

Application Note



Testing all Samples from Xilinx Vitis AI Library 2.0 on Trenc Electronic board TE0808 SoM + TEBF0808 Carrier

Zdeněk Pohl, Lukáš Kohout, Jiří Kadlec
zdenek.pohl@utia.cas.cz, kohoutl@utia.cas.cz, kadlec@utia.cas.cz

Revision history

Rev.	Date	Author	Description
01	30.11.2022	Z.P.	Document creation

Contents

1	Description.....	1
2	Requirements	1
3	How to Build Vitis AI Library Samples and Install Models.....	1
4	Tested Demos	3
4.1	Demo: 3Dsegmentation	3
4.2	Demo: bcc.....	4
4.3	Demo: c2d2_lite	5
4.4	Demo: centerpoint.....	7
4.5	Demo: classification	7
4.6	Demo: CLOCs.....	12
4.7	Demo: covid19segmentation.....	13
4.8	Demo: dpu_task/fadnet.....	14
4.9	Demo: dpu_task/psmnet	16
4.10	Demo: dpu_task/ssr.....	17
4.11	Demo: dpu_task/yolov3	18
4.12	Demo: facedetect.....	19
4.13	Demo: facefeature	20
4.14	Demo: facelandmark.....	21
4.15	Demo: facequality5pt	23
4.16	Demo: fairmot	24
4.17	Demo: graph_runner/platinum_graph_runner	25
4.18	Demo: graph_runner/resnet50_graph_runner.....	26
4.19	Demo: graph_runner/resnet50_graph_runner_py	27
4.20	Demo: graph_runner/tfssd_gridanchor_nms_op_graph_runner	28
4.21	Demo: hourglass.....	29
4.22	Demo: lanedetect.....	30
4.23	Demo: medicaldetection	31
4.24	Demo: medicalsegcell.....	32
4.25	Demo: medicalsegmentation.....	33
4.26	Demo: multitask	34
4.27	Demo: multitaskv3	35
4.28	Demo: openpose	36
4.29	Demo: platedetect.....	37
4.30	Demo: platinum	38
4.31	Demo: pmg	39
4.32	Demo: pointpainting.....	40
4.33	Demo: pointpillars	41
4.34	Demo: pointpillars_nuscenes	42
4.35	Demo: polypsegmentation	43
4.36	Demo: posedetect.....	44
4.37	Demo: rcan	45
4.38	Demo: refinedet	47
4.39	Demo: reid	48
4.40	Demo: retinaface	49
4.41	Demo: RGBD Segmentation	50
4.42	Demo: segmentation.....	51
4.43	Demo: solo	52
4.44	Demo: ssd	53
4.45	Demo: tfssd	54
4.46	Demo: ultrafast	55
4.47	Demo: yolov2.....	56

4.48	Demo: yolov3.....	56
4.49	Demo: yolov4.....	57
4.50	Demo: yolovx.....	58
5	References	59

Acknowledgement

Acknowledgement to the project StorAlge and the corresponding Czech institutional support project No. 8A21009.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 101007321. The JU receives support from the European Union's Horizon 2020 research and innovation programme and France, Belgium, Czech Republic, Germany, Italy, Sweden, Switzerland, Turkey.

1 Description

This document provides tutorial how to setup and run all Vitis AI Library 2.0 samples on Trezz TE0808 SoM attached to TEBF0808 carrier board.

2 Requirements

1. Hardware:
 - a. Trezz TE0808 SoM installed on TEBF0808 and power source.
 - b. Display Port Cable.
 - c. Display port monitor with FHD support.
 - d. USB webcam with USB cable, tested with See3CAM_CU30 - 3.4 Mpix Low Light USB Camera (Color).
 - e. Ethernet UTP cable.
 - f. 16GB SD card
2. Software:
 - a. Finished "Test 3: Vitis-AI Demo" [1] example from TE0808 StarterKit Vitis AI Tutorial, i.e. it is possible to run `dpu_trd (resnet50)` demo.
 - b. SD Card image created in "Test 3: Vitis-AI Demo" written to 16GB SD card and tested on `resnet50` demo.

3 How to Build Vitis AI Library Samples and Install Models

1. Get scripts `init.sh` and `vitis_ai_library_sample_build_all.sh`
2. Edit `init.sh` script and set correct paths to:
 - a. Vitis AI github repository path (see Vitis AI Starterkit Tutorial, the path should be `~/vitis_ai_2_0/`):

```
VITIS_AI_DIR=~/vitis_ai_2_0/
```

- b. Path to installed platform `SYSROOT` (see Vitis AI Starterkit Tutorial, the path should be `~/work/te0808_24_240/StarterKit_pfm`):

```
PLATFORM_SYSROOTS_DIR=~/work/te0808_24_240/StarterKit_pfm
```

3. Start downloading support files and building all examples:

```
./vitis_ai_library_sample_build_all.sh all
```

4. Connect UTP and power cable to TE0808+TEBF0808. Power on the board.
5. Connect your PC to TE0808 using SFTP.
6. Copy all 'samples' folder content to board using SFTP:

Copy all content of:

```
~/vitis_ai_2_0/demo/Vitis_AI_Library/samples
```

to target board TE0808 folder:

```
/home/root
```

7. In PC open folder `~/vitis_ai_2_0/models/AI-Model-Zoo/` and use script to get all available precompiled models from Xilinx, call:

```
python3 downloader.py
```

when asked for input fill: "all", then enter "0" for all and enter "2" for zcu102 & zcu104 & kv260. Wait until all models in form of tar.gz archives are downloaded.

IMPORTANT: Vitis library has error in one of 'yaml' metafiles. Before download process is started it is needed to fix it:

In folder model-list/pt_pointpainting_nuscenec_2.0

Open 'model.yaml' file for editing and replace complete line:

```
"download link: download link"
```

With line

```
download link:  
https://www.xilinx.com/bin/public/openDownload?filename=pointpainting_nuscenec_40000_64  
_0_pt-zcu102_zcu104_kv260-r2.0.0.tar.gz
```

8. Connect to target board TE0808 and create folder for models:

```
/usr/share/vitis_ai_library/models
```

9. Copy all downloaded *.tar.gz files to TE0808 board using SFTP to folder:

```
/usr/share/vitis_ai_library/models
```

10. Open ssh terminal to TE0808 board and continue on target board.
11. (Optional step) Set correct date and time:

```
date -s "2 OCT 2006 18:00:00"  
hwclock --systohc
```

12. Go to /usr/share/vitis_ai_library/models and extract all:

```
cat *.tar.gz | tar xvzf - -i
```

13. (Optional step) Remove archives to save space on SD card:

```
rm *.tar.gz
```

14. Set environment variables:

```
export XLNX_VART_FIRMWARE=/mnt/sd-mmcb1k1p1/dpu.xclbin
```

DISPLAY must be set only when X11 forwarding is NOT used:

```
export DISPLAY=:0.0
```

15. (recommended step) Test on one example - facedetect:

- a. Open readme file located in individual folder, as an example we use /home/root/facedetect, and find names of required models to run this demo, in this case readme says the valid models are:

Valid model name:
densebox_320_320
densebox_640_360

- b. Run example from file (as instructed in readme):

```
cd /home/root/facedetect
./test_jpeg_facedetect densebox_320_320 sample_facedetect.jpg
```

See result in file: 0_sample_facedetect_result.jpg at the same directory.

- c. Run example using webcam:

1. Connect screen using Display Port.
2. Check if linux desktop manager can be seen on screen, if not, reboot the board.
3. Connect USB webcam.
4. Run application:

```
./test_video_facedetect densebox_320_320 0 -t 1
```

or

```
./test_video_facedetect densebox_640_360 0 -t 1
```

NOTE: Parameter -t says how many threads will be used.
--

4 Tested Demos

The demos are described on Xilinx Web:

<https://docs.xilinx.com/r/2.0-English/ug1354-xilinx-ai-sdk/Model-Samples>

In following subsection we present results of running most of them. Each demo lists available models, then the command tested is presented. After that the input and output is shown and summary at the end describes what we have found. We usually execute test_jpeg_* on images and test_video_* using USB webcam. Other testing binaries 'test_performance_' and 'test_accuracy_' are not tested. Performance figures can be found on Xilinx web referenced above.

4.1 Demo: 3Dsegmentation

Xilinx description: The 3D segmentation library can support the SalsaNext model, which is used for the uncertainty-aware semantic segmentation of a full 3D LiDAR point cloud in real-time. SalsaNext is the next version of SalsaNet which has an encoder-decoder architecture, where the encoder unit has a set of ResNet blocks and the decoder unit combines upsampled features from the residual blocks.

Models:

salsanext_pt
salsanext_v2_pt

Command:

```
./test_jpeg_3Dsegmentation salsanext_pt scan_x.txt scan_y.txt scan_z.txt  
scan_remission.txt
```

Input:

scan_x.txt scan_y.txt scan_z.txt scan_remission.txt

Output:

0_3Dsegmentation_result.txt



```
192.168.211.130 - PuTTY  
root@petalinux:~/3Dsegmentation# ./test_jpeg_3Dsegmentation salsanext_pt scan_x.t  
xt scan_y.txt scan_z.txt scan_remission.txt  
width 2048 height 64  
scan_x.txt 127405  
scan_y.txt 127405  
scan_z.txt 127405  
scan_remission.txt 127405  
WARNING: Logging before InitGoogleLogging() is written to STDERR  
I0314 16:41:13.089134 1307 test_jpeg_3Dsegmentation.cpp:109] batch: 0  
root@petalinux:~/3Dsegmentation# █
```

Summary:

It is not very clear without further investigation what is meaning of input and output values.

4.2 Demo: bcc

Xilinx description: Bayesian Crowd Counting is a neural network that is used for counting the number of individuals in a crowd. The input is a picture of a crowd whose size you would like to estimate. The output is the estimated number of individuals in the crowd.

Models:

bcc_pt

Command:

```
./test_jpeg_bcc bcc_pt sample_bcc.jpg
```

Input:



Output:

Count of people in the crowd.

```
192.168.211.130 - PuTTY
root@petalinux:~/bcc# ./test_jpeg_bcc bcc_pt sample_bcc.jpg
WARNING: Logging before InitGoogleLogging() is written to STDERR
I0314 15:35:14.435221 1231 demo.hpp:1183] batch: 0 image: sample_bcc.jpg
I0314 15:35:14.435367 1231 process_result.hpp:33] RESULT: 2062
root@petalinux:~/bcc#
```

Summary:

Counts number of people in crowd in presented input image. Output is shown in terminal.

4.3 Demo: c2d2_lite

Xilinx description: Colonoscopy Coverage Deficiency via Depth algorithm, or C2D2, is a machine learning-based approach for improving colonoscopy coverage. The C2D2 network is a cascading structure. The inputs are 300 serialized gray images and the output is coverage. The C2D2_Lite_0_pt model is responsible for extracting the features of each image and the C2D2_Lite_1_pt model predicts a coverage value by inputting the characteristics of 300 pictures.

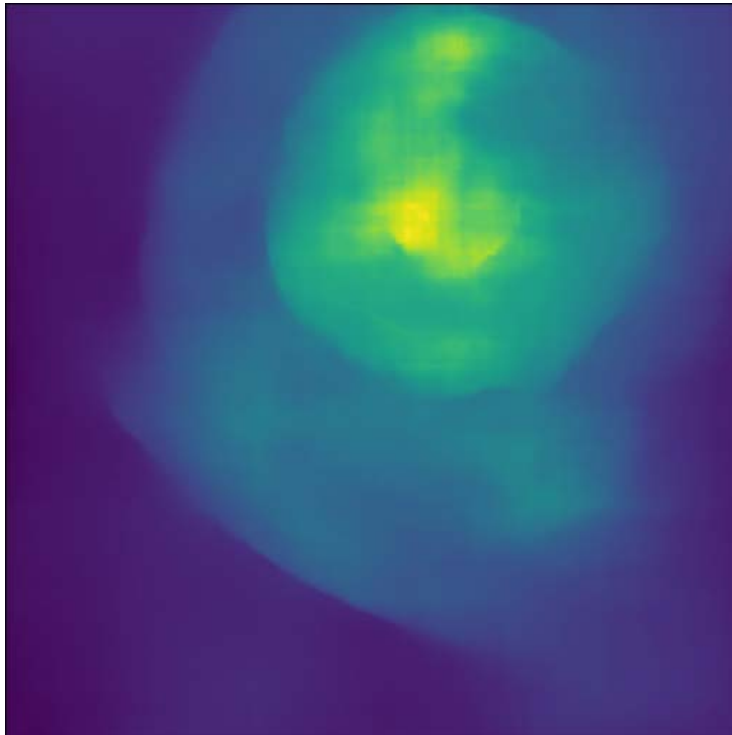
Models:

C2D2_Lite_0_pt
C2D2_Lite_1_pt

Command:

```
./test_jpeg_c2d2_lite C2D2_Lite_0_pt C2D2_Lite_1_pt image.list
```

Input (one of 300 images shown):



Output:

```
192.168.211.130 - PuTTY
root@petalinux:~/c2d2_lite# ./test_jpeg_c2d2_lite C2D2_Lite_0_pt C2D2_Lite_1_pt im
age.list
0.332031
root@petalinux:~/c2d2_lite#
```

Summary:

Demo takes list of 300 images and presents result as a number in terminal. Without more detailed study of the example is not clear what the input and output is.

4.4 Demo: centerpoint

Xilinx description: 4D radar is a high-resolution long-range radar sensor that not only detects the distance, relative speed, and azimuth of objects, but also their height above the road level. Unlike LiDAR, it works well in all weather conditions, including fog and heavy rain. A state-of-the-art anchor-free 3D object detector CenterPoint is used. It is trained on the 4D radar data of the open dataset Astyx. Because the annotated samples are limited and the 4D radar point clouds are sparse, the 3D bounding box prediction is naturally not so good. It is observed that although vehicles near ego cars could be correctly detected, there are still some false positive predictions and some objects at longer distances that could not be detected. 4D radar object detection and fusion with camera image could boost the performance by a large margin.

Models:

```
centerpoint_0_pt
centerpoint_1_pt
```

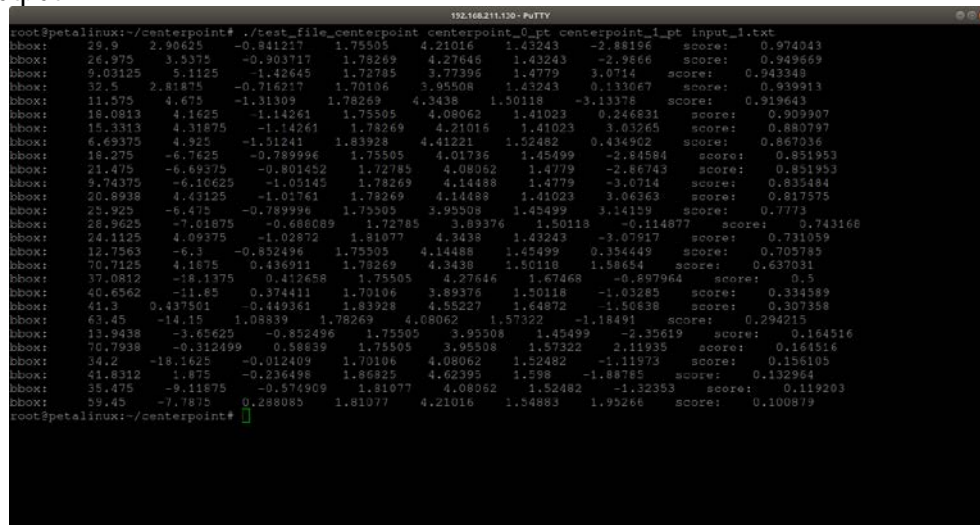
Command:

```
./test_file_centerpoint centerpoint_0_pt centerpoint_1_pt input_1.txt
```

Input:

```
input_1.txt
```

Output:



```
192.168.211.120: PuTTY
root@petalinux:~/centerpoint# ./test_file_centerpoint centerpoint_0_pt centerpoint_1_pt input_1.txt
bbox: 29.9 2.90625 -0.841217 1.75505 4.21016 1.43243 -2.88196 score: 0.974043
bbox: 26.975 3.5375 -0.903717 1.78269 4.27646 1.43243 -2.9666 score: 0.949669
bbox: 9.03125 5.1125 -1.42645 1.72785 3.77396 1.4779 3.0714 score: 0.943348
bbox: 32.5 2.81875 -0.716217 1.70106 3.95508 1.43243 0.133067 score: 0.939913
bbox: 11.575 4.675 -1.31309 1.78269 4.3438 1.50118 -3.13378 score: 0.919643
bbox: 18.0813 4.1625 1.14261 1.75505 4.08062 1.41023 0.246821 score: 0.909907
bbox: 15.3313 4.31875 -1.14961 1.78269 4.21016 1.41023 3.49265 score: 0.880797
bbox: 6.69375 4.925 -1.91241 1.83928 4.43221 1.52482 0.434902 score: 0.867036
bbox: 18.2075 -6.7625 -0.789996 1.75505 4.01736 1.45499 -2.64584 score: 0.851953
bbox: 21.475 -6.69375 -0.801452 1.72785 4.08062 1.4779 -2.86743 score: 0.851953
bbox: 9.74375 -6.10625 -1.05145 1.78269 4.14488 1.4779 -3.0714 score: 0.835484
bbox: 20.8938 4.43125 -1.01761 1.78269 4.14488 1.41023 3.06363 score: 0.817575
bbox: 25.925 -6.475 -0.789996 1.75505 3.95508 1.45499 3.14159 score: 0.7773
bbox: 28.9025 -7.01875 -0.686089 1.72785 3.89376 1.50118 -0.114877 score: 0.743166
bbox: 34.1125 4.95375 -1.02872 1.81077 4.3438 1.43243 -3.07917 score: 0.731059
bbox: 12.7563 -6.1 -0.852496 1.75505 4.14488 1.54999 0.354449 score: 0.705785
bbox: 70.7125 4.1875 0.436911 1.78269 4.3438 1.50118 1.58654 score: 0.637031
bbox: 37.0812 -18.1375 0.412658 1.75505 4.27646 1.67468 -0.897964 score: 0.5
bbox: 40.6562 -11.85 0.374411 1.70106 3.89376 1.50118 -1.03285 score: 0.334589
bbox: 41.9 0.437561 -0.449361 1.83928 4.55227 1.64872 -1.50828 score: 0.307358
bbox: 63.45 -14.15 1.08339 1.78269 4.08062 1.57322 -1.18491 score: 0.294215
bbox: 13.9438 -3.65625 -0.852496 1.75505 3.95508 1.45499 -2.35619 score: 0.164516
bbox: 70.7938 -0.312499 0.58839 1.75505 3.95508 1.57322 2.11935 score: 0.164516
bbox: 34.2 -18.1625 -0.012409 1.70106 4.08062 1.52482 -1.11973 score: 0.156105
bbox: 41.8312 1.875 -0.236498 1.86825 4.62395 1.598 -1.88785 score: 0.132964
bbox: 35.475 -9.11875 -0.574909 1.81077 4.08062 1.52482 -1.32353 score: 0.119203
bbox: 59.45 -7.7815 0.288085 1.81077 4.21016 1.54883 1.99266 score: 0.100819
root@petalinux:~/centerpoint#
```

Summary:

It is not very clear without further investigation what is meaning of input and output values.

4.5 Demo: classification

Xilinx description: The Classification library is used to classify images. Such neural networks are trained on ImageNet for ILSVRC and they can identify the objects from a thousand classes. The Vitis AI Library integrates networks including, but not limited to, ResNet18, ResNet50, Inception_v1, Inception_v2, Inception_v3, Inception_v4, VGG, mobilenet_v1, mobilenet_v2, and Squeezenet into Xilinx libraries. The input is a picture with an object and the output is the top-K most probable category.

Models:

resnet50
resnet18
inception_v1
inception_v2
inception_v3
inception_v4
mobilenet_v2
squeezenet——different test_ application needed, see below
inception_resnet_v2_tf
inception_v1_tf
inception_v2_tf
inception_v3_tf
inception_v4_2016_09_09_tf
mobilenet_v1_0_25_128_tf
mobilenet_v1_0_5_160_tf
mobilenet_v1_1_0_224_tf
mobilenet_v2_1_0_224_tf
mobilenet_v2_1_4_224_tf
mobilenet_edge_0_75_tf
mobilenet_edge_1_0_tf
resnet_v1_101_tf
resnet_v1_152_tf
resnet_v1_50_tf
resnet_v2_101_tf
resnet_v2_152_tf
resnet_v2_50_tf
vgg_16_tf
vgg_19_tf
MLPerf_resnet50_v1.5_tf
resnet50_tf2
inception_v3_tf2
mobilenet_1_0_224_tf2
squeezenet_pt——different test_ application needed, see below
resnet50_pt
inception_v3_pt
efficientNet-edgetpu-S_tf
efficientNet-edgetpu-M_tf
efficientNet-edgetpu-L_tf
efficientnet-b0_tf2
ofa_resnet50_0_9B_pt
person-orientation_pruned_558m_pt——classification of person orientation, see below
ofa_depthwise_res50_pt
mobilenet_v3_small_1_0_tf2

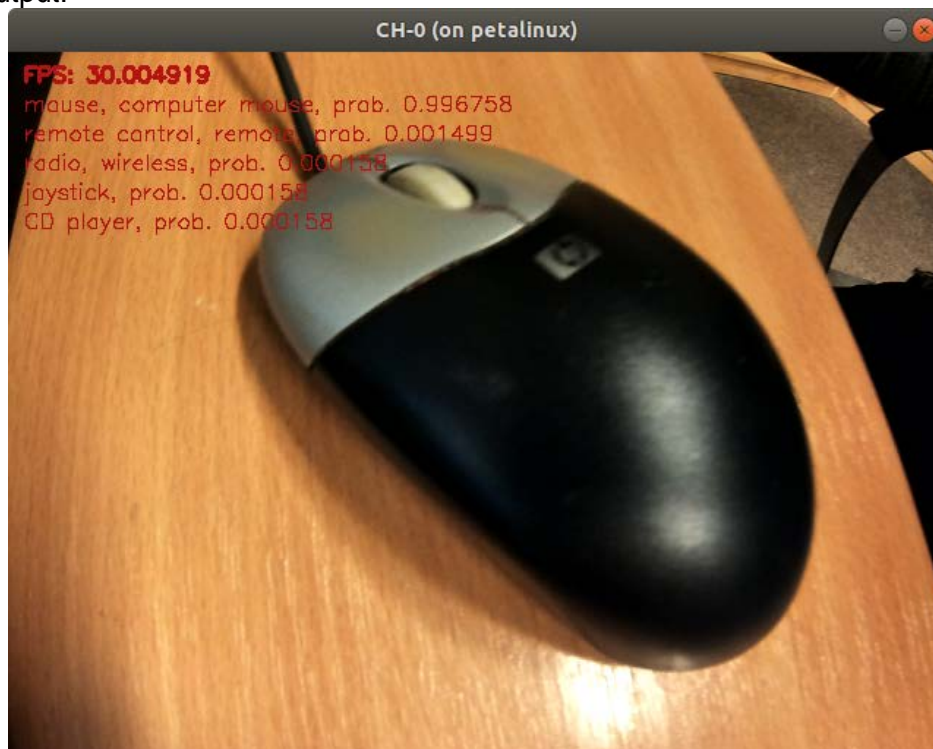
Command:

```
./test_video_classification resnet50 0 -t 1
```

Input:

USB webcam

Output:



Models:

squeezenet
squeezenet_pt

Command:

```
./test_jpeg_classification_squeezenet_squeezenet_pt sample_classification.jpg
```

Input:



Output:

```
192.168.211.130 - PuTTY
root@petalinux:~/classification# ./test_jpeg_classification_squeezenet_squeeze
t_pt sample_classification.jpg
squeezenet_pt 109 0.941306
squeezenet_pt 955 0.0433217
squeezenet_pt 5 0.00502098
squeezenet_pt 390 0.00331745
squeezenet_pt 328 0.00243838
root@petalinux:~/classification#
```

Models:

person-orientation_pruned_558m_pt

Command:

./test_jpeg_classification person-orientation_pruned_558m_pt sample_orientation.jpg

Input:



Output:

```
root@petalinux:~/classification# ./test_jpeg_classification person-orientation_pruned_558m_pt sample_o
rientation.jpg
WARNING: Logging before InitGoogleLogging() is written to STDERR
I0101 01:05:06.614094 1079 demo.hpp:1183] batch: 0 image: sample_orientation.jpg
I0101 01:05:06.614243 1079 process_result.hpp:24] r.index 1 Right, r.score 0.729709
I0101 01:05:06.614472 1079 process_result.hpp:24] r.index 0 Left, r.score 0.268445
I0101 01:05:06.614589 1079 process_result.hpp:24] r.index 2 Front, r.score 0.00180877
I0101 01:05:06.614712 1079 process_result.hpp:24] r.index 3 Back r.score 3.75398e-05
root@petalinux:~/classification#
```

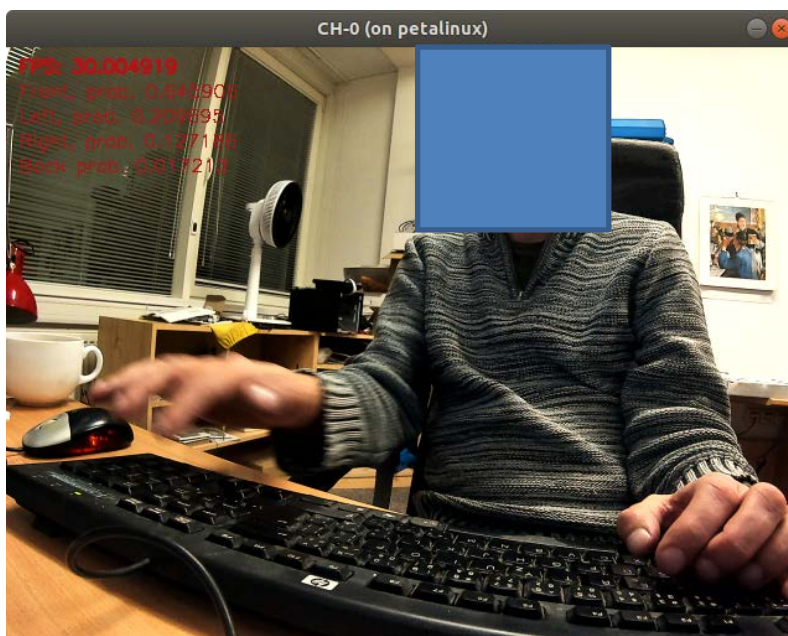
Command:

```
./test_video_classification person-orientation_pruned_558m_pt 0 -t 1
```

Input:

USB webcam

Output:



Summary:

Many different models can be used to classify image or classify on video from USB webcam. Result is shown in text overlay or in terminal. Models squeezenet and squeezenet_pt can run only on images and result is shown in terminal. Model person-

orientation_pruned_558m_pt is classifying orientation of person in image/video. Result is in terminal or text overlay.

4.6 Demo: CLOCs

Xilinx description: CLOCs is a novel Camera-LiDAR fusion method for 3D object detection in autonomous driving. Being fed with the predictions from the 2D detection pipeline (with camera image as input) and 3D detection pipeline (with LiDAR point cloud as input) in parallel, a light-weight fusion network is trained to fuse the 2D/3D prediction properly and refine the scores of the 3D detection results. CLOCs decouples the 2D/3D pipelines in the fusion framework, making it convenient to adopt different 2D/3D pipelines to strike a balance between accuracy and efficiency. The following images show the result of CLOCs.

Models:

```
model_0: clocs_yolox_pt
model_1: clocs_pointpillars_kitti_0_pt
model_2: clocs_pointpillars_kitti_1_pt
model_3: clocs_fusion_cnn_pt
```

Command:

```
./test_bin_clocs clocs_yolox_pt clocs_pointpillars_kitti_0_pt
clocs_pointpillars_kitti_1_pt clocs_fusion_cnn_pt ./000002.txt
```

Input:

Likely all files 000002.*

set of binary data (likely LiDAR) + corresponding camera image



Output:

Bounding boxes

```
192.168.211.130 - PuTTY
root@petalinux:~/clocs# ./test_bin_clocs clocs_yolox_pt clocs_pointpillars_kitti_0_pt clocs_pointpillars_kitti_1_pt clocs_fusion_cnn_pt ./000002.txt
batch 0
label:0 bbox: 34.7859 -3.22827 -2.10944 1.6252 3.9 1.512 1.59344 score:0.743168
label:0 bbox: 37.5671 -3.7788 -2.12793 1.63794 3.99249 1.50024 1.81219 score:0.0503306
root@petalinux:~/clocs#
```

Summary:

Demo requires set of image + LiDAR data to find bounding boxes coordinates.

4.7 Demo: covid19segmentation

Xilinx description: The Covid19 segmentation library can support the COVID-Net model which is a deep convolutional neural network design tailored for the detection of COVID-19 cases from chest X-ray (CXR) images.

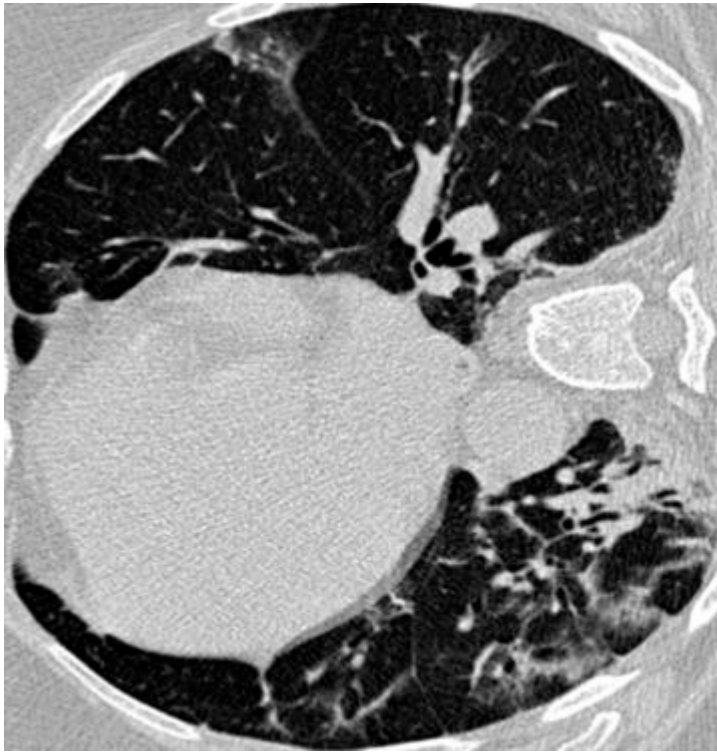
Models:

FPN-resnet18_covid19-seg_pt

Command:

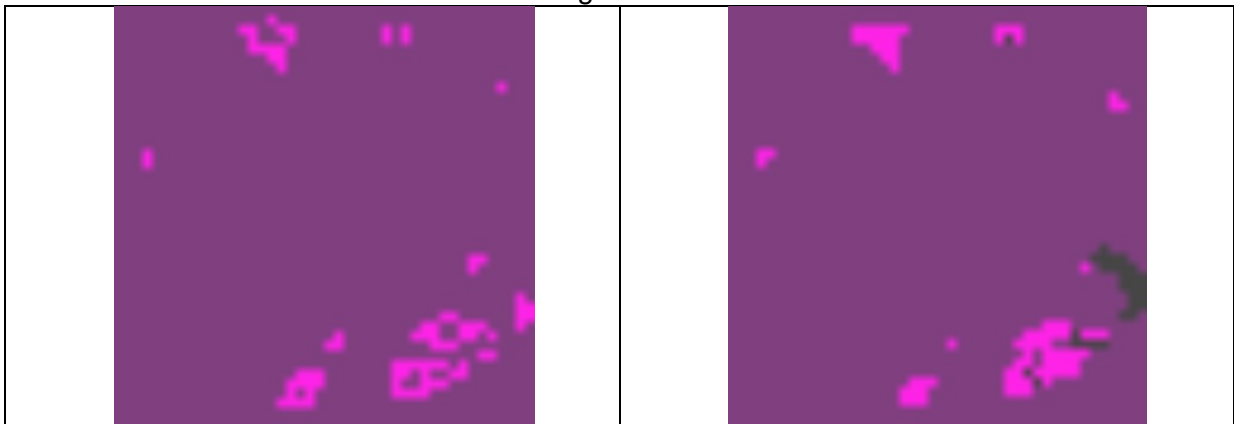
```
./test_jpeg_covid19segmentation FPN-resnet18_covid19-seg_pt
samples_covid19segmentation.jpg
```

Input:



Output:

Classification and infected area images



Summary:

Demo is returning classification result and infected area images.

4.8 Demo: dpu_task/fadnet

Xilinx description: FADNet is a model used for depth estimation. It is a fast and accurate network for disparity estimation. It has three main features:

- It exploits efficient 2D-based correlation layers with stacked blocks to preserve fast computation.
- It combines the residual structures to make the deeper model easier to learn.
- It contains multiscale predictions to exploit a multiscale weight scheduling training technique to improve the accuracy.

Models:

```
FADNet_0_pt
FADNet_1_pt
FADNet_2_pt
```

Pruned models:

```
FADNet_pruned_0_pt
FADNet_pruned_1_pt
FADNet_pruned_2_pt
```

Command:

```
./demo_fadnet demo_fadnet_left.png demo_fadnet_right.png
```

or with pruned model:

```
env
FADNET_MODEL_0=/usr/share/vitis_ai_library/models/FADNet_pruned_0_pt/FADNet_pruned_0_pt.xmodel
FADNET_MODEL_1=/usr/share/vitis_ai_library/models/FADNet_pruned_1_pt/FADNet_pruned_1_pt.xmodel
FADNET_MODEL_2=/usr/share/vitis_ai_library/models/FADNet_pruned_2_pt/FADNet_pruned_2_pt.xmodel ./demo_fadnet demo_fadnet_left.png demo_fadnet_right.png
```

Input:



Output:



Summary:

Output is depth estimation from input stereo images.

4.9 Demo: dpu_task/psmnet

Xilinx description: PSMNet is a pyramid stereo matching network that can be used for depth estimation. It consists of two main modules: spatial pyramid pooling and 3D CNN. The spatial pyramid pooling module takes advantage of the capacity of global context information by aggregating context in different scales and locations to form a cost volume. The 3D CNN regularizes cost volume using stacked multiple hourglass networks with intermediate supervision.

Models:

PSMNet_0_int
PSMNet_1_int
PSMNet_2_int

Command:

```
./demo_psmnet demo_psmnet_left.png demo_psmnet_right.png
```

Input:



Output:

-

```
192.168.211.130 - PuTTY
root@petalinux:~/dpu_task/psmnet# ./demo_psmnet demo_psmnet_left.png demo_psmnet_right.png
WARNING: Logging before InitGoogleLogging() is written to STDERR
F1201 12:39:57.775204 1207 serialize_v2.cpp:701] [UNILog][FATAL][XIR_READ_PB_FAILURE][failed to read pb file] file = /usr/share/vitis_ai_library/models/PSMNet_pruned_0_pt/PSMNet_pruned_0_pt.xmodel
*** Check failure stack trace: ***
Aborted
root@petalinux:~/dpu_task/psmnet#
```

Summary:

This demo is not working as neither „PSMNet_0_int“ nor „PSMNet_pruned_0_int“ are not available.

4.10 Demo: dpu_task/ssr

Xilinx description: Specular reflections that often appear in the endoscopy images can disturb the surgeon’s observation and judgment. The SSR model is an end-to-end network that can be used to remove the specular reflections in the endoscopy images thereby improving the image quality.

Models:

ssr_pt

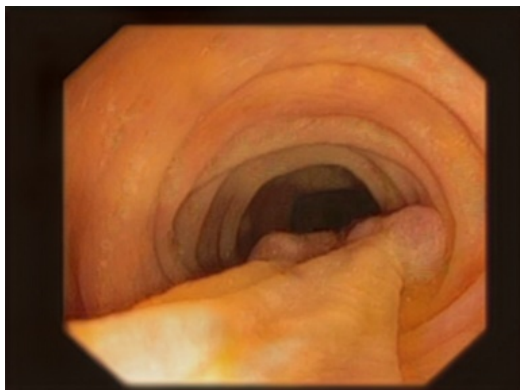
Command:

./demo_ssr /usr/share/vitis_ai_library/models/ssr_pt/ssr_pt.xmodel sample_ssr.bmp

Input:



Output:



Summary:

Demo removes reflections in image.

4.11 Demo: dpu_task/yolov3

Xilinx description: YOLOv3 is a neural network used to detect objects. The input is a picture with one or more objects and the output is a vector of the result struct which is composed of the detected information.

Models:

yolov3_voc

Command:

```
./demo_yolov3 /usr/share/vitis_ai_library/models/yolov3_voc/yolov3_voc.xmodel \  
demo_yolov3.jpg
```

Input:



Output:



Summary:
Demo shows yolov3 detecting objects in image.

4.12 Demo: facedetect

Xilinx description: The Face Detection library uses the DenseBox neural network to detect human faces. The input is a picture with the faces you want to detect and the output is a vector containing the information of each detection box.

Models:

densebox_320_320
densebox_640_360

Command:

`./test_video_facedetect densebox_640_360 0 -t 1`

Input:

USB webcam

Output:



Summary:

Demo detects faces in video stream or in images.

4.13 Demo: facefeature

Xilinx description: The face feature models are used for face recognition. They can extract the features of a person's face. The output of these models is 512 features. If you have two different images and you want to know if they are of the same person, use these models to extract features of the two images, and then use calculation functions and mapped functions to get the similarity of the two images.

Models:

```
facerec_resnet20  
facerec_resnet64  
facerec-resnet20_mixed_pt
```

Command:

```
./test_jpeg_facefeature facerec_resnet20 sample_facefeature.jpg
```

Input:



Output:



Output:



Summary:

Demo detects face landmarks in images.

4.15 Demo: facequality5pt

Xilinx description: The Face Quality library uses the face quality network to detect the quality score of a face. If a face is clear and a front face, the score is high. On the contrary, a blurry or side face will get a low score. The scores range from 0 to 1. It also provides face landmark positions. The input is a face that is detected by the face detect network and the output contains a quality score and five landmark key points.

Models:

face-quality
face-quality_pt

Command:

```
./test_jpeg_facequality5pt face-quality sample_facequality5pt.jpg
```

Input:



Output:



Summary:
Demo evaluates face quality and return also landmark points.

4.16 Demo: fairmot

Xilinx description: Fairmot is a multi-task model that can detect and get the re-ID features of the detected object at the same time. FairMot detects the person in the picture and provides the features of the detected target. This model can be used for tracking.

Models:

FairMot_pt

Command:

```
./test_video_fairmot FairMot_pt 0 -t 1
```

Input:

USB webcam

Output:



Summary:

Demo detects people in video and tracks them.

4.17 Demo: graph_runner/platenum_graph_runner

Xilinx description: The Plate Recognition library uses a classification network to recognize license plate numbers (Chinese license plates only). The input is a picture of the license plate that is detected by plate detect. The output is a structure containing license plate number information. The following image shows the result of the plate recognition.

Models:

plate_num

Command:

```
./platenum_graph_runner  
/usr/share/vitis_ai_library/models/plate_num/plate_num.xmodel sample_platenum.jpg
```

Input:



Output:

```
192.168.211.130 - PUTTY
root@petalinux:~/graph_runner/platenum_graph_runner# ./platenum_graph_runner /usr/share/vitis_ai_
library/models/plate_num/plate_num.xmodel sample_platenum.jpg
batch_index: 0
plate_color: Blue
plate_number: zhd22211
root@petalinux:~/graph_runner/platenum_graph_runner#
```

Summary:

Demo reads chinese licence plates. Demonstrates Vitis AI Library API_3 based on Graph_runner.

4.18 Demo: graph_runner/resnet50_graph_runner

Xilinx description: resnet50 classification using Vitis AI Library API_3 based on Graph_runner.

Models:

resnet50

Command:

```
./resnet50_graph_runner /usr/share/vitis_ai_library/models/resnet50/resnet50.xmodel
sample_classification.jpg
```

Input:



Output:

```
root@petalinux:~/graph_runner/resnet50_graph_runner# ./resnet50_graph_runner /usr/share/vitis_ai_
library/models/resnet50/resnet50.xmodel sample_classification.jpg
batch_index: 0
score[109] = 0.982666 text: brain coral,
score[373] = 0.00850172 text: coral reef,
score[353] = 0.00662115 text: jackfruit, jak, jack,
score[337] = 0.000543497 text: puffer, pufferfish, blowfish, globefish,
score[390] = 0.000329648 text: eel,
root@petalinux:~/graph_runner/resnet50_graph_runner#
```

Summary:

Image classification, model resnet50, demonstrates API_3 (graph runner).

4.19 Demo: graph_runner/resnet50_graph_runner_py

Xilinx description: Resnet50 clasification using Vitis AI Library API_3 based on Graph_runner. Demo is implemented using Python.

Models:

resnet50

Command:

```
python3 resnet50.py /usr/share/vitis_ai_library/models/resnet50/resnet50.xmodel
sample_classification.jpg
```

Input:



Output:

```
192.168.211.130 - PuTTY
root@petalinux:~/graph_runner/resnet50_graph_runner_py# python3 resnet50.py /usr/share/vitis_ai_1
library/models/resnet50/resnet50.xmodel sample_classification.jpg
Top[0] 109 brain coral
Top[1] 973 coral reef
Top[2] 955 jackfruit, jak, jack
Top[3] 397 puffer, pufferfish, blowfish, globefish
Top[4] 390 sea
root@petalinux:~/graph_runner/resnet50_graph_runner_py#
```

Summary:

Python implementation of previous demo, classification, model resnet50, demonstrates API_3 (graph runner).

4.20 Demo: graph_runner/tfssd_gridanchor_nms_op_graph_runner

Xilinx description: The SSD Detection library is commonly used with the SSD neural network. SSD is a neural network that is used to detect objects. The input is a picture with some objects you want to detect. The output is a vector of the resulting structure containing the information of each detection box. The following image shows the result of SSD detection. API_3 Graph runner is used.

Models:

ssd_mobilenet_v1_coco_tf_trt_op_b4096

Command:

tfssd_gridanchor_nms_op_graph_runner
ssd_mobilenet_v1_coco_tf_trt_op_b4096.xmodel sample_tfssd.jpg

Input:

-

Output:

-

Summary:

In demo is missing input image. If one provided it runs but says nothing.

4.21 Demo: hourglass

Xilinx description: The Hourglass library is used to detect the posture of the human body. It is represented by an array of 16 joint points. Joint points are arranged in order:

0 - r ankle, 1 - r knee, 2 - r hip, 3 - l hip, 4 - l knee, 5 - l ankle,
6 - pelvis, 7 - thorax, 8 - upper neck, 9 - head top, 10 - r wrist,
11 - r elbow, 12 - r shoulder, 13 - l shoulder, 14 - l elbow, 15 - l wrist

This network can detect the posture of only one person in the input image. The input of the network is 256x256.

Models:

hourglass-pe_mpii

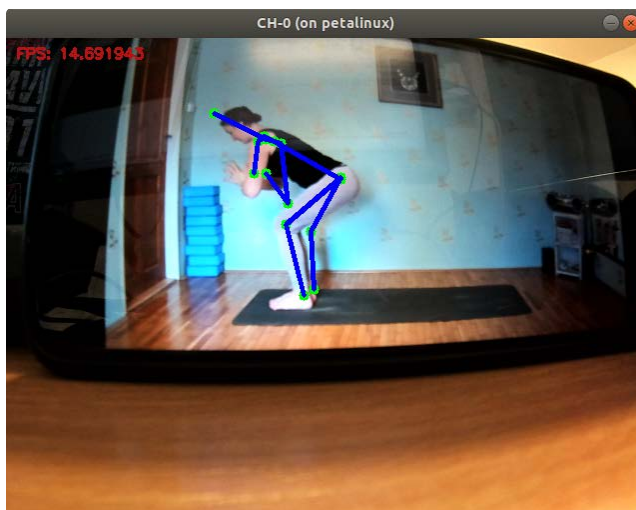
Command:

```
./test_video_hourglass hourglass-pe_mpii 0 -t1
```

Input:

USB Webcam

Output:



Summary:

Demo runs on video or images and detects body posture. Results are not very reliable and fast.

4.22 Demo: lanedetect

Xilinx description: The Road Line Detection library is used to draw lane lines in ADAS applications. Each lane line is represented by a number representing the category. A vector<Point> is used to draw the lane line. In the test code, a color map is used. Different types of lane lines are represented by different colors. The point is stored in the container vector, and the polygon interface `cv::polyline()` of OpenCV is used to draw the lane line.

Models:

vpgnet_pruned_0_99

Command:

```
./test_video_lanedetect vpgnet_pruned_0_99 0 -t 1
```

Input:

USB Webcam

Output:



Summary:

Demo runs on video or images detects lanes. It doesn't work very well for us. Possible reasons are: strong radial distortion of our camera or different road lines used in training data (chinese vs US vs europe).

4.23 Demo: medicaldetection

Xilinx description: The RefineDet model is based on vgg16. It is used for medical detection and can detect five types of diseases, namely, BE, cancer, HGD, polyp, and suspicious from an input endoscopy image like the Endoscopy Disease Detection and Segmentation database (EDD2020).

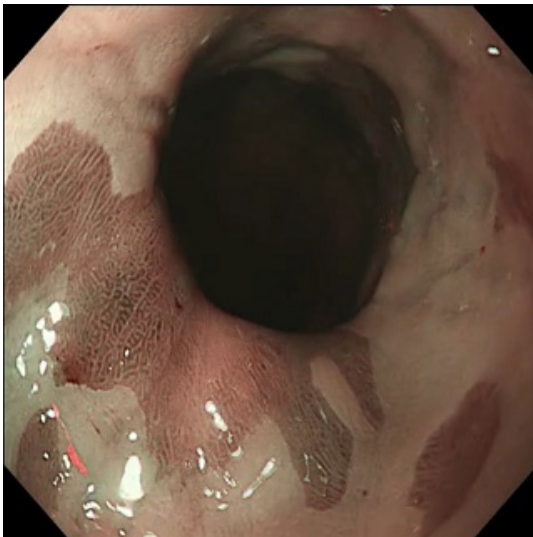
Models:

RefineDet-Medical_EDD_tf

Command:

```
./test_jpeg_medicaldetection RefineDet-Medical_EDD_tf  
sample_medicaldetection.jpg
```

Input:



Output:

```
192.168.211.130 - PuTTY
root@petalinux:~/medicaldetection# ./test_jpeg_medicaldetection RefineDet-Medical_EDD_tf sample_medicaldetection.jpg
WARNING: Logging before InitGoogleLogging() is written to STDERR
I1201 16:22:35.840950 1122 demo.hpp:1183] batch: 0 image: sample_medicaldetection.jpg
I1201 16:22:35.841382 1122 process_result.hpp:39] RESULT: BE 0.990099 -3.35321 72.3176 229.08 297.6
I1201 16:22:35.841666 1122 process_result.hpp:39] RESULT: BE 0.933227 7.67506 236.273 40.1888 309.044
I1201 16:22:35.841809 1122 process_result.hpp:39] RESULT: suspicious 0.90479 28.2807 142.651 105.215 226.621
I1201 16:22:35.841953 1122 process_result.hpp:39] RESULT: BE 0.847918 236.762 219.754 303.208 318.879
I1201 16:22:35.842097 1122 process_result.hpp:39] RESULT: BE 0.775535 288.085 88.6029 318.715 139.103
root@petalinux:~/medicaldetection#
```

Summary:

Demo runs on video or jpeg images. Testing video referred in readme is not present.

4.24 Demo: mediacsegcell

Xilinx description: The nucleus is an organelle present within all eukaryotic cells, including human cells. Aberrant nuclear shape can be used to identify cancer cells, for example, pap smear tests for the diagnosis of cervical cancer. Medical segmentation cell models offer nuclear segmentation in digital microscopic tissue images which can enable extraction of high-quality features for nuclear morphometric and other analyses in computational pathology. The following images show the results of cell segmentation.

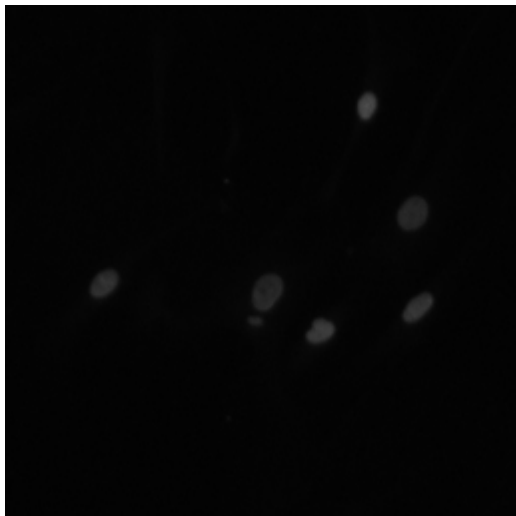
Models:

medical_seg_cell_tf2

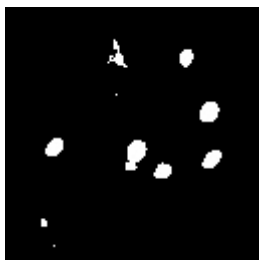
Command:

```
./test_jpeg_mediacsegcell medical_seg_cell_tf2 sample_mediacsegcell.png
```

Input:



Output:



Summary:

Demo runs on video or jpeg images, however video is not present.

4.25 Demo: medicalsegmentation

Xilinx description: Endoscopy is a common clinical procedure for the early detection of cancers in hollow organs such as nasopharyngeal cancer, esophageal adenocarcinoma, gastric cancer, colorectal cancer, and bladder cancer. Accurate and temporally consistent localization and segmentation of diseased region-of-interests enable precise quantification and mapping of lesions from clinical endoscopy videos, which is critical for monitoring and surgical planning.

The medical segmentation model is used to classify diseased region-of-interests in the input image. It can be classified into many categories, including BE, cancer, HGD, polyp, and suspicious.

libmedicalsegmentation is a segmentation library that can be used in the segmentation of multiclass diseases in endoscopy. It offers simple interfaces for developers to deploy segmentation tasks on Xilinx FPGAs. The following is an example of medical segmentation, where the goal is to mark the diseased region.

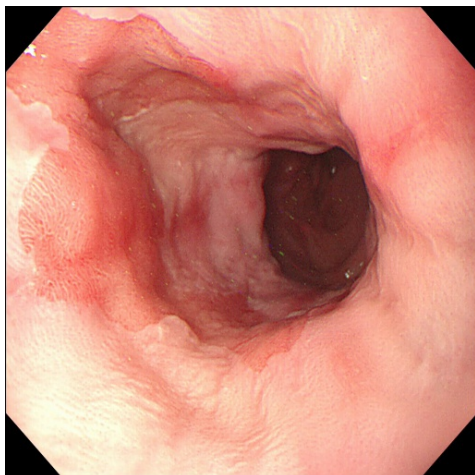
Models:

FPN_Res18_Medical_segmentation

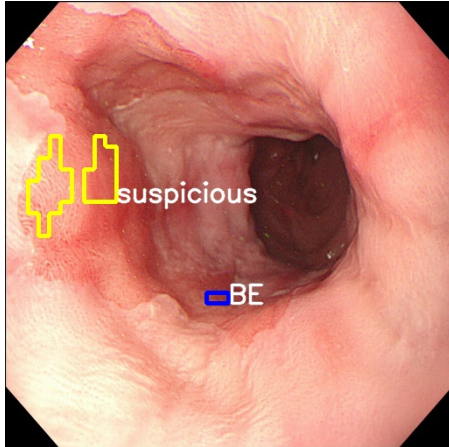
Command:

```
./test_jpeg_medicalsegmentation FPN_Res18_Medical_segmentation  
sample_medicalsegmentation.jpg
```

Input:



Output:



Summary:

Demo runs with jpeg images and result is written to jpeg file.

4.26 Demo: multitask

Xilinx description: The MultiTask library is appropriate for a model that has multiple subtasks. The MultiTask model in the Vitis AI Library has two subtasks: semantic segmentation and SSD detection. The following table lists the MultiTask models supported by the Vitis AI Library.

Models:

multi_task
MT-resnet18_mixed_pt

Command:

```
./test_video _multitask multi_task 0 -t 1
```

Input:

USB webcam

Output:



Summary:

Demo runs with jpeg images or video.

4.27 Demo: multitaskv3

Xilinx description: MultiTask V3 aims to do different tasks in autonomous driving scenarios simultaneously while achieving good performance and efficiency. The tasks include object detection, segmentation, lane detection, drivable area segmentation, and depth estimation, which are important components of the autonomous driving perception module.

Models:

multi_task_v3_pt

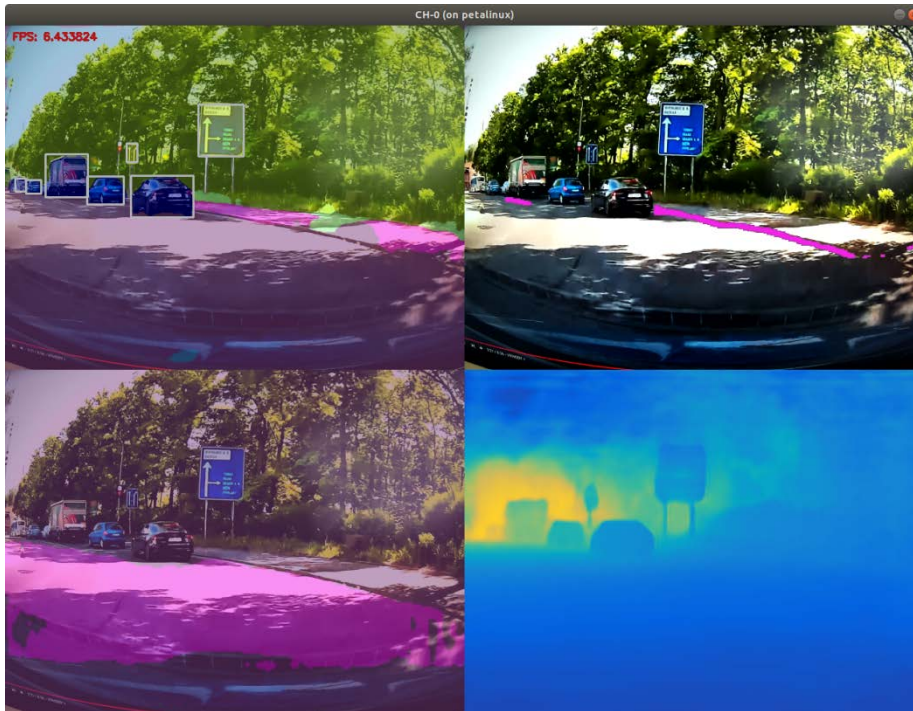
Command:

```
./test_video_multitaskv3 multi_task_v3_pt 0 -t 1
```

Input:

USB webcam

Output:



Summary:

Demo runs with jpeg images or webcam video.

4.28 Demo: openpose

Xilinx description: The Openpose Detection library is used to detect the posture of the human body. The posture is represented by an array of 14 key points as shown below:

0: head, 1: neck, 2: L_shoulder, 3:L_elbow, 4: L_wrist, 5: R_shoulder,
6: R_elbow, 7: R_wrist, 8: L_hip, 9: L_knee, 10: L_ankle, 11: R_hip,
12: R_knee, 13: R_ankle

The input of the network is 368x368.

Models:

openpose_pruned_0_3

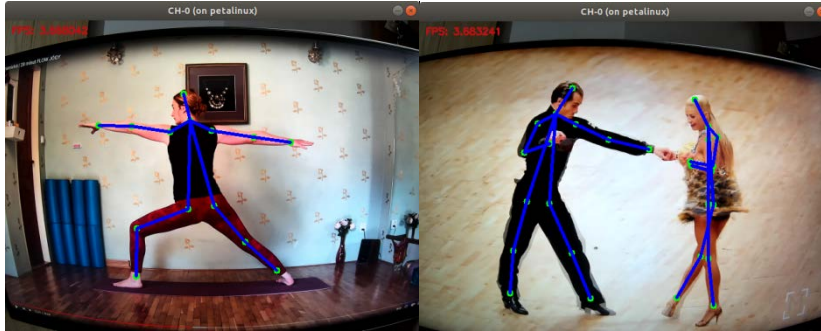
Command:

```
./test_video_openpose openpose_pruned_0_3 0 -t 1
```

Input:

USB webcam

Output:



Summary:

Demo runs with jpeg images or webcam video.

4.29 Demo: platedetect

Xilinx description: The Plate Detection library uses the DenseBox neural network to detect license plates. The input is a picture of the vehicle that is detected by the SSD and the output is a structure containing the plate location information.

Models:

plate_detect

Command:

```
./test_video_platedetect plate_detect 0 -t 1
```

Input:

USB webcam

Output:



Summary:

Demo runs with jpeg images or video. It is not working very well with our licence plates.

4.30 Demo: platenum

Xilinx description: The Plate Recognition library uses a classification network to recognize license plate numbers (Chinese license plates only). The input is a picture of the license plate that is detected by plate detect. The output is a structure containing license plate number information.

Models:

plate_num

Command:

```
./test_video_platenum plate_num 0 -t 1
```

Input:



Output:

```
192.168.211.130 - PuTTY
root@petalinux:~/platenum# ./test_jpeg_platenum plate_num samples_platenum.jpg
WARNING: Logging before InitGoogleLogging() is written to STDERR
I1201 19:05:43.811182 9440 demo.hpp:1183] batch: 0 image: samples_platenum.jpg
I1201 19:05:43.812283 9440 process_result.hpp:24] result.width 288 result.height 96
result.plate_color Blue result.plate_number zheDZ2211
root@petalinux:~/platenum#
```

Summary:

Demo runs with jpeg images or video. Works only with chinese license plates.

4.31 Demo: pmg

Xilinx description: PMG model can be used for fine-grained goods product recognition, for example, RP2K dataset. The model is Resnet18-based and the detailed model structure is shown in the picture below. On rp2k dataset, this model can achieve 96.4% top-1 float accuracy with 13.82M parameters and 2.28G Flops. Model final deployment and quantized top-1 accuracies are 96.19% and 96.18%, respectively.

Models:

pmg_pt

Command:

```
./test_jpeg_pmg pmg_pt sample_pmg.jpg
```

Input:



Output:

```
192.168.211.130 - PuTTY
root@petalinux:~/pmg# ./test_jpeg_pmg pmg_pt sample_pmg.jpg
WARNING: Logging before InitGoogleLogging() is written to STDERR
I1201 19:09:41.535651 9461 demo.hpp:1183] batch: 0 image: sample_pmg.jpg
I1201 19:09:41.535807 9461 process_result.hpp:33] RESULT: 133
root@petalinux:~/pmg#
```

Summary:

Demo runs with jpeg images and performs classification.

4.32 Demo: pointpainting

Xilinx description: For AD/ADAS systems, sensor-fusion algorithms play a significant role in providing high-quality perception and increasing the safety level for driving. PointPainting provides a sensor-fusion framework that takes advantage of 2D semantic segmentation and 3D object detection models. First, a network is applied to the camera images for semantic segmentation. Based on the semantic information and calibration information (on camera and LiDAR), the LiDAR point clouds are projected to the images and fused with the semantic information to get the painted point clouds. Finally, the painted point clouds are consumed by the 3D object detector to achieve better perception. PMG model can be used for fine-grained goods product recognition, for example, RP2K dataset. The model is Resnet18-based and the detailed model structure is shown in the picture below. On rp2k dataset, this model can achieve 96.4% top-1 float accuracy with 13.82M parameters and 2.28G Flops. Model final deployment and quantized top-1 accuracies are 96.19% and 96.18%, respectively.

Models:

```
seg_model: semanticfpn_nuimage_576_320_pt  
pointpillars_model_0: pointpainting_nuscenes_40000_64_0_pt  
pointpillars_model_1: pointpainting_nuscenes_40000_64_1_pt
```

Command:

```
./test_bin_pointpainting semanticfpn_nuimage_576_320_pt  
pointpainting_nuscenes_40000_64_0_pt pointpainting_nuscenes_40000_64_1_pt  
./sample_pointpainting.info
```

Input:

.info file contains references to camera and lidar data

Output:

```
192.168.211.130 - PUTTY
root@petalinux:~/pointpainting# ./test_bin_pointpainting semanticfpn_nuimage_376_320_pt_pointpainting_nuscenes_400
00_64_pt_pointpainting_nuscenes_40000_64_1_pt ./sample_pointpainting.info
batch: 0
label: 0 bbox: -6.36332 -23.8247 -2.50183 1.95018 4.32805 1.61833 3.39155 1.8125 7.75 score: 0.843895
label: 1 bbox: -7.192 22.072 -1.22268 2.79013 8.78905 3.36601 3.01655 0 0 score: 0.851953
label: 1 bbox: -7.688 27.1276 -1.24364 2.6145 6.73778 3.09355 2.95409 0 0 score: 0.743168
label: 7 bbox: -16.9891 14.5705 -1.62501 0.663449 0.640378 1.99149 1.11591 0.0625 0 score: 0.562177
label: 7 bbox: -14.5091 21.08 -1.01547 0.663449 0.640378 1.87083 3.2575 -0.0625 -0.125 score: 0.531209
label: 7 bbox: -15.7469 17.6695 -1.12531 0.623253 0.560132 1.87083 1.67841 0 0 score: 0.453262
label: 7 bbox: -14.2589 20.038 -0.928795 0.663449 0.640378 1.75748 0.863908 0 0 score: 0.377541
label: 7 bbox: -15.6854 19.1574 -0.358795 0.663449 0.60158 1.75748 5.26409 0.0625 0.0625 score: 0.307356
root@petalinux:~/pointpainting#
```

Summary:
Demo runs with data from camera and lidar.

4.33 Demo: pointpillars

Xilinx description: Object detection in point clouds is an important aspect of many robotics applications such as autonomous driving. The pointpillars model is a novel deep network and encoder that can be trained end-to-end on LiDAR point clouds. It offers the best architecture for 3D object detection from LiDAR.

Models:

PointNet: pointpillars_kitti_12000_0_pt

RPN: pointpillars_kitti_12000_1_pt

Command:

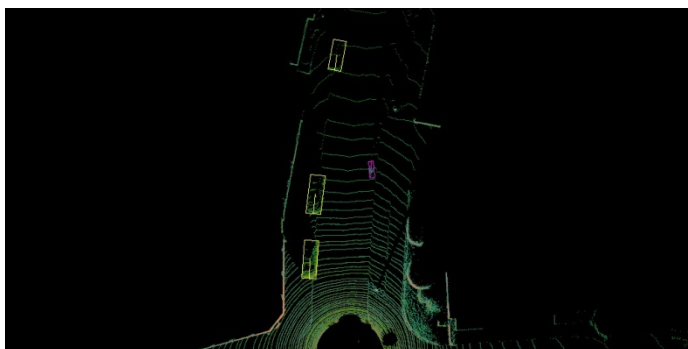
```
env XLNX_POINTPILLARS_PRE_MT=1 ./test_bin_pointpillars
pointpillars_kitti_12000_0_pt pointpillars_kitti_12000_1_pt sample_pointpillars.bin
sample_pointpillars.png
```

Input:



+ binary LiDAR data

Output:



Summary:

Demo runs with data from image and binary lidar data, returns objects found in lidar and image data.

4.34 Demo: pointpillars_nuscenes

Xilinx description: PointPillars is an efficient network for real-time 3D object detection on the point cloud. Trained on the nuScenes dataset, this model gives 3D bounding boxes and speed prediction for ten classes (including some kinds of vehicles, pedestrians, barriers, and traffic cones) in the surround-view range. With multisweep point clouds as input, PointPillars can achieve higher accuracy of 3D object detection and speed estimation at the cost of increasing the complexity of the pre-processing part.

Models:

model_0: pointpillars_nuscenes_40000_64_0_pt
model_1: pointpillars_nuscenes_40000_64_1_pt

Command:

```
./test_bin_pointpillars_nuscenes pointpillars_nuscenes_40000_64_0_pt  
pointpillars_nuscenes_40000_64_1_pt ./sample_pointpillars_nuscenes.info
```

Input:

Binary data listed in .info file

Output:

```
root@petalinux:~/pointpillars_nuscenes# ./test_bin_pointpillars_nuscenes pointpillars_nuscenes_40000_64_0_pt_pointpillars_nuscenes_40000_64_1_pt ./sample_pointpillars_nuscenes.info
batch : 0
label: 0 bbox: -0.743999 -23.8727 -2.17882 1.83202 4.32805 1.61833 3.14159 0 5.4375 score: 0.889797
label: 1 bbox: -17.112 44.336 -0.507348 3.05965 9.9593 3.36681 5.97069 0 0 score: 0.320821
label: 1 bbox: 46.9198 7.192 -2.00296 2.45609 6.32956 2.56464 0.8825 0 0 score: 0.320821
root@petalinux:~/pointpillars_nuscenes#
```

Summary:

Demo runs with data from binary data, returns coordinates of objects found (bboxes).

4.35 Demo: polypsegmentation

Xilinx description: HarDNet-MSEG is a new convolution neural network for polyp segmentation. It consists of a backbone and a decoder. The backbone is a low memory traffic CNN called HarDNet68, which has been successfully applied to various CV tasks including image classification, object detection, multi-object tracking, and semantic segmentation. The decoder part is inspired by the Cascaded Partial Decoder, which is known for fast and accurate salient object detection.

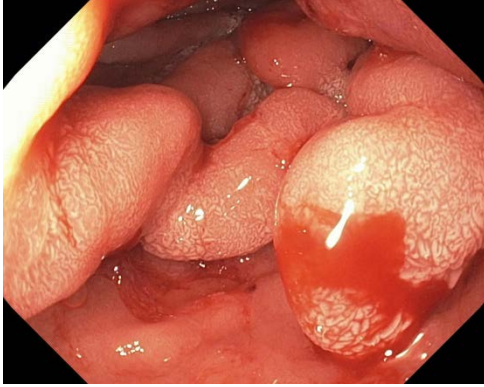
Models:

HardNet_MSeg_pt

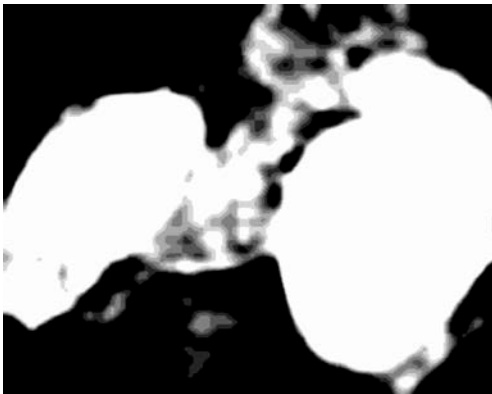
Command:

./test_jpeg_polypsegmentation HardNet_MSeg_pt sample_polypsegmentation.png

Input:



Output:



Summary:

Demo runs with jpeg or video, however video is not available.

4.36 Demo: posedetect

Xilinx description: The Pose Detection library is used to detect the posture of the human body. This library includes a neural network that can identify 14 key points on the human body (you can use our SSD detection library). The input is a picture that is detected by the pedestrian detection neural network. The output is a structure containing the coordinates of each point. The following image shows the result of pose detection.

Models:

sp_net

Command:

```
./test_video_posedetect_with_ssd 0 -t 1
```

Input:

USB webcam

Output:



Summary:

Demo provides test_jpeg and test_video executables. Both do expect output image from person detector. To run it together with ssd detector use test_video_posedetect_with_ssd as shown above.

4.37 Demo: rcan

Xilinx description: RCAN model is a super-resolution network. The corresponding high-resolution image is reconstructed from the low-resolution image. Based on the original image, the length and width are enlarged by two times. It has important application value in the fields of monitoring equipment, satellite images, and medical imaging. The following images show the result of RCAN. The image is still clear after zooming in.

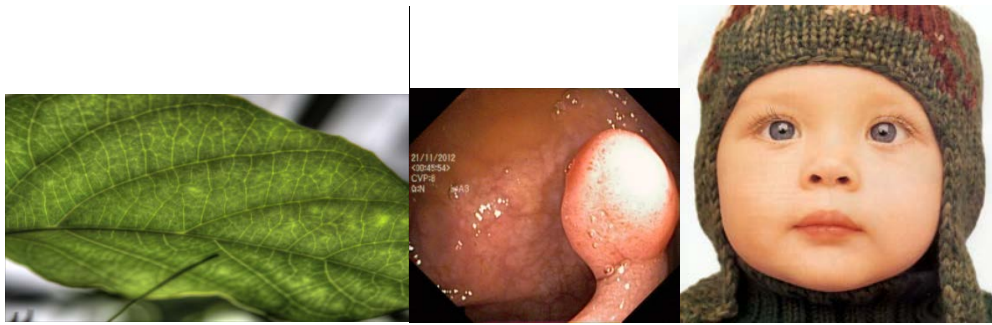
Models:

```
rcan_pruned_tf
drunet_pt
SESR_S_pt
```

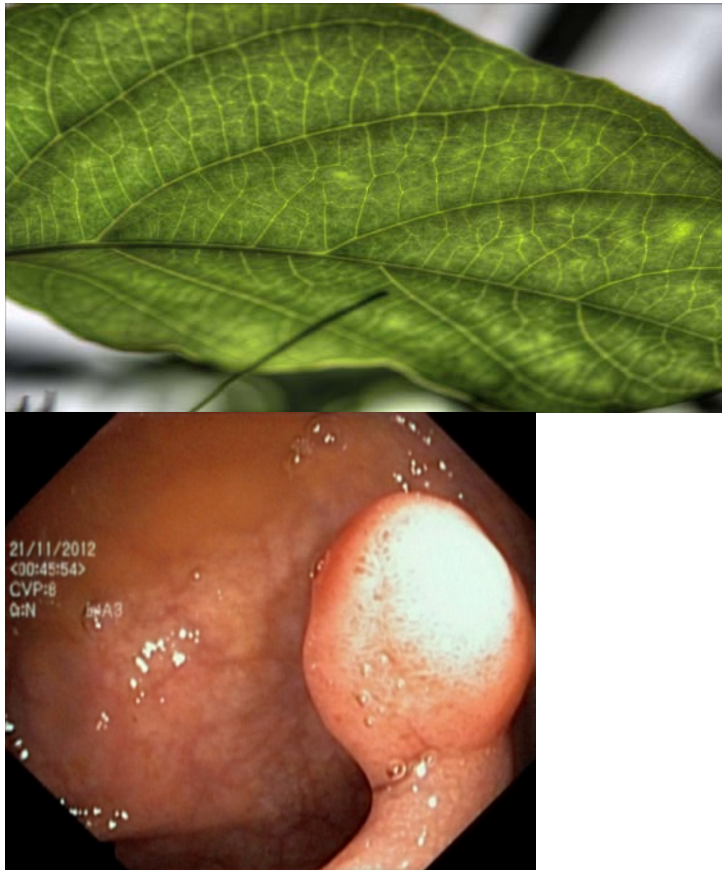
Command:

```
./test_jpeg_rcan rcan_pruned_tf sample_rcan.png
./test_jpeg_rcan drunet_pt sample_drunet.png
./test_jpeg_rcan SESR_S_pt sample_sesr.png
```

Input:



Output:





Summary:

Input image is processed and output image is 2x bigger in each dimension while still sharp.

4.38 Demo: refinedet

Xilinx description: RefinedDet is a neural network that is used to detect human bodies. The input is a picture with some individuals that you would like to detect. The output is a vector of the resulting structure that contains each box's information.

Models:

```
refinedet_baseline  
refinedet_pruned_0_8  
refinedet_pruned_0_92  
refinedet_pruned_0_96  
refinedet_VOC_tf
```

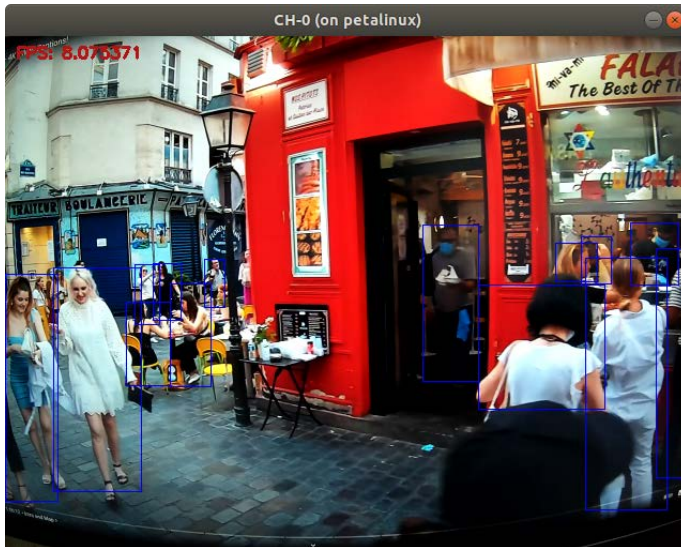
Command:

```
./test_video_refinedet refinedet_baseline 0 -t 1
```

Input:

USB webcam

Output:



Summary:

Detection of people in video or jpeg.

4.39 Demo: reid

Xilinx description: The task of person re-identification is to identify a person of interest at any time or place. This is done by extracting the image feature and comparing the features. Images of the same person should have similar features and have small feature distances, while images of different persons have large feature distances. Given a queried image and a pile of candidate images, the image that has the smallest feature distance is identified as the same person as the queried image. The following table lists the ReID detection models supported by the Vitis AI Library.

Models:

personreid-res50_pt
personreid-res18_pt
facereid-large_pt
facereid-small_pt

Command:

```
./test_jpeg_reid personreid-res50_pt sample_reid_001.jpg sample_reid_002.jpg  
./test_jpeg_reid facereid-large_pt face_reid_001.jpg face_reid_002.jpg
```

Input:



or



Output:

```
root@petalinux:~/reid# ./test_jpeg_reid personreid-res50_pt sample_reid_001.jpg sample_reid_002.jpg
WARNING: Logging before InitGoogleLogging() is written to STDERR
I1201 21:56:04.927155 753 test_jpeg_reid.cpp:91] batch: 0
I1201 21:56:04.927256 753 test_jpeg_reid.cpp:92] distmat : 0.567

root@petalinux:~/reid# ./test_jpeg_reid facereid-large_pt face_reid_001.jpg face_reid_002.jpg
WARNING: Logging before InitGoogleLogging() is written to STDERR
I1201 21:56:22.796289 756 test_jpeg_reid.cpp:91] batch: 0
I1201 21:56:22.796387 756 test_jpeg_reid.cpp:92] distmat : 1.021

root@petalinux:~/reid#
```

Summary:

Input are two images of person or of faces. Output to terminal is their similarity.

4.40 Demo: retinaface

Xilinx description: This retinaface network is used to detect human face and face landmarks. The input is a picture with some faces you would like to detect and the output contains face positions, scores, and landmarks of faces.

Models:

retinaface

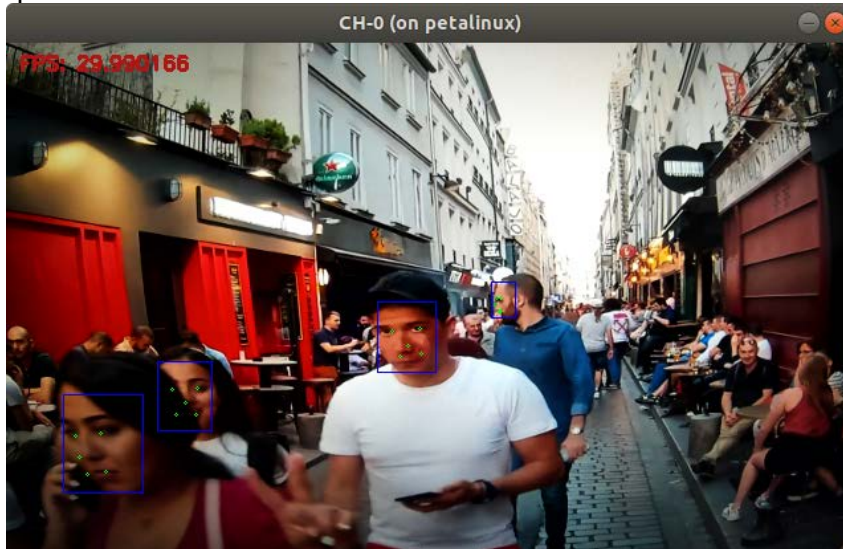
Command:

```
./test_video_retinaface retinaface 0 -t 1
```

Input:

USB webcam

Output:



Summary:

Detects faces and face landmarks in jpegs or video.

4.41 Demo: RGBD Segmentation

Xilinx description: SA-Gate is a neural network that is used for indoor segmentation. The input is a pair of an RGB image and an HHA map generated with the depth map. The output is a heat map where each pixel is predicted with a semantic category, like chair, bed, and other objects typically found indoors.

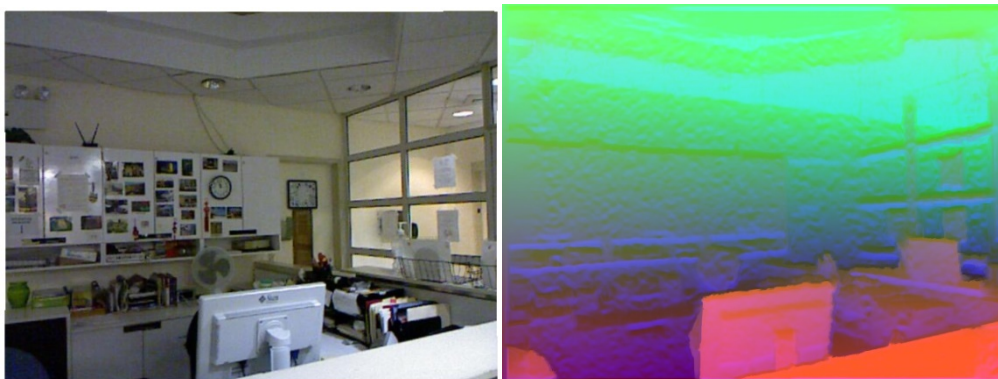
Models:

SA_gate_pt

Command:

```
./test_jpeg_RGBDsegmentation SA_gate_pt sample_rgbsegmentation_bgr.jpg  
sample_rgbsegmentation_hha.jpg
```

Input:



Output:

-

Summary:

Demo cannot work as model is not preset between precompiled models loaded in this tutorial.

4.42 Demo: segmentation

Xilinx description: Semantic segmentation assigns a semantic category to each pixel in the input image, that is, it identifies pixels as part of an object, say, a car, a road, a tree, a horse, etc. libsegmentation is a segmentation library that can be used in ADAS applications. It offers simple interfaces for a developer to deploy segmentation tasks on a Xilinx® FPGA.

The following is an example of semantic segmentation, where "blue-gray" denotes the sky, "green" denotes trees, "red" denotes people, "dark blue" denotes cars, "plum" denotes the road, and "gray" denotes structures.

Models:

```
fpn
semantic_seg_citys_tf2
UNET_chaos-CT_pt
FPN-resnet18_Endov
SemanticFPN_cityscapes_pt
ENet_cityscapes_pt
mobilenet_v2_cityscapes_tf
SemanticFPN_Mobilenetv2_pt
```

Command:

```
./test_video_segmentation fpn 0 -t 1
```

Input:

USB webcam

Output:



Summary:

Test shows semantic segmentation on video or jpeg.

4.43 Demo: solo

Xilinx description: Segment objects by locations (SOLO) is a simple and flexible framework applied for accomplishing instance segmentation in digital image processing and computer vision tasks. It is based on the notion of "instance categories" for instance segmentation in which each pixel within an instance of an object is assigned a category based on its location and size.

Models:

solo_pt

Command:

```
./test_video_solo solo_pt 0 -t 1
```

Input:

USB webcam

Output:

-

Summary:

Demo cannot work as model is not preset between precompiled models loaded in this tutorial.

4.44 Demo: ssd

Xilinx description: The SSD Detection library is commonly used with the SSD neural network. SSD is a neural network that is used to detect objects. The input is a picture with some objects you want to detect. The output is a vector of the resulting structure containing the information of each detection box. The following image shows the result of SSD detection.

Models:

```
ssd_pedestrian_pruned_0_97  
ssd_traffic_pruned_0_9  
ssd_adas_pruned_0_95  
ssd_mobilenet_v2  
mlperf_ssd_resnet34_tf
```

Command:

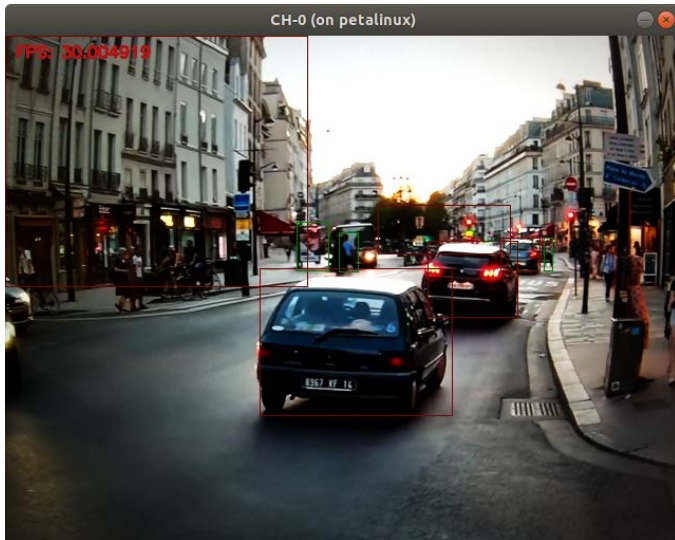
```
./test_video_ssd ssd_pedestrian_pruned_0_97 0 -t 1  
./test_video_ssd ssd_traffic_pruned_0_9 0 -t 1
```

Input:

USB webcam

Output:





Summary:

Object detection. Different models detect different objects.

4.45 Demo: tfssd

Xilinx description: The SSD Detection library is commonly used with the SSD neural network. SSD is a neural network that is used to detect objects. The input is a picture with some objects you want to detect. The output is a vector of the resulting structure containing the information of each detection box. The following image shows the result of SSD detection.

Models:

```
ssd_mobilenet_v1_coco_tf
ssd_mobilenet_v2_coco_tf
ssd_resnet_50_fpn_coco_tf
ssd_inception_v2_coco_tf
ssdlite_mobilenet_v2_coco_tf
mlperf_ssd_resnet34_tf -----not working
```

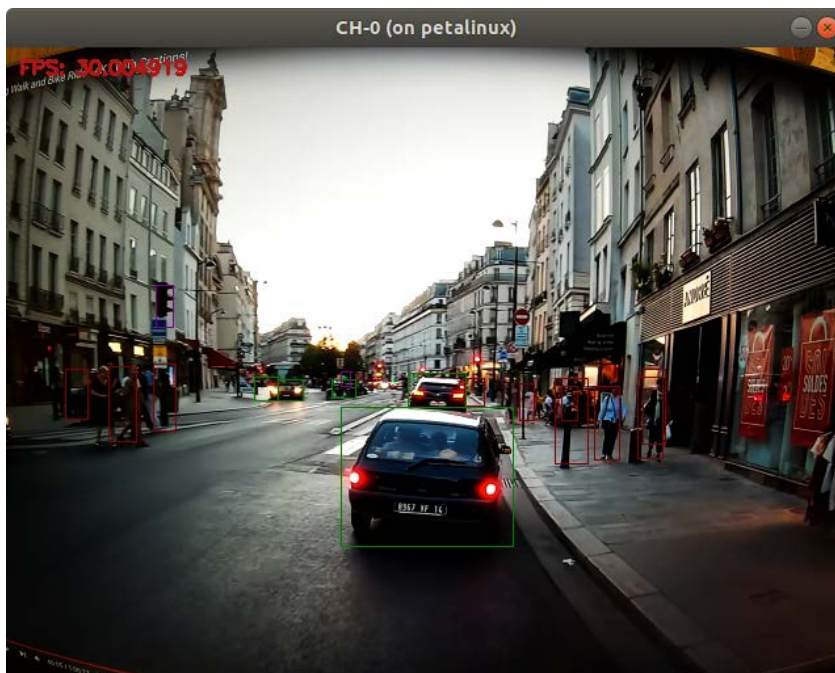
Command:

```
./test_video_tfssd ssd_mobilenet_v1_coco_tf 0 -t 1
./test_video_tfssd ssd_mobilenet_v2_coco_tf 0 -t 1
./test_video_tfssd ssd_resnet_50_fpn_coco_tf 0 -t 1
```

Input:

USB webcam

Output:



Summary:

SSD Object detection using tensorflow models.

4.46 Demo: ultrafast

Xilinx description: UltraFast Road Line Detection is a lane detection method that treats the process of lane detection as a row-based selection problem using global features. It can run at high FPS with comparable performance. The input is an image with a lane in it and the output is a structure holding the lane information. The following image shows the result of the UltraFast road line detection.

Models:

ultrafast_pt

Command:

```
./test_jpeg_ultrafast ultrafast_pt sample_ultrafast.jpg
```

Input:



Output:



Summary:

Detection of road lanes, file to file.

4.47 Demo: yolov2

Xilinx description: YOLOv2 does the same thing as YOLOv3, which is an upgraded version of YOLOv2. The following table lists the YOLOv2 detection models supported by the Vitis AI Library.

Models:

yolov2_voc
yolov2_voc_pruned_0_66
yolov2_voc_pruned_0_71
yolov2_voc_pruned_0_77

Command:

-

Input:

-

Output:

-

Summary:

Skipped as it does the same as YOLOv3

4.48 Demo: yolov3

Xilinx description: YOLOv3 is a neural network used to detect objects. The input is a picture with one or more objects and the output is a vector of the result structure which is composed of the detected information.

Models:

```
yolov3_adas_pruned_0_9  
yolov3_voc  
yolov3_bdd  
yolov3_voc_tf
```

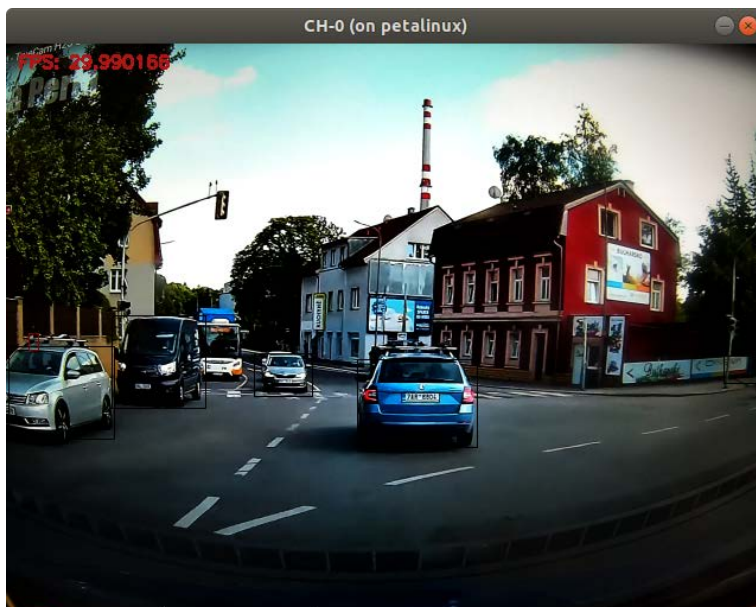
Command:

```
./test_video_yolov3 yolov3_adas_pruned_0_9 0 -t 1
```

Input:

USB webcam

Output:



Summary:

YOLOv3 detector.

4.49 Demo: yolov4

Xilinx description: YOLOv4 is an upgraded version of YOLOv3 and does the same thing as YOLOv3. The following table lists the YOLOv4 detection models supported by the Vitis AI Library.

Models:

```
yolov4_leaky_spp_m  
yolov4_leaky_spp_m_pruned_0_36
```

Command:

```
./test_video_yolov4 yolov4_leaky_spp_m 0 -t 1
```

Input:

USB webcam

Output:



Summary:

YOLOv4 detector.

4.50 Demo: yolovx

Xilinx description: YOLOX is an anchor-free version of YOLO, with a simpler design but better performance. It aims to bridge the gap between the research and industrial communities. Based on the best general detection framework of YOLOX, and modified the TSD-YOLOX network for Traffic Sign Detection. The input size of the model is 640*640, and the output is the score and coordinates of the object.

Models:

tsd_yolox_pt

Command:

```
./test_video_yolovx tsd_yolox_pt 0 -t 1
```

Input:

USB webcam

Output:



Summary:

YOLOvx detector, detects traffic signs.

5 References

- [1] TE0808 Starterkit Vitis AI Tutorial, Trenz Electronic Wiki: <https://wiki.trenz-electronic.de/display/PD/TE0808+Starterkit+Vitis+AI+Tutorial>