Category:Artificial intelligence expansion packages

Contents

Stable: 17.06.2020 - 15:26 / Revision: 12.12.2019 - 12:27

Category:Artificial intelligence expansion packages

A quality version of this page, approved on *17 June 2020*, was based off this revision.

This category groups together all articles related to software expansion packages about artificial intelligence.

Subcategories

This category has only the following subcategory.

- Artificial intelligence sample apps (11 P)

Pages in category "Artificial intelligence expansion packages"

The following 4 pages are in this category, out of 4 total.

- X-LINUX-AI OpenSTLinux Expansion Package
- X-LINUX-AI application samples zoo
- How to run Coral Edge TPU inference using Python TensorFlow Lite API
- How to compile model and run inference on Coral Edge TPU using STM32MP1

Stable: 08.02.2021 - 12:42 / Revision: 04.02.2021 - 09:05

A quality version of this page, approved on *8 February 2021*, was based off this revision.

Contents

1 Article purpose

This article aims at describing how to run an inference on Coral Edge TPU using STM32MP1 microprocessor devices.

Information

This article provides a simple example. Other methods exist that might be better adapted to your development constraints. Feel free to explore them.

2 Prerequisites

2.1 Installing the Edge TPU compiler

To perform an inference on the Coral Edge TPU hardware, we need to convert our TensorFlow Lite model into an Edge TPU model. This can be achieved by using the Edge TPU compiler. To do this, install the Edge TPU compiler on your host computer.

Information

The Edge TPU compiler is provided only for 64-bit architectures. Thus, it is not possible to install it on the STM32MP1 target board. Besides, you must ensure that your host computer uses a Debian-based Linux system.

Install the Coral Edge TPU compiler by running the following commands on your host computer:

```
 PC $> curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
 PC $> echo "deb https://packages.cloud.google.com/apt coral-edgetpu-stable main" | sudo
tee /etc/apt/sources.list.d/coral-edgetpu.list
 PC $> sudo apt-get update
 PC $> sudo apt-get install edgetpu-compiler
```

Enter the following command to check that the compiler has been installed:

```
 PC $> edgetpu_compiler -v
 Edge TPU Compiler version 2.x.y
```

3 Fetching a Coral Edge TPU compiled model

3.1 Coral compiled models

Coral [1] offers a wide set of quantized and compiled models for demonstration purposes. Ready-to-use models are available from https://coral.ai/models/.

3.2 Compiling your own model

The Edge TPU compiler role is to convert one or several TensorFlow Lite models into Edge TPU compatible models. It takes as an argument your *.tflite* model and returns a *.tflite* Edge TPU compatible model. You can pass multiple models as arguments (each separated with a space). They are then co-compiled and share the Edge TPU 8 Mbytes of RAM for parameter data caching.

Be aware that not all the operations supported on TensorFlow Lite are supported on Edge TPU. While building your own model architecture, check the operations and the layers supported by the Edge TPU compiler on https://coral.ai/docs/edgetpu/models-intro/#supported-operations.

If your model architecture uses unsupported operations and does not meet all the requirements, then only the first portion of the model will be executed on the Edge TPU compiler. Starting from the first node in your model graph where an unsupported operation occurs, all operations are run on the CPU of the target board even if an Edge TPU supported operation occurs. The Edge TPU compiler cannot partition the model more than once.

Information

If an important percentage of your model is executed on the CPU, you have to expect a significant degradation of the inference speed compared to a model that is entirely executed on the Edge TPU compiler. To achieve a maximal speed, use only Edge TPU supported operations in your model

To compile your *.tflite* model, execute the following command:

```
PC $> edgetpu_compiler your_model_1.tflite your_model_1.tflite ...
```

If you do not specify the *-out_dir* option, the compiled model is saved in the current directory with the *input_filename_edgetpu.tflite* name. Another *.log* file that provides information about data caching and memory consumption is stored in the Edge TPU RAM.

3.3 Example: compiling an object detection model

The object detection model used is the *ssd_mobilenet_v1_coco_quant.tflite* downloaded from the **Coral** [1] website and compiled for the Coral Edge TPU using the steps detailed below:

```
PC $> wget http://storage.googleapis.com/download.tensorflow.org/models/tflite
/coco_ssd_mobilenet_v1_1.0_quant_2018_06_29.zip
PC $> unzip ./coco_ssd_mobilenet_v1_1.0_quant_2018_06_29.zip
 Archive:  ./coco_ssd_mobilenet_v1_1.0_quant_2018_06_29.zip
           inflating: detect.tflite
           inflating: labelmap.txt
```

Now that the model has been downloaded and extracted successfully, it is time to compile it using the Edge TPU compiler:

```
PC $> edgetpu_compiler detect.tflite
```

3.4 Sending your model to the target

In your board workspace directory, create two main directories to organize our workflow:

```
Board $> cd /usr/local && mkdir -p workspace
Board $> cd /usr/local/workspace && mkdir -p models
```

Then transfer your compiled model from the **host computer** to the *models* directory in the **board** workspace:

```
PC $> scp path/to/your/compiled_model_edgetpu.tflite root@<board_ip_address>:/usr/local
/workspace/models/
```

Now that your workspace is ready with the compiled model file, it is time to see how to run an inference using the C++ benchmark application.

4 Running the inference

4.1 Installing the benchmark application

Configure the AI OpenSTLinux package, then install X-LINUX-AI components for this application:

Warning

The software package is provided AS IS, and by downloading it, you agree to be bound to the terms of the software license agreement (SLA). The detailed content licenses can be found here.

```
Board $> apt-get install tflite-edgetpu-benchmark
```

4.2 Execute the benchmark on the model

Now that our compiled model is loaded on the board, it is time to run an inference using the benchmark example. This example aims at measuring your model average inference time on a predefined number of 25 loops. To do this, execute the following command:

```
Board $> cd /usr/local/demo-ai/benchmark/tflite-edgetpu/
Board $> ./tflite_edgetpu_benchmark -m /usr/local/workspace/models/your_compiled_model.
tflite -l 25
```

The first inference may take longer since the model is being loaded on the Coral Edge TPU RAM. This time is not taken into account in the average inference time.

4.3 Customizing your application

You can adapt your application to your development constraints and requirements.

To build a prototype of your application using Python, go through Image classification Python example or Object detection Python example.

To run your application using the C++ API, refer to the Image classification C++ example or Object detection C++ example.

5 References

- [1.01.1] Coral AI

Linux® is a registered trademark of Linus Torvalds.

Random Access Memory (Early computer memories generally hadserial access. Memories where any given address can be accessed when desired were then called "random access" to distinguish them from the memories where contents can only be accessed in a fixed order. The term is used today for volatile random-acces ssemiconductor memories.)

Central processing unit

Artificial Intelligence

Application programming interface

Stable: 04.03.2021 - 14:33 / Revision: 04.03.2021 - 14:32

A quality version of this page, approved on *4 March 2021*, was based off this revision.

Contents

1 Article purpose

This article describes how to run an inference on the STM32MP1 using a Google Coral EdgeTPU device and the Python TensorFlow Lite API. It is an example based on an image classification application.

Information

There are many ways to achieve this result; this article provides a simple example. You are free to explore other methods that are better adapted to your development constraints.

2 Libedgetpu and TensorFlow Lite Python APIs

The Artificial Intelligence expansion package X-LINUX-AI comes with TensorFlow Lite Python APIs and the libedgetpu (providing the support of the Coral Edge TPU) that has been rebuilt from source to be compatible with the embedded TensorFlow Lite runtime.

In the next section we explore, with a basic image-classification example, how to inference your models on the board using the Coral EdgeTPU device.

3 Running an inference on Coral EdgeTPU using the TensorFlow Lite Python API

3.1 Installing prerequisites on the target

Warning

The software package is provided AS IS, and by downloading it, you agree to be bound to the terms of the software license agreement (SLA). The detailed content licenses can be found here.

After having configured the AI OpenSTLinux package, you can install the X-LINUX-AI components and the packages needed to run our example.

The main packages are Python Numpy[1], Python OpenCV[2], Python TensorFlow Lite runtime[3] and libedgetpu

```
Board $> apt-get install python3-numpy python3-opencv python3-tensorflow-lite libedgetpu
```

3.2 Preparing the workspace on the target

```
Board $> cd /usr/local/ && mkdir -p workspace
Board $> cd workspace && mkdir -p models testdata
```

In this example, we use the mobilenet_v1_1.0_224_quant_edgetpu.tflite model to classify download images, accompanied by the labels file from the Coral[4] website.

```
Board $> wget https://github.com/google-coral/edgetpu/raw/master/test_data/mobilenet_v1_1.
0_224_quant_edgetpu.tflite -O models/mobilenet_v1_1.0_224_quant_edgetpu.tflite
Board $> wget https://github.com/google-coral/edgetpu/raw/master/test_data
/imagenet_labels.txt -O models/labels.txt
Board $> wget https://github.com/google-coral/edgetpu/raw/master/test_data/bird.bmp -O
testdata/bird.bmp
```

Information

You can run your own model but you have to make sure that your *.tflite* model is compiled for inferencing on Coral EdgeTPU. Refer first to Compile your custom model

3.3 Running the inference

Here is a simple python script example to execute a NN inference on the Google Coral Edge TPU.

```
#!/usr/bin/python3
#
# Copyright (c) 2020 STMicroelectronics. All rights reserved.
#
# This software component is licensed by ST under BSD 3-Clause license,
# the "License"; You may not use this file except in compliance with the
```

```
# License. You may obtain a copy of the License at:
#                        opensource.org/licenses/BSD-3-Clause

import sys
import numpy as np
import tflite_runtime.interpreter as tflite
import time
import cv2

label_file = "/usr/local/workspace/models/labels.txt"
with open( label_file, 'r') as  f :
          labels = [ line.strip() for line in f.readlines() ]
model_file = "/usr/local/workspace/models/mobilenet_v1_1.0_224_quant_edgetpu.tflite"

#Create the interpreter and allocate tensors
interpreter = tflite.Interpreter(model_path = model_file, experimental_delegates = [tflite
.load_delegate('libedgetpu-max.so.2')])
interpreter.allocate_tensors()

#Getting the model input and output details
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
height = input_details[0]['shape'][1]
width = input_details[0]['shape'][2]

#Read the picture, convert from BGR to RGB encoding,
#resized to fit the size of the model input and have their
#dimensions expanded by one
image = cv2.imread(sys.argv[1])
nn_img_rgb = cv2.cvtColor(np.array(image), cv2.COLOR_BGR2RGB)
nn_img_rgb_resized = cv2.resize(nn_img_rgb, (width, height))
input_data = np.expand_dims(nn_img_rgb_resized, axis=0)

#Set the input data, execute the first inference (that could take longer
#since the model is being loaded on the Coral Edge TPU RAM)
interpreter.set_tensor(input_details[0]['index'], input_data)
start = time.perf_counter()
interpreter.invoke()
inference_time = time.perf_counter() - start
print("1st inference:", inference_time, "s")

# Execute the second inference and measure the inference duration
start = time.perf_counter()
interpreter.invoke()
inference_time = time.perf_counter() - start
print("2nd inference:", inference_time, "s")

#Print the results
results = np.squeeze(interpreter.get_tensor(output_details[0]['index']))
top_k = results.argsort()[-5:][::-1]
for i in top_k:
    print('{0:08.6f}'.format(float(results[i]*100/255.0))+":", labels[i])
print("\n")
```

Copy this python script to the target: PC $> scp path/to/your/script/classify_on_stm32mp1.py root@<board_ip_address>:/usr/local/workspace

3.4 Running the inference from the board on the Coral Edge TPU

```
Board $> cd /usr/local/workspace
Board $> python3 classify_on_stm32mp1.py testdata/bird.bmp
inference time:  0.11815726400163 84 ms
```

```
88.627451: 20  chickadee
4.313725: 19  magpie
3.137255: 18  jay
2.745098: 21  water ouzel, dipper
1.176471: 14  junco, snowbird
```

Information

The first inference may take longer since the model is being loaded on the Coral Edge TPU RAM

4 References

- Numpy
- OpenCV
- TensorFlow Lite
- Coral AI

Application programming interface

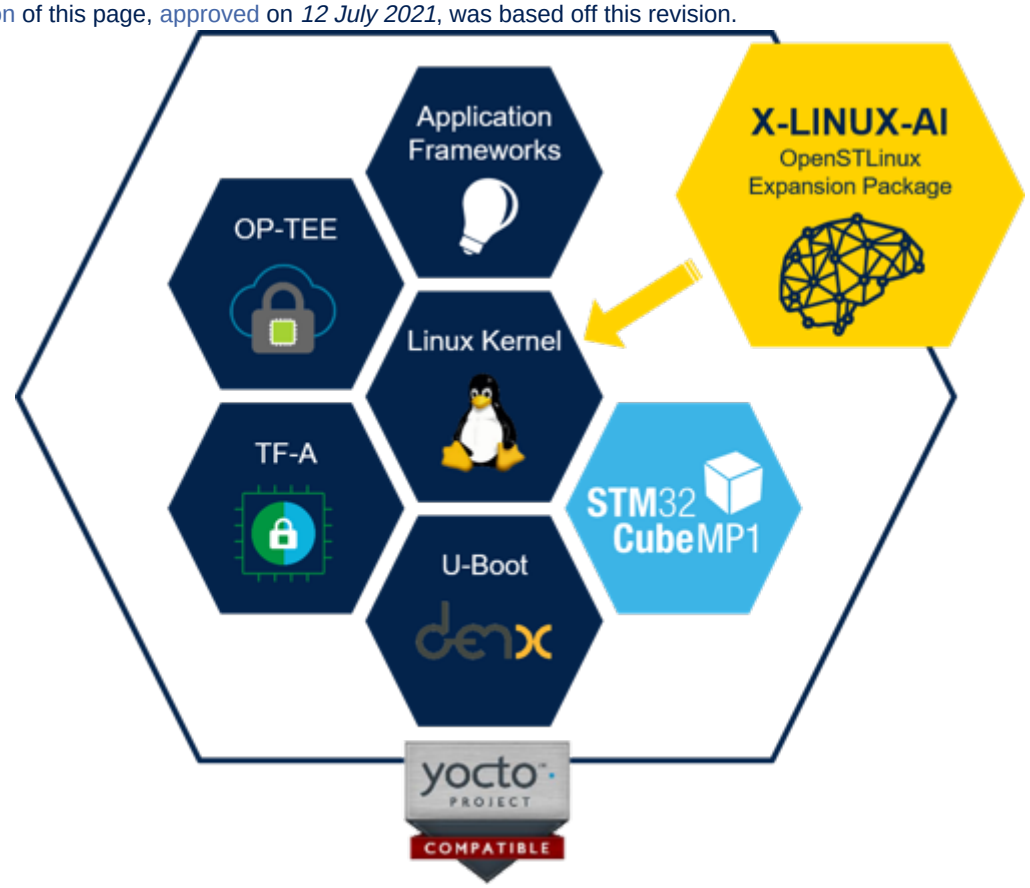
Artificial Intelligence

Neural Network

Random Access Memory (Early computer memories generally hadserial access. Memories where any given address can be accessed when desired were then called "random access" to distinguish them from the memories where contents can only be accessed in a fixed order. The term is used today for volatile random-acces ssemiconductor memories.)

Stable: 12.07.2021 - 13:12 / Revision: 12.07.2021 - 12:57

A quality version of this page, approved on *12 July 2021*, was based off this revision.



X-LINUX-AI is an STM32 MPU OpenSTLinux Expansion Package that targets artificial intelligence for STM32MP1 Series devices.

It contains Linux AI frameworks, as well as application examples to get started with some basic use cases such as computer vision (CV).

It is composed of an OpenEmbedded meta layer, named **meta-st-stm32mpu-ai**, to be added on top of the STM32MP1 Distribution Package.

It brings a complete and coherent easy-to-build / install environment to take advantage of AI on STM32MP1 Series devices.

Contents

1 Versions

1.1 X-LINUX-AI v2.1.0

Information

This version is compatible with Yocto Project® build system Thud and Dunfell and has been validated against the OpenSTLinux ecosystem release v2.1.0 , ecosystem release v2.0.0 , ecosystem release v1.2.0 and validated on STM32MP157x-DKx, STM32MP157x-EV1 and STM32MP157 Avenger96[1] boards.

1.1.1 Contents

- /!\ New /!\ **TensorFlow Lite**[2] **2.4.1**
- **Coral Edge TPU**[3] accelerator support
  - /!\ New /!\ **libedgetpu 2.4.1** aligned with TensorFlow Lite 2.4.1 (built from source)
- /!\ New /!\ **armNN**[4] **20.11**
- /!\ New /!\ **OpenCV**[5] **4.1.x**
- **Python**[6] **3.8.x** (enabling Pillow module)
- Support **STM32MP15xF**[7] devices operating at up to **800MHz**
- Application samples
  - C++ / Python image classification using TensorFlow Lite based on MobileNet v1 quantized model
  - C++ / Python object detection using TensorFlow Lite based on COCO SSD MobileNet v1 quantized model
  - C++ / Python image classification using Coral Edge TPU based on MobileNet v1 quantized model and compiled for the Coral Edge TPU
  - C++ / Python object detection using Coral Edge TPU based on COCO SSD MobileNet v1 quantized model and compiled for the Coral Edge TPU
  - C++ image classification using armNN TensorFlow Lite parser based on MobileNet v1 float model
  - C++ object detection using armNN TensorFlow Lite parser based on COCO SSD MobileNet v1 quantized model
  - /!\ New /!\ C++ face recognition using TensorFlow Lite models capable of recognizing the face of a known (enrolled) user (available on demand)

Warning

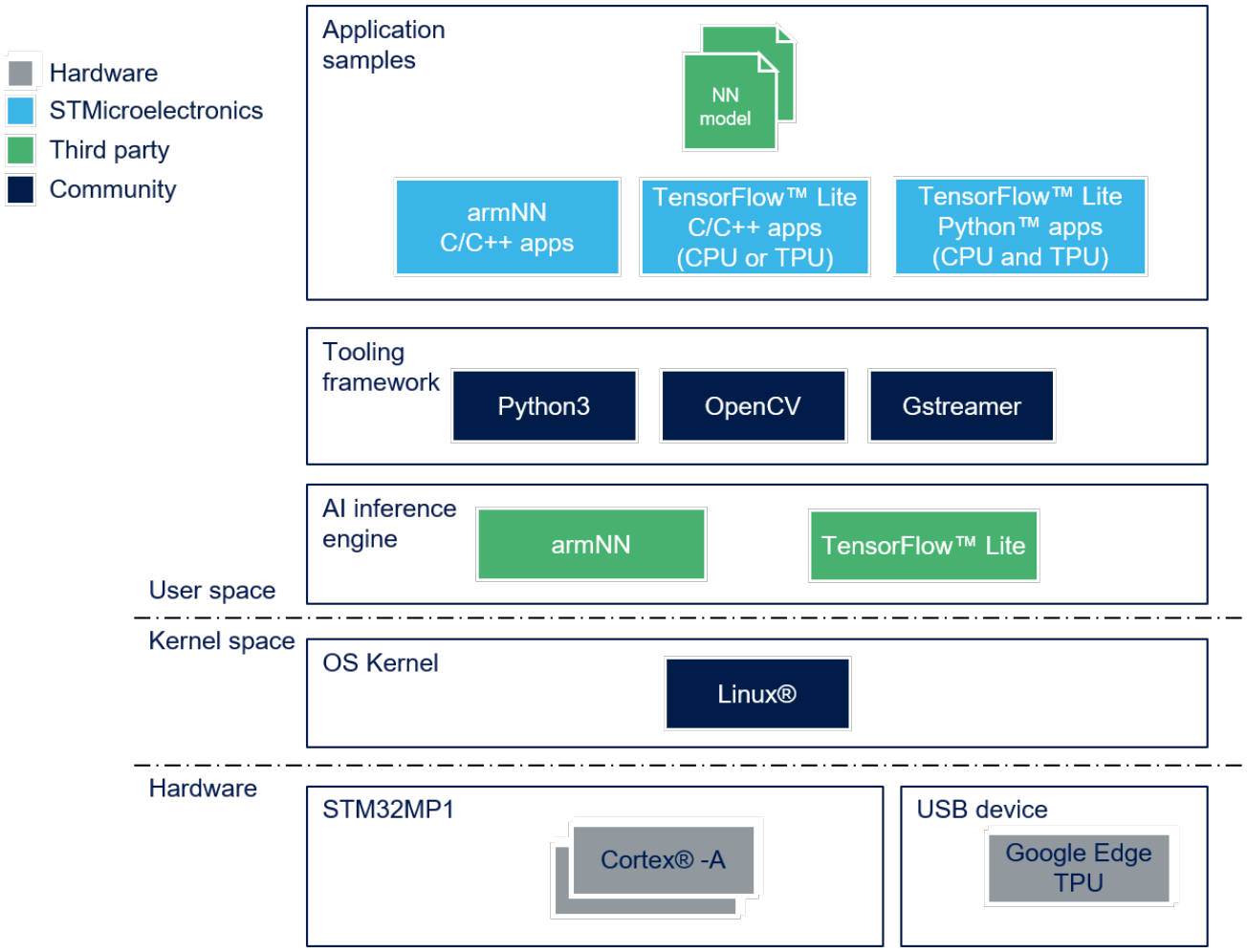
The face recognition binary is available on demand. Please contact the local STMicroelectronics support for more information about this application or send a request to edge.ai@st.com

1.1.2 Validated hardware

As any software expansion package, the X-LINUX-AI is supported on all STM32MP1 Series and it has been validated on the following boards:

- **STM32MP157C-DK2**[8]
- **STM32MP157C-EV1**[9]
- **STM32MP157A-EV1**[10]
- **STM32MP157 Avenger96 board**[1]

1.1.3 Software structure



1.2 X-LINUX-AI v2.0.0

Information

This version has been validated against the OpenSTLinux ecosystem release v2.0.0 and validated on STM32MP157x-DKx and STM32MP157x-EV1 boards.

1.2.1 Contents

- **TensorFlow Lite**[2] **2.2.0**
- **Coral Edge TPU**[3] accelerator support
- **armNN**[4] **20.05**
- **OpenCV**[5] **4.1.x**
- **Python**[6] **3.8.x** (enabling Pillow module)
- Support **STM32MP15xF**[7] devices operating at up to **800MHz**
- Python and C++ application samples
  - Image classification using TensorFlow Lite based on MobileNet v1 quantized model

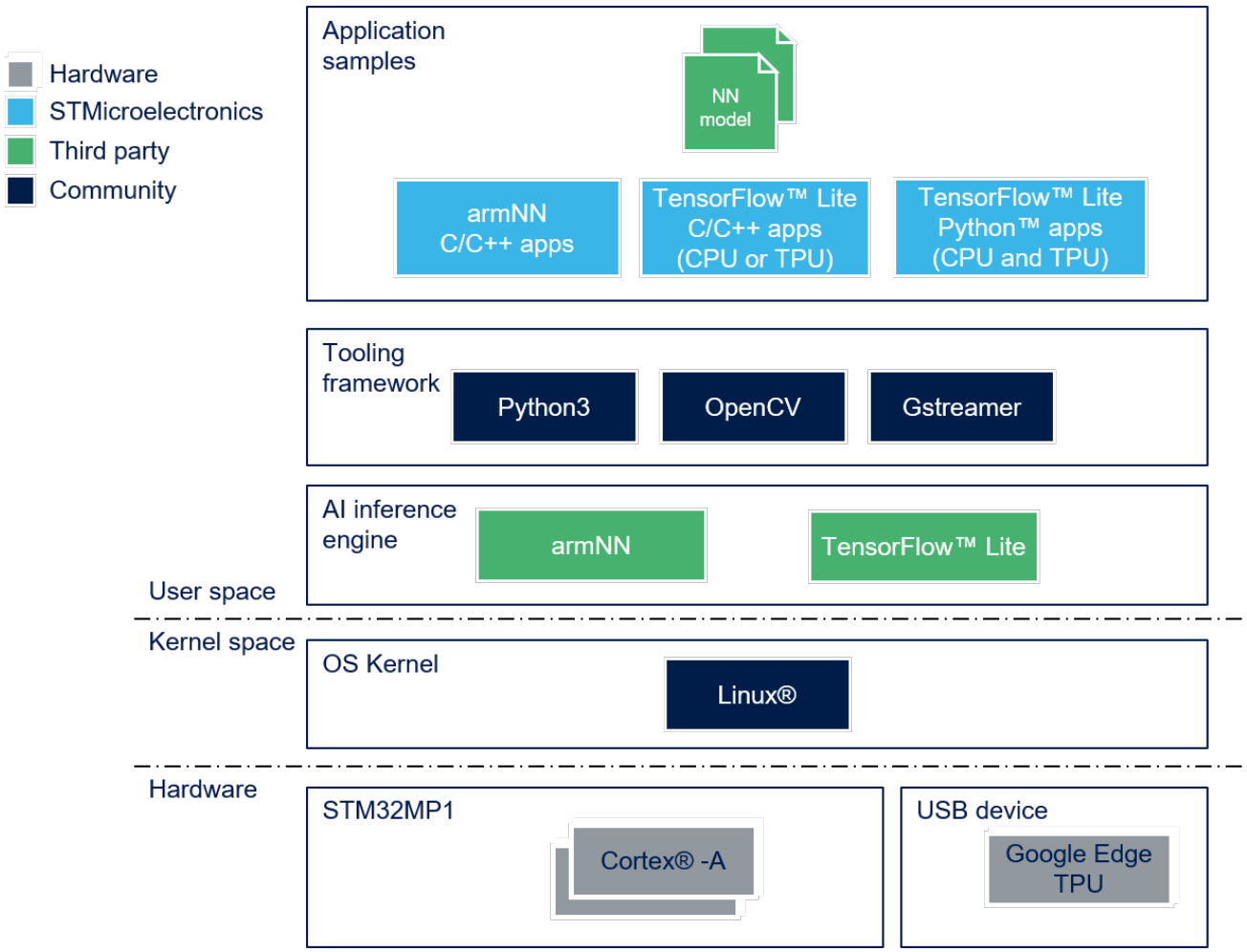
- Object detection using TensorFlow Lite based on COCO SSD MobileNet v1 quantized model
- Image classification using Coral Edge TPU based on MobileNet v1 quantized model and compiled for the Coral Edge TPU
- Object detection using Coral Edge TPU based on COCO SSD MobileNet v1 quantized model and compiled for the Coral Edge TPU
- Image classification using armNN TensorFlow Lite parser based on MobileNet v1 float model
- Object detection using armNN TensorFlow Lite parser based on COCO SSD MobileNet v1 quantized model

1.2.2 Validated hardware

As any software expansion package, the X-LINUX-AI is supported on all STM32MP1 Series and it has been validated on the following boards:

- **STM32MP157C-DK2**[8]
- **STM32MP157C-EV1**[9]
- **STM32MP157A-EV1**[10]

1.2.3 Software structure



2 Install from the OpenSTLinux AI package repository

Information

The STMicroelectronics packages repository service is provided for evaluation purposes only, its content may be updated at any time without notice and is therefore not approved for use in production.

All the generated X-LINUX-AI packages are available from the OpenSTLinux AI package repository service hosted at the non-browsable URL http://extra.packages.openstlinux.st.com/AI.

This repository contains AI packages that can be simply installed using **apt-*** utilities, which the same as those used on a Debian system:

- the **main** group contains the selection of AI packages whose installation is automatically tested by STMicroelectronics
- the **updates** group is reserved for future uses such as package revision update.

You can install them individually or by package group.

2.1 Prerequisites

ST boards prerequisites:

- Flash the Starter Package on your SDCard

  For OpenSTLinux ecosystem release v2.1.0 and **ecosystem release v2.0.0** :

  STM32MP157x-DKx Starter Package procedure

  or

  STM32MP157x-EV1 Starter Package procedure

  For OpenSTLinux ecosystem release v1.2.0 :

  STM32MP157x-DKx Starter Package procedure

  or

  STM32MP157x-EV1 Starter Package procedure

- Your board has an internet connection either through the network cable or through a WiFi connection.

Information

If your internet access depends on a proxy server, you should define the http_proxy environment variable with the following command before any `apt-*` commands:

```
Board $> export http_proxy='http://<proxy url>:<proxy port>/'
```

Avenger96 board prerequisites:

- The Avenger96 board starter image supporting OpenSTLinux v2.1.0 must be flashed on to your SD Card

  OpenSTLinux-2.1 based on Yocto Dunfell LTS and Linux 5.4.56 - v6.5 Starter Image

- Your board has an internet connection either through the network cable or through a WiFi connection.

Information

If your internet access depends on a proxy server, you should define the http_proxy environment variable with the following command before any `apt-*` commands:

```
Board $> export http_proxy='http://<proxy url>:<proxy port>/'
```

2.2 Configure the AI OpenSTLinux package repository

Once the board is booted, execute the following command in the console in order to configure the AI OpenSTLinux package repository:

```
For ecosystem release  v2.1.0 :

Board $> wget http://extra.packages.openstlinux.st.com/AI/2.1/pool/config/a/apt-
openstlinux-ai/apt-openstlinux-ai_1.0_armhf.deb
Board $> dpkg -i apt-openstlinux-ai_1.0_armhf.deb

For ecosystem release  v2.0.0 :
Board $> wget http://extra.packages.openstlinux.st.com/AI/2.0/pool/config/a/apt-
openstlinux-ai/apt-openstlinux-ai_1.0_armhf.deb
Board $> dpkg -i apt-openstlinux-ai_1.0_armhf.deb


For ecosystem release  v1.2.0 :
Board $> wget http://extra.packages.openstlinux.st.com/AI/1.2/pool/config/a/apt-
openstlinux-ai/apt-openstlinux-ai_1.0_armhf.deb
Board $> dpkg -i apt-openstlinux-ai_1.0_armhf.deb
```

Then synchronize the AI OpenSTLinux package repository.

```
Board $> apt-get update
```

2.3 Install AI packages

Warning

The software package is provided AS IS, and by downloading it, you agree to be bound to the terms of the software license agreement (SLA). The detailed content licenses can be found here.

2.3.1 Install all X-LINUX-AI packages

Command	Description
	Install all the X-LINUX-AI

Command	Description
`apt-get install packagegroup-x-linux-ai`	packages (TensorFlow Lite, Edge TPU, armNN, application samples and tools)

2.3.2 Install AI framework related packages

Command	Description
`apt-get install packagegroup-x-linux-ai-tflite`	Install X-LINUX-AI packages related to TensorFlow Lite framework (including application samples)
`apt-get install packagegroup-x-linux-ai-tflite-edgetpu`	Install X-LINUX-AI packages related to the Edge TPU framework (including application samples)
`apt-get install packagegroup-x-linux-ai-armnn-tflite`	Install X-LINUX-AI packages related to the armNN framework (including application samples)

2.3.3 Install individual packages

X-LINUX-AI v2.1.0 packages

Command	Description
`apt-get install arm-compute-library`	Install Arm Compute Library (ACL)
	Install Arm

Command	Description
`apt-get install arm-compute-library-tools`	Compute Library utilities (graph examples and benchmarks)
`apt-get install armnn`	Install arm Neural Network SDK (armNN)
`apt-get install armnn-tensorflow-lite`	Install armNN TensorFlow Lite parser
`apt-get install armnn-tensorflow-lite-examples`	Install armNN TensorFlow Lite examples
`apt-get install armnn-tfl-cv-apps-image-classification-c++`	Install C++ image classification example using armNN TensorFlow Lite parser
`apt-get install armnn-tfl-cv-apps-object-detection-c++`	Install C++ object detection example using armNN TensorFlow Lite parser
`apt-get install armnn-tools`	Install armNN utilitites such as unitary tests
`apt-get install python3-tensorflow-lite`	Install Python TensorFlow Lite inference engine
`apt-get install python3-tensorflow-lite-edgetpu`	Install Python TensorFlow Lite inference engine for Edge TPU
	Install Edge TPU libraries and the

Command	Description
`apt-get install tensorflow-lite-edgetpu`	USB rules
`apt-get install tensorflow-lite-tools`	Install Tensorflow Lite utilities
`apt-get install tflite-cv-apps-edgetpu-image-classification-c++`	Install C++ image classification example using Coral Edge TPU TensorFlow Lite API
`apt-get install tflite-cv-apps-edgetpu-image-classification-python`	Install Python image classification example using Coral Edge TPU TensorFlow Lite API
`apt-get install tflite-cv-apps-edgetpu-object-detection-c++`	Install C++ object detection example using Coral Edge TPU TensorFlow Lite API
`apt-get install tflite-cv-apps-edgetpu-object-detection-python`	Install Python object detection example using Coral Edge TPU TensorFlow Lite API
`apt-get install tflite-cv-apps-image-classification-c++`	Install C++ image classification using TensorFlow Lite
`apt-get install tflite-cv-apps-image-classification-python`	Install Python image classification example using TensorFlow Lite
	Install C++ object

Command	Description
`apt-get install tflite-cv-apps-object-detection-c++`	detection example using TensorFlow Lite
`apt-get install tflite-cv-apps-object-detection-python`	Install Python object detection example using TensorFlow Lite
`apt-get install tflite-edgetpu-benchmark`	Install benchmark application for Edge TPU models
`apt-get install tflite-models-coco-ssd-mobilenetv1`	Install TensorFlow Lite COCO SSD Mobilenetv1 model
`apt-get install tflite-models-coco-ssd-mobilenetv1-edgetpu`	Install TensorFlow Lite COCO SSD Mobilenetv1 model for Edge TPU
`apt-get install tflite-models-mobilenetv1`	Install TensorFlow Lite Mobilenetv1 model
`apt-get install tflite-models-mobilenetv1-edgetpu`	Install TensorFlow Lite Mobilenetv1 model for Edge TPU

X-LINUX-AI v2.0.0 packages

Command	Description
`apt-get install arm-compute-library`	Install Arm Compute Library (ACL)
`apt-get install arm-compute-library-tools`	Install Arm Compute Library utilities (graph examples and benchmarks)

Command	Description
`apt-get install armnn`	Install arm Neural Network SDK (armNN)
`apt-get install armnn-tensorflow-lite`	Install armNN TensorFlow Lite parser
`apt-get install armnn-tensorflow-lite-examples`	Install armNN TensorFlow Lite examples
`apt-get install armnn-tfl-benchmark`	Install armNN benchmark application for TensorFlow Lite models
`apt-get install armnn-tfl-cv-apps-image-classification-c++`	Install C++ image classification example using armNN TensorFlow Lite parser
`apt-get install armnn-tfl-cv-apps-object-detection-c++`	Install C++ object detection example using armNN TensorFlow Lite parser
`apt-get install armnn-tools`	Install armNN utilitites such as unitary tests
`apt-get install libedgetpu1`	Install Edge TPU libraries and the USB rules
`apt-get install python3-tensorflow-lite`	Install Python TensorFlow Lite inference engine
	Install Python

Command	Description
`apt-get install python3-tensorflow-lite-edgetpu`	TensorFlow Lite inference engine for Edge TPU
`apt-get install tensorflow-lite-tools`	Install Tensorflow Lite utilities
`apt-get install tflite-cv-apps-edgetpu-image-classification-c++`	Install C++ image classification example using Coral Edge TPU TensorFlow Lite API
`apt-get install tflite-cv-apps-edgetpu-image-classification-python`	Install Python image classification example using Coral Edge TPU TensorFlow Lite API
`apt-get install tflite-cv-apps-edgetpu-object-detection-c++`	Install C++ object detection example using Coral Edge TPU TensorFlow Lite API
`apt-get install tflite-cv-apps-edgetpu-object-detection-python`	Install Python object detection example using Coral Edge TPU TensorFlow Lite API
`apt-get install tflite-cv-apps-image-classification-c++`	Install C++ image classification using TensorFlow Lite
`apt-get install tflite-cv-apps-image-classification-python`	Install Python image classification example using TensorFlow Lite

Command	Description
`apt-get install tflite-cv-apps-object-detection-c++`	Install C++ object detection example using TensorFlow Lite
`apt-get install tflite-cv-apps-object-detection-python`	Install Python object detection example using TensorFlow Lite
`apt-get install tflite-edgetpu-benchmark`	Install benchmark application for Edge TPU models
`apt-get install tflite-models-coco-ssd-mobilenetv1`	Install TensorFlow Lite COCO SSD Mobilenetv1 model
`apt-get install tflite-models-coco-ssd-mobilenetv1-edgetpu`	Install TensorFlow Lite COCO SSD Mobilenetv1 model for Edge TPU
`apt-get install tflite-models-mobilenetv1`	Install TensorFlow Lite Mobilenetv1 model
`apt-get install tflite-models-mobilenetv1-edgetpu`	Install TensorFlow Lite Mobilenetv1 model for Edge TPU

Information

If you need more information about how to use apt-* utilities check the Package repository for OpenSTLinux distribution article.

3 Re-generate X-LINUX-AI OpenSTLinux distribution

With the following procedure, you can re-generate the complete distribution enabling the X-LINUX-AI expansion package. This procedure is mandatory if you want to update frameworks by yourself, or if you want to modify the application samples. For further details, please expand the contents...

3.1 Download the STM32MP1 Distribution Package

For ecosystem release v2.1.0 :
Install the STM32MP1 Distribution Package v2.1.0, **but do not initialize the OpenEmbedded environment (do not source the envsetup.sh).**

For ecosystem release v2.0.0 :

Install the STM32MP1 Distribution Package v2.0.0, **but do not initialize the OpenEmbedded environment (do not source the envsetup.sh).**

For ecosystem release v1.2.0 :

Install the STM32MP1 Distribution Package v1.2.0, **but do not initialize the OpenEmbedded environment (do not source the envsetup.sh).**

3.2 Install X-LINUX-AI environment for ST boards

- **Clone the meta-st-stm32mpu-ai git repositories**

Warning

The software package is provided AS IS, and by downloading it, you agree to be bound to the terms of the software license agreement (SLA). The detailed content licenses can be found here.

```
For X-LINUX-AI v2.1.0:

PC $> cd <Distribution Package installation directory>/layers/meta-st
PC $> git clone https://github.com/STMicroelectronics/meta-st-stm32mpu-ai.git -b v2.1.0

For X-LINUX-AI v2.0.0:
PC $> cd <Distribution Package installation directory>/layers/meta-st
PC $> git clone https://github.com/STMicroelectronics/meta-st-stm32mpu-ai.git -b v2.0.0
```

- **Set up the build environment**

```
PC $> cd ../..
PC $> DISTRO=openstlinux-weston MACHINE=stm32mp1 BSP_DEPENDENCY='layers/meta-st/meta-st-
stm32mpu-ai' source layers/meta-st/scripts/envsetup.sh
```

3.3 Install X-LINUX-AI environment for Avenger96 board

- Clone the meta-av96 and meta-st-stm32mpu-ai git repositories

Warning

The software package is provided AS IS, and by downloading it, you agree to be bound to the terms of the software license agreement (SLA). The detailed content licenses can be found here.

```
For X-LINUX-AI v2.1.0:

PC $> cd <Distribution Package installation directory>/layers
PC $> git clone https://github.com/dh-electronics/meta-av96.git -b av96_v65
PC $> cd <Distribution Package installation directory>/layers/meta-st
PC $> git clone https://github.com/STMicroelectronics/meta-st-stm32mpu-ai.git -b v2.1.0

For X-LINUX-AI v2.0.0:
PC $> git clone https://github.com/dh-electronics/meta-av96.git -b av96_v62
PC $> cd <Distribution Package installation directory>/layers/meta-st
PC $> git clone https://github.com/STMicroelectronics/meta-st-stm32mpu-ai.git -b v2.0.0
```

- Set up the build environment

```
PC $> cd ../..
PC $> META_LAYER_ROOT=layers DISTRO=openstlinux-weston MACHINE=stm32mp1-av96 BSP_DEPENDENC
Y='layers/meta-st/meta-st-stm32mp-addons layers/meta-st/meta-st-stm32mpu-ai' source layers
/meta-st/scripts/envsetup.sh
```

3.4 Build the image

```
PC $> bitbake st-image-ai
```

Information

Note that building the image could take long time depending on the host computer performance.

3.5 Flash the built image

Follow this link to see how to flash the built image.

4 How to use the X-LINUX-AI Expansion Package

4.1 Material needed

To use the X-LINUX-AI OpenSTLinux Expansion Package, choose one of the following materials:

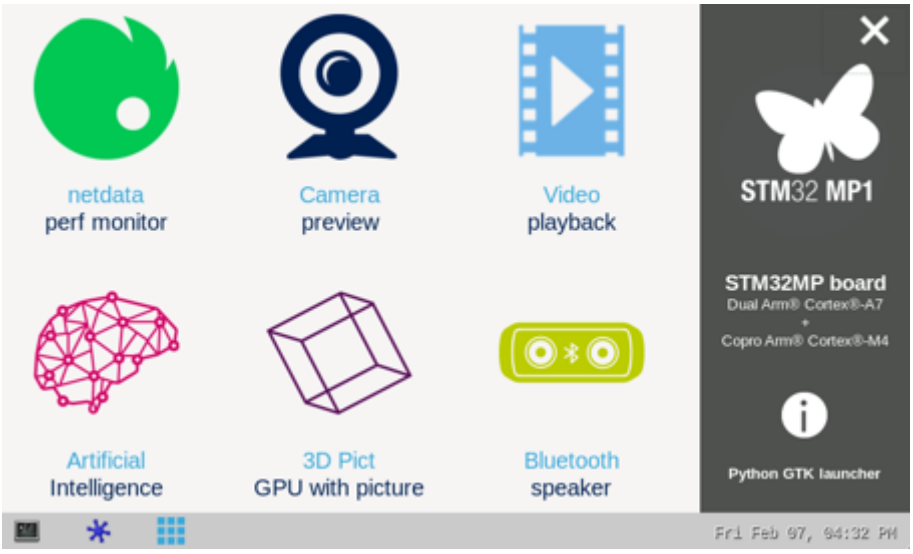
- **STM32MP157C-DK2**[8] + an UVC USB WebCam
- **STM32MP157C-EV1**[9] with the built in OV5640 parallel camera
- **STM32MP157A-EV1**[10] with the built in OV5640 parallel camera
- **STM32MP157 Avenger96 board**[1] + an UVC USB WebCam or the OV5640 CSI Camera mezzanine board[11]

Optional:

- **Coral USB Edge TPU**[3] accelerator

4.2 Boot the OpenSTlinux Starter Package

At the end of the boot sequence, the demo launcher application appears on the screen.



4.3 Install the X-LINUX-AI

Warning

The software package is provided AS IS, and by downloading it, you agree to be bound to the terms of the software license agreement (SLA). The detailed content licenses can be found here.

After having configured the AI OpenSTLinux package you can install the X-LINUX-AI components.

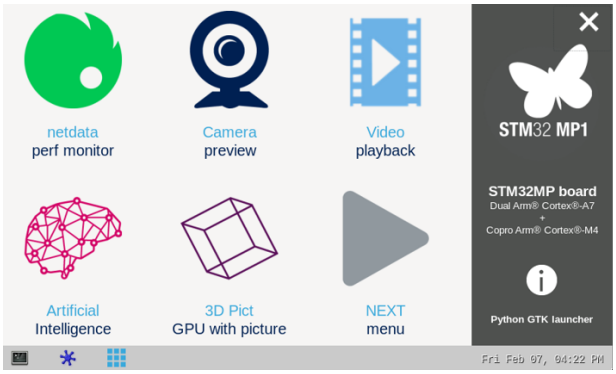
```
Board $> apt-get install packagegroup-x-linux-ai
```

And restart the demo launcher

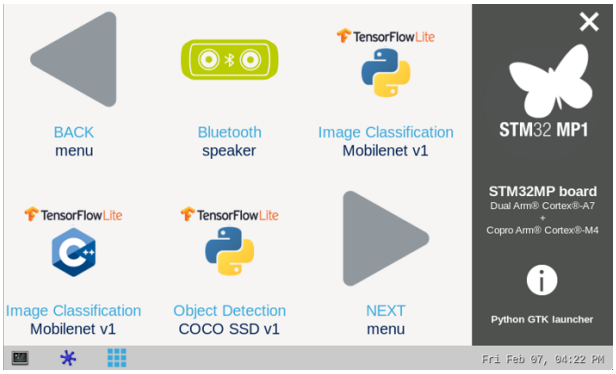
```
Board $> systemctl restart weston@root
```

4.4 Launch an AI application sample

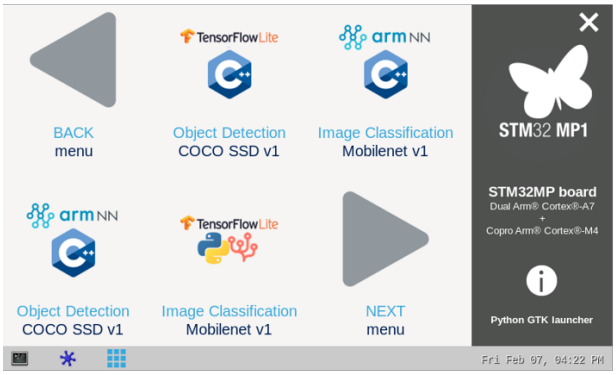
Once the demo launcher is restarted, notice that it is slightly different because new AI application samples have been installed. The demo laucher has the following appearance, and you can navigate into the different screens by using the **NEXT** or **BACK** buttons.



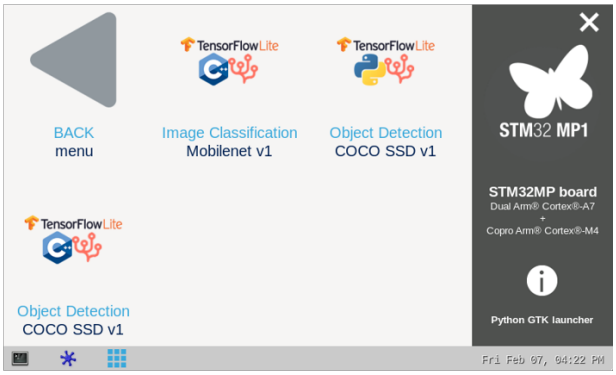
screen #1



screen #2



screen #3



screen #4

Screens 2, 3 and 4 contain AI application samples that are described within dedicated article available in the X-LINUX-AI application samples zoo page.

4.5 Enjoy running your own NN models

The above examples provide application samples to demonstrate how to execute models easily on the STM32MP1.

You are free to update the C/C++ application or Python scripts for your own purposes, using your own NN models.

Source code locations are provided in application sample pages.

5 References

- 1.01.11.2 Avenger96
- 2.02.1 TensorFlow Lite
- 3.03.13.2 Coral Edge TPU
- 4.04.1 armNN
- 5.05.1 OpenCV
- 6.06.1 Python
- 7.07.1 STM32MP1 series
- 8.08.18.2 STM32MP157C-DK2
- 9.09.19.2 STM32MP157C-EV1
- 10.010.110.2 STM32MP157A-EV1
- OV5640 CSI D3Camera board

Artificial Intelligence

Microprocessor Unit

Linux® is a registered trademark of Linus Torvalds.

Computer Vision

Arm® is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

Automatic current limit (LCD power improvement solution)

Software development kit (A programming package that enables a programmer to develop applications for a specific platform.)

Application programming interface

Board support package

USB Video Class

Multi Speed Internal oscillator (STM32 clock source)

Neural Network

Stable: 04.03.2021 - 14:33 / Revision: 04.03.2021 - 14:31

A quality version of this page, approved on *4 March 2021*, was based off this revision.

All X-LINUX-AI application samples are listed in this page.

1 TensorFlow Lite application samples

- Face recognition using TensorFlow Lite C++ API
- Image classification using TensorFlow Lite C++ API
- Object detection using TensorFlow Lite C++ API

- Image classification using TensorFlow Lite Python runtime
- Object detection using TensorFlow Lite Python runtime

2 Coral Edge TPU application samples

- Image classification using Coral Edge TPU TensorFlow Lite C++ API
- Object detection using using Coral Edge TPU TensorFlow Lite C++ API

- Image classification using Coral Edge TPU TensorFlow Lite Python runtime
- Object detection using using Coral Edge TPU TensorFlow Lite Python runtime

3 armNN application samples

- Image classification using armNN TensorFlow Lite parser
- Object detection using armNN TensorFlow Lite parser

Artificial Intelligence