

BUILDING THE MICRO SD

The hackster tutorial below has been followed in order to get the HW and the Linux distribution:

<https://www.hackster.io/AlbertaBeef/ultra96-v2-building-the-foundational-designs-e4315f>

As the WIC wifi driver was not working, the following advice has been followed:

Known build issues

I often get an error during the petalinux build phase, caused by a failure to access packages from various github repositories. If this occurs, re-check your internet connection, and re-start the build as follows:

```
$ cd ~/Avnet_2022_2/petalinux/u96v2_sbc_base_2022_2
$ petalinux-build
```

I usually keep doing this until the build succeeds.

For the Ultra96-V2, I often am not able to build the following openamp-fw-* packages:

- openamp-fw-echo-testd
- openamp-fw-mat-muld
- openamp-fw-rpc-demo

In order to workaround this issue, I simply remove them from the petalinux-image definition. To do this, edit the following file:

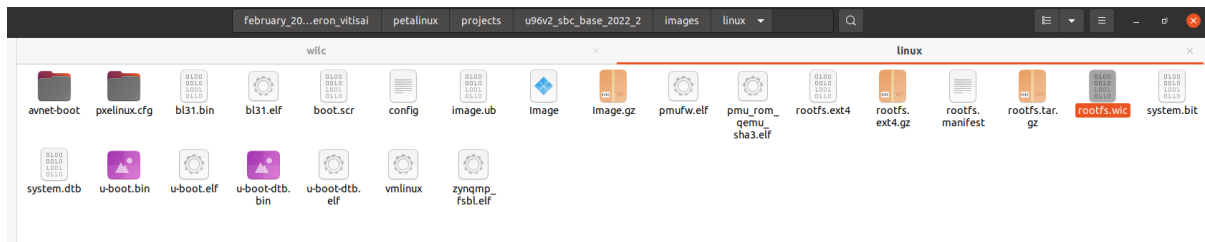
```
~/Avnet_2022_2/petalinux/u96v2_sbc_base_2022_2/project-spec/meta-avnet/recipes-core/images/petalinux-image-minimal.bbappend
```

Remove the "openamp-fw-*" packages from the "IMAGE_INSTALL:append:u96v2-sbc" entry:

```
IMAGE_INSTALL:append:u96v2-sbc = "\
    . . .
"
```

And the process has gone smoothly.

In the end, the "WIC" file is in the folder below:



It gets flashed on a micro SD with the following commands:

Give “`sudo fdisk -l`” to find out what is the micro SD:

```

Disk /dev/sdc: 59.7 GiB, 64088965120 bytes, 125173760 sectors
Disk model: STORAGE DEVICE
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x4e0663ea

Device      Boot      Start      End Sectors  Size Id Type
/dev/sdc1   *            2048 2000895 1998848   976M 83 Linux
/dev/sdc2             2000896 7340031 5339136   2.6G 83 Linux
caccollillo@caccollillo-OMEN-25L-Desktop-GT12-1xxx: ~/Documents/february_2025_test_bergeron_vitisai/petalinux/projects/u96v2_sbc_base_2022_2/images$ sudo fdisk -l

```

CREATING A BOOTABLE IMAGE ON A MICRO-SD

Go in the petalinux image folder and create the BOOT.BIN boot file as per instructions below:

- Change directory into **images/linux** within the lab1 project.

```
cd images/linux
```

- Once in the project enter:

```
$ petalinux-package --boot --fsbl zynqmp_fsbl.elf --u-boot u-boot.elf --pmufw pmufw.elf --fpga system.bit
```

```

trainag@training-VirtualBox: ~/AvnetTTC-u96/MPSoC_Petalinux/2021.2/Labi/images/linux$ petalinux-package --boot --fsbl zynqmp_fsbl.elf --u-boot u-boot.elf --pmufw pmufw.elf --fpga system.bit
[INFO] Sourcing buildtools
[INFO] Getting system flash information...
INFO: File in BOOT BIN: "/home/training/AvnetTTC-u96/MPSoC_Petalinux/2021.2/Labi/images/linux/zynqmp_fsbl.elf"
INFO: File in BOOT BIN: "/home/training/AvnetTTC-u96/MPSoC_Petalinux/2021.2/Labi/images/linux/pmufw.elf"
INFO: File in BOOT BIN: "/home/training/AvnetTTC-u96/MPSoC_Petalinux/2021.2/Labi/images/linux/system.bit"
INFO: File in BOOT BIN: "/home/training/AvnetTTC-u96/MPSoC_Petalinux/2021.2/Labi/images/linux/bl31.elf"
INFO: File in BOOT BIN: "/home/training/AvnetTTC-u96/MPSoC_Petalinux/2021.2/Labi/images/linux/system.dtb"
INFO: File in BOOT BIN: "/home/training/AvnetTTC-u96/MPSoC_Petalinux/2021.2/Labi/images/linux/u-boot.elf"
INFO: Generating zynqmp binary package BOOT.BIN...

***** Xilinx Bootgen v2021.2
***** Build date : Sep 30 2021-06:08:18
** Copyright 1986-2021 Xilinx, Inc. All Rights Reserved.

[INFO] : Bootimage generated successfully

[INFO] Binary is ready.
WARNING: Unable to access the TFTPBOOT folder /tftpboot!!!
WARNING: skip file copy to TFTPBOOT folder!!!
trainag@training-VirtualBox: ~/AvnetTTC-u96/MPSoC_Petalinux/2021.2/Labi/images/linux$

```

Figure 6 Generate **BOOT.BIN**

Create a bootable micro-SD card as per instructions in the following:

14. Now that the SD card is in the Virtual machine, we can use fdisk and disks to create and format our partitions. Open disks:

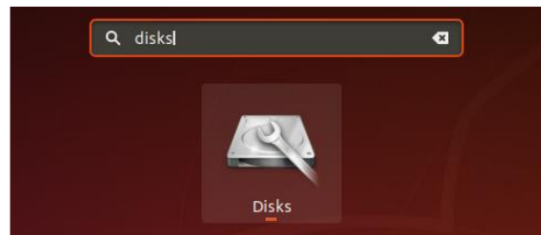


Figure 8 Open Disk Application

15. In disks we can see our SD card:

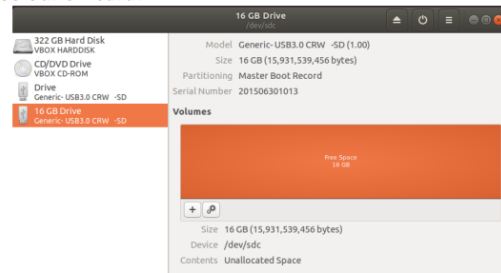


Figure 9 SD card in Disk Application

Reach Further™

Note if you already have a partition on the drive, we recommend you delete the partitions before continuing. Simply press the “minus” button (only shows if you have partitions) on the selected partition to delete them. Continue if you have “Free Space”

16. We are going to create the first Partition as FAT32. Simply press the plus button. We recommend that you have at least 1 GB for the boot files and leave the rest for the Linux OS.

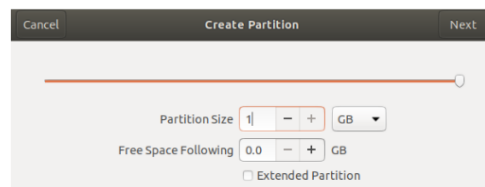


Figure 10 Create new partition 1GB

17. Next add a name for the boot partition. Leave the settings as default and press Create when done.

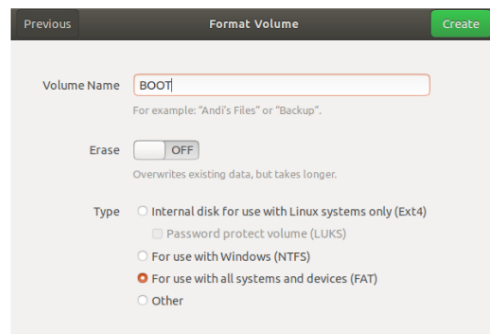


Figure 11 Name the new partition and FAT32 type

18. Now we create the second partition in the same way. Click on the free space and then the plus button.

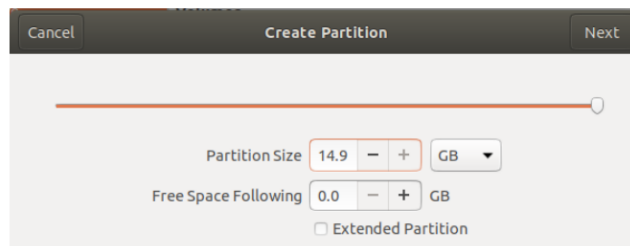


Figure 12 Create second partition with remaining free space

www.ultra96.org

Page 8

Install PetaLinux 2021.2 on Ultra96 - Lab 1



We are going to use the rest of the free space on the SD card for the Linux OS. Click next.

19. Name the Linux File System for the partition and **select the file system type as Ext4** click create once done.

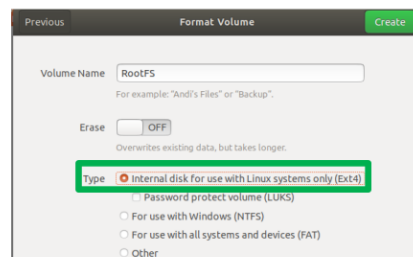


Figure 13 Name the Linux File System and EXT4

20. Open a terminal and enter:

```
sudo fdisk -l
```

This command will show all the drives on the virtual machine. Look for the SD card as we need the device name.

Device	Boot	Start	End	Sectors	Size	Id	Type
/dev/sdc1		2048	1955839	1953792	954M	c	W95 FAT32 (LBA)
/dev/sdc2		1955840	31115263	29159424	13.9G	83	Linux

Figure 14 Check in fdisk

21. Navigate into the images/Linux folder of our PetaLinux project. In our case

```
cd ~/AvnetTTC/home/training/AvnetTTC-u96/MPSoc_PetaLinux/2021.2/Lab1
```

21. Navigate into the images/Linux folder of our PetaLinux project. In our case

```
cd ~/AvnetTTC/home/training/AvnetTTC-u96/MPSoc_PetaLinux/2021.2/Lab1
```

22. Next is to enter the command below. Make sure the correct partition is selected. Our ext4 partition is mounted to /dev/sdc2. This took just under an hour to complete

```
sudo dd if=images/linux/rootfs.ext4 of=/dev/sdc2 status=progress
```

```
training@training-VirtualBox:~/AvnetTTC-u96/MPSoc_PetaLinux/2021.2/Lab1$ sudo dd if=images/linux/rootfs.ext4 of=/dev/sdc2 status=progress
[sudo] password for training:
Sorry, try again.
[sudo] password for training:
2343449888 bytes (2.3 GB, 2.2 GiB) copied, 3534 s, 663 kB/s
457779240 records in
457779240 records out
2343829504 bytes (2.3 GB, 2.2 GiB) copied, 3534.62 s, 663 kB/s
```

Figure 15 Flash Completed

While we wait for the flashing process to complete, we are going to copy the boot files into the boot partition that we created earlier. The boot files that we need to copy over is the boot.scr, BOOT.BIN and image.ub.

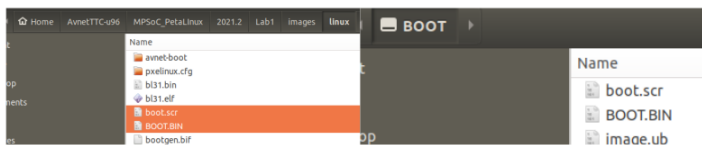


Figure 16 Copy the boot files in the BOOT partition

www.ultra96.org

Page 9

Insert the micro-SD card and boot as follows:

1. Power down the Ultra96 board by pressing and holding the power down, then change the Boot Mode switch on Ultra96 to 01 for SD Card as shown below.



Figure 17 Set switch to SD

2. Plug-in the USB Cable to the USB to JTAG/UART pod
3. Open Terra Term and connect to the serial port at 115200, No Parity and 1 Stop
4. Power on the Ultra96 by pressing the power on button. You should see the u-boot and PetaLinux boot messages scroll in the terminal window.

The port to use for the serial terminal is /dev/ttyUSB1.

COMBINING DESIGNS INTO A SINGLE FILE

In this article is shown how to combine several platforms together by using overlays:

<https://www.hackster.io/AlbertaBeef/ultra96-v2-combining-designs-into-a-single-platform-9d2572>

It is a solution to reprogram the FPGA with Linux still running:

<https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18841645/Solution+Zynq+PL+Programming+With+FPGA+Manager>

ADDING VITIS AI SUPPORT

In this article is shown how to add support to Vitis AI and to get a DPU working:

<https://www.hackster.io/AlbertaBeef/ultra96-v2-adding-support-for-vitis-ai-3-0-704799>

And in:

https://www.hackster.io/AlbertaBeef/ultra96-v2-adding-support-for-ros2-8ba68d?auth_token=6d7e7ae101ef33c3b16e9a5aa25a23b4&

Pre-Built SD image

The following link provided a pre-built image for the Ultra96-V2

- <http://avnet.me/avnet-u96v2-2022.2-sdimage>
(2023/05/10, md5sum = de17c497334da903790d702a5fae8f51)

- - - - -

A link to a prebuilt image is provided:

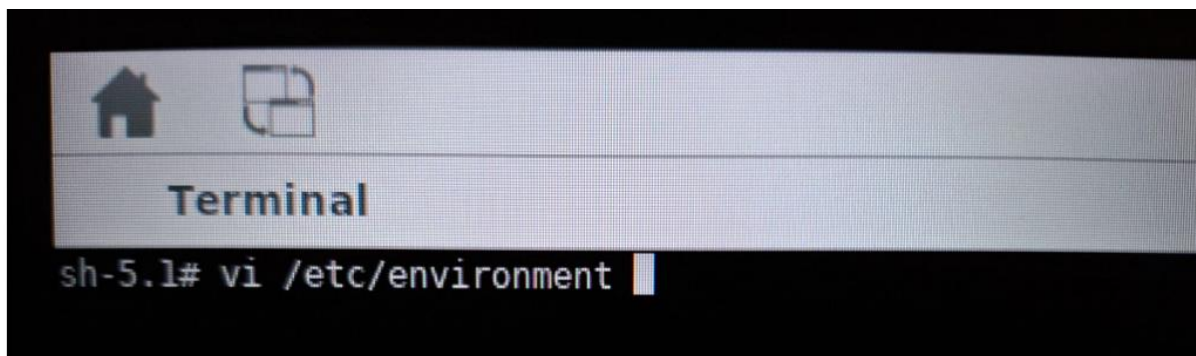
<https://avtinc.sharepoint.com/teams/ET-Downloads/Shared%20Documents/Forms/AllItems.aspx?id=%2Fteams%2FET%2DDo>

[wnloads%2FShared%20Documents%2Fxtm%5Fteam%2Fbergeron%2FVITIS%2Fvitis%5Fai%5F3%5F0%5Favnet%2Favnet%2Du96v2%5Fsb%2Dv2022%2E2%2D2023%2D05%2D10%2Ezip&parent=%2Fteams%2FET%2DDownloads%2FShared%20Documents%2Fxtm%5Fteam%2Fbergeron%2FVITIS%2Fvitis%5Fai%5F3%5F0%5Favnet&p=true&ga=1](#)

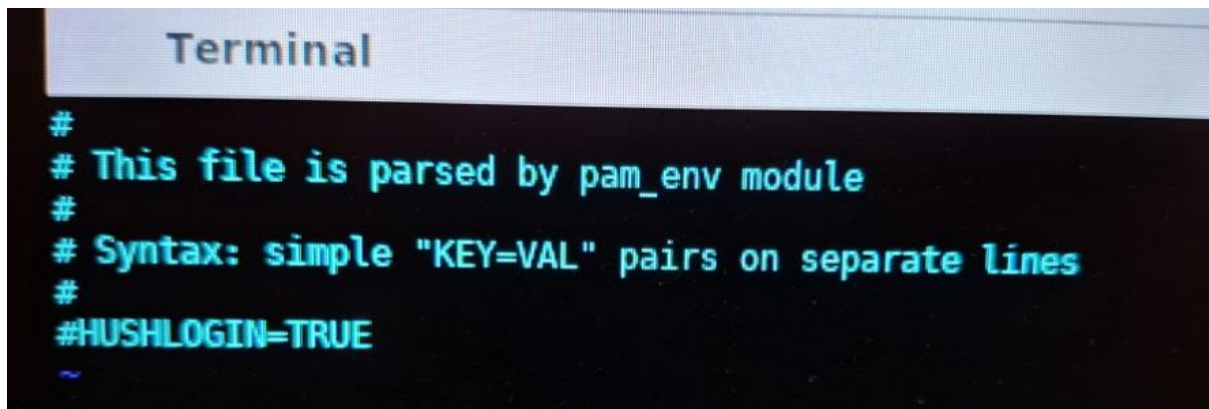
Once downloaded and extracted, it gets written to a MicroSD with Balena Etcher.

When the board boots up with a screen connected and a USB camera, a mouse and a keyboard, xserver (the graphic server) does not start.

To get it working give to the command line:



And use vi to edit the file adding:



The line with hushlogin.

Reboot the board and it will fix the graphics.

INSTALLING VITIS AI ON THE HOST

Starting with this tutorial:

<https://beetlebox.org/vitis-ai-using-tensorflow-and-keras-tutorial-part-2/>

Instructions about how to install the required SW must be followed:

<https://github.com/Xilinx/Vitis-AI/tree/3.0/docs>

The screenshot shows a web browser displaying the 'Host Installation Instructions' page for Vitis AI 3.0. The page is part of a documentation site with a dark sidebar on the left containing navigation links. The main content area has a white background and includes a title, an introductory paragraph, two installation options, a list of supported host types, an important note, and sections for prerequisites and preparing for installation.

Vitis™ AI
AMD
3.0

Search docs

SETUP AND INSTALL
Release Notes
System Requirements

Host Install Instructions

Pre-requisites
Preparing for the Installation
Docker Install and Verification
Clone The Repository
Leverage Vitis AI Containers

QUICK START GUIDES
Zynq™ Ultrascale+™
Versal™ VCK5000 Development Card
Versal™ AI Core VCK190

WORKFLOW AND COMPONENTS
Overview
DPU IP Details and System Integration
Vitis™ AI Model Zoo
Developing a Model for Vitis AI
Deploying a Model with Vitis AI

ADDITIONAL INFORMATION
Vitis™ AI User Guides & IP Product Guides
Vitis™ AI Developer Tutorials
Third-party Inference Stack Integration

Host Installation Instructions

The purpose of this page is to provide the developer with guidance on the installation of Vitis™ AI tools on the development host PC. Instructions for installation of Vitis AI on the target are covered separately in the Quickstart documentation for the respective target.

There are two primary options for installation:

[Option1] Directly leverage pre-built Docker containers available from Docker Hub: [xilinx/vitis-ai](https://hub.docker.com/r/xilinx/vitis-ai).

[Option2] Build a custom container to target your local host machine.

In addition, Vitis AI supports three host types:

- CPU-only with no GPU acceleration
- CUDA-capable GPUs
- AMD ROCm™ GPUs

Important

These installation instructions are only relevant for the current release of Vitis AI. If your intention is to pull a Docker container for a historic release you must leverage one of the previous release [Docker containers](#).

Pre-requisites

- Confirm that your development machine meets the minimum [Host System Requirements](#).
- Confirm that you have at least 100GB of free space in the target partition.

Preparing for the Installation

Refer to the relevant section (CPU-only, ROCm, CUDA) below to prepare your selected host for Docker installation.

QUICK START GUIDES

Zynq™ Ultrascale+™

Versal™ VCK5000 Development Card

Versal™ AI Core VCK190

WORKFLOW AND COMPONENTS

Overview

DPU IP Details and System Integration

Vitis™ AI Model Zoo

Developing a Model for Vitis AI

Deploying a Model with Vitis AI

ADDITIONAL INFORMATION

Vitis™ AI User Guides & IP Product Guides

Vitis™ AI Developer Tutorials

Third-party Inference Stack Integration

IP and Tools Compatibility

Frequently Asked Questions

Branching and Tagging Strategy

RESOURCES AND SUPPORT

Technical Support

Additional Resources

RELATED AMD SOLUTIONS

DPU-PYNQ

FINN & Brevitas

Inference Server

Unified Inference Frontend

Ryzen™ AI Developer Guide ~July 29

Vitis™ AI ONNX Runtime Execution Provider

Vitis™ Video Analytics SDK

CPU-only Host Initial Preparation

CPU hosts require no special preparation.

ROCm GPU Host Initial Preparation

For ROCm hosts, developers need to install ROCm. Vitis AI 3.0 supports ROCm v5.4.1.

The below steps describe the installation of ROCm for Ubuntu 20.04 hosts. If you are leveraging a different host operating systems, please refer to [the ROCm Installation Guide](#).

```
sudo apt-get update
wget https://repo.radeon.com/amdgpu-install/5.4.1/ubuntu/focal/amdgpu-install_5.4.50401-1_all.deb
sudo apt-get install ./amdgpu-install_5.4.50401-1_all.deb
sudo amdgpu-install --usecase=rocm
```

Add yourself to the render or video group using the following instructions:

```
sudo usermod -a -G render $LOGNAME
-OR-
sudo usermod -a -G video $LOGNAME
```

Note

Use of the video group is recommended for all ROCm-supported operating systems.

Add the user to the render group for Ubuntu 20.04:

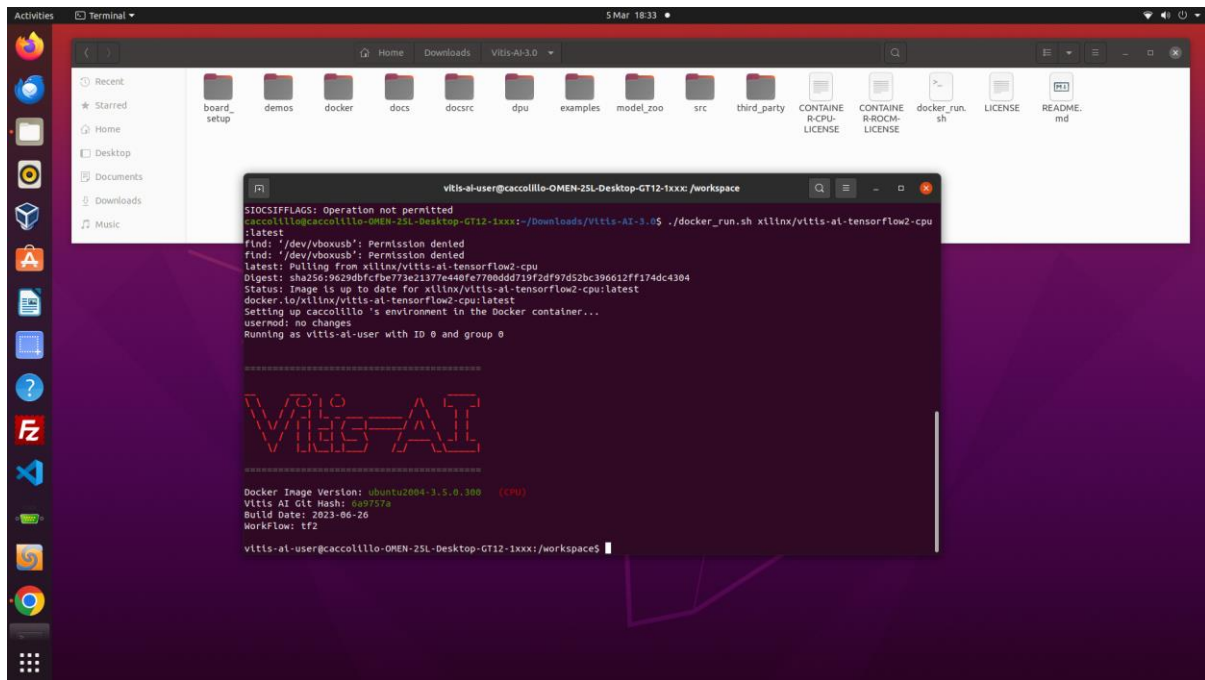
```
sudo usermod -a -G render $LOGNAME
```

Make sure to reboot the machine after installing the ROCm kernel package to force the new kernel to load on reboot. You can verify the ROCm kernel is loaded by typing the following command at a prompt:

```
lsmod | grep amdgpu
```

You may also refer to the [ROCm Docker installation documentation](#) for further details.

Once done, the docker image can be started:



VITIS AI TUTORIAL

This tutorial shows how to start from an h5 model:

<https://github.com/Xilinx/Vitis-AI-Tutorials/tree/3.0/Tutorials/RESNET18/>

<https://www.xilinx.com/developer/articles/medical-ai-application-acceleration-with-xilinx-vitis-ai.html>

<https://community.element14.com/products/roadtest/b/blog/posts/amd-xilinx-kria-kv260-vision-ai-starter-kit-software>

Activate tensorflow 2 (conda activate ...):



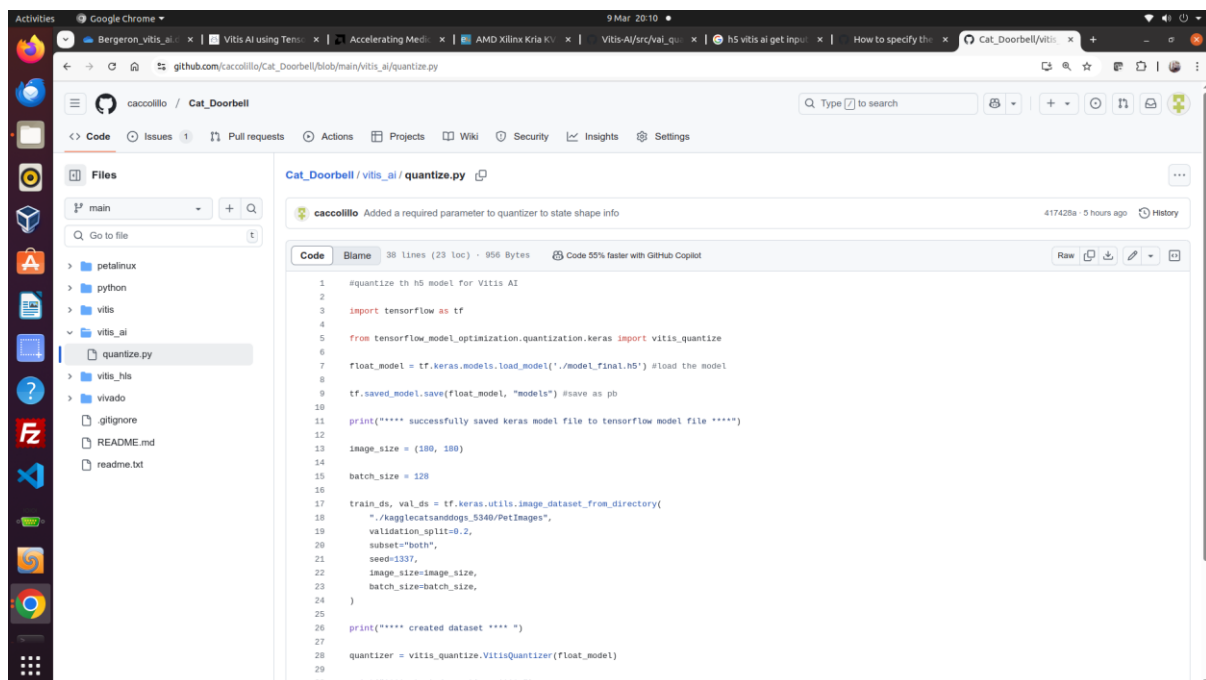
VITIS AI QUANTIZER

The h5 model has to be quantized first:

<https://docs.amd.com/r/en-US/ug1414-vitis-ai/Quantize>

https://github.com/Xilinx/Vitis-AI/tree/3.0/src/vai_quantizer/vai_q_tensorflow2.x

A script has been created to quantize the h5 model:

A screenshot of a web browser displaying a GitHub repository page for 'caccollilo / Cat_Doorbell'. The repository is viewed on the 'Code' tab, showing the file 'vitis_ai / quantize.py'. The file has 38 lines of code, 23 localizations, and 956 bytes. A commit message from 'caccollilo' is visible: 'Added a required parameter to quantizer to state shape info'. The code itself is a Python script that quantizes a Keras H5 model using Vitis AI. It includes imports for TensorFlow, Keras, and Vitis AI quantization. The script loads a model, saves it as a TensorFlow SavedModel, and then uses the Vitis AI quantizer to create a quantized model. The quantized model is saved as 'quantized_model.h5'. The script also includes a dataset loading section for training and validation.

```
1 #quantize th h5 model for Vitis AI
2
3 import tensorflow as tf
4
5 from tensorflow_model_optimization.quantization.keras import vitis_quantize
6
7 float_model = tf.keras.models.load_model('./model_final.h5') #load the model
8
9 tf.saved_model.save(float_model, "models") #save as pb
10
11 print("**** successfully saved keras model file to tensorflow model file ****")
12
13 image_size = (180, 180)
14
15 batch_size = 128
16
17 train_ds, val_ds = tf.keras.utils.image_dataset_from_directory(
18     './agglcatsanddogs_5340/PetImages',
19     validation_split=0.2,
20     subset="both",
21     seed=1337,
22     image_size=image_size,
23     batch_size=batch_size,
24 )
25
26 print("**** created dataset **** ")
27
28 quantizer = vitis_quantize.VitisQuantizer(float_model)
29
30 print("**** started quantizer **** ")
```

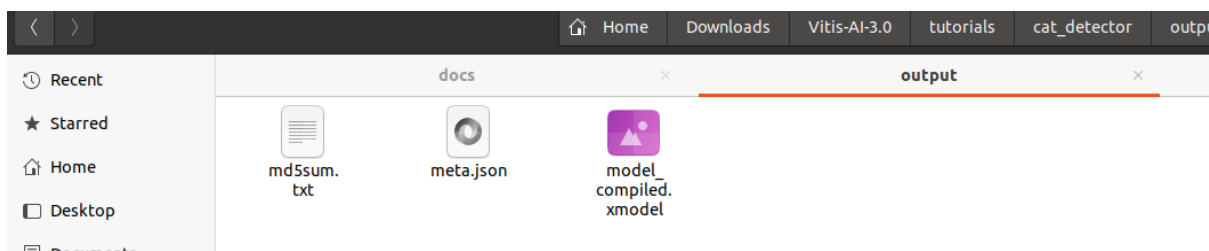
Once started, it creates the quantized model.

Then in the docker, launch the following command to compile the model:

```
vai_c_tensorflow2 -m ./quantized_model.h5 -a
/opt/vitis_ai/compiler/arch/DPUCZDX8G/KV260/arch.json -o output/ -n
model_compiled --options '{"mode':'normal','save_kernel':",
'input_shape':'128,180,180,3'}"
```

```
(vitis-ai-tensorflow) vitis-ai-user@accolillo-OPEN-ZSL-Desktop-GT12-xxx:/workspace/tutorials/cat_detector$ val_c_tensorflow2 -n ./quantized_model.hs -a /opt/vitis-ai/compiler/arch/DPU CZDX8G_KV260/arch
.json -o output/ -n model_compiled --options '{"mode":"normal","save_kernel":true,"input_shape":"128,180,180,3"}'
*****
* Vitis AI Compilation - Xilinx Inc.
*****
[INFO] Namespace(batchsize=1, inputs shape=[128,180,180,3], layout='NHWC', model_files=['./quantized_model.hs'], model_type='tensorflow2', named_inputs_shape=None, out_filename='/tmp/model_compiled_DPU
CZDX8G_ISA1_B4096_org.xmodel', proto=None)
in shapes: [[128, 180, 180, 3]]
[INFO] tensorflow2 model: /workspace/tutorials/cat_detector/quantized_model.hs
[INFO] keras version: 2.12.0
[INFO] Tensorflow Keras model type: functional
[INFO] parse raw model :100% | 42/42 [00:00:00:00, 23516.32it/s]
[INFO] infer shape (NHWC) :100% | 71/71 [00:00:00:00, 635.24it/s]
[INFO] perform level-0 opt :100% | 1/1 [00:00:00:00, 279.56it/s]
[INFO] perform level-1 opt :100% | 2/2 [00:00:00:00, 500.99it/s]
[INFO] infer shape (NHWC) :100% | 75/75 [00:00:00:00, 802.15it/s]
[INFO] generate xmodel : 0% | 0/75 [00:00:00:00, 71it/s]
WARNING: Logging before initGoogleLogging() is written to STDERR
420250309 00:21:29.277642 2738 tool_function.cpp:171] [UNILOG][WARNING] The operator named rescaling, type: Rescaling, is not defined in XIR. XIR creates the definition of this operator automatically. Y
ou should specify the shape and the data type of the output tensor of this operation by set_attr("shape", std::vector<int>) and set_attr("data_type", std::string)
[INFO] generate xmodel :100% | 75/75 [00:00:00:00, 2313.46it/s]
[INFO] dump xmodel: /tmp/model_compiled_DPU CZDX8G_ISA1_B4096_org.xmodel
[UNILOG][WARNING] The operator named rescaling, type: Rescaling, is not defined in XIR. XIR creates the definition of this operator automatically. You should specify the shape and the data type of the ou
put tensor of this operation by set_attr("shape", std::vector<int>) and set_attr("data_type", std::string)
[UNILOG][INFO] compile mode: dpu
[UNILOG][INFO] debug mode: null
[UNILOG][INFO] Target architecture: DPU CZDX8G_ISA1_B4096
[UNILOG][INFO] Graph name: model, with op num: 135
[UNILOG][INFO] begin to compile...
[UNILOG][WARNING] xir::Op(name = quant_add_recomputation_0, type = add) is inserted to do the recomputation for xir::Op(name = quant_add, type = add).
[UNILOG][WARNING] xir::Op(name = quant_add_1_recomputation_0, type = add) is inserted to do the recomputation for xir::Op(name = quant_add_1, type = add).
[UNILOG][INFO] Total device subgraph number 6, DPU subgraph number 2
[UNILOG][INFO] compile done.
[UNILOG][INFO] The meta json is saved to "/workspace/tutorials/cat_detector/output/meta.json"
[UNILOG][INFO] The compiled xmodel is saved to "/workspace/tutorials/cat_detector/output/model_compiled.xmodel"
[UNILOG][INFO] The compiled xmodel's md5sum is 8d25254fdec296df35d3e4787fc8074f, and has been saved to "/workspace/tutorials/cat_detector/output/md5sum.txt"
```

And a folder gets created with the desired output:



RUNNING THE VITIS AI MODEL

To run an xmodel the vart library is needed, and it can be used either in C++ or Python.

https://xilinx.github.io/inference-server/0.1.0/xmodel_example_python.html

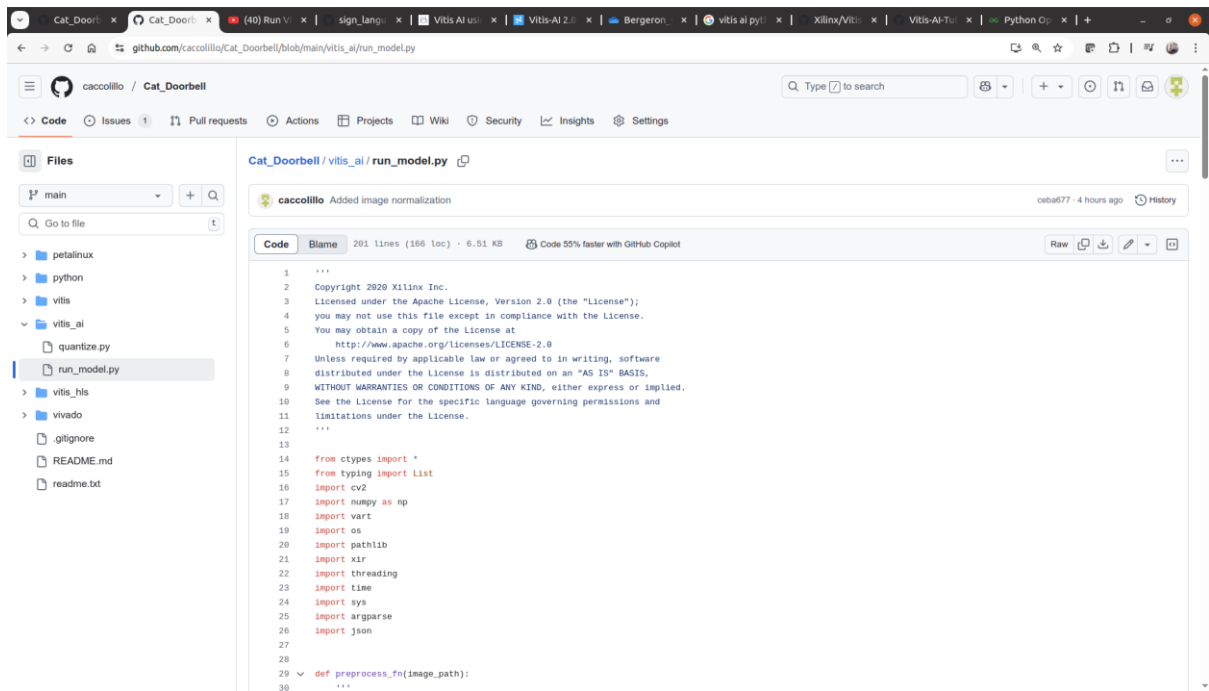
<https://beetlebox.org/vitis-ai-using-tensorflow-and-keras-tutorial-part-9/>

https://github.com/beetleboxorg/sign_language_mnist

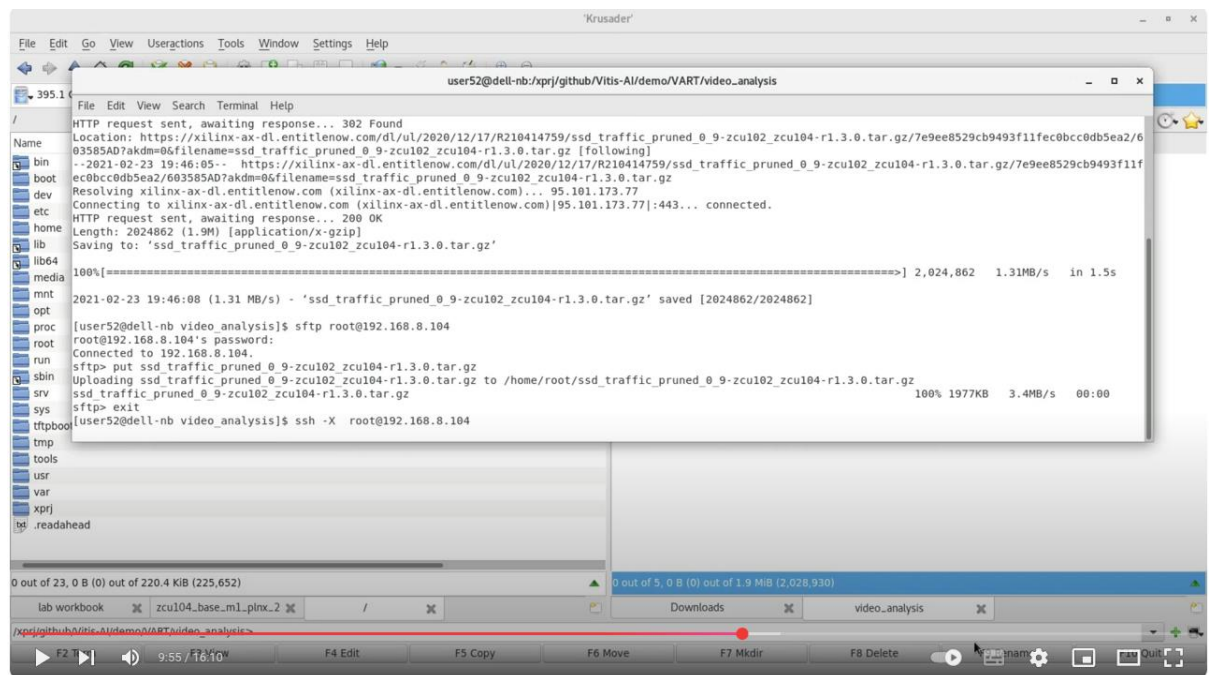
https://github.com/beetleboxorg/sign_language_mnist/blob/master/target/sign_language_app.py

A script has been prepared, inspired by the one at:

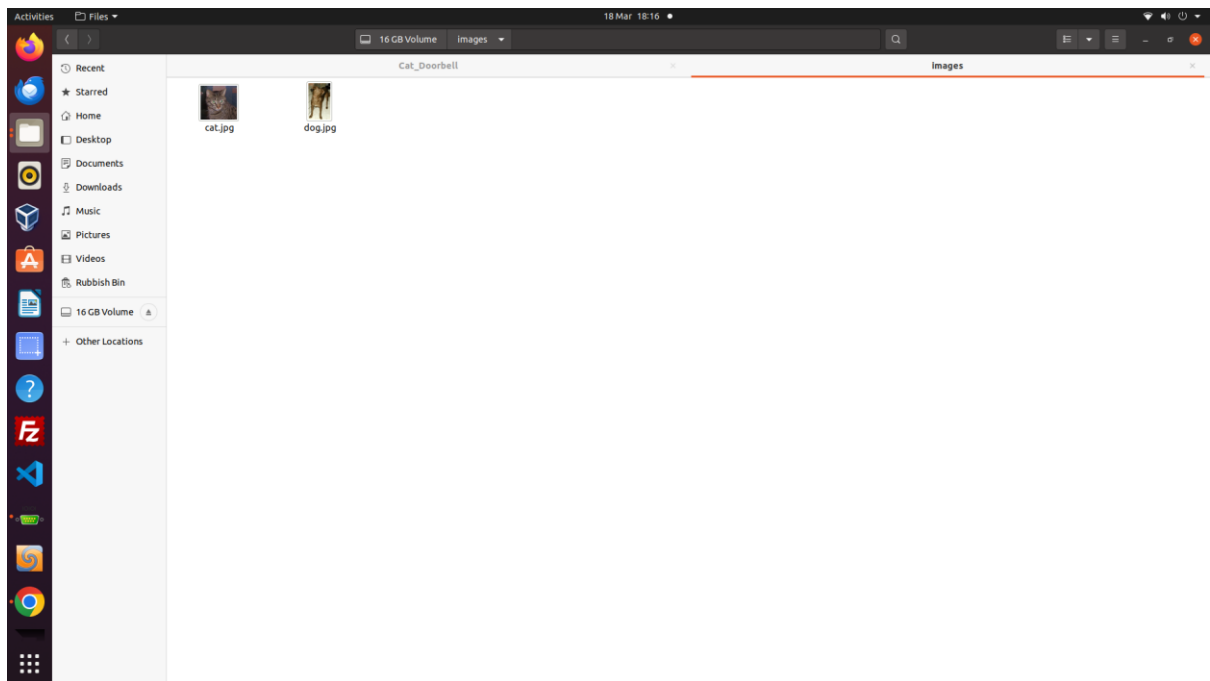
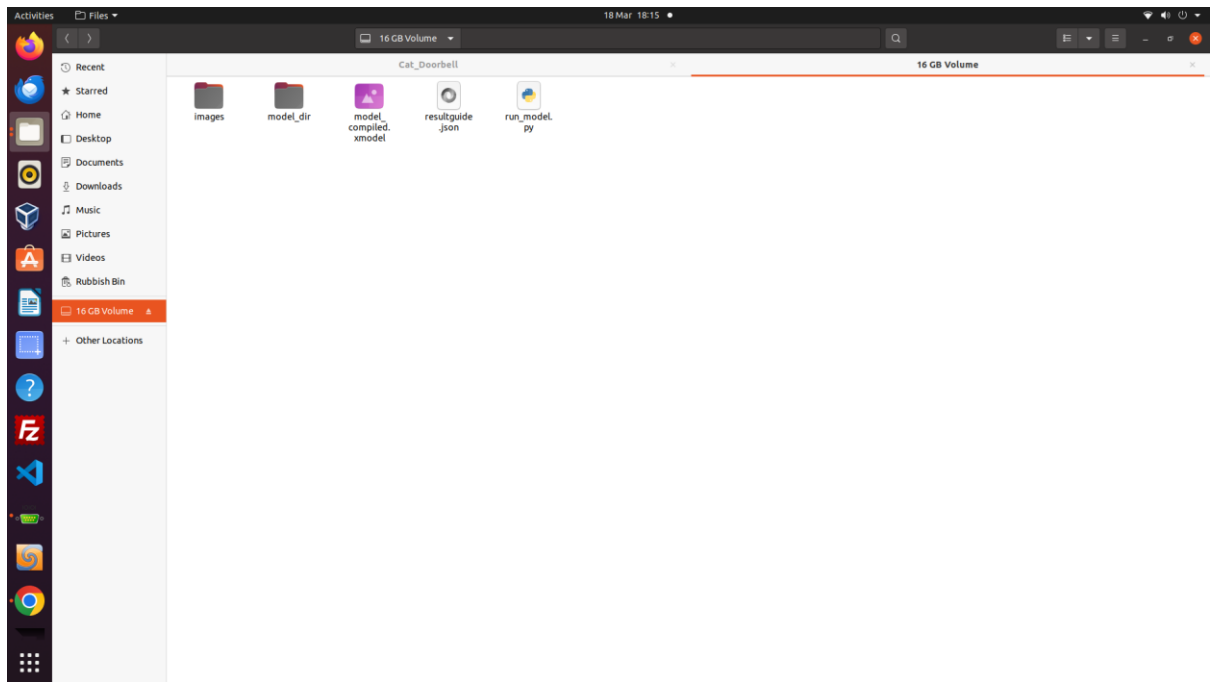
https://github.com/beetleboxorg/sign_language_mnist/blob/master/target/sign_language_app.py

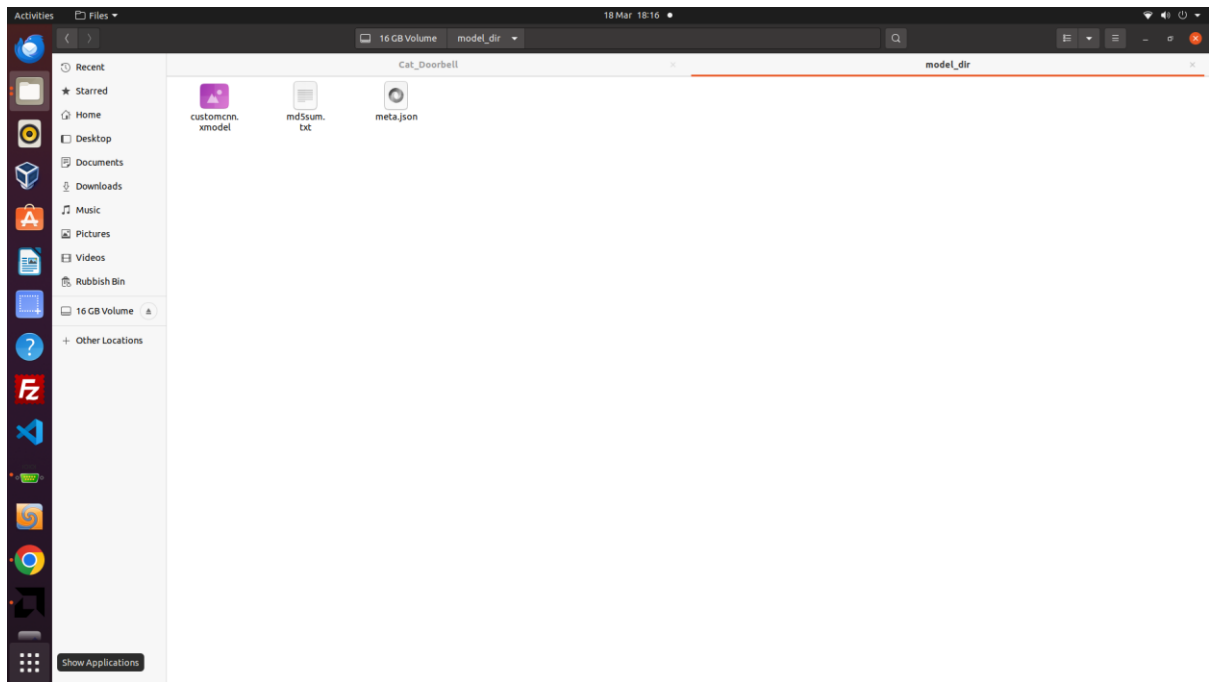


Copy the model and the script on the board either with ssh or sftp:



By creating an USB pen organized as follows:





And mounting it, giving the following commands:

```
cd /run/media/sda1
```

```
python3 ./run_model.py
```

The script fails as the model has been compiled for the wrong DPU:



By searching online for the error code, it seems as if an error in the Vitis AI docker has been made, when compiling the quantized model:

https://adaptivesupport.amd.com/s/article/DPU-fingerprint-ERROR?language=en_US

https://adaptivesupport.amd.com/s/question/0D54U00006wDmkzSAC/info-post-about-dpu-fingerprint?language=en_US

https://xilinx.github.io/kria-apps-docs/kv260/2022.1/build/html/docs/aibox/docs/customize_ai_models-aib.html

As adviced in the first link, by giving the “xdputil query” command on the U96 board:

```
u96v2-sbc-2022-2 login: root
root@u96v2-sbc-2022-2:~# xdputil query
{
  "DPU IP Spec":{
    "DPU Core Count":1,
    "IP version":"v4.1.0",
    "generation timestamp":"2023-02-21 21:30:00",
    "git commit id":"7d32c41",
    "git commit time":2023022121,
    "regmap":"litol version"
  },
  "VAI Version":{
    "libvart-runner.so":"Xilinx vart-runner Version: 3.0.0-c5d2bd43d951c174185d728b8e5bcda3869e0b39 2023-04-10-20:06:16 ",
    "libvitis-ai-library-dpu-task.so":"Xilinx vitis-ai-library dpu task Version: 3.0.0-c5d2bd43d951c174185d728b8e5bcda3869e0b39 2023-01-13 06:58:30 [UTC] ",
    "libxir.so":"Xilinx xir Version: xir-c5d2bd43d951c174185d728b8e5bcda3869e0b39 2023-04-03-14:21:20",
    "target_factory":"target-factory.3.0.0 c5d2bd43d951c174185d728b8e5bcda3869e0b39"
  },
  "kernels":[
    {
      "DPU Arch":"DPUCZDX8G_ISA1_B2304_0101000016010405",
      "DPU Frequency (MHz)":200,
      "IP Type":"DPU",
      "Load Parallel":2,
      "Load augmentation":"enable",
      "Load minus mean":"disable",
      "Save Parallel":2,
      "XRT Frequency (MHz)":200,
      "cu_addr":"0xb0000000",
      "cu_handle":"0xaaaa170e7a0",
      "cu_idx":0,
      "cu_mask":1,
      "cu_name":"DPUCZDX8G:DPUCZDX8G_1",
      "device_id":0,
      "fingerprint":"0x101000016010405",
      "name":"DPU Core 0"
    }
  ]
}
```

We get the fingerprint:

"DPU Arch":"DPUCZDX8G_ISA1_B2304_0101000016010405",
"fingerprint":"0x101000016010405",

A new folder U96v2 is created on the docker and the json file is created:

```
vitis-ai-user@caccolillo-OMEN-25L-Desktop-GT12-1xxx: /opt/vitis_ai/compiler/arch/DPUCZDX8G/U96v2
(vitis-ai-tensorflow2) vitis-ai-user@caccolillo-OMEN-25L-Desktop-GT12-1xxx:/opt/vitis_ai/compiler/arch/DPUCZDX8G/U96v2$ cat arch.json
{
  "fingerprint":"0x101000016010405"
}
(vitis-ai-tensorflow2) vitis-ai-user@caccolillo-OMEN-25L-Desktop-GT12-1xxx:/opt/vitis_ai/compiler/arch/DPUCZDX8G/U96v2$
```

The model is compiled again:

vai_c_tensorflow2 -m ./quantized_model.h5 -a
/opt/vitis_ai/compiler/arch/DPUCZDX8G/U96v2/arch.json -o output/ -n
model_compiled --options
"{ 'mode': 'normal', 'save_kernel': '', 'input_shape': '128,180,180,3' }"


```
vitis-ai-user@caccollilo-OMEN-25L-Desktop-GT12-1xxx: /workspace/tutorials/cat_detector
(vitis-ai-tensorflow2) vitis-ai-user@caccollilo-OMEN-25L-Desktop-GT12-1xxx:/opt/vitis_ai/compiler/arch/DPUCZDX8G/U96v25 cat arch.json
{
  "fingerprint": "0x101000010010405"
}
(vitis-ai-tensorflow2) vitis-ai-user@caccollilo-OMEN-25L-Desktop-GT12-1xxx:/opt/vitis_ai/compiler/arch/DPUCZDX8G/U96v25 cd /workspace/tutorials/cat_detector/
(vitis-ai-tensorflow2) vitis-ai-user@caccollilo-OMEN-25L-Desktop-GT12-1xxx:/workspace/tutorials/cat_detector$ val_c_tensorflow2 -n ./quantized_model.h5 -a /opt/vitis_ai/compiler/arch/DPUCZDX8G/U96v25/arch.json -o output -n model_compiled --options '{"mode":"normal","save_kernel":"","input_shape":"128,180,180,3"}'
*****
* Vitis AI Compilation - Xilinx Inc.
*****
[INFO] Namespace(batchsize=1, inputs_shape=['128,180,180,3'], layout='NHWC', model_files=['./quantized_model.h5'], model_type='tensorflow2', named_inputs_shape=None, out_filename='/tmp/model_compiled_0x101000010010405.org.xmodel', proto=None)
to shape: [[128, 180, 180, 3]]
[INFO] tensorflow2 model: /workspace/tutorials/cat_detector/quantized_model.h5
[INFO] keras version: 2.12.0
[INFO] tensorflow Keras model type: functional
[INFO] parse raw model : 100% | 42/42 [00:00<00:00, 24148.15it/s]
[INFO] infer shape (NHWC) : 100% | 71/71 [00:00<00:00, 691.88it/s]
[INFO] perform level-0 opt : 100% | 1/1 [00:00<00:00, 262.54it/s]
[INFO] perform level-1 opt : 100% | 2/2 [00:00<00:00, 1036.78it/s]
[INFO] infer shape (NHWC) : 100% | 75/75 [00:00<00:00, 816.59it/s]
[INFO] generate xmodel : 0% | 0/75 [00:00<?, ?it/s]
WARNING: Logging before InitGoogleLogging() is written to STDERR
w20250323 04:05:01.719713 163 tool_function.cpp:171] [UNILOG][WARNING] The operator named rescaling, type: Rescaling, is not defined in XIR. XIR creates the definition of this operator automatically. You should specify the shape and the data_type of the output tensor of this operation by set_attr('shape', std::vector<int>) and set_attr('data_type', std::string)
[INFO] generate xmodel : 100% | 75/75 [00:00<00:00, 1876.48it/s]
[INFO] dump xmodel: /tmp/model_compiled_0x101000010010405.org.xmodel
[UNILOG][WARNING] The operator named rescaling, type: Rescaling, is not defined in XIR. XIR creates the definition of this operator automatically. You should specify the shape and the data_type of the output tensor of this operation by set_attr('shape', std::vector<int>) and set_attr('data_type', std::string)
[INFO] tensorflow2 model: /workspace/tutorials/cat_detector/quantized_model.h5
[UNILOG][INFO] Compile mode: dpu
[UNILOG][INFO] Debug mode: null
[UNILOG][INFO] Target architecture: DPUCZDX8G_ISA1_B2304_0101000010010405
[UNILOG][INFO] Graph name: model, with op num: 135
[UNILOG][INFO] Begin to compile...
[UNILOG][WARNING] xir::Op(name = quant_add_recomputation_0, type = add) is inserted to do the recomputation for xir::Op(name = quant_add, type = add).
[UNILOG][WARNING] xir::Op(name = quant_add_1_recomputation_0, type = add) is inserted to do the recomputation for xir::Op(name = quant_add_1, type = add).
[UNILOG][INFO] Total device subgraph number 6, DPU subgraph number 2
[UNILOG][INFO] Compile done.
[UNILOG][INFO] The meta.json is saved to "/workspace/tutorials/cat_detector/output/meta.json"
[UNILOG][INFO] The compiled xmodel is saved to "/workspace/tutorials/cat_detector/output/model_compiled.xmodel"
[UNILOG][INFO] The compiled xmodel's md5sum is 8f47b59a5ee3ecc48bd7460781sfafai, and has been saved to "/workspace/tutorials/cat_detector/output/md5sum.txt"
(vitis-ai-tensorflow2) vitis-ai-user@caccollilo-OMEN-25L-Desktop-GT12-1xxx:/workspace/tutorials/cat_detector$
```

And placed in the USB memory stick, overwriting the wrong one.

And mounting it, giving the following commands:

```
cd /run/media/sda1
```

```
python3 ./run_model.py
```

The script starts this time and runs the DPU:

```
root@u96v2-sbc-2022-2:/run/media/sda1# python3 ./run_model.py
Command line options:
--image_dir : /images
--threads : 1
--model : model_dir/customcnn.xmodel
--results : ./resultguide.json
--custom_results : custom_images
WARNING: Logging before InitGoogleLogging() is written to STDERR
w20250323 10:41:09.715732 853 tool_function.cpp:171] [UNILOG][WARNING] The operator named rescaling, type: Rescaling, is not defined in XIR. XIR creates the definition of this operator by set_attr('shape', std::vector<int>) and set_attr('data_type', std::string)
Pre-processing 2 images...
Starting 1 threads...
Throughput=10.91 fps, total frames = 2, time=0.1834 seconds
Traceback (most recent call last):
  File "/run/media/sda1/./run_model.py", line 200, in <module>
    main()
  File "/run/media/sda1/./run_model.py", line 105, in main
    app(args.image_dir,args.threads,args.model, args.results)
  File "/run/media/sda1/./run_model.py", line 154, in app
    prediction = classes[out_q[1]]
IndexError: list index out of range
root@u96v2-sbc-2022-2:/run/media/sda1#
```

But it then crashes at around row 200.

Some debugging is needed.

At least now, the DPU is working.

It looks like there is an error over the way in which the output of the model run on the DPU is taken.

To get a better understanding of how to run it, it has been decided to follow the tutorial below:

https://www.pixela.co.jp/products/pickup/dev/ai/vitisai_ai_4_custom_model_en.html

MNIST VITIS AI MODEL TUTORIAL

To get a better understanding of how to run a trained model, it has been decided to follow the tutorial below:

https://www.pixela.co.jp/products/pickup/dev/ai/vitisai_ai_4_custom_model_en.html

The training script wasn't working, so this one has been used instead:

https://adaptivesupport.amd.com/s/question/0D52E00006rCKYqSAO/any-idea-why-this-is-producing-a-shape-error-in-quantization?language=en_US

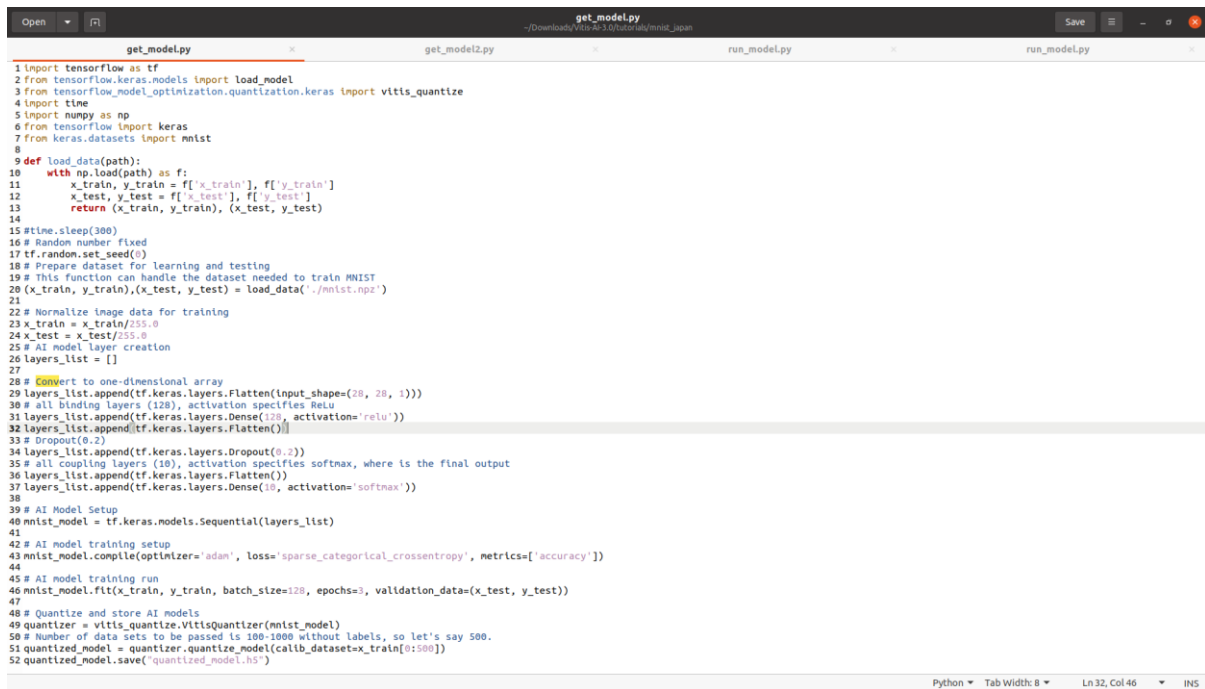
The mnist dataset could not be downloaded, so the mnist dataset has been downloaded manually:

<https://www.kaggle.com/datasets/vikramtiwari/mnist-numpy>

And to load it, the function below has been used:

```
10 def load_data(path):
11     with np.load(path) as f:
12         x_train, y_train = f['x_train'], f['y_train']
13         x_test, y_test = f['x_test'], f['y_test']
14         return (x_train, y_train), (x_test, y_test)
15
16
17
18 #time.sleep(300)
19 # Random number fixed
20 tf.random.set_seed(0)
21
22 # Prepare dataset for learning and testing
23 # This function can handle the dataset needed to train MNIST
24
25
26
27 (x_train, y_train), (x_test, y_test) = load_data('./mnist.npz')
28
```

The script below has worked, generating a quantized model:



```
1 import tensorflow as tf
2 from tensorflow.keras.models import load_model
3 from tensorflow_model_optimization.quantization.keras import vitis_quantize
4 import time
5 import numpy as np
6 from tensorflow import keras
7 from keras.datasets import mnist
8
9 def load_data(path):
10     with np.load(path) as f:
11         x_train, y_train = f['x_train'], f['y_train']
12         x_test, y_test = f['x_test'], f['y_test']
13         return (x_train, y_train), (x_test, y_test)
14
15 #time.sleep(300)
16 # Random number fixed
17 tf.random.set_seed(0)
18 # Prepare dataset for learning and testing
19 # This function can handle the dataset needed to train MNIST
20 (x_train, y_train), (x_test, y_test) = load_data('./mnist.npz')
21
22 # Normalize image data for training
23 x_train = x_train/255.0
24 x_test = x_test/255.0
25 # AI model layer creation
26 layers_list = []
27
28 # Convert to one-dimensional array
29 layers_list.append(tf.keras.layers.Flatten(input_shape=(28, 28, 1)))
30 # all binding layers (128), activation specifies ReLU
31 layers_list.append(tf.keras.layers.Dense(128, activation='relu'))
32 layers_list.append(tf.keras.layers.Flatten())
33 # Dropout(0.2)
34 layers_list.append(tf.keras.layers.Dropout(0.2))
35 # all coupling layers (10), activation specifies softmax, where is the final output
36 layers_list.append(tf.keras.layers.Flatten())
37 layers_list.append(tf.keras.layers.Dense(10, activation='softmax'))
38
39 # AI Model Setup
40 mnist_model = tf.keras.models.Sequential(layers_list)
41
42 # AI model training setup
43 mnist_model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
44
45 # AI model training run
46 mnist_model.fit(x_train, y_train, batch_size=128, epochs=3, validation_data=(x_test, y_test))
47
48 # Quantize and store AI models
49 quantizer = vitis_quantize.VitisQuantizer(mnist_model)
50 # Number of data sets to be passed is 100-1000 without labels, so let's say 500.
51 quantized_model = quantizer.quantize_model(calib_dataset=x_train[0:500])
52 quantized_model.save('quantized_model.h5')
```

The `flatten()` function has been used to get it working.

KERAS MINIRESNET FINALLY WORKING

As a starting point, the following tutorial has been used:

https://github.com/Xilinx/Vitis-AI-Tutorials/tree/3.0/Tutorials/Keras_GoogleNet_ResNet/

The files of interest have been:

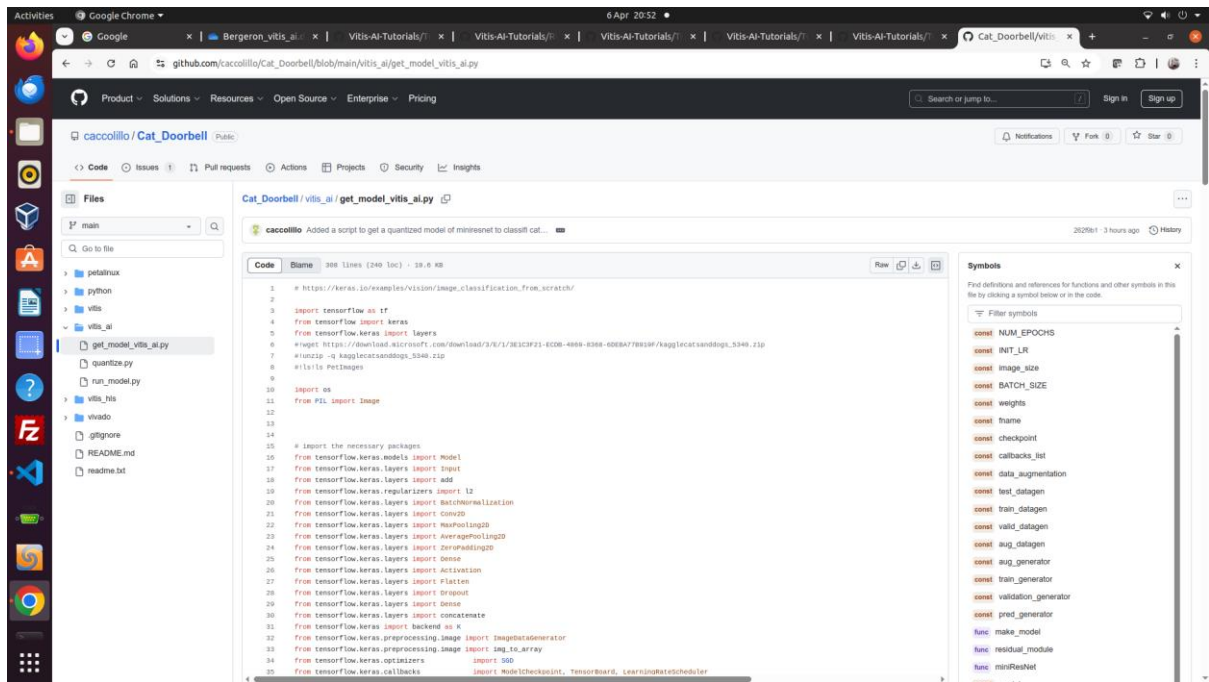
https://github.com/Xilinx/Vitis-AI-Tutorials/blob/3.0/Tutorials/Keras_GoogleNet_ResNet/files/code/custom_cnn.py

And:

https://github.com/Xilinx/Vitis-AI-Tutorials/blob/3.0/Tutorials/Keras_GoogleNet_ResNet/files/code/train_fmnnist.py

Where the dataset has been created starting from images of cats and dogs borrowed by kaggle, and reworking the script.

The script has been saved in:



At the end of the training, by using the following datasets available on kaggle:

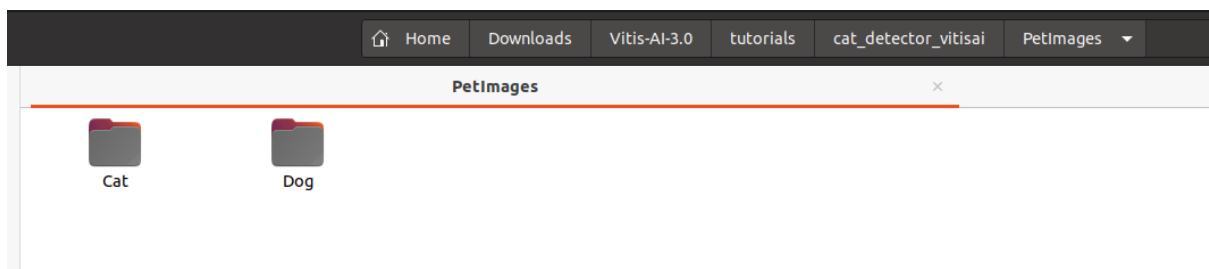
<https://www.kaggle.com/c/dogs-vs-cats>

<https://www.kaggle.com/datasets/tongpython/cat-and-dog>

<https://www.kaggle.com/datasets/karakaggle/kaggle-cat-vs-dog-dataset>

<https://www.kaggle.com/datasets/bhavikjikadara/dog-and-cat-classification-dataset>

By saving the cat and dog pictures in the folders shown below:



By launching the training script, we get the following performances:

```

=====
Total params: 888,910
Trainable params: 876,552
Non-trainable params: 12,358

WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g., tf.keras.optimizers.legacy
Epoch 1/10
803/803 [=====] - 5871s 7s/step - loss: 1.1657 - accuracy: 0.6167 - val_loss: 1.0453 - val_accuracy: 0.6985
Epoch 2/10
803/803 [=====] - 6715s 8s/step - loss: 0.9894 - accuracy: 0.7398 - val_loss: 0.9376 - val_accuracy: 0.7768
Epoch 3/10
803/803 [=====] - 7215s 9s/step - loss: 0.8737 - accuracy: 0.8090 - val_loss: 0.8971 - val_accuracy: 0.7961
Epoch 4/10
803/803 [=====] - 7571s 9s/step - loss: 0.7959 - accuracy: 0.8444 - val_loss: 0.7628 - val_accuracy: 0.8572
Epoch 5/10
803/803 [=====] - 7769s 10s/step - loss: 0.7228 - accuracy: 0.8748 - val_loss: 0.7391 - val_accuracy: 0.8668
Epoch 6/10
803/803 [=====] - 8336s 10s/step - loss: 0.6695 - accuracy: 0.8948 - val_loss: 0.6930 - val_accuracy: 0.8824
Epoch 7/10
803/803 [=====] - 8005s 10s/step - loss: 0.6184 - accuracy: 0.9117 - val_loss: 0.6770 - val_accuracy: 0.8943
Epoch 8/10
803/803 [=====] - 8483s 11s/step - loss: 0.5830 - accuracy: 0.9274 - val_loss: 0.6554 - val_accuracy: 0.8988
Epoch 9/10
803/803 [=====] - 8204s 10s/step - loss: 0.5418 - accuracy: 0.9408 - val_loss: 0.6062 - val_accuracy: 0.9133
Epoch 10/10
803/803 [=====] - 8343s 10s/step - loss: 0.5105 - accuracy: 0.9495 - val_loss: 0.6328 - val_accuracy: 0.9142
Training completed. Press Enter to continue...

```

At the end, an hdf5file is created, as per the description in the tutorial:

https://github.com/Xilinx/Vitis-AI-Tutorials/blob/3.0/Tutorials/Keras_GoogleNet_ResNet/README.md

And it gets compiled:

```

vitis-ai-user@caccollilo-OMEN-25L-Desktop-GT12-1xxx:/workspace/tutorials/cat_detector_vitisai$ val_c_tensorflow2 -n ./quantized.h5 -a /opt/vitis_ai/compiler/arch/DPU CZDX8G/U96V2/arch.json -o output/ -n model_compiled
=====
* Vitis AI Compilation - Xilinx Inc.
=====
[INFO] Namespace(batchsize=1, inputs_shape=None, layout='NHWC', model_files=['./quantized.h5'], model_type='tensorflow', named_inputs_shape=None, out_filename='/tmp/model_compiled_0x101000016010405_org.xmodel', proto=None)
[INFO] tensorflow2 model: /workspace/tutorials/cat_detector_vitisai/quantized.h5
[INFO] keras version: 2.12.0
[INFO] Tensorflow Keras model type: functional
[INFO] parse raw model :100%|
[INFO] infer shape (NHWC) :100%|
[INFO] perform level-0 opt :100%|
[INFO] perform level-1 opt :100%|
[INFO] infer shape (NHWC) :100%|
[INFO] generate xmodel :100%|
[INFO] dump xmodel: /tmp/model_compiled_0x101000016010405_org.xmodel
[UNILOG][INFO] compile mode: dpu
[UNILOG][INFO] Debug mode: null
[UNILOG][INFO] Target architecture: DPU CZDX8G ISA1_B2304_0101000016010405
[UNILOG][INFO] Graph name: resnet, with op num: 778
[UNILOG][INFO] Begin to compile...
[UNILOG][INFO] Total device subgraph number 3, DPU subgraph number 1
[UNILOG][INFO] Compile done.
[UNILOG][INFO] The meta json is saved to "/workspace/tutorials/cat_detector_vitisai/output/meta.json"
[UNILOG][INFO] The compiled xmodel is saved to "/workspace/tutorials/cat_detector_vitisai/output/model_compiled.xmodel"
[UNILOG][INFO] The compiled xmodel's md5sum is c4595a3cecf3b11f62b2912ebff230, and has been saved to "/workspace/tutorials/cat_detector_vitisai/output/md5sum.txt"
vitis-ai-user@caccollilo-OMEN-25L-Desktop-GT12-1xxx:/workspace/tutorials/cat_detector_vitisai$

```

