My Workflow –

Links of drive and complete source code –

https://drive.google.com/drive/folders/1mUY8SWf4Yxap5B0x0D2gu-com1XdXBt_?usp=sharing

Machine worked out :

1. Training of ML model : Google Colab (T4 GPU)
2. Quantise & Compile model : IPC lab, Docker image : Vitis-ai-pytorch-cpu

Versions :

1. Vitis ai : 3.5
2. Framework : Pytorch-cpu

ML model :

1. Architecture : Ultralytics Yolov5 (small, segmentation)
2. Framework: Pytorch
3. Model and utility library: Ultralytics Yolov5 - https://github.com/ultralytics/yolov5.git

//--//

Flow –

1. Understand the Problem statement, Dataset format.
    a. Mine was Instance-Segmentation of "Containers".
    b. Used Roboflow dataset - https://universe.roboflow.com/ds/MRggnf5QK1?key=WDpUCKltpJ
    c. Download the dataset in format – "Yolov5-pytorch"

2. Training of ML model – [Google Colab T4 GPU].
   Ipynb file which runs this flow– *"Training_yolov5.ipynb"*

    a. Reference - https://colab.research.google.com/github/roboflow-ai/yolov5-custom-training-tutorial/blob/main/yolov5-custom-training.ipynb
       This is Roboflow training tutorial colab file. We used this to train our ML model.
    b. We modified the above colab to cater for "instance segmentation".
        i. Refered : Ultralytics github repo : https://github.com/ultralytics/yolov5.git
        ii. Carried workflow to mimic segmentation. i.e., using yolov5s-seg COCO trained weights as initial weights for Transfer learning, used CLI calls to do training, validation, inferencing with mode = segment.
    c. Challenge faced while training –
       For my dataset which has labels of both Image-segments (array of points to form polygon) and detection (Bounding-boxes info x1,y1,width,height) for all images. Some images had only Bounding-boxes not segments. Because of which the training was throwing warning (not training):
       "WARNING ⚠ Box and segment counts should be equal, but got len(segments) = 188, len(boxes) = 203. To resolve this only boxes will be

used and all segments will be removed. To avoid this please supply either a **detect or segment dataset**, not a **detect-segment mixed dataset**."

Solution :
Prepared a simple python code to read json of all labels and find those images who's len(segments) != len(boxes) and printed it. It was around 43 images which later I manually removed using terminal command rm.

d. Transfer learning –
    i. Some learning (from previous workflow errors) – Xilinx doesn't support SiLU activation function. So, in yolo architecture the SiLU need to be replaced to some other non-linear activation function i.e. ReLU, Leaky-ReLU. Reference - https://github.com/Xilinx/Vitis-AI/issues/1252

       Need to modify **./models/Common.py and ./models/experiment.py** where the architecture definitions are used.

       nn.SiLU → nn.LeakyReLU(0.1, inplace=True)

    ii. Now using CLI interface train.py is run, with yolov5s-seg model for 100 epochs, initial weights = COCO dataset train weights for segmentation. Image size is 640x640. And in batch of 16.
    iii. After training for 100 epochs, the best model file can be found in ./runs/train-seg/exp/weights/best.pt
    iv. Get the model evaluated using test dataset. And obtain the evaluation metrics of our trained model.
    v. This model file along with model class definitions becomes a complete pytorch model. Download this .pt for further workflow in Vitis-ai environment.

3. Model importing into Vitis-ai environment – [IPC lab. Docker image : vitis-ai-pytorch-cpu]
   Ipynb file which runs this flow– *"MyNotebook_Quantisation_Compilation.ipynb"*

    a. Run docker. Create a new working directory for the project. Activate vitis-ai-pytorch environment. And open a fresh jupyter nootbook here.
    b. Setup the Ultralytics Yolov5 library in this working directory.
    c. Need to modify –
        i. SiLU function to be replaced with leakyReLU in common.py and experiment.py
        ii. Forward method definition to be modified in yolo.py for quantisation.
    d. Import the best.pt using DetectMultiBackend method.
    e. Do a sanity check after model import. By a simple inference of any test image.

4. Quantize and Compile model to the target DPU
   Ipynb file which runs this flow– *"MyNotebook_Quantisation_Compilation.ipynb"*

    a. "from pytorch_nndct.apis import torch_quantizer" this method does the quantisation. Import it in the jupyter book.
    b. Quantise in Calib-mode.
    c. Do evaluation and a simple output, as it has to be done for drop-outs reset.

d. Now, run qunatise in Test-mode. And perform step c.
e. Export the xmodel.
f. Now compile the model for target DPU.
g. Final xmodel is available in ./quantize_result/yolov5_kv260.xmodel