Deep Neural Network On FPGA

Final Presentation

Students:

Amit Shtober

Alon Nemirovsky

Supervisors:

Ina Rivkin

Oz Shmueli

Winter 2020-2021

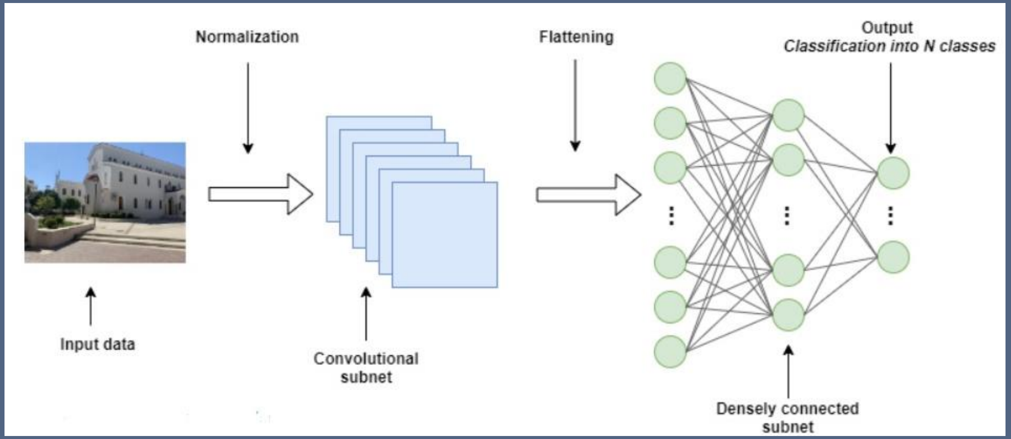
Project Goals

Evaluate and validate the Xilinx Vitis-AI ecosystem :

1. Understanding Vitis-AI ecosystem
2. Running full flow from TensorFlow to the FPGA
   - Step #1 – Using trained model from Xilinx Zoo (resnet50)
   - Step #2 – Using untrained model from the Zoo (MNIST), Modify it and run the flow in order to fully understand the ecosystem ability
3. Evaluate and modify, when possible, each step in the process of the ecosystem
4. Creating a tutorial for running the flows step by step and indicate problems we have encountered in the process

Deep Neural Network

- Creation of Neural Network was inspired by the human brain and the way it functions
- Works according to the algorithm and can predict a solution based on experience
- The system has process layers that indicates on the depth of the Neural Network
- This method is very popular and highly used today



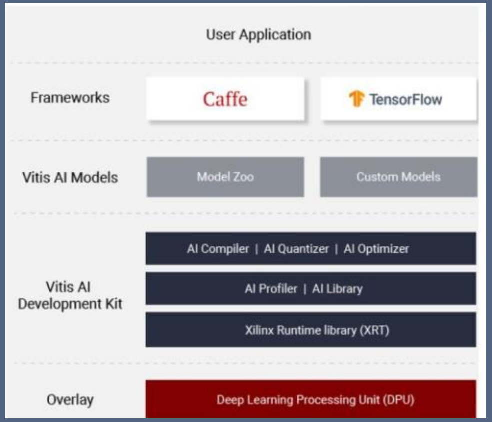
Creating The Network

Full process of creating working neural network Includes:

- Creating the model
- Train it using CPU/GPU
- Optimize the model
- Inference which can be done on various machines such as host using GPU or dedicated Hardware
- Create and manufacture the dedicated hardware which will run the Inference stage

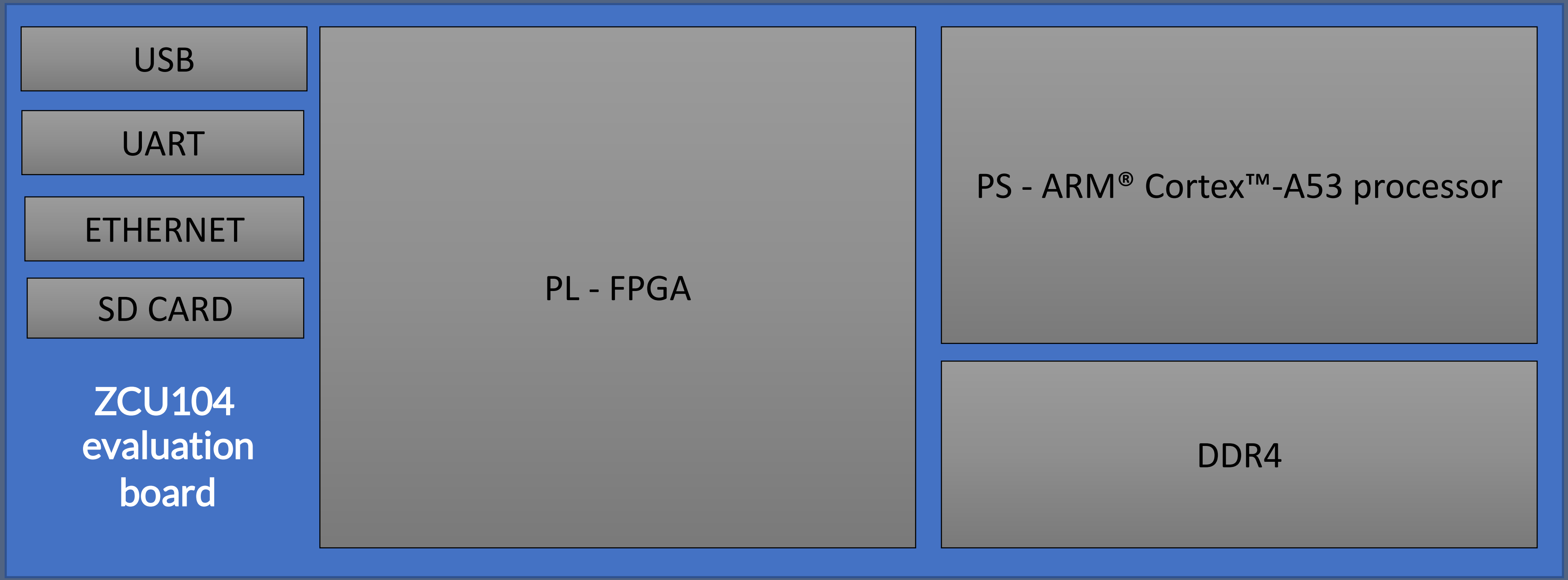
Vitis-AI Ecosystem

- Xilinx offers a full ecosystem which helps deal with those problems. It includes:
  - Dedicated IP which called ***DPU***
  - **AI Zoo** - Repository which includes optimized deep learning models. Great for quick start
  - **Quantizer -** Takes a floating-point model and performs pre-processing and then quantizes the weights/biases and activations to the given bit width
  - **Compiler -** Maps a network model into a highly optimized DPU instruction sequence

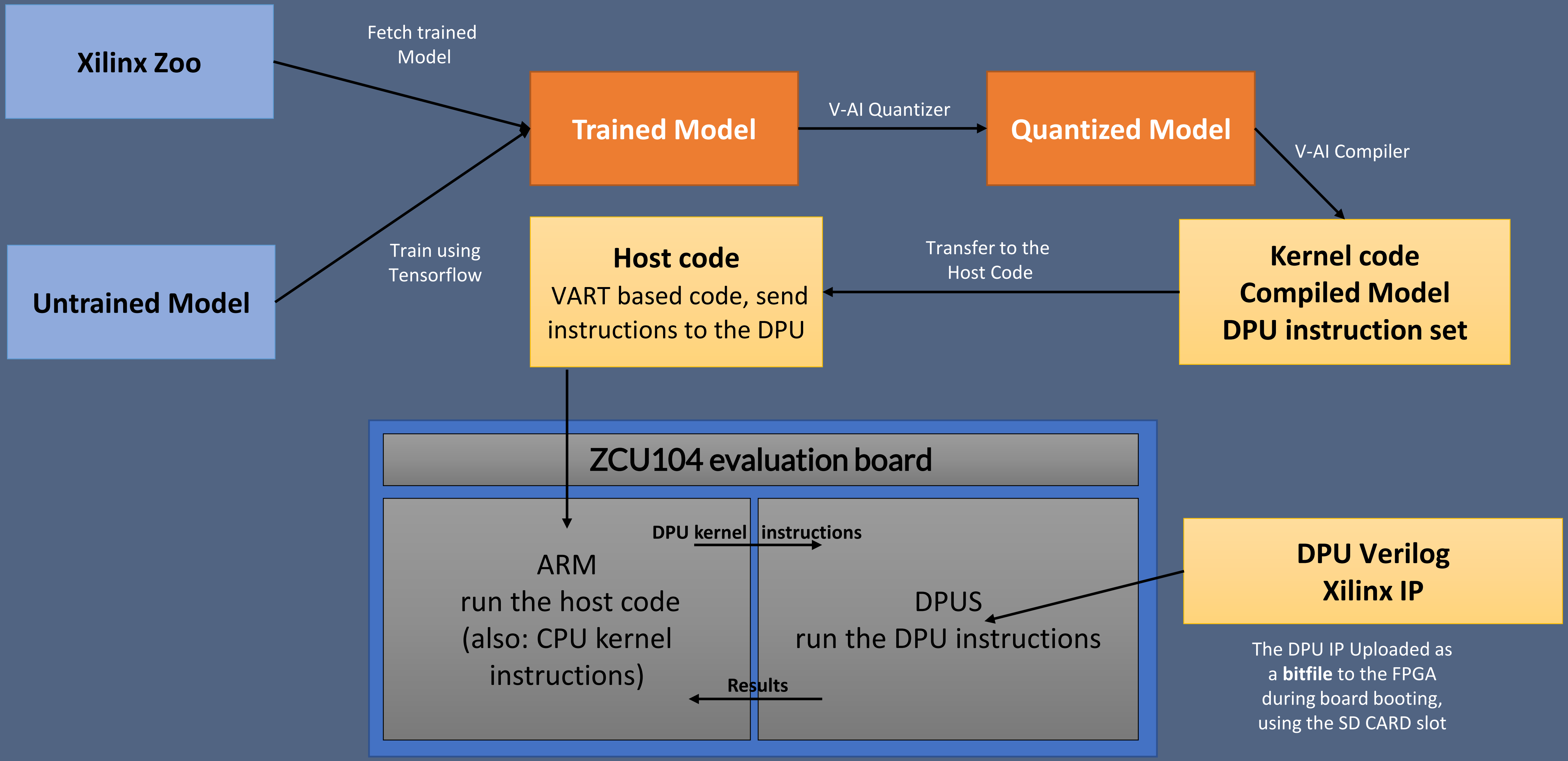


ZCU104 evaluation board

- An integrated circuit designed to be configured by a designer using HDL
- Great for parallel tasks (adders and multipliers)



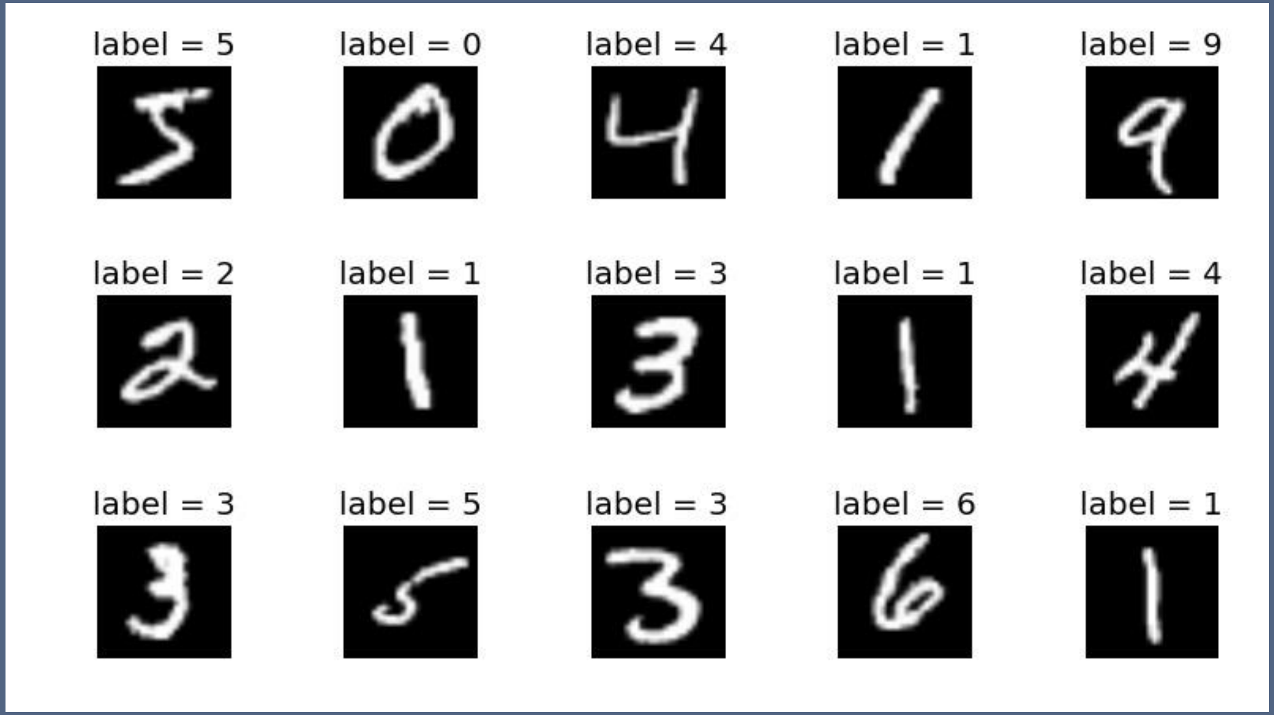
Full Flow - Block Diagram



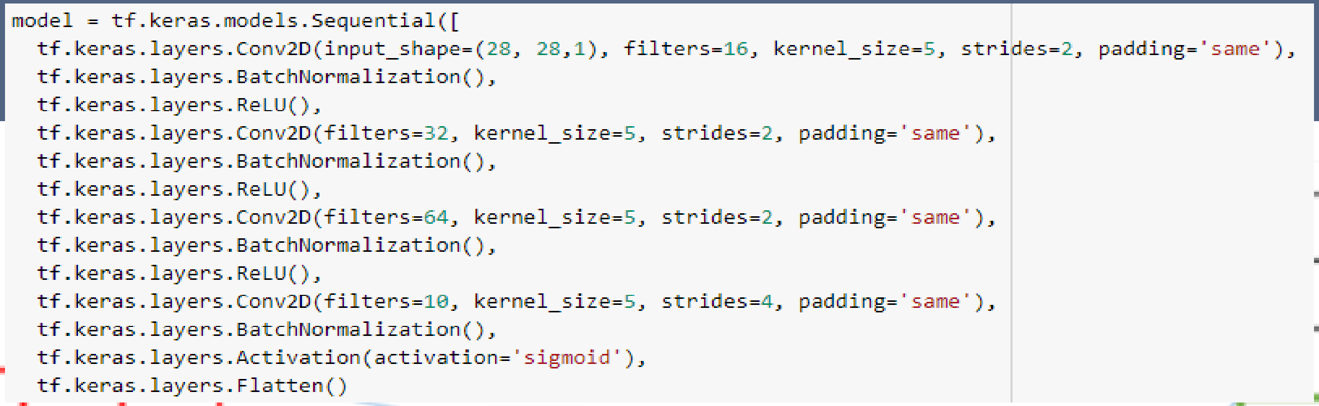
The DPU IP Uploaded as a **bitfile** to the FPGA during board booting, using the SD CARD slot

MNIST Classification Model

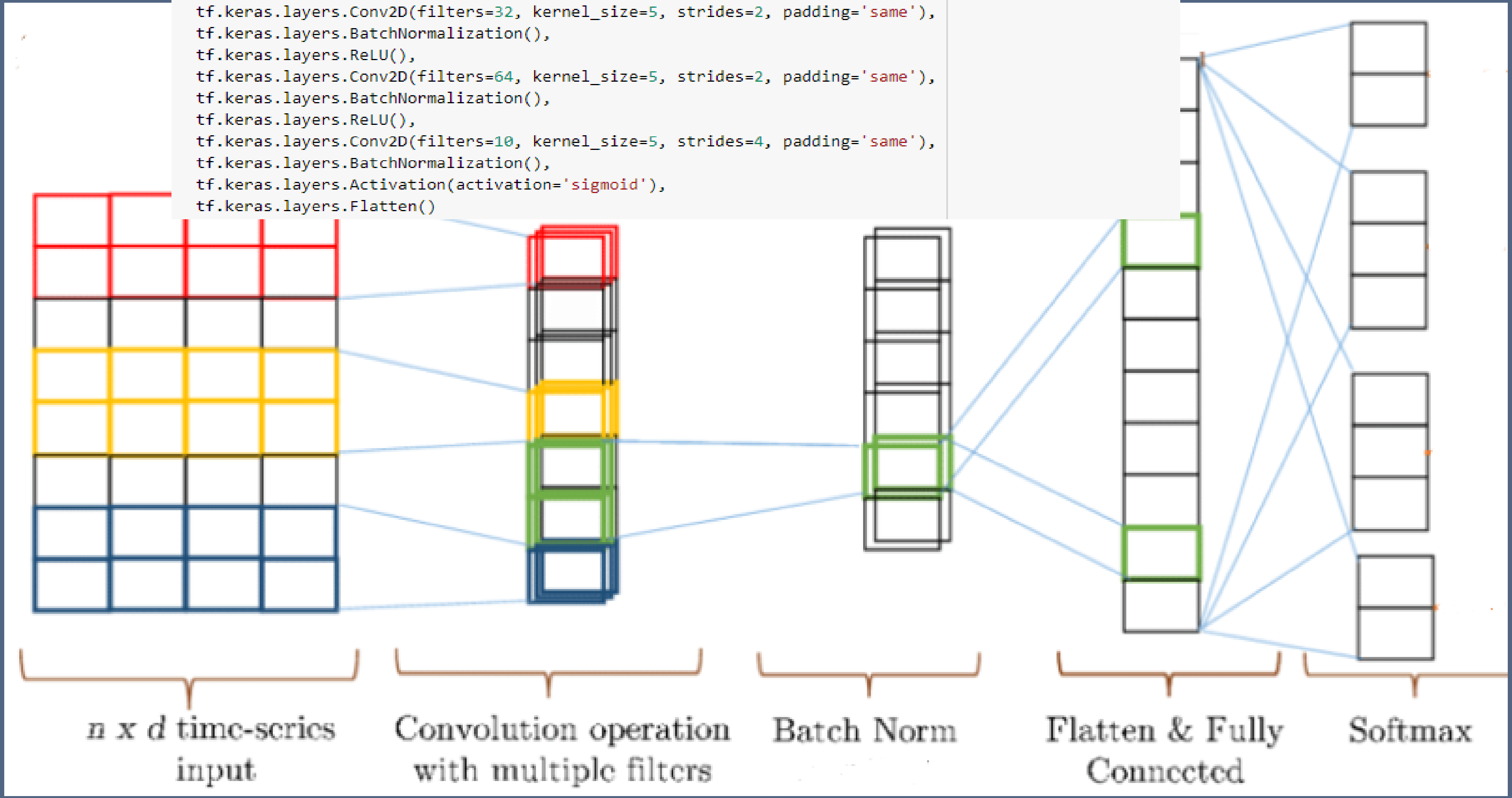
- The MNIST handwritten digits dataset is a publicly available dataset that contains a total of 70k 8bit grayscale images each of which are 28pixels x 28pixels



CNN



```
model = tf.keras.models.Sequential([
  tf.keras.layers.Conv2D(input_shape=(28, 28,1), filters=16, kernel_size=5, strides=2, padding='same'),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.ReLU(),
  tf.keras.layers.Conv2D(filters=32, kernel_size=5, strides=2, padding='same'),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.ReLU(),
  tf.keras.layers.Conv2D(filters=64, kernel_size=5, strides=2, padding='same'),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.ReLU(),
  tf.keras.layers.Conv2D(filters=10, kernel_size=5, strides=4, padding='same'),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation(activation='sigmoid'),
  tf.keras.layers.Flatten()
```


n x d time-series input
Convolution operation with multiple filters
Batch Norm
Flatten & Fully Connected
Softmax

Insights and problems of the ecosystem

Initialize docker environment on the VM in order to use an exists image with Xilinx tools

- Demands large storage
- VirtualBox does not support FMA instructions, switched to VMware

Create new model from scratch in TS and freeze it

Freezing the model in TS is complex. Demanded research in order to find a suitable way to do it

Insights and problems of the ecosystem

Quantize and Compile the model using Xilinx tools

- The model output files works only using TensorFlow 1.x
- The ecosystem having trouble use fully connected layers, therefore we used our own convolution layers.
- **Reason: Xilinx tools transform the FC layers into Conv layers, Thus exceed the maximum input size.**

Insights and problems of the ecosystem

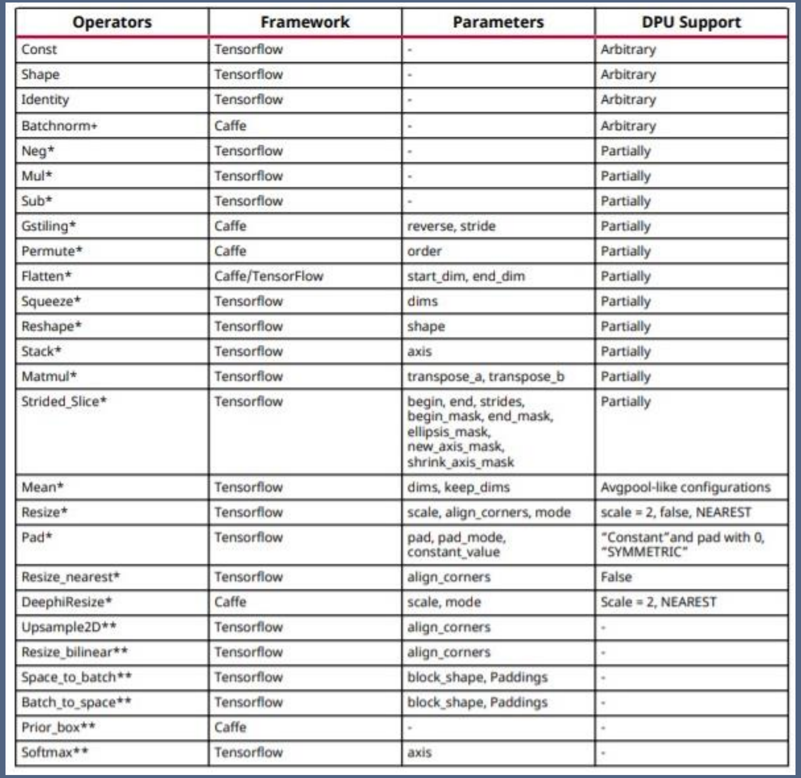
Quantize and Compile the model using Xilinx tools

- Dense-softmax layer failed in quantization stage. Used sigmoid layer as a workaround.
  Reason: The Quantizer is not compatible to work with this layer's output.
- The sigmoid layer couldn't run on the DPU, only on the CPU

Insights and problems of the ecosystem

Between the CPU and the DPU

- Some layers can run on the CPU, some on the DPU. It depends on the Hardware capabilities!

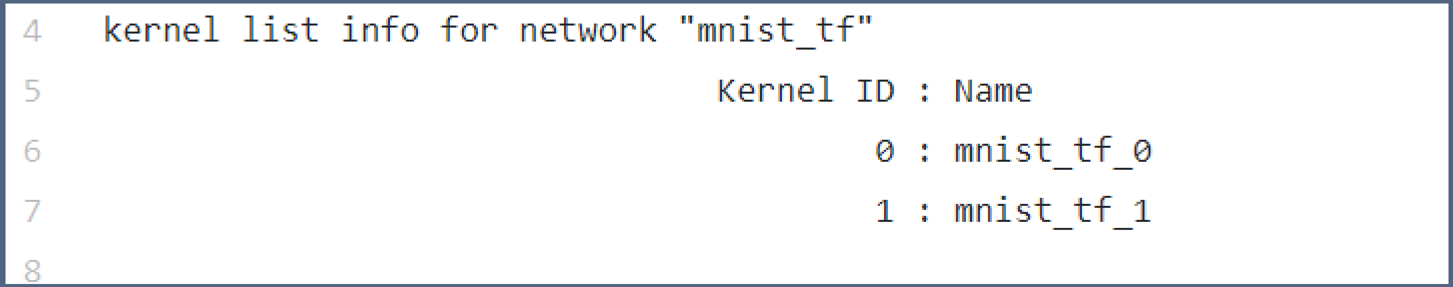


Operators	Framework	Parameters	DPU Support
Const	Tensorflow	-	Arbitrary
Shape	Tensorflow	-	Arbitrary
Identity	Tensorflow	-	Arbitrary
Batchnorm+	Caffe	-	Arbitrary
Neg*	Tensorflow	-	Partially
Mul*	Tensorflow	-	Partially
Sub*	Tensorflow	-	Partially
Gstiling*	Caffe	reverse, stride	Partially
Permute*	Caffe	order	Partially
Flatten*	Caffe/TensorFlow	start_dim, end_dim	Partially
Squeeze*	Tensorflow	dims	Partially
Reshape*	Tensorflow	shape	Partially
Stack*	Tensorflow	axis	Partially
Matmul*	Tensorflow	transpose_a, transpose_b	Partially
Strided_Slice*	Tensorflow	begin, end, strides, begin_mask, end_mask, ellipsis_mask, new_axis_mask, shrink_axis_mask	Partially
Mean*	Tensorflow	dims, keep_dims	Avgpool-like configurations
Resize*	Tensorflow	scale, align_corners, mode	scale = 2, false, NEAREST
Pad*	Tensorflow	pad, pad_mode, constant_value	"Constant"and pad with 0, "SYMMETRIC"
Resize_nearest*	Tensorflow	align_corners	False
DeephiResize*	Caffe	scale, mode	Scale = 2, NEAREST
Upsample2D**	Tensorflow	align_corners	-
Resize_bilinear**	Tensorflow	align_corners	-
Space_to_batch**	Tensorflow	block_shape, Paddings	-
Batch_to_space**	Tensorflow	block_shape, Paddings	-
Prior_box**	Caffe	-	-
Softmax**	Tensorflow	axis	-

Insights and problems of the ecosystem

Between the CPU and the DPU

- The Vitis-AI compiler will figure out which layers can run on each processor
- It will export two type of kernels:



```
kernel list info for network "mnist_tf"
                               Kernel ID : Name
                                       0 : mnist_tf_0
                                       1 : mnist_tf_1

```

- Data flow: ...CPU->DDR->DPU->DDR...

Evaluations And Modifications

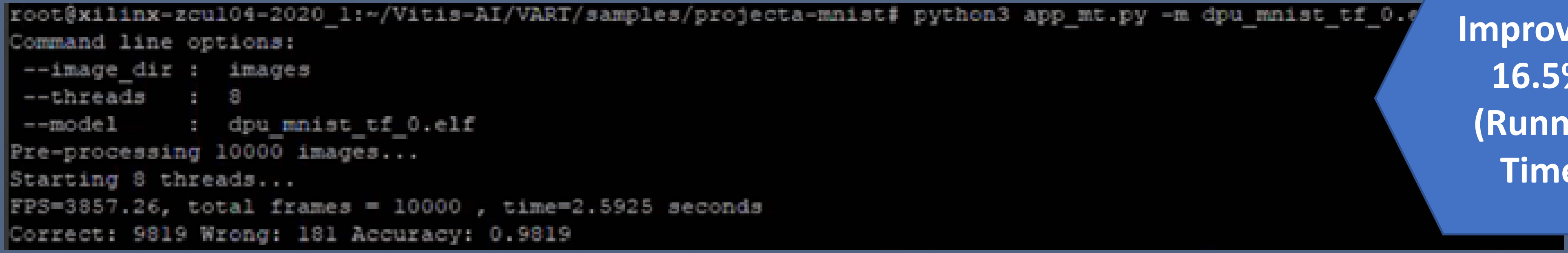
Optional modifications in the process

- Number of calibration iterations in the quantization stage
  - Increasing the number of iteration and accordingly the number of the images
- Built in quantization methods – method 0/1
- Number of threads
- Change the weight/activation bit width from 8 bit to 16 bit

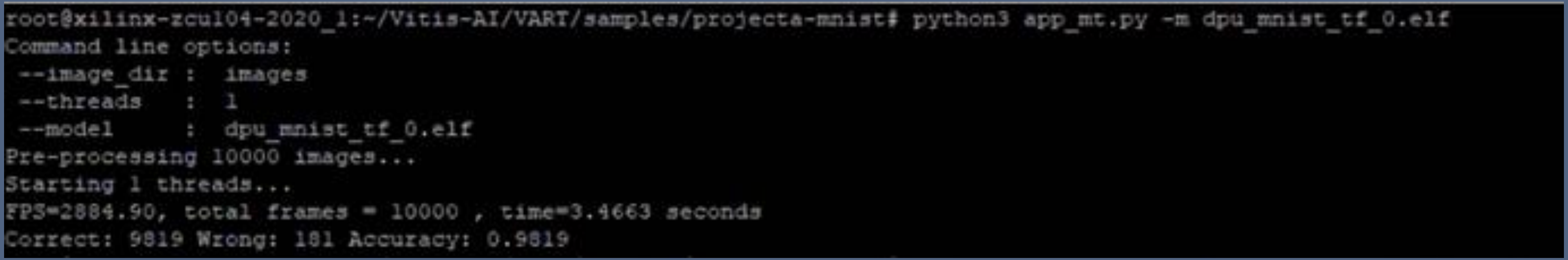
Optimal Running Time Modification

Number of threads

- Change the number of threads which run the inference stage on the board. We have changed it in order to optimize the running time



```
root@xilinx-zcu104-2020_1:~/Vitis-AI/VART/samples/projecta-mnist# python3 app_mt.py -m dpu_mnist_tf_0.e
Command line options:
 --image_dir :  images
 --threads   :  8
 --model     :  dpu_mnist_tf_0.elf
Pre-processing 10000 images...
Starting 8 threads...
FPS=3857.26, total frames = 10000 , time=2.5925 seconds
Correct: 9819 Wrong: 181 Accuracy: 0.9819
```



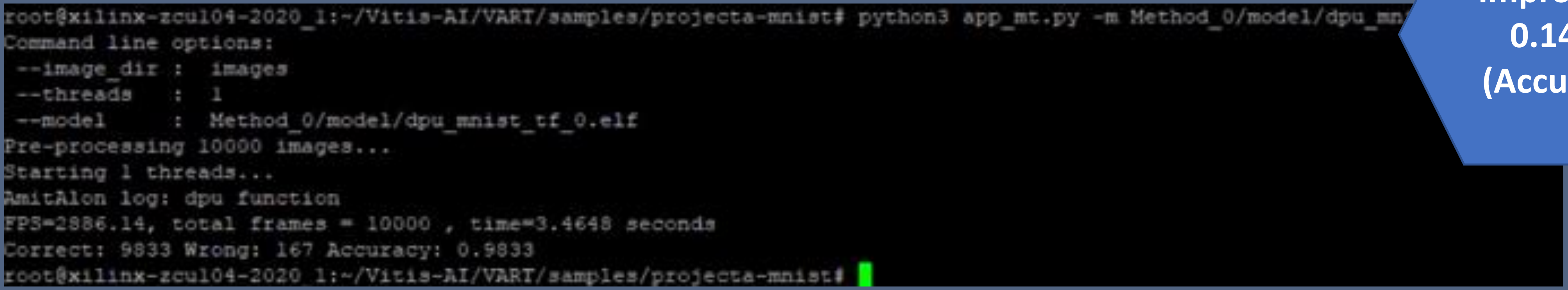
```
root@xilinx-zcu104-2020_1:~/Vitis-AI/VART/samples/projecta-mnist# python3 app_mt.py -m dpu_mnist_tf_0.elf
Command line options:
 --image_dir :  images
 --threads   :  1
 --model     :  dpu_mnist_tf_0.elf
Pre-processing 10000 images...
Starting 1 threads...
FPS=2884.90, total frames = 10000 , time=3.4663 seconds
Correct: 9819 Wrong: 181 Accuracy: 0.9819
```

Improve of 16.5%! (Running Time)

Optimal Accuracy Modification

Quantization method

- There are two methods of quantization, big range of the quantization values and smaller one. By default, the board choose the smaller one. We chose the method that have the bigger range of quantization values



```
root@xilinx-zcu104-2020_1:~/Vitis-AI/VART/samples/projecta-mnist# python3 app_mt.py -m Method_0/model/dpu_mn
Command line options:
 --image_dir :  images
 --threads   :  1
 --model     :  Method_0/model/dpu_mnist_tf_0.elf
Pre-processing 10000 images...
Starting 1 threads...
AmitAlon log: dpu function
FPS=2886.14, total frames = 10000 , time=3.4648 seconds
Correct: 9833 Wrong: 167 Accuracy: 0.9833
root@xilinx-zcu104-2020_1:~/Vitis-AI/VART/samples/projecta-mnist#
```

Improve of 0.14%! (Accuracy)

Problems, Issues and Insights

We faced many Issues in the modification process:

- Changing from 8 bit width to 16 bit width is not possible. The board can work with width of only 8 bit
- We found that the relevant modification is possible only in the quantization stage.

- Accuracy can be improved in substantial rate on method 0 quantization
- Running time can be improved in substantial rate on the right quantity of threads due to its parallelism

Performance

PC - Intel(R) Xeon(R) CPU @ 2.20GHz, 2 Cores

- **Accuracy 98.25%**
- **Time:** $\mathbf{2.221[\mathit{Sec}], 0.02221\left[\frac{\mu Sec}{Pic}\right]}$

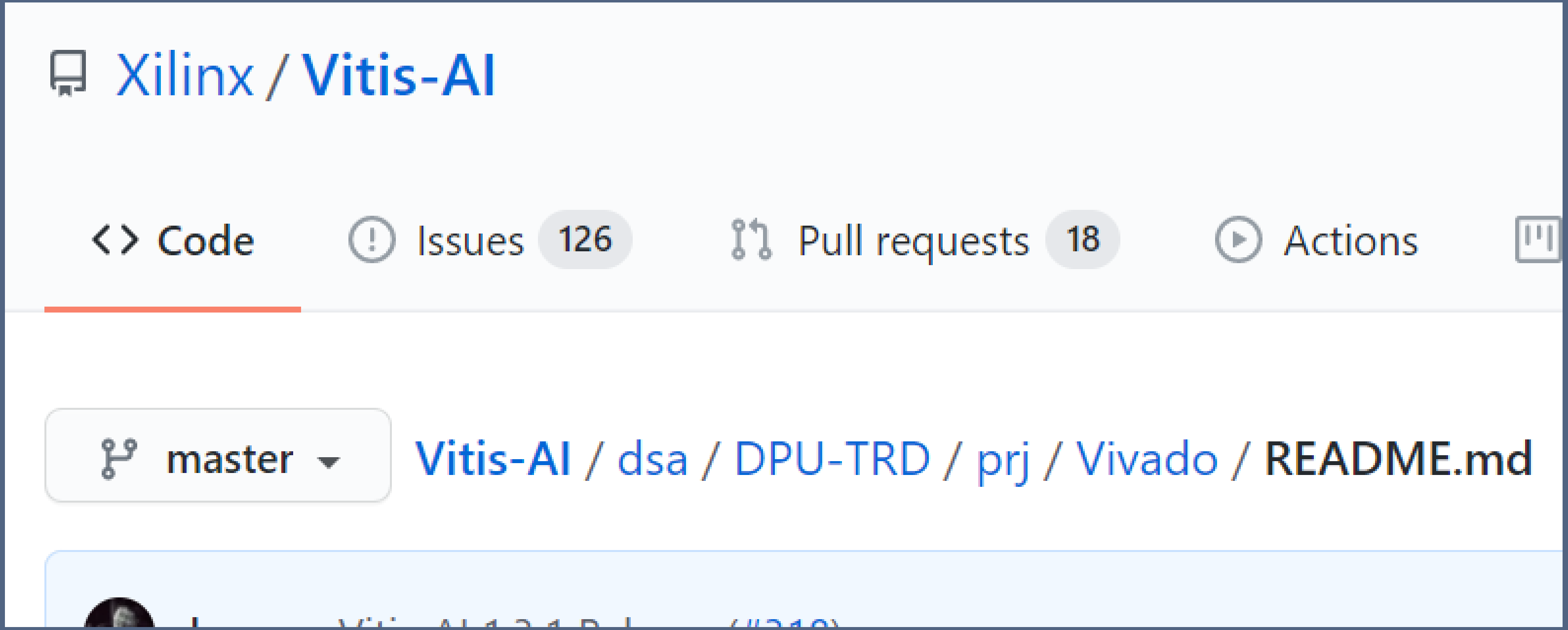
ZCU104 - Quad-core ARM® Cortex™-A53 + DPU

- **Accuracy 98.33%**
- **Time:** $\mathbf{2.592[\mathit{Sec}], 0.02592\left[\frac{\mu Sec}{Pic}\right]}$

- ZCU 104 can achieve higher accuracy! However, it’s slower compared to inference on a regular PC
- On the board we can make modifications in many points between the model and the inference, for example the quantization stage. Less trivial to do on basic Collab

Compile and deploy DPU’s RTL (POC)

- POC in order to see the complexity of compiling and deploying the DPU’s RTL into the ZCU-104 board


Xilinx / Vitis-AI

Vitis-AI / dsa / DPU-TRD / prj / Vivado / README.md

Compile and deploy DPU’s RTL (POC) - Insights

- The only available guide that’s exists is for a different board (ZCU-102)
- Requires a lot of adjustments and changes In **each** step in the process:
  - RTL - Compiling
  - Configuring and build the Peta-Linux project
  - Creating the boot image
  - Configure the DPU itself
- It is better to be performed on ZCU-102 board
- In any Case - it’s a long and complex process. Even after Success compiling, encountered problems of compatibility with the board

Conclusions

- In some cases, the board can achieve better accuracy than normal computer in the inference stage due to his dedicated hardware
- Xilinx gives AI ability, even to unexperienced people (although it could be limited)
- Creating your own Custom model is very complex. Many adjustments have to be done in order to successfully run it on the board
- There is a lack of updated information on Xilinx ecosystem. Therefore, it’s hard to solve problems that arise during the process

Future work

- Test and validate the ecosystem on different framework (caffe)
- Compile and deploy DPU’s RTL (POC)
- Combine the board with real life applications