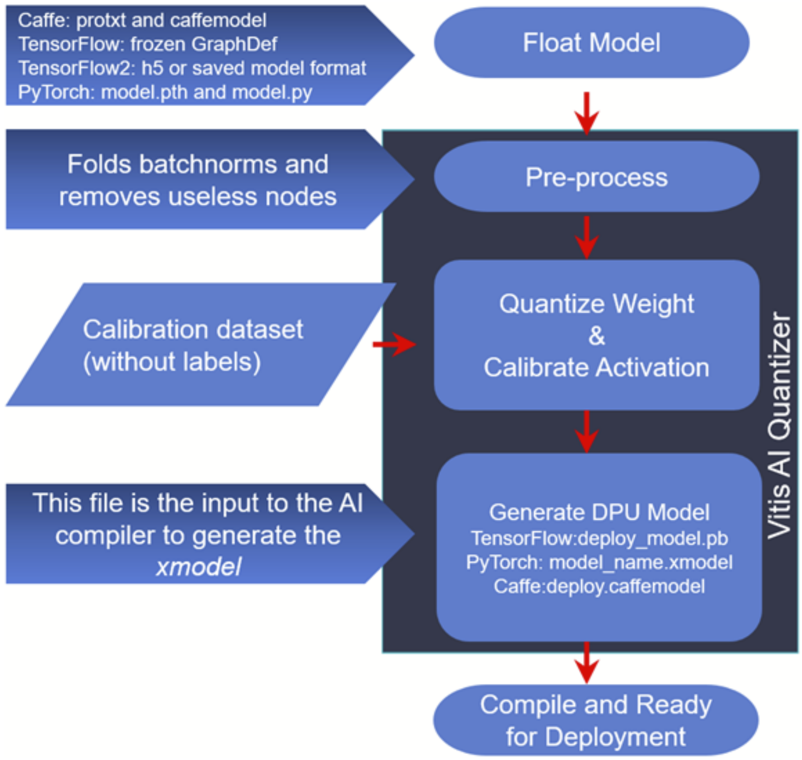
Part C. Compile the Models

We will describe how to compile models (generating the xmodel files that program the DPU), compiling manually the models. The general compiling flow is described in the following figure depending on the input framework used (TensorFlow, Caffe, PyTorch).



The **Xilinx Model Zoo** (https://github.com/Xilinx/AI-Model-Zoo) is a repository of free pre-trained deep learning models, optimized for inference deployment on Xilinx™ platforms.

Note: It is important to know the correlation between models and applications. This table includes a non-exhaustive list of applications that were verified with corresponding models from the model zoo.

Application	**model name**	**model zoo name**
face detection	densebox_640_360	cf_densebox_wider_360_640_1.11G_1.4
resnet50 (caffe)	resnet50	cf_resnet50_imagenet_224_224_7.7G_1.4
resnet50 (tensorflow)	resnet_v1_50_tf	tf_resnetv1_50_ imagenet_224_224_6.97G_1.4

Step 1 - Installing the Vitis-AI

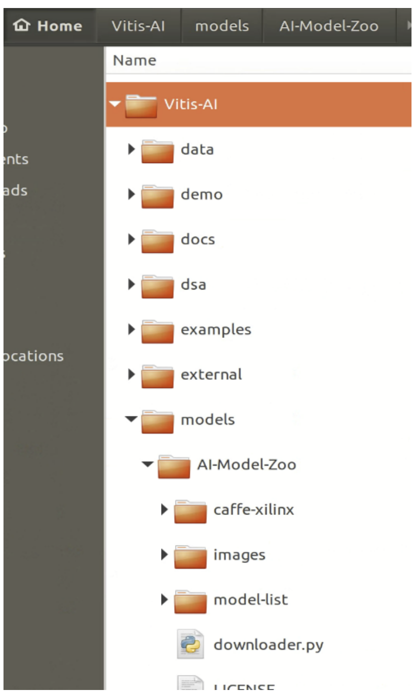
The first step is to clone the “v1.4” branch of the Vitis-AI repository

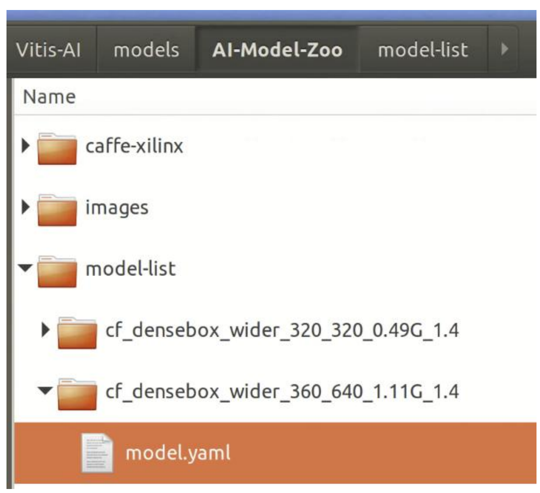
```
$ git clone -b v1.4 https://github.com/Xilinx/Vitis-AI Vitis-AI_1.4
$ cd Vitis-AI_1.4
$ export VITIS_AI_HOME=$PWD
```

Step 2 - Inspect the models

The second step is to inspect the model.yaml for the specific model from the Xilinx Model Zoo. For example, for the 640x360 version of the densebox model:

```
$ cd models/AI-Model-Zoo/
$ cat model-list/cf_densebox_wider_360_640_1.11G_1.4/model.yaml
```





```yaml
description: face detection model.
input size: 360*640
float ops: 1.11G
task: face detection
framework: caffe
prune: 'no'
version: 1.4
files:
- name: cf_densebox_wider_360_640_1.11G_1.4
  type: float & quantized
  board: GPU
  download link: https://www.xilinx.com/bin/public/openDownload?filename=cf_densebox_wider_360_640_1.11G_1.4.zip
  checksum: e7a2fb60638909db368ab6bb6fa8283e
- name: densebox_640_360
  type: xmodel
  board: zcu102 & zcu104 & kv260
  download link: https://www.xilinx.com/bin/public/openDownload?filename=densebox_640_360-zcu102_zcu104_kv260-r1.4.0.tar.gz
  checksum: 101bce699b9dada0e97fdf0c95aa809f
- name: densebox_640_360
  type: xmodel
  board: vck190
  download link: https://www.xilinx.com/bin/public/openDownload?filename=densebox_640_360-vck190-r1.4.0.tar.gz
  checksum: 101c3c36dec1ffd9291126fcd365fbc0
- name: densebox_640_360
  type: xmodel
```

We can see that Xilinx provides several versions of the model, including:

> float & quantized: pre-quantized model, used as source for compilation
> zcu102 & zcu104 & kv260 : pre-built model binaries for zcu102/zcu104 boards
> etc…

Since our board is not present (ultra96v2) we download the float and quantized model.

Step 3 - Download the models

The third step is to download the source archive for the model and extract it. We will download caffe´s model of densebox and resnet50, and tensorflow model of resnet50.

```
$ wget
https://www.xilinx.com/bin/public/openDownload?filename=cf_densebox_wider_360_640_1.11G_1.4.zip -O cf_densebox_wider_360_640_1.11G_1.4.zip
$ unzip cf_densebox_wider_360_640_1.11G_1.4.zip
```

Do the same for the other models that you want to compile

```
$ wget
https://www.xilinx.com/bin/public/openDownload?filename=cf_resnet50_imagenet_224_224_7.7G_1.4.zip -O
cf_resnet50_imagenet_224_224_7.7G_1.4.zip
$ unzip cf_resnet50_imagenet_224_224_7.7G_1.4.zip
```

```
$ wget
https://www.xilinx.com/bin/public/openDownload?filename=tf_resnetv1_50_imagenet_224_224_6.97G_1.4.zip -O
tf_resnetv1_50_imagenet_224_224_6.97G_1.4.zip
$ unzip tf_resnetv1_50_imagenet_224_224_6.97G_1.4.zip
```

After downloading and uncompress, you will see the three folders.

Step 4 - Copy the architecture description

Copy the architecture file (arch.json) for your hardware platform. For the pre-built images, this file can be found in the BOOT partition of the design's SD card.

```
$ cp {path_to_arch_json}/arch.json .
```

This file should contain content similar to the following:


```
{"fingerprint":"..."}
```

This is some kind of encrypted content, that identifies the DPU configuration for the design.

We leave a copy in shared repository under the name arch_ultra96v2_lab3.json

For the “u96v2” (which have the B2304_LR DPU), the following arch.json can also be used:

```
{"target":"DPUCZDX8G_ISA0_B2304_MAX_BG2"}
```

The encrypted version is:

```
{"fingerprint":"0x1000020F6014405"}
```

Step 5 - Launch the Vitis-AI docker container

5.1. If not done so already, pull version 1.4.916 of the docker container with the following command (it is already downloaded):

Check the installed dockers

```
$ docker images
```

```
(base) osboxes@osboxes:~/Vitis-AI/lab3$ docker images
REPOSITORY         TAG          IMAGE ID       CREATED          SIZE
xilinx/vitis-ai    2.5          41ed5c7e3331   9 months ago     19.5GB
xilinx/vitis-ai    1.4.1.978    806328cdfc06   17 months ago    17.7GB
hello-world        latest       d1165f221234   2 years ago      13.3kB
(base) osboxes@osboxes:~/Vitis-AI/lab3$
```

In case it is not present, you can download it

```
docker pull xilinx/vitis-ai:1.4.1.978
```

5.2 Launch version 1.4.1.978 of the Vitis-AI docker from the Vitis-AI directory:

```
$ cd $VITIS_AI_HOME
$ sh -x docker_run.sh xilinx/vitis-ai:1.4.1.978
```

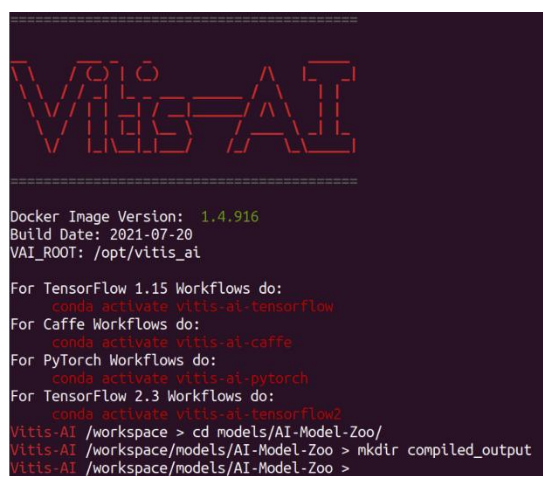
5.2. When prompted, read all the license notification messages, and press ENTER to accept the license terms.

5.3. Navigate to the AI-Model-Zoo directory

```
$ cd models/AI-Model-Zoo
```

5.4. Create a directory for the compiled models

```
$ mkdir compiled_output
```



```
Vitis-AI /workspace/models/AI-Model-Zoo > ls -l
total 402680
drwxr-xr-x  14 vitis-ai-user vitis-ai-group      4096 Feb 22 11:57 caffe-xilinx
drwxrwxr-x   6 vitis-ai-user vitis-ai-group      4096 Jun  8  2021 cf_densebox_wider_360_640_1.11G_1.4
-rw-rw-r--   1 vitis-ai-user vitis-ai-group   4629181 Jul 20  2021 cf_densebox_wider_360_640_1.11G_1.4.zip
drwxrwxr-x   6 vitis-ai-user vitis-ai-group      4096 Jun  8  2021 cf_resnet50_imagenet_224_224_7.7G_1.4
-rw-rw-r--   1 vitis-ai-user vitis-ai-group 217139307 Jul 20  2021 cf_resnet50_imagenet_224_224_7.7G_1.4.zip
drwxr-xr-x   2 vitis-ai-user vitis-ai-group      4096 Mar 20 15:22 compiled_output
-rw-r--r--   1 vitis-ai-user vitis-ai-group      3701 Feb 22 11:57 downloader.py
drwxr-xr-x   2 vitis-ai-user vitis-ai-group      4096 Feb 22 11:57 images
-rw-r--r--   1 vitis-ai-user vitis-ai-group     11356 Feb 22 11:57 LICENSE
drwxr-xr-x 113 vitis-ai-user vitis-ai-group     12288 Feb 22 11:57 model-list
-rw-r--r--   1 vitis-ai-user vitis-ai-group    260015 Feb 22 11:57 README.md
drwxrwxr-x   6 vitis-ai-user vitis-ai-group      4096 Jun 10  2021 tf_resnetv1_50_imagenet_224_224_6.97G_1.4
-rw-rw-r--   1 vitis-ai-user vitis-ai-group 190247152 Jul 20  2021 tf_resnetv1_50_imagenet_224_224_6.97G_1.4.zip
Vitis-AI /workspace/models/AI-Model-Zoo >
```

Step 6 - Create a generic recipe to compile models

6.1. Create a generic recipe for compiling a caffe model, by creating a script named “compile_cf_model.sh” with the following content

```
model_name=$1
modelzoo_name=$2
vai_c_caffe \
--prototxt ./${modelzoo_name}/quantized/deploy.prototxt \
--caffemodel ./${modelzoo_name}/quantized/deploy.caffemodel \
--arch ./arch.json \
--output_dir ./compiled_output/${model_name} \
--net_name ${model_name}
```

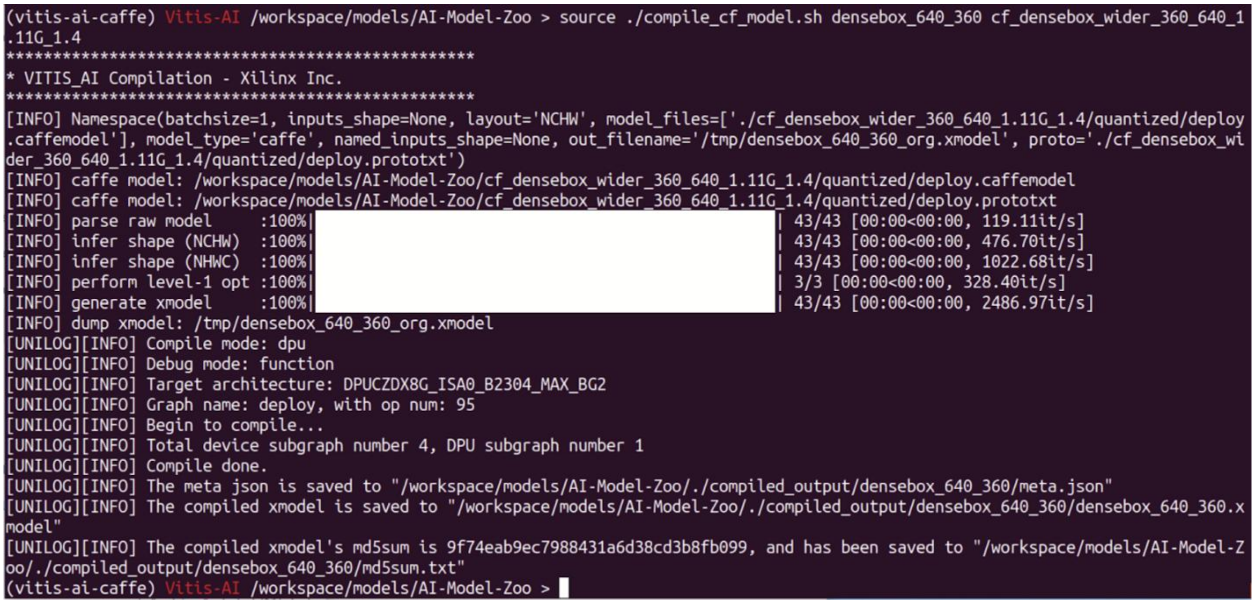
6.2. Create a generic recipe for compiling a tensorflow model, by creating a script called “compile_tf_model.sh” with the following content

```
model_name=$1
modelzoo_name=$2
vai_c_tensorflow \
--frozen_pb ./${modelzoo_name}/quantized/quantize_eval_model.pb \
--arch ./arch.json \
--output_dir ./compiled_output/${model_name} \
--net_name ${model_name}
```

Step 7 - Compile the models using the created scripts

7.1. To compile the caffe model used by the face_detection application, invoke the generic script we just created as follows:

```
$ conda activate vitis-ai-caffe
(vitis-ai-caffe) $
source ./compile_cf_model.sh densebox_640_360
cf_densebox_wider_360_640_1.11G_1.4
```



```
(vitis-ai-caffe) Vitis-AI /workspace/models/AI-Model-Zoo > source ./compile_cf_model.sh densebox_640_360 cf_densebox_wider_360_640_1
.11G_1.4
**************************************************
* VITIS_AI Compilation - Xilinx Inc.
**************************************************
[INFO] Namespace(batchsize=1, inputs_shape=None, layout='NCHW', model_files=['./cf_densebox_wider_360_640_1.11G_1.4/quantized/deploy
.caffemodel'], model_type='caffe', named_inputs_shape=None, out_filename='/tmp/densebox_640_360_org.xmodel', proto='./cf_densebox_wi
der_360_640_1.11G_1.4/quantized/deploy.prototxt')
[INFO] caffe model: /workspace/models/AI-Model-Zoo/cf_densebox_wider_360_640_1.11G_1.4/quantized/deploy.caffemodel
[INFO] caffe model: /workspace/models/AI-Model-Zoo/cf_densebox_wider_360_640_1.11G_1.4/quantized/deploy.prototxt
[INFO] parse raw model     :100%|                    | 43/43 [00:00<00:00, 119.11it/s]
[INFO] infer shape (NCHW)  :100%|                    | 43/43 [00:00<00:00, 476.70it/s]
[INFO] infer shape (NHWC)  :100%|                    | 43/43 [00:00<00:00, 1022.68it/s]
[INFO] perform level-1 opt :100%|                    | 3/3 [00:00<00:00, 328.40it/s]
[INFO] generate xmodel     :100%|                    | 43/43 [00:00<00:00, 2486.97it/s]
[INFO] dump xmodel: /tmp/densebox_640_360_org.xmodel
[UNILOG][INFO] Compile mode: dpu
[UNILOG][INFO] Debug mode: function
[UNILOG][INFO] Target architecture: DPUCZDX8G_ISA0_B2304_MAX_BG2
[UNILOG][INFO] Graph name: deploy, with op num: 95
[UNILOG][INFO] Begin to compile...
[UNILOG][INFO] Total device subgraph number 4, DPU subgraph number 1
[UNILOG][INFO] Compile done.
[UNILOG][INFO] The meta json is saved to "/workspace/models/AI-Model-Zoo/./compiled_output/densebox_640_360/meta.json"
[UNILOG][INFO] The compiled xmodel is saved to "/workspace/models/AI-Model-Zoo/./compiled_output/densebox_640_360/densebox_640_360.x
model"
[UNILOG][INFO] The compiled xmodel's md5sum is 9f74eab9ec7988431a6d38cd3b8fb099, and has been saved to "/workspace/models/AI-Model-Z
oo/./compiled_output/densebox_640_360/md5sum.txt"
(vitis-ai-caffe) Vitis-AI /workspace/models/AI-Model-Zoo >
```

7.2. To compile the caffe model used by the resnet50 application, invoke the generic script we just created as follows:

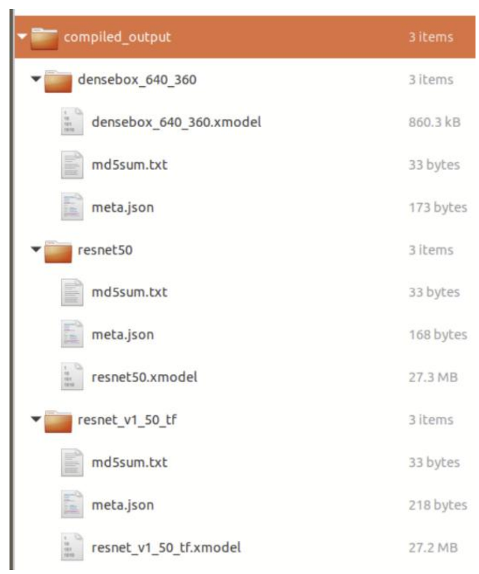
```
$ conda activate vitis-ai-caffe
(vitis-ai-caffe) $
source ./compile_cf_model.sh resnet50
cf_resnet50_imagenet_224_224_7.7G_1.4
```

7.3. To compile the tensorflow model used by the resnet50 application, invoke the generic script we just created as follows:

```
$ conda activate vitis-ai-tensorflow
(vitis-ai-tensorflow) $
source ./compile_tf_model.sh resnet_v1_50_tf
tf_resnetv1_50_imagenet_224_224_6.97G_1.4
```

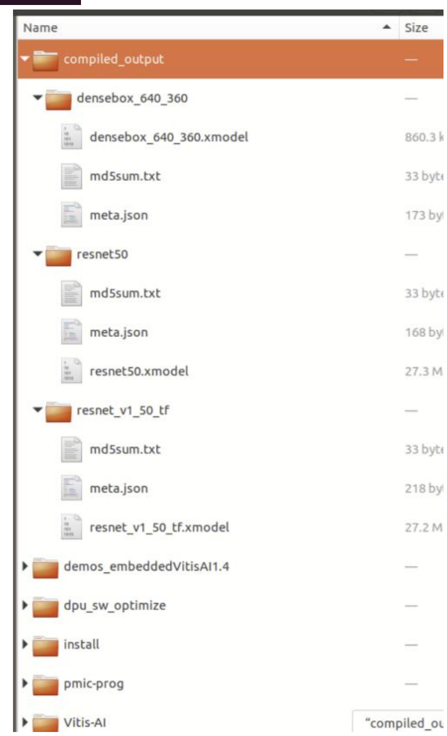
7.4. Verify the contents of the directory with the tree utility:

```
$ tree compiled_output
```



```
(vitis-ai-caffe) Vitis-AI /workspace/models/AI-Model-Zoo > tree compiled_output/
compiled_output/
├── densebox_640_360
│   ├── densebox_640_360.xmodel
│   ├── md5sum.txt
│   └── meta.json
├── resnet50
│   ├── md5sum.txt
│   ├── meta.json
│   └── resnet50.xmodel
└── resnet_v1_50_tf
    ├── md5sum.txt
    ├── meta.json
    └── resnet_v1_50_tf.xmodel

3 directories, 9 files
(vitis-ai-caffe) Vitis-AI /workspace/models/AI-Model-Zoo >
```



Exit the tools docker using exit

```
$ exit
```

Step 8 - Test Compiled model

Copy the three generated models to the embedded device (UltraV2 board) and analyze and test the three generated models.

Hint1: Connect to the wifi and then using scp or graphically copy the compiled_output to the board.

8.1. Test the two resnet50 models

Hint2. In order to test the two resnet50 models, analyze the example from **“3.6e.1. Launch the resnet50_mt_py application”**

Open the ~/Vitis-AI/demo/VART/resnet50_mt_py/resnet50.py python file

Analyze what is doing this example

Compare the results using the resnet50 from caffe and tensorflow

Hint3. Some screens capture with results.

```
root@u96v2-sbc-base-2021-1:~/Vitis-AI/demo/VART/resnet50_mt_py# python3 ./resnet50.py 8 /usr/share/vitis_ai_library/models/resnet50/resnet50.xmodel
FPS=26.96, total frames = 2880.00 , time=106.827189 seconds
root@u96v2-sbc-base-2021-1:~/Vitis-AI/demo/VART/resnet50_mt_py# python3 ./resnet50.py 8 ~/compiled_output/resnet50/resnet50.xmodel
FPS=26.95, total frames = 2880.00 , time=106.849430 seconds
root@u96v2-sbc-base-2021-1:~/Vitis-AI/demo/VART/resnet50_mt_py#
```

```
root@u96v2-sbc-base-2021-1:~/Vitis-AI/demo/VART/resnet50_mt_py# python3 ./resnet50.py 8 ~/compiled_output/resnet_v1_50_tf/resnet_v1_50_tf.xmodel
FPS=29.56, total frames = 2880.00 , time=97.417513 seconds
```

8.2 Test the densebox_640_360models

Hint1. Analyse the example “3.6.f. Launch the Vitis-AI-Library based sample applications”.

```
$ cd ~/Vitis-AI/demo/Vitis-AI-Library/samples/facedetect
$ ./test_video_facedetect densebox_640_360 0
```

OPTIONAL: Analyze the C++ code: test_video_facedetect.cpp and process_result.hpp. Change the color of the bounding boxes

Hint2. Recall the basic rectangle functions

```
// our rectangle...
cv::Rect rect(x, y, width, height);
// and its top left corner...
cv::Point pt1(x, y);
// and its bottom right corner.
cv::Point pt2(x + width, y + height);
// These two calls...
cv::rectangle(img, pt1, pt2, cv::Scalar(0, 255, 0));
// essentially do the same thing
cv::rectangle(img, rect, cv::Scalar(0, 255, 0))
```

8.2.3. Run the example using the recently generated xmodel (modify accordingly the execution)

Hint3: You will need to copy prototxt model from:

/usr/share/vitis_ai_library/models/densebox_640_360

Step 9 (optional) - Test another model from ModelZoo

You can play with roughly a hundred precompiled models.
This step gives extra points for your grade.

Hint: Search on the internet for examples using Ultra96 boards.

EXTRA POINTS (difficult): Use input images from an external USB camera, connected to the Ultra96 board. You can use whatever model / use case you want.