Course Project Report

Hardware-software codesign

Group members: Praneet Thakur (B19CSE066) , Priyansh (B19CSE067)

Github link: https://github.com/priyansh2011/HW-SW-Co-Design

Contents:

We performed two tasks one is image classification and the other is text classification on the datasets, Cifar10 and IMBD movie reviews.

Image Classification - Cifar10

About dataset - The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.
Models used -
All the models used except Resnet, Inception, and Xception are lightweight models specifically made for devices like mobiles. Model 1 i.e. CNN is a simple Convolution Network heuristically designed. The model shuffleNet was implemented from a paper and is made for mobile devices. Link-https://arxiv.org/abs/1707.01083

1. CNN
2. Resnet
3. Inception
4. Xception
5. EfficientNetB0
6. EfficientNetB1
7. EfficientNetB2
8. MobileNetV1
9. MobileNetV2
10. DenseNet169
11. DenseNet121
12. nasNetMobile
13. shuffleNet

Text Classification - IMDB dataset

IMDB dataset having 50K movie reviews for natural language processing or Text analytics.
This is a dataset for binary sentiment classification containing substantially more data than previous benchmark datasets. It contains a set of 25,000 highly polar movie reviews for training and 25,000 for testing.

Models used -

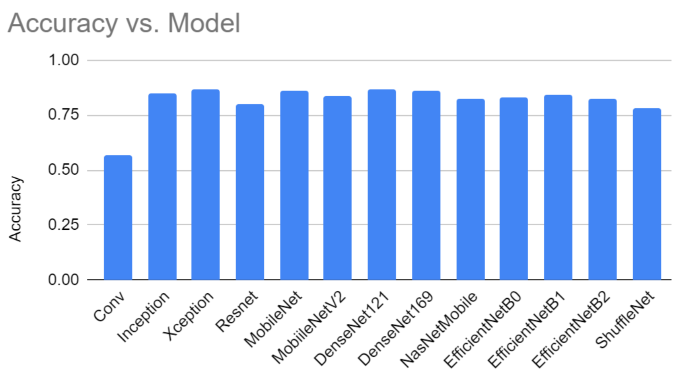
1. Transformer
2. Switch Transformer
3. FNet classifier
4. Bidirectional LSTM

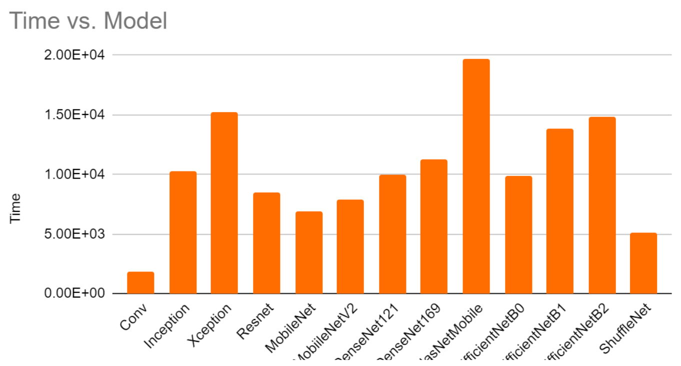
The Transformer and Switch Transformer models are known to be computationally intensive and require significant computing resources, while the FNet classifier and Bidirectional LSTM models are more computationally efficient.

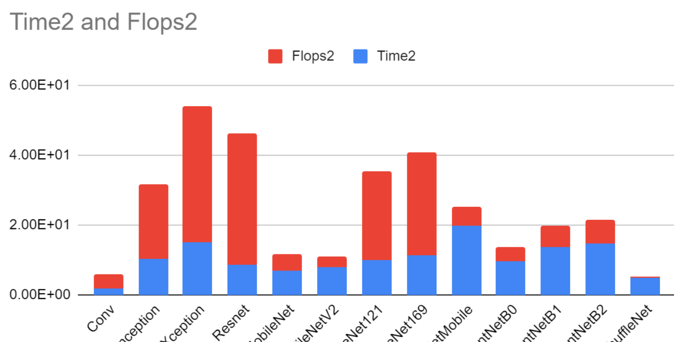
Results-

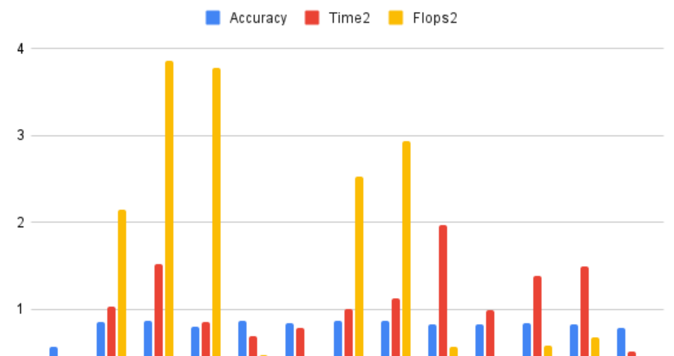
Model	Accuracy	Time(s)	Giga Flops
CNN	0.5693	1.79E+03	4.10E-02
Inception	0.8460	1.03E+04	2.14E-01
Xception	0.8645	1.53E+04	3.86E-01
Resnet	0.8004	8.51E+03	3.78E-01
MobileNet	0.8626	6.93E+03	4.67E-02
MobileNetV2	0.8364	7.88E+03	3.00E-02
DenseNet121	0.8658	9.97E+03	2.53E-01
DenseNet169	0.8633	1.13E+04	2.94E-01
NasNetMobile	0.8265	1.97E+04	5.60E-02
efficientNetB0	0.8295	9.84E+03	3.95E-02
efficientNetB1	0.8418	1.39E+04	5.84E-02
efficientNetB2	0.8232	1.49E+04	6.74E-02

Time2 and Flops2 are basically time/1000 and gflops*10 respectively



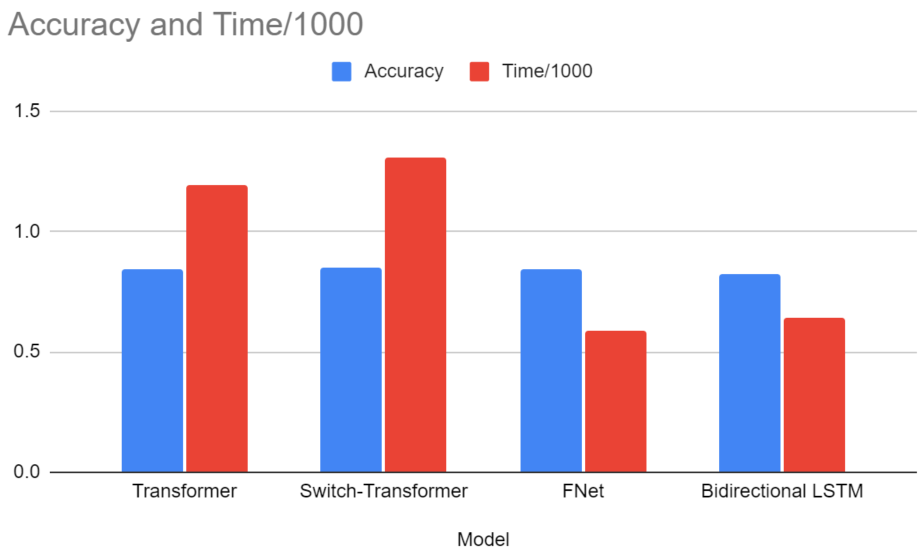






Text classification:

Model	Accuracy	Time
Transformer	0.8415	1193 s
Switch-Transformer	0.8514	1304 s
FNet	0.8408	586 s
Bidirectional LSTM	0.8244	644 s



PyNQ-Z2

We tried to use the PyNQ Z2 board to simulate our models of image and text classification using FPGA (Field Programmable Gate Arrays).

FPGAs (Field-Programmable Gate Arrays) have several advantages for implementing deep learning models, including:

1. High performance: FPGAs can provide high performance with low latency because they can be programmed to perform specific computations in parallel. This is particularly beneficial for deep learning models, which involve many matrix multiplications and other operations that can be parallelized.

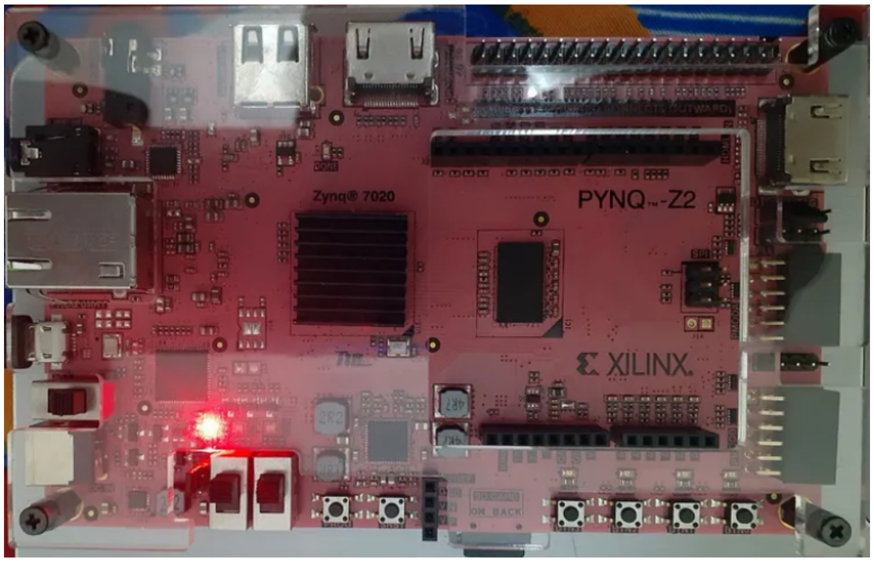
2. Flexibility: FPGAs can be reprogrammed to perform different computations, making them more flexible than application-specific integrated circuits (ASICs) that are designed for specific tasks. This allows for experimentation with different neural network architectures and the ability to adapt to changing computational needs.

3. Energy efficiency: FPGAs can perform computations with high energy efficiency because they only consume power when they are actively performing computations. This makes them a good choice for deep learning applications that require low power consumption.

4. Low latency: FPGAs have low latency because they can perform computations on the chip without the need for communication with other parts of the system. This can be especially beneficial for real-time applications like autonomous vehicles or robotics.

5. Cost-effective: FPGAs can be more cost-effective than other hardware options, such as GPUs or custom ASICs, because they are more readily available and require less specialized knowledge to program.

Overall, FPGAs offer a compelling combination of high performance, flexibility, energy efficiency, low latency, and cost-effectiveness that make them a promising option for deep learning applications.



We referred to the following link for setting up the PyNQ Z2 board:
https://medium.com/@ayushdixithere/configuring-internet-connection-and-downloading-packages-in-xilinx-pynq-z2-and-zcu-104-board-3e085dda30df

We completed the setup properly but we ended up running into some difficulties. We were having some trouble accessing the jupyter notebook sometimes. Also to run our models, we required using keras-tensorflow library. So we tried downloading the library on the PyNQ board using pip but it showed some error while downloading stating version mismatch. Then we tried to download TensorFlow using directly gz file using wget. But while using this we get the error: ERROR: tensorflow-1.1.0-cp34-cp34m-linux_armv6l.whl is not a supported wheel on this platform. And we tried this for other versions as well but the issue persists.
We tried manually downloading separate versions but ultimately we could not resolve the error. Therefore we weren't able to simulate our models using FPGA on PyNQ Z2 board.

After this exercise, we got the idea of how to use pynq board for python development. If only we could have downloaded TensorFlow, we would have been able to run the models. We can also run any ML and DL model and normal python usage on the board.

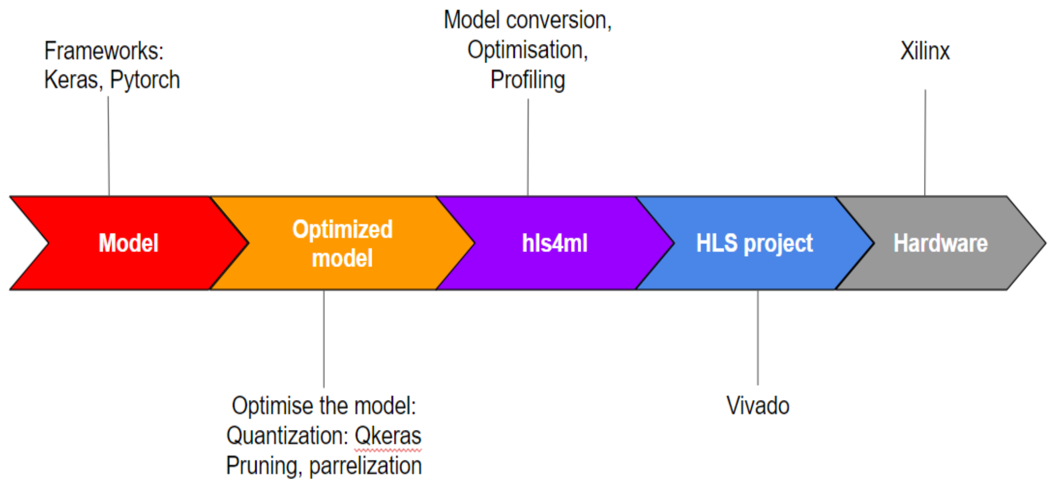
Hls4ml

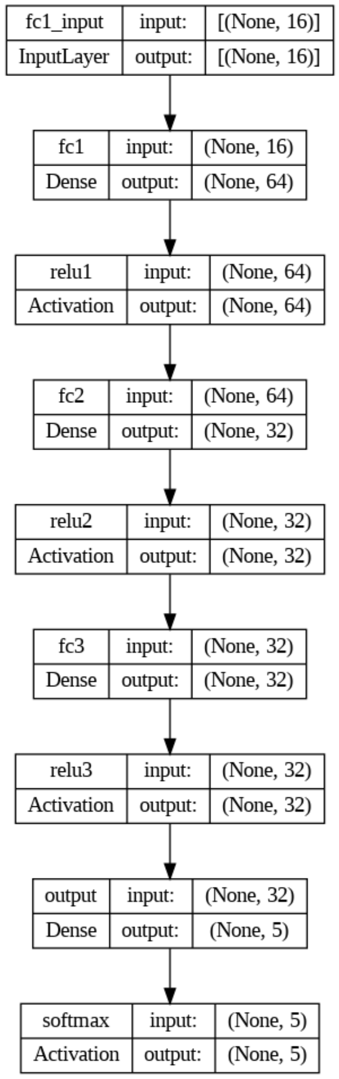
hls4ml is a package for translating neural networks to FPGA firmware for inference with extremely low latency on FPGAs

- https://github.com/hls-fpga-machine-learning/hls4ml

High Level Synthesis for machine learning

Workflow:


Frameworks: Keras, Pytorch
Model conversion, Optimisation, Profiling
Xilinx
Model
Optimized model
hls4ml
HLS project
Hardware
Optimise the model: Quantization: Qkeras Pruning, parrelization
Vivado


fc1_input | input: | [(None, 16)]
InputLayer | output: | [(None, 16)]
fc1 | input: | (None, 16)
Dense | output: | (None, 64)
relu1 | input: | (None, 64)
Activation | output: | (None, 64)
fc2 | input: | (None, 64)
Dense | output: | (None, 32)
relu2 | input: | (None, 32)
Activation | output: | (None, 32)
fc3 | input: | (None, 32)
Dense | output: | (None, 32)
relu3 | input: | (None, 32)
Activation | output: | (None, 32)
output | input: | (None, 32)
Dense | output: | (None, 5)
softmax | input: | (None, 5)
Activation | output: | (None, 5)

Dataset used: jet tagging dataset by openML
Description- Identify jets of particles from the LHC, created for the study of ultra low latency inference. Use 16 high level features to identify the 5 jet classes: quark (q), gluon (g), W boson (w), Z boson (z), or top quark (t).

Jet = collimated 'spray' of particles.

Our work:

We designed and synthesized a simple neural network for this. The structure is as follows. First, we synthesized this.

Then we further used various optimizations on this as the above image suggests:

The optimisations being:

1. **Quantization1:** We changed the precision of the weights of the first layer to 6 bits. Initially, it was ap_fixed<16,6>. We set the precision to 8 bits. We set 8 bits width with 2 integer bits i.e ap_fixed<8,2>. This model was

compiled and synthesised using hls4ml.

2. **Quantization1+Prallelisation:** Then we played around with parallelisation using the reuse factor set to now 2 instead of 1 for every layer. The reuse factor implements parallelization for multiplications. This was done on the same model above. This final model was also compiled and synthesised using hls4ml.

3. **Compression or Pruning:** Trained the keras model again but this time also used pruning as callbacks. We used Tensorflow model optimization sparsity to train a sparse model. The target sparsity was 75%. This model was also compiled and synthesised using hls4ml

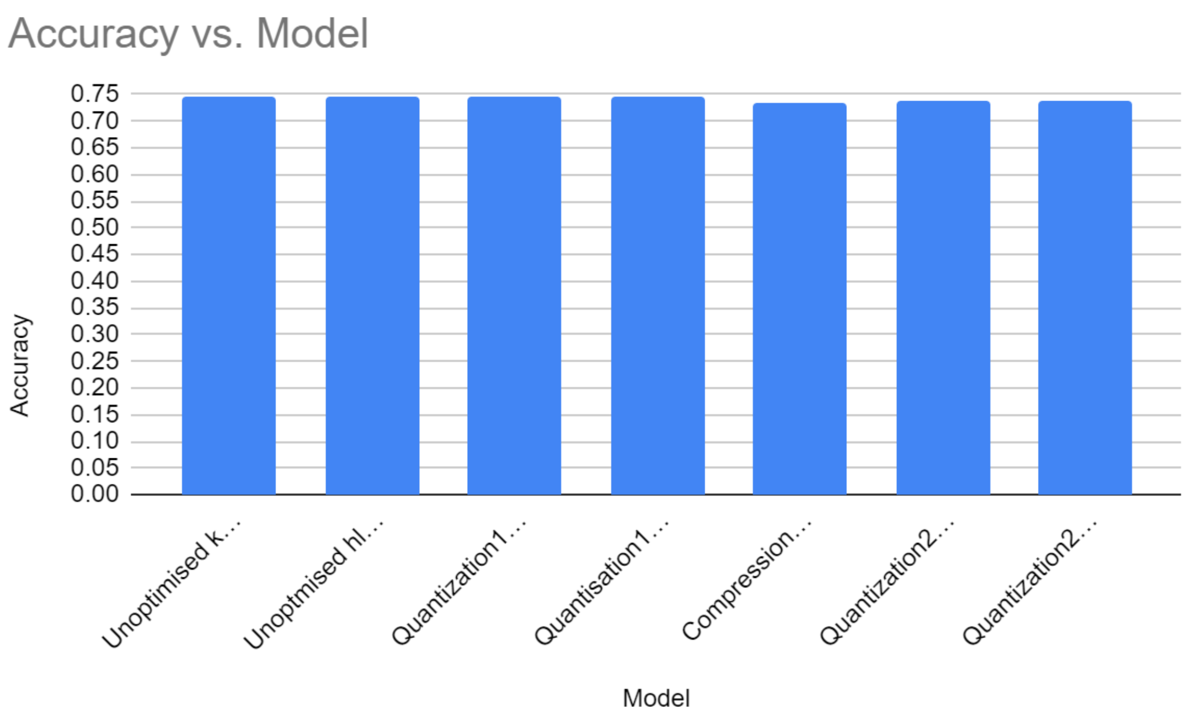
4. **Quantization2:** This time we used Qkeras. QKeras is "Quantized Keras" for deep heterogeneous quantization of ML models. Here instead of Dense, we use QDense, QActivation instead of Activation and so on. The precision for weights used was width 6 bits, with 0 integer bits. This model was trained, compiled, and synthesised.

The number of epochs used for all the models was 20. And the other hyperparameters were the same throughout all the models.

All the models were synthesized for the **device:xcu250-figd2104-2L-e, product family: virtexuplus**

Results:

Model	**Accuracy**
Unoptimised keras (Model-1)	0.7457650602409639
Unoptmised hls4ml (Model-1)	0.7456686746987952
Quantization1 hls4ml	0.7452650602409638
Quantisation1+Prallelisation hls4ml (Model-1.2)	0.7456686746987952
Compression or Pruning hls4ml (model-2)	0.7351265060240963
Quantization2 qkeras(Model-3)	0.7390903614457831
Quantization2 + pruning hls4ml (Model-3)	0.7391265060240964



Synthesis Results:

Model-1	
Performance Estimates	Utilization estimates

```
================================================================
== Performance Estimates
================================================================
+ Timing:
    * Summary:
    +--------+---------+----------+------------+
    |  Clock |  Target | Estimated| Uncertainty|
    +--------+---------+----------+------------+
    |ap_clk  | 5.00 ns | 4.371 ns |   0.62 ns  |
    +--------+---------+----------+------------+

+ Latency:
    * Summary:
    +---------+---------+-----------+-----------+-----+-----+----------+
    |  Latency (cycles) |   Latency (absolute)  |  Interval | Pipeline |
    |   min   |   max   |    min    |    max    | min | max |   Type   |
    +---------+---------+-----------+-----------+-----+-----+----------+
    |       10|       10| 50.000 ns | 50.000 ns |    1|    1| function |
    +---------+---------+-----------+-----------+-----+-----+----------+
```

```
================================================================
== Utilization Estimates
================================================================
* Summary:
+---------------------+---------+-------+---------+---------+------+
|         Name        | BRAM_18K| DSP48E|    FF   |   LUT   | URAM |
+---------------------+---------+-------+---------+---------+------+
|DSP                  |        -|      -|        -|        -|     -|
|Expression           |        -|      -|        0|        6|     -|
|FIFO                 |        -|      -|        -|        -|     -|
|Instance             |        4|   2904|    10541|   110305|     -|
|Memory               |        -|      -|        -|        -|     -|
|Multiplexer          |        -|      -|        -|       36|     -|
|Register             |        -|      -|     3341|        -|     -|
+---------------------+---------+-------+---------+---------+------+
|Total                |        4|   2904|    13882|   110347|     0|
+---------------------+---------+-------+---------+---------+------+
|Available SLR        |     1344|   3072|   864000|   432000|   320|
+---------------------+---------+-------+---------+---------+------+
|Utilization SLR (%)  |    ~0   |     94|        1|       25|     0|
+---------------------+---------+-------+---------+---------+------+
|Available            |     5376|  12288|  3456000|  1728000|  1280|
+---------------------+---------+-------+---------+---------+------+
|Utilization (%)      |    ~0   |     23|    ~0   |        6|     0|
+---------------------+---------+-------+---------+---------+------+
```

Model-1.2

Performance Estimates

```
================================================================
== Performance Estimates
================================================================
+ Timing:
    * Summary:
    +--------+---------+----------+------------+
    |  Clock |  Target | Estimated| Uncertainty|
    +--------+---------+----------+------------+
    |ap_clk  | 5.00 ns | 4.365 ns |   0.62 ns  |
    +--------+---------+----------+------------+

+ Latency:
    * Summary:
    +---------+---------+-----------+-----------+-----+-----+----------+
    |  Latency (cycles) |   Latency (absolute)  |  Interval | Pipeline |
    |   min   |   max   |    min    |    max    | min | max |   Type   |
    +---------+---------+-----------+-----------+-----+-----+----------+
    |       13|       13| 65.000 ns | 65.000 ns |    2|    2| function |
    +---------+---------+-----------+-----------+-----+-----+----------+
```

Utilization estimates

```
================================================================
== Utilization Estimates
================================================================
* Summary:
+---------------------+---------+-------+---------+---------+------+
|         Name        | BRAM_18K| DSP48E|    FF   |   LUT   | URAM |
+---------------------+---------+-------+---------+---------+------+
|DSP                  |        -|      -|        -|        -|     -|
|Expression           |        -|      -|        0|        6|     -|
|FIFO                 |        -|      -|        -|        -|     -|
|Instance             |        4|   2117|    15540|   122332|     -|
|Memory               |        -|      -|        -|        -|     -|
|Multiplexer          |        -|      -|        -|       60|     -|
|Register             |        -|      -|     3934|        -|     -|
+---------------------+---------+-------+---------+---------+------+
|Total                |        4|   2117|    19474|   122398|     0|
+---------------------+---------+-------+---------+---------+------+
|Available SLR        |     1344|   3072|   864000|   432000|   320|
+---------------------+---------+-------+---------+---------+------+
|Utilization SLR (%)  |    ~0   |     68|        2|       28|     0|
+---------------------+---------+-------+---------+---------+------+
|Available            |     5376|  12288|  3456000|  1728000|  1280|
+---------------------+---------+-------+---------+---------+------+
|Utilization (%)      |    ~0   |     17|    ~0   |        7|     0|
+---------------------+---------+-------+---------+---------+------+
```

Model-2

Performance Estimates

```
================================================================
== Performance Estimates
================================================================
+ Timing:
    * Summary:
    +--------+---------+----------+------------+
    |  Clock |  Target | Estimated| Uncertainty|
    +--------+---------+----------+------------+
    |ap_clk  | 5.00 ns | 3.965 ns |   0.62 ns  |
    +--------+---------+----------+------------+

+ Latency:
    * Summary:
    +---------+---------+-----------+-----------+-----+-----+----------+
    |  Latency (cycles) |   Latency (absolute)  |  Interval | Pipeline |
    |   min   |   max   |    min    |    max    | min | max |   Type   |
    +---------+---------+-----------+-----------+-----+-----+----------+
    |        9|        9| 45.000 ns | 45.000 ns |    1|    1| function |
    +---------+---------+-----------+-----------+-----+-----+----------+
```

Utilization estimates

```
================================================================
== Utilization Estimates
================================================================
* Summary:
+---------------------+---------+-------+---------+---------+------+
|         Name        | BRAM_18K| DSP48E|    FF   |   LUT   | URAM |
+---------------------+---------+-------+---------+---------+------+
|DSP                  |        -|      -|        -|        -|     -|
|Expression           |        -|      -|        0|        6|     -|
|FIFO                 |        -|      -|        -|        -|     -|
|Instance             |        4|    919|     2687|    29721|     -|
|Memory               |        -|      -|        -|        -|     -|
|Multiplexer          |        -|      -|        -|       36|     -|
|Register             |        -|      -|     3292|        -|     -|
+---------------------+---------+-------+---------+---------+------+
|Total                |        4|    919|     5979|    29763|     0|
+---------------------+---------+-------+---------+---------+------+
|Available SLR        |     1344|   3072|   864000|   432000|   320|
+---------------------+---------+-------+---------+---------+------+
|Utilization SLR (%)  |    ~0   |     29|    ~0   |        6|     0|
+---------------------+---------+-------+---------+---------+------+
|Available            |     5376|  12288|  3456000|  1728000|  1280|
+---------------------+---------+-------+---------+---------+------+
|Utilization (%)      |    ~0   |      7|    ~0   |        1|     0|
+---------------------+---------+-------+---------+---------+------+
```

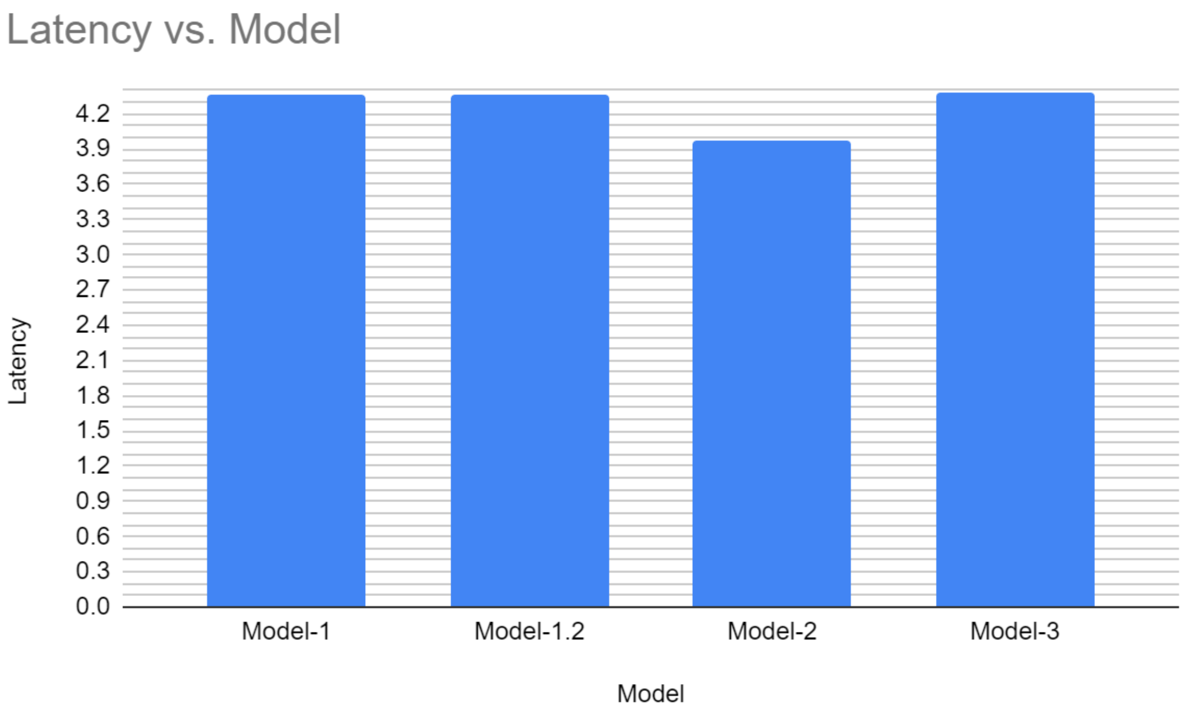
Model-3

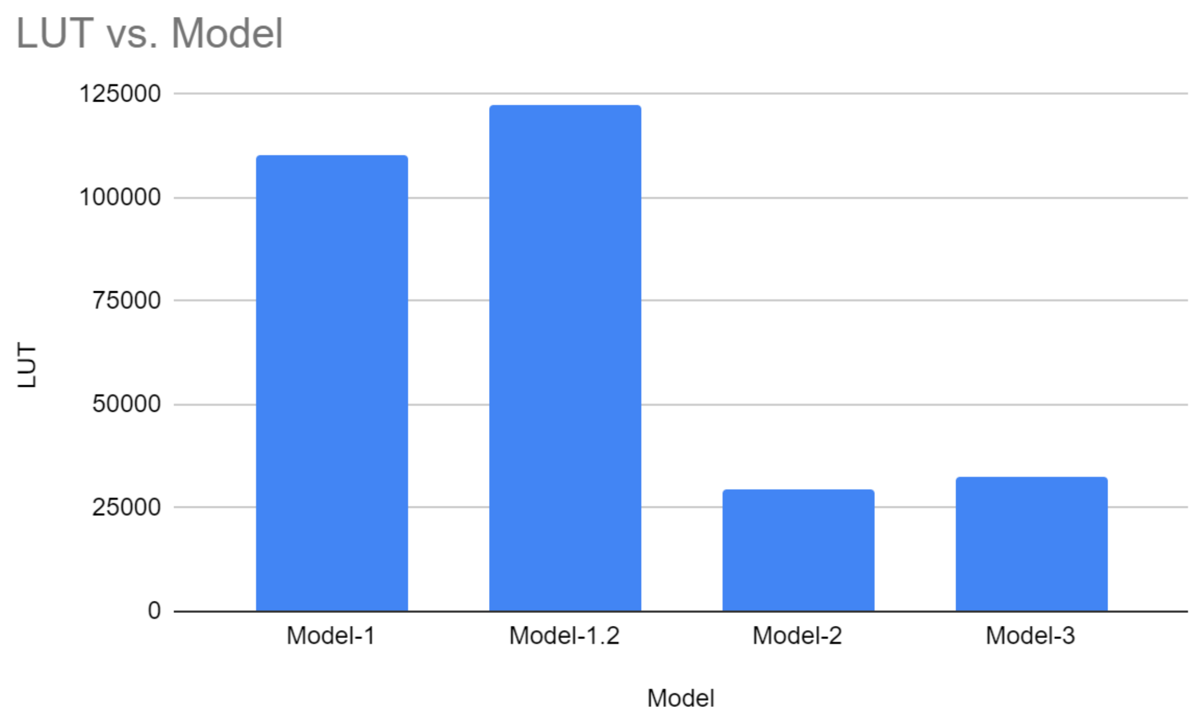
Performance Estimates

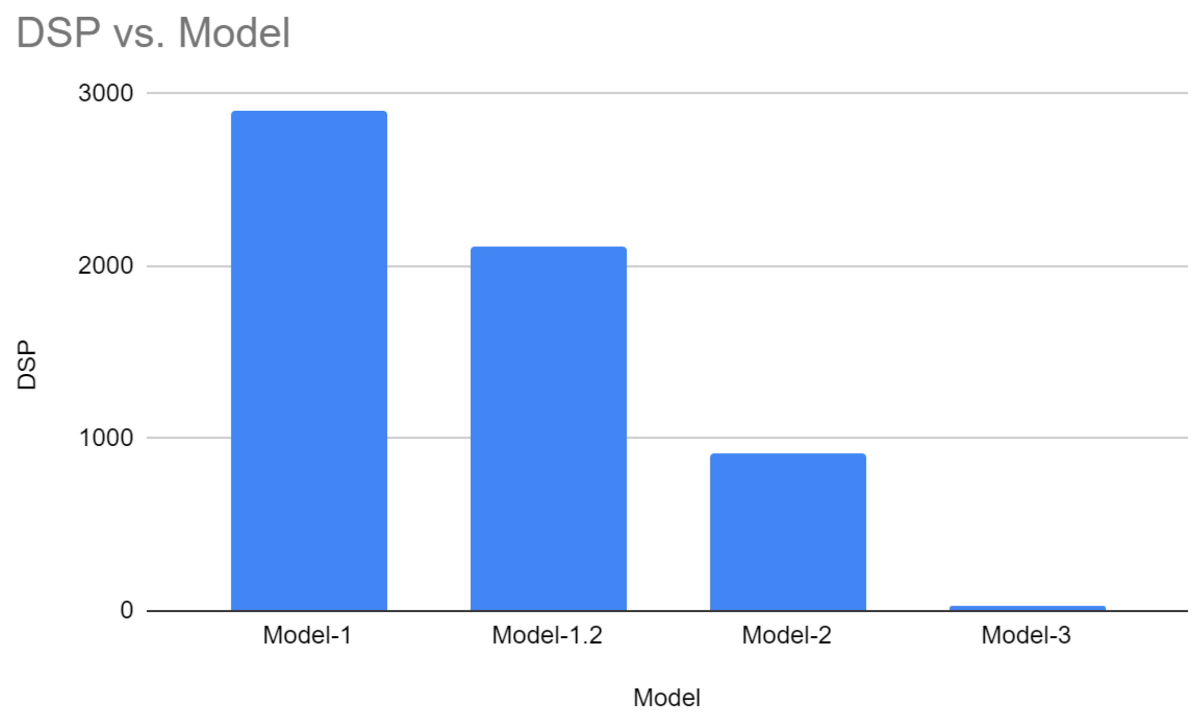
```
================================================================
== Performance Estimates
================================================================
+ Timing:
    * Summary:
    +--------+---------+----------+------------+
    |  Clock |  Target | Estimated| Uncertainty|
    +--------+---------+----------+------------+
    |ap_clk  | 5.00 ns | 4.372 ns |   0.62 ns  |
    +--------+---------+----------+------------+

+ Latency:
    * Summary:
    +---------+---------+-----------+-----------+-----+-----+----------+
    |  Latency (cycles) |   Latency (absolute)  |  Interval | Pipeline |
    |   min   |   max   |    min    |    max    | min | max |   Type   |
    +---------+---------+-----------+-----------+-----+-----+----------+
    |        8|        8| 40.000 ns | 40.000 ns |    1|    1| function |
    +---------+---------+-----------+-----------+-----+-----+----------+
```

Utilization estimates

```
================================================================
== Utilization Estimates
================================================================
* Summary:
+---------------------+---------+-------+---------+---------+------+
|         Name        | BRAM_18K| DSP48E|    FF   |   LUT   | URAM |
+---------------------+---------+-------+---------+---------+------+
|DSP                  |        -|      -|        -|        -|     -|
|Expression           |        -|      -|        0|        6|     -|
|FIFO                 |        -|      -|        -|        -|     -|
|Instance             |        4|     35|       93|    32265|     -|
|Memory               |        -|      -|        -|        -|     -|
|Multiplexer          |        -|      -|        -|       36|     -|
|Register             |        -|      -|     2827|        -|     -|
+---------------------+---------+-------+---------+---------+------+
|Total                |        4|     35|     2920|    32307|     0|
+---------------------+---------+-------+---------+---------+------+
|Available SLR        |     1344|   3072|   864000|   432000|   320|
+---------------------+---------+-------+---------+---------+------+
|Utilization SLR (%)  |    ~0   |      1|    ~0   |        7|     0|
+---------------------+---------+-------+---------+---------+------+
|Available            |     5376|  12288|  3456000|  1728000|  1280|
+---------------------+---------+-------+---------+---------+------+
|Utilization (%)      |    ~0   |    ~0 |    ~0   |        1|     0|
+---------------------+---------+-------+---------+---------+------+
```




LUT vs. Model
125000
100000
75000
50000
25000
0
LUT
Model-1
Model-1.2
Model-2
Model-3
Model


DSP vs. Model
3000
2000
1000
0
DSP
Model-1
Model-1.2
Model-2
Model-3
Model

Analysis:

All the results are as expected, DSP were to decrease in model 1.2 as we use reuse factor 2. Thus, more parallelization. Though the latency remained almost constant.

For model 1.2 as the precision is 6 bits, Vivado hls has moved multiplication operations from DSPs into LUTs, reducing the "critical" resource usage. Which is reflected.

As far as model-2 is concerned, here we used 75% sparsity and thus many weights became 0. Vivado utilizes this to further reduce DSPs. When Vivado HLS notices an operation like y = 0 * x it can avoid placing a DSP for that operation. This impact is the biggest.

This model was trained with low-precision quantization, and 75% pruning and 6 bit precision. We are able to see that we have saved a lot of resources compared to model-1. If we compare this model with model-2 we can see that Vivado hls has moved multiplication operations from DSPs into LUTs, reducing the "critical" resource usage.

Latency remained almost the same for all the models. But we can see the resource utilizations being optimized. But this optimization has reduced the accuracy of the model but not at a huge margin but a tiny margin which can be traded off with the resource performance improvements.

Steps to run hls4ml:

Follow these steps for smooth setup of the environment, the process is long and if using the process given we face a lot of errors.

Step1-In a linux based system do the following:

- `git clone https://github.com/hls-fpga-machine-learning/hls4ml-tutorial.git`
- `cd hls4ml-tutorial`
- `docker build --build-arg NB_USER=jovyan --build-arg NB_UID=1000 . -f`

`Step2- Check if we got the image by doing`

- `docker images`

Step3 - Get vivado from the EEL_7210_2023@10.9.1.79 server.

Step4- Now, we have our image ready. We can create a container by doing the following:

- `docker run -it -t -v $PWD:/<directory in which Vivado is saved>:/workspace <image_id> bash`

Step5 - At this point we would be inside the container. Then add the vivado to the path. Assuming you downloaded 2019.2 version. Add this to the path:/Vivado/2019.2/bin.

- `export PATH=$PATH:`/Vivado/2019.2/bin:

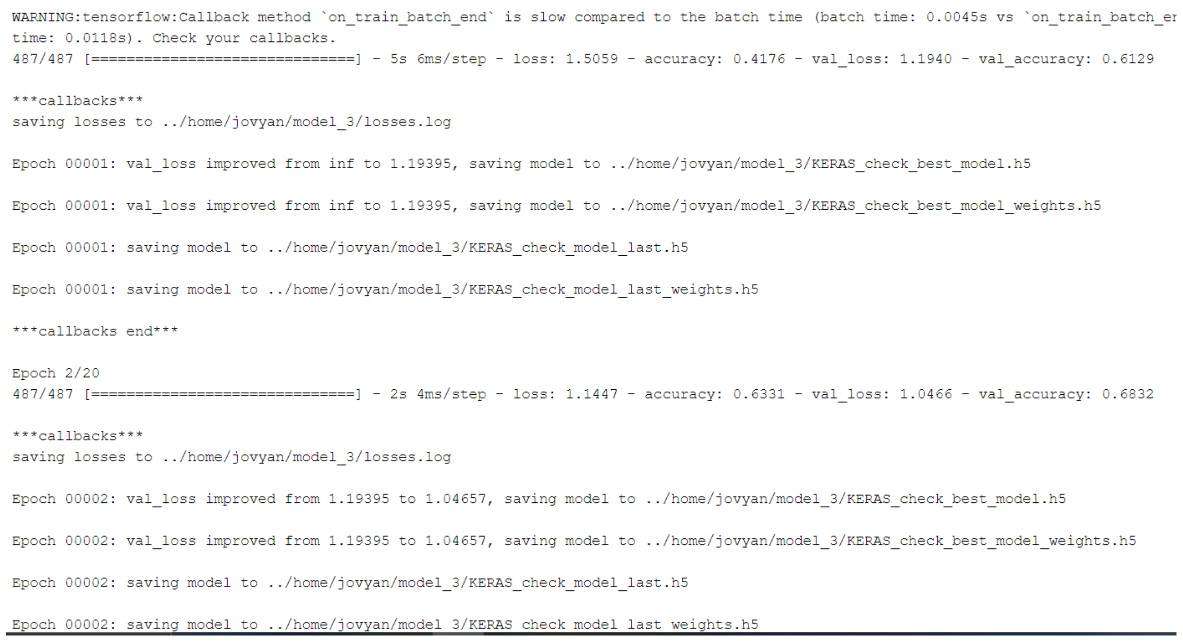
Step6 - Then we have our environment ready, we can now use hls4ml as well as synthesise the models.

Step7 - Run the files from workspace directory. Make sure inside the code files, you save all the generated files from code in ../home/jovyan directory.

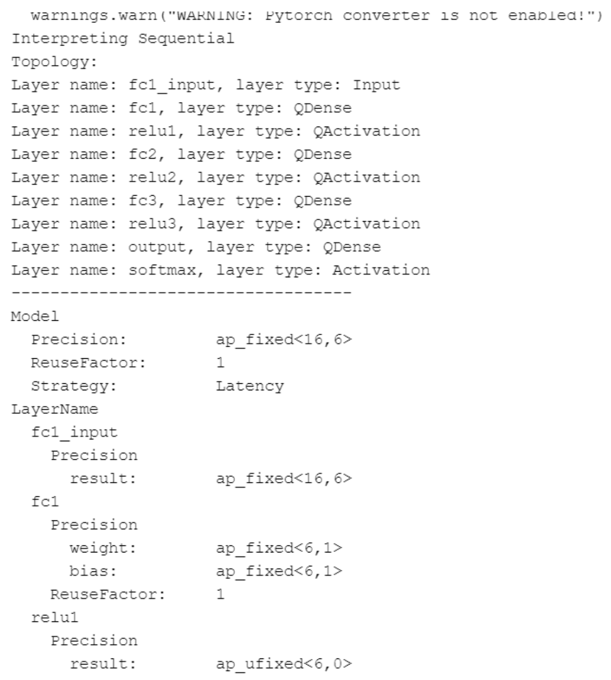
To run code do: `python <code file name>.py`

One thing to note is that we would have issues when trying to synthesize Convolution Neural Network codes.

Some logs screenshots:



```
WARNING:tensorflow:Callback method `on_train_batch_end` is slow compared to the batch time (batch time: 0.0045s vs `on_train_batch_er
time: 0.0118s). Check your callbacks.
487/487 [==============================] - 5s 6ms/step - loss: 1.5059 - accuracy: 0.4176 - val_loss: 1.1940 - val_accuracy: 0.6129

***callbacks***
saving losses to ../home/jovyan/model_3/losses.log

Epoch 00001: val_loss improved from inf to 1.19395, saving model to ../home/jovyan/model_3/KERAS_check_best_model.h5

Epoch 00001: val_loss improved from inf to 1.19395, saving model to ../home/jovyan/model_3/KERAS_check_best_model_weights.h5

Epoch 00001: saving model to ../home/jovyan/model_3/KERAS_check_model_last.h5

Epoch 00001: saving model to ../home/jovyan/model_3/KERAS_check_model_last_weights.h5

***callbacks end***

Epoch 2/20
487/487 [==============================] - 2s 4ms/step - loss: 1.1447 - accuracy: 0.6331 - val_loss: 1.0466 - val_accuracy: 0.6832

***callbacks***
saving losses to ../home/jovyan/model_3/losses.log

Epoch 00002: val_loss improved from 1.19395 to 1.04657, saving model to ../home/jovyan/model_3/KERAS_check_best_model.h5

Epoch 00002: val_loss improved from 1.19395 to 1.04657, saving model to ../home/jovyan/model_3/KERAS_check_best_model_weights.h5

Epoch 00002: saving model to ../home/jovyan/model_3/KERAS_check_model_last.h5

Epoch 00002: saving model to ../home/jovyan/model_3/KERAS_check_model_last_weights.h5
```



```
  warnings.warn("WARNING: Pytorch converter is not enabled!")
Interpreting Sequential
Topology:
Layer name: fc1_input, layer type: Input
Layer name: fc1, layer type: QDense
Layer name: relu1, layer type: QActivation
Layer name: fc2, layer type: QDense
Layer name: relu2, layer type: QActivation
Layer name: fc3, layer type: QDense
Layer name: relu3, layer type: QActivation
Layer name: output, layer type: QDense
Layer name: softmax, layer type: Activation
-----------------------------------
Model
  Precision:         ap_fixed<16,6>
  ReuseFactor:       1
  Strategy:          Latency
LayerName
  fc1_input
    Precision
      result:        ap_fixed<16,6>
  fc1
    Precision
      weight:        ap_fixed<6,1>
      bias:          ap_fixed<6,1>
    ReuseFactor:     1
  relu1
    Precision
      result:        ap_ufixed<6,0>
```

```
WARNING: [SYN 201-103] Legalizing function name 'linear<ap_fixed,ap_fixed<16,6,0,0,0>,linear_config9>' to
'linear_ap_fixed_ap_fixed_16_6_0_0_0_linear_config9_s'.
WARNING: [SYN 201-103] Legalizing function name 'relu<ap_fixed,ap_ufixed<6,0,0,0,0>,relu_config10>' to
'relu_ap_fixed_ap_ufixed_6_0_0_0_0_relu_config10_s'.
WARNING: [SYN 201-103] Legalizing function name 'dense_latency<ap_ufixed,ap_fixed,config11>.0.0.0.0.0.0.0' to
'dense_latency_ap_ufixed_ap_fixed_config11_0_0_0_0_0_0'.
WARNING: [SYN 201-103] Legalizing function name 'linear<ap_fixed,ap_fixed<16,6,0,0,0>,linear_config12>' to
'linear_ap_fixed_ap_fixed_16_6_0_0_0_linear_config12_s'.
WARNING: [SYN 201-103] Legalizing function name 'softmax_latency<ap_fixed,ap_fixed,softmax_config13>' to
'softmax_latency_ap_fixed_ap_fixed_softmax_config13_s'.
INFO: [HLS 200-10] ----------------------------------------------------------------
INFO: [HLS 200-42] -- Implementing module 'dense_latency_0_0_0_0_0_0_0_0_0_0_0_0_0_0_0_0_0_0_0_0_0_0_0_2'
INFO: [HLS 200-10] ----------------------------------------------------------------
INFO: [SCHED 204-11] Starting scheduling ...
INFO: [SCHED 204-61] Pipelining function 'dense_latency.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0..2'.
INFO: [SCHED 204-61] Pipelining result : Target II = 1, Final II = 1, Depth = 1.
INFO: [SCHED 204-11] Finished scheduling.
INFO: [HLS 200-111]  Elapsed time: 338.26 seconds; current allocated memory: 505.795 MB.
INFO: [BIND 205-100] Starting micro-architecture generation ...
INFO: [BIND 205-101] Performing variable lifetime analysis.
INFO: [BIND 205-101] Exploring resource sharing.
INFO: [BIND 205-101] Binding ...
INFO: [BIND 205-100] Starting global binding ...
INFO: [BIND 205-100] Finished micro-architecture generation.
INFO: [HLS 200-111]  Elapsed time: 1.2 seconds; current allocated memory: 510.050 MB.
INFO: [HLS 200-10] ----------------------------------------------------------------
INFO: [HLS 200-42] -- Implementing module 'linear_ap_fixed_ap_fixed_16_6_0_0_0_linear_config3_s'
INFO: [HLS 200-10] ----------------------------------------------------------------
```

Contributions:

1. Image classification - Priyansh (B19CSE067)
2. Text Classification - Praneet Thakur (B19CSE066)
3. Pynq-z2 - Praneet Thakur (B19CSE066)
4. Hls4ml - Priyansh (B19CSE067)