Output</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Quantised CNN

Floating-point weights

Fixed-point weights (2-bits)

GPU

- Avoid quantising from trained floating-point weights
  - Massive reduction in accuracy
- Altered our training process to train with quantisation limitations

FPGA

Example of kernel quantised training

```
Quantise Kernel/Weights
nb = 2 bits signed
```

```
| 0.34 | -0.98 | 0.48 |
| 0.04 | -0.56 | -0.12 |
```

```
| 0.5  | -1   | 0.5  |
| 0    | -0.5 | 0    |
```

Forward-pass

Straight Through Estimator

clip(w, -1, 0.5)

Backpropagation

Example of kernel quantised training
Quantised CNN

Classification Accuracy (%)

- Floating Point: 99.2%
- 2-Bit Quantised: 88.2%
- Trained 2-Bit Quantised: 97.2%
Preliminary Proposed System

**Programmable Logic**

- **Transmit Side IP**
  - Symbol Generation
  - Pulse Shaping
  - Interpolation Filter Chain

- **Receive Side IP**
  - Matched Filtering
  - Decimation Filter Chain

**RF Data Converter (DAC)**
- Interpolation
- NCO

**RF Data Converter (ADC)**
- Decimation
- NCO

**Convolutional Neural Network**
- Output 4
- Dense 1x4
- Dense 128
- Conv 16x2x3
- Conv 64x1x3
- IQ Input 2x128

**DDR**
- Filter Weights

**ARM PS**
- Configure (Jupyter)
- Visualise (Jupyter)

**PYNQ**
- Python productivity on Zynq
- Dynamically change modulation scheme
- Visualise the CNN decision making in real-time

**RFSoc**
- (Zynq UltraScale+ ZCU111 Evaluation Platform)

**Single chip transmit/receive**
Proposed Workflow

- Train quantised weights on DL frameworks
  - Tensorflow/Keras
  - PyTorch
  - MATLAB Deep Learning Toolbox
- Load weights into MATLAB System Objects
  - Configurable System Objects
  - Adjustable CNN dimensions
  - Simulate quantised network using Simulink
- Generate HDL with HDL Coder
  - Integrate with other MATLAB HDL IP
  - Generate HDL for both CNN & communications applications
- Integrate with PYNQ overlay in Vivado
  - Generate PYNQ bitstream for deployment
  - Interface with Jupyter notebook
Thank you!

Questions can be answered at the poster.

Feel free to come and discuss with us :)