Design credit Report

Title: Deep Learning Executions for Mobile Devices

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

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 -

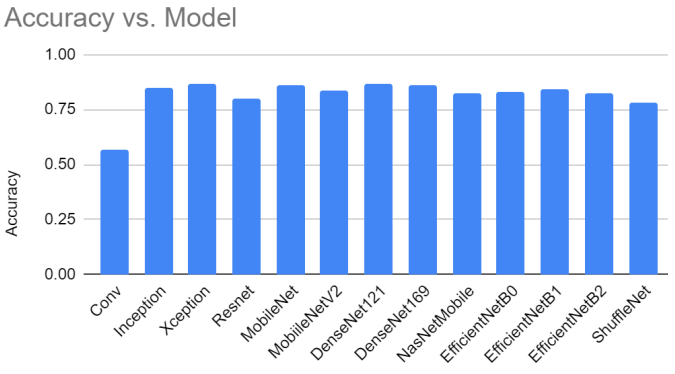
1. Transformer
2. Switch Transformer
3. FNet classifier
4. Bidirectional LSTM

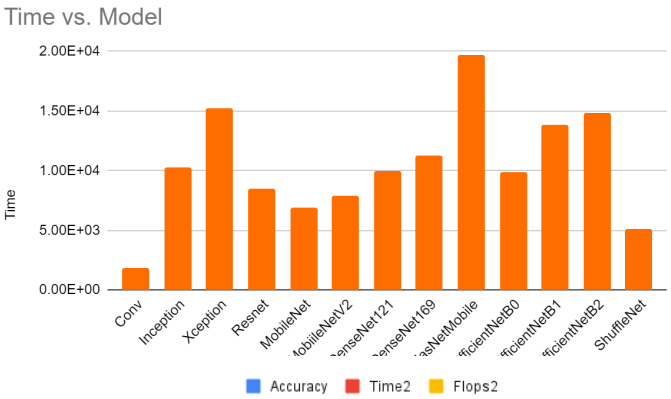
The Transformer and Switch Transformer models are known to be computationally intensive and require significant computing resources, while the FNet classifier and Bidirectional LSTM models are more computationally efficient.

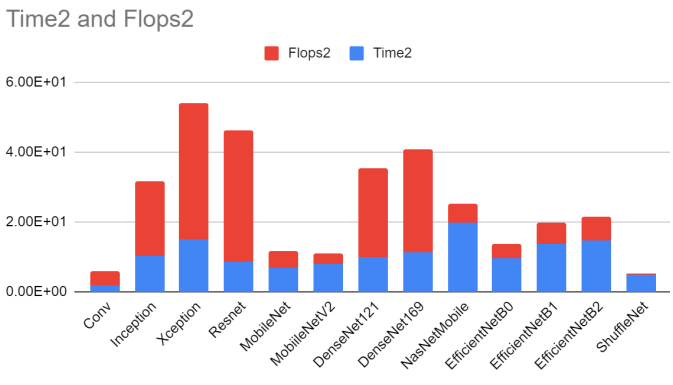
Results-

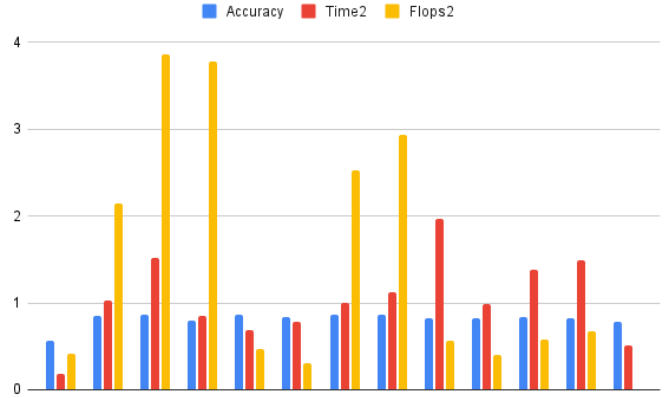
Model	Accuracy	Time(s)	Giga Flops
CNN	0.5693	1.79E+03	4.10E-02
Inception	0.8460	1.03E+04	2.14E-01
Xception	0.8645	1.53E+04	3.86E-01
Resnet	0.8004	8.51E+03	3.78E-01
MobileNet	0.8626	6.93E+03	4.67E-02
MobileNetV2	0.8364	7.88E+03	3.00E-02
DenseNet121	0.8658	9.97E+03	2.53E-01
DenseNet169	0.8633	1.13E+04	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



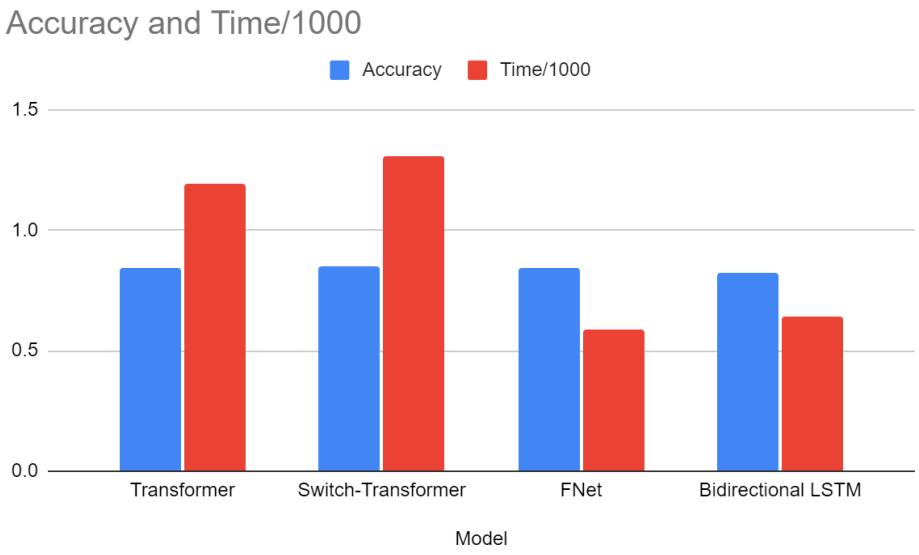






Text classification:

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.

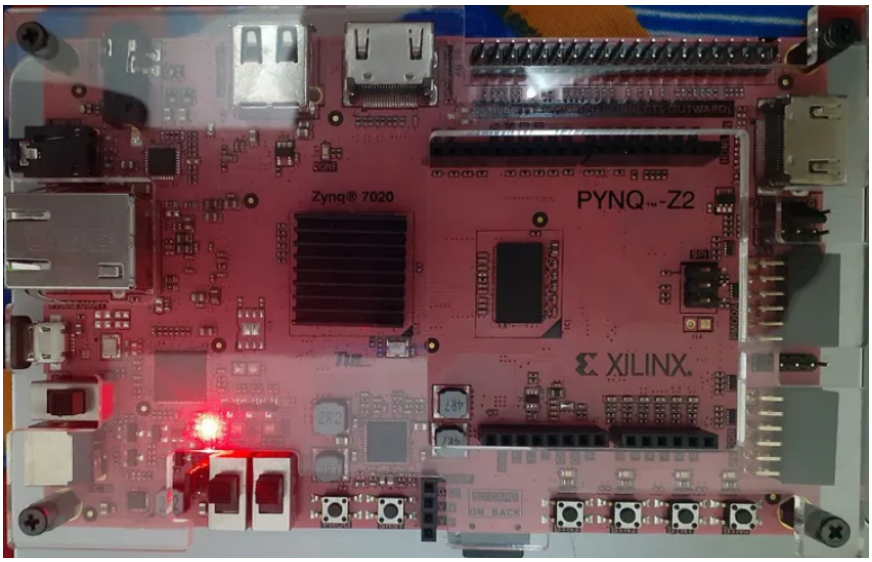
2. Flexibility: FPGAs can be reprogrammed to perform different computations, making them more flexible than application-specific integrated circuits (ASICs) that are designed for specific tasks. This allows for experimentation with different neural network architectures and the ability to adapt to changing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



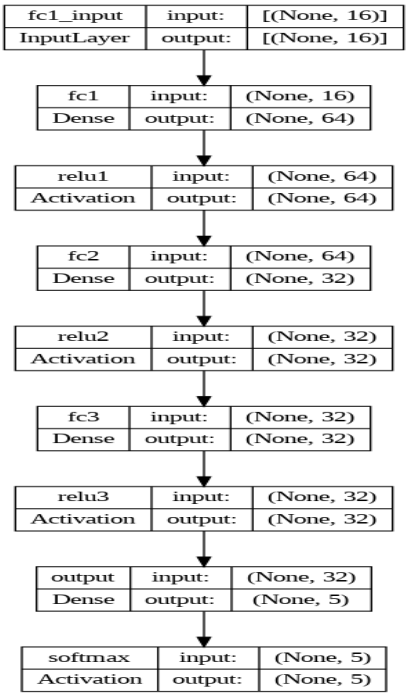
We referred to the following link for setting up the PyNQ Z2 board:
https://medium.com/@ayushdixithere/configuring-internet-connection-and-downloading-packages-in-xilinx-pynq-z2-and-zcu-104-board-3e085dda30df

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

We also did some work using hls4ml(Link-https://github.com/fastmachinelearning/hls4ml). Using this we synthesised a simple neural network trained on jet tagging data from OpenML. The structure of the model used is as follows:



The model was compiled and run with both keras and hls4ml. The accuracy results for both of them are as follows:

Keras Accuracy: 0.687289156626506
hls4ml Accuracy: 0.6872048192771084

As we can see the reduction is not much, but to further optimise for a light device we can first quantize the model and then synthesize using hls4ml.

```
---------------------------------
Configuration
Model
  Precision:         ap_fixed<16,6>
  ReuseFactor:       1
  Strategy:          Latency
---------------------------------
Interpreting Sequential
Topology:
Layer name: fc1_input, layer type: InputLayer, input shapes: [[None, 16]], output shape: [None, 16]
Layer name: fc1, layer type: Dense, input shapes: [[None, 16]], output shape: [None, 64]
Layer name: relu1, layer type: Activation, input shapes: [[None, 64]], output shape: [None, 64]
Layer name: fc2, layer type: Dense, input shapes: [[None, 64]], output shape: [None, 32]
Layer name: relu2, layer type: Activation, input shapes: [[None, 32]], output shape: [None, 32]
Layer name: fc3, layer type: Dense, input shapes: [[None, 32]], output shape: [None, 32]
Layer name: relu3, layer type: Activation, input shapes: [[None, 32]], output shape: [None, 32]
Layer name: output, layer type: Dense, input shapes: [[None, 32]], output shape: [None, 5]
Layer name: softmax, layer type: Softmax, input shapes: [[None, 5]], output shape: [None, 5]
Creating HLS model
plotting the hls4ml model
Compiling and predicting the hls4ml model
Writing HLS project
Done
Comparing both the models
Keras  Accuracy: 0.687289156626506
hls4ml Accuracy: 0.6872048192771084
Synthesing...
```

We then synthesized the model using hls4ml.
Some of the logs are:

```
Synthesing...

****** Vivado(TM) HLS - High-Level Synthesis from C, C++ and SystemC v2019.2 (64-bit)
  **** SW Build 2708876 on Wed Nov  6 21:39:14 MST 2019
  **** IP Build 2700528 on Thu Nov  7 00:09:20 MST 2019
    ** Copyright 1986-2019 Xilinx, Inc. All Rights Reserved.

source /workspace/Vivado/2019.2/scripts/vivado_hls/hls.tcl -notrace
INFO: [HLS 200-10] Running '/workspace/Vivado/2019.2/bin/unwrapped/lnx64.o/vivado_hls'
INFO: [HLS 200-10] For user 'jovyan' on host 'f5c10941e709' (Linux_x86_64 version 5.4.0-52-generic) on Sun Apr 23 10:12:11
INFO: [HLS 200-10] In directory '/home/jovyan/model_1/hls4ml_prj'
Sourcing Tcl script 'build_prj.tcl'
INFO: [HLS 200-10] Creating and opening project '/home/jovyan/model_1/hls4ml_prj/myproject_prj'.
INFO: [HLS 200-10] Adding design file 'firmware/myproject.cpp' to the project
INFO: [HLS 200-10] Adding test bench file 'myproject_test.cpp' to the project
INFO: [HLS 200-10] Adding test bench file 'firmware/weights' to the project
INFO: [HLS 200-10] Adding test bench file 'tb_data' to the project
INFO: [HLS 200-10] Creating and opening solution '/home/jovyan/model_1/hls4ml_prj/myproject_prj/solution1'.
INFO: [XFORM 203-101] Allowed max sub elements number after partition is 4096.
INFO: [XFORM 203-1161] The maximum of name length is set into 60.
INFO: [HLS 200-10] Setting target device to 'xcu250-figd2104-2L-e'
INFO: [SYN 201-201] Setting up clock 'default' with a period of 5ns.
***** C/RTL SYNTHESIS *****
INFO: [SCHED 204-61] Option 'relax_ii_for_timing' is enabled, will increase II to preserve clock frequency constraints.
INFO: [HLS 200-10] Analyzing design file 'firmware/myproject.cpp' ...
```

```
.
INFO: [XFORM 203-622] Instantiating function 'nnet::dense_latency<ap_fixed<16, 6, (ap_q_mode)5, (ap_o_mode)3, 0>, ap_fixed<16, 6, (ap_q_mode)5,
(ap_o_mode)3, 0>, config8>.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0'(firmware/nnet_utils/nnet_dense_latency.h:45:23) to 'nnet::de
nse_latency<ap_fixed<16, 6, (ap_q_mode)5, (ap_o_mode)3, 0>, ap_fixed<16, 6, (ap_q_mode)5, (ap_o_mode)3, 0>, config8>.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0
.0.0.0.0.0.0.0.0.0.0.0.0.0.0' at call site (firmware/nnet_utils/nnet_dense.h:46->firmware/myproject.cpp:80) by setting 'biases[27].V' to 'b8.V
.27'.
INFO: [XFORM 203-622] Instantiating function 'nnet::dense_latency<ap_fixed<16, 6, (ap_q_mode)5, (ap_o_mode)3, 0>, ap_fixed<16, 6, (ap_q_mode)5,
(ap_o_mode)3, 0>, config8>.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0'(firmware/nnet_utils/nnet_dense_latency.h:45:23) to 'nnet::
dense_latency<ap_fixed<16, 6, (ap_q_mode)5, (ap_o_mode)3, 0>, ap_fixed<16, 6, (ap_q_mode)5, (ap_o_mode)3, 0>, config8>.0.0.0.0.0.0.0.0.0.0.0.0.0.0
.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0' at call site (firmware/nnet_utils/nnet_dense.h:46->firmware/myproject.cpp:80) by setting 'biases[28].V' to '
b8.V.28'.
INFO: [XFORM 203-622] Instantiating function 'nnet::dense_latency<ap_fixed<16, 6, (ap_q_mode)5, (ap_o_mode)3, 0>, ap_fixed<16, 6, (ap_q_mode)5,
(ap_o_mode)3, 0>, config8>.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0'(firmware/nnet_utils/nnet_dense_latency.h:45:23) to 'nnet
::dense_latency<ap_fixed<16, 6, (ap_q_mode)5, (ap_o_mode)3, 0>, ap_fixed<16, 6, (ap_q_mode)5, (ap_o_mode)3, 0>, config8>.0.0.0.0.0.0.0.0.0.0.0.0.0
.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0' at call site (firmware/nnet_utils/nnet_dense.h:46->firmware/myproject.cpp:80) by setting 'biases[29].V'
to 'b8.V.29'.
INFO: [XFORM 203-622] Instantiating function 'nnet::dense_latency<ap_fixed<16, 6, (ap_q_mode)5, (ap_o_mode)3, 0>, ap_fixed<16, 6, (ap_q_mode)5,
(ap_o_mode)3, 0>, config8>.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0'(firmware/nnet_utils/nnet_dense_latency.h:45:23) to 'nn
et::dense_latency<ap_fixed<16, 6, (ap_q_mode)5, (ap_o_mode)3, 0>, ap_fixed<16, 6, (ap_q_mode)5, (ap_o_mode)3, 0>, config8>.0.0.0.0.0.0.0.0.0.0.0.0
.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0' at call site (firmware/nnet_utils/nnet_dense.h:46->firmware/myproject.cpp:80) by setting 'biases[30]
.V' to 'b8.V.30'.
```

```
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_0_2'
INFO: [HLS 200-10] ----------------------------------------------------------------
INFO: [SCHED 204-11] Starting scheduling ...
INFO: [SCHED 204-61] Pipelining function 'dense_latency.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0..2'.
INFO: [SCHED 204-61] Pipelining result : Target II = 1, Final II = 1, Depth = 2.
INFO: [SCHED 204-11] Finished scheduling.
INFO: [HLS 200-111]  Elapsed time: 426.97 seconds; current allocated memory: 553.059 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: 12.69 seconds; current allocated memory: 625.125 MB.
INFO: [HLS 200-10] ----------------------------------------------------------------
INFO: [HLS 200-42] -- Implementing module 'relu_ap_fixed_ap_fixed_16_6_5_3_0_relu_config4_s'
INFO: [HLS 200-10] ----------------------------------------------------------------
INFO: [SCHED 204-11] Starting scheduling ...
INFO: [SCHED 204-61] Pipelining function 'relu<ap_fixed,ap_fixed<16,6,5,3,0>,relu_config4>'.
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: 7.59 seconds; current allocated memory: 628.644 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] Finished micro-architecture generation.
INFO: [HLS 200-111]  Elapsed time: 0.59 seconds; current allocated memory: 630.154 MB.
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_0_1'
INFO: [HLS 200-10] ----------------------------------------------------------------
INFO: [SCHED 204-11] Starting scheduling ...
INFO: [SCHED 204-61] Pipelining function 'dense_latency.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0..1'.
```

After synthesis we generated the performance estimates again using hls4ml which are as follows:

```
================================================================
== Vivado HLS Report for 'myproject'
================================================================
* Date:           Sun Apr 23 10:23:26 2023

* Version:        2019.2 (Build 2704478 on Wed Nov 06 22:10:23 MST 2019)
* Project:        myproject_prj
* Solution:       solution1
* Product family: virtexuplus
* Target device:  xcu250-figd2104-2L-e


================================================================
== Performance Estimates
================================================================
+ Timing:
    * Summary:
    +--------+---------+----------+------------+
    |  Clock |  Target | Estimated| Uncertainty|
    +--------+---------+----------+------------+
    |ap_clk  | 5.00 ns | 3.985 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|   3237|    11671|   112525|     -|
|Memory               |        -|      -|        -|        -|     -|
|Multiplexer          |        -|      -|        -|       36|     -|
|Register             |        -|      -|     3341|        -|     -|
+---------------------+---------+-------+---------+---------+------+
|Total                |        4|   3237|    15012|   112567|     0|
+---------------------+---------+-------+---------+---------+------+
|Available SLR        |     1344|   3072|   864000|   432000|   320|
+---------------------+---------+-------+---------+---------+------+
|Utilization SLR (%)  |    ~0   |    105|        1|       26|     0|
+---------------------+---------+-------+---------+---------+------+
|Available            |     5376|  12288|  3456000|  1728000|  1280|
+---------------------+---------+-------+---------+---------+------+
|Utilization (%)      |    ~0   |     26|    ~0   |        6|     0|
+---------------------+---------+-------+---------+---------+------+
```

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 - Get vivado from the EEL_7210_2023@10.9.1.79 server.

Step3 - Then add the vivado to the path. Assuming you downloaded 2019.2 version. Add this to the path:/Vivado/2019.2/bin

Step4 - Then we have our environment ready, we can now use hls4ml as well as synthesise the model.

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