Machine learning Demos Guide

IMX8X AI_ML Reference Design

Date: February, 2021 | Version 3.1

CONTENTS

FIGURES

ACRONYMS AND ABBREVIATIONS

Definition/Acronym/Abbreviation	Description
cd	Change directory
scp	Secure copy over the network
dfl	Default
Wi-Fi	Wireless fidelity
LTE	Long-Term Evolution
ML	Machine Learning
SVM	Support Vector Machine
CNN	Convolutional Neural Network
ARM NN	ARM Nueral Network
AI_ML board	Artificial intelligence and Machine Learning board featuring the NXP i.MX 8X MPU
AES	Advanced Encryption Standard
AHAB	Advanced High Assurance Boot
AWS	Amazon Web Services
BSP	Board Support Package
CA	Certificate Authority
CAAM	Cryptographic Acceleration and Assurance Module
CMS	Cryptographic Message Syntax
CSF	Command Sequence File
CSR	Certificate Signing Request
CST	Code Signing Tool
DCD	Device Configuration Data
GG	AWS Greengrass
OS	Operating System
OTP	One-Time Programmable
PKI	Public Key Infrastructure
SA	Signature Authority
SCFW	SCU Firmware
SDP	Serial Download Protocol
SECO	Security Controller
SPL	Secondary Program Loader
SRK	Super Root Key
SSK	Security Starter Kit
TPM	Trusted Platform Module
USB	Universal Serial Bus
TLS	Transport Layer Security
RSA	Rivest–Shamir–Adleman
IoT	Internet of Things
HSM	Hardware Security Module

PKCS#11	PKCS#11 (Public Key Cryptography Standards) defines an API to communicate with cryptographic security tokens such as smart cards, USB keys and HSMs
HW	Hardware
MQTT	Message Queuing Telemetry Transport
SSL	Secure Sockets Layer
SHA	Secure Hash Algorithm
SDK	Software Development Kit
ECC	Elliptic Curve Cryptography
ARN	Amazon Resource Name
SECO	SEcurity COntroller
FW	Firmware
NV RAM	Non Volatile Random Access Memory
API	Application Programming Interface
UUID	Universally Unique Identifier
SCP	Secure Copy Protocol
SCU	System Control Unit
IAM	AWS Identity and Access Management
TSS	TPM2 Software Stack
SVM	Support Vector Machine
CNN	Convolutional Neural Network
tf	Tensorflow

1 INTRODUCTION

1.1 Purpose of the Document

Purpose of this document is to use / understand / flash / demonstrate interfaces on iMX8ML_RD firmware.

1.2 About the system

This system contains iMX8X/iMX8M reference design with multiple interfaces. This is used for Machine learning experience.

1.3 Intended Audience

This document is for developers and end-users who want to understand /demonstrate Machine Learning Demos to run on iMX8X AIML and iMX8M THOR96 boards.

1.4 Prerequisites

Below are the list of Hardware and Software needed to demonstrate the Machine Learning Demos :

- x86 Host system having Linux Ubuntu 16.04 LTS installed (for developers to build Yocto image)
- For building machine-learning components, minimum 250 GB disk space is recommended.
- Basic understanding of Linux commands
- Setup will require following:
    - AI_ML Board/Thor96 Board
    - SD-card -16GB
    - MicroUSB debug cable
- Linux PC (Minicom for serial console)
- Internet connectivity (Wi-Fi/Ethernet) of Board and Linux PC should be on same Network
- Webcam or Mezzanine D3 Camera
- USB HUB / Mouse / Keyboard
- HDMI Display with HDMI connector
- Ethernet or Wi-Fi with Internet Connectivity (for audio google API Demo Only)

2 ML DEMOS BACKGROUND

To demonstrate board's capabilities for Machine Learning, few Audio and Video related ML demos are implemented. These demos mainly depend on OpenCV, Tensorflow, Caffe, ARM NN and few python packages. All video ML demos require video source (webcam or D3 Mezzanine based OV5640 camera) to capture live stream and perform some action on it. Moreover, Audio demos capture audio from DMIC or any other USB MIC and perform speech recognition on it.

All Demos are located inside home folder of board under "ARROW_DEMOS" name.

2.1 Copy Demos to SD Card

(If we have constraint of size of board and want to copy demos to USB or another partition then one should perform these steps else one should notperform these steps.)

The original firmware image is of size 7GB on SD card. The remaining space inside SDcard can be used as storage device for ML demos. To do so, create FAT or EXT4 partition on SDcard. ext4 partition is recommended as it is default Linux file system.
To do so follow below procedure.

- Flash SDcard with the required AIML or THOR96 firmware release. (Firmware release version must be BETA release 0.3 or above). (Refer User Guide for this.)
- In Linux HOST OS, open "disk" utility and see SDcard partitions in it. You can see image as under:

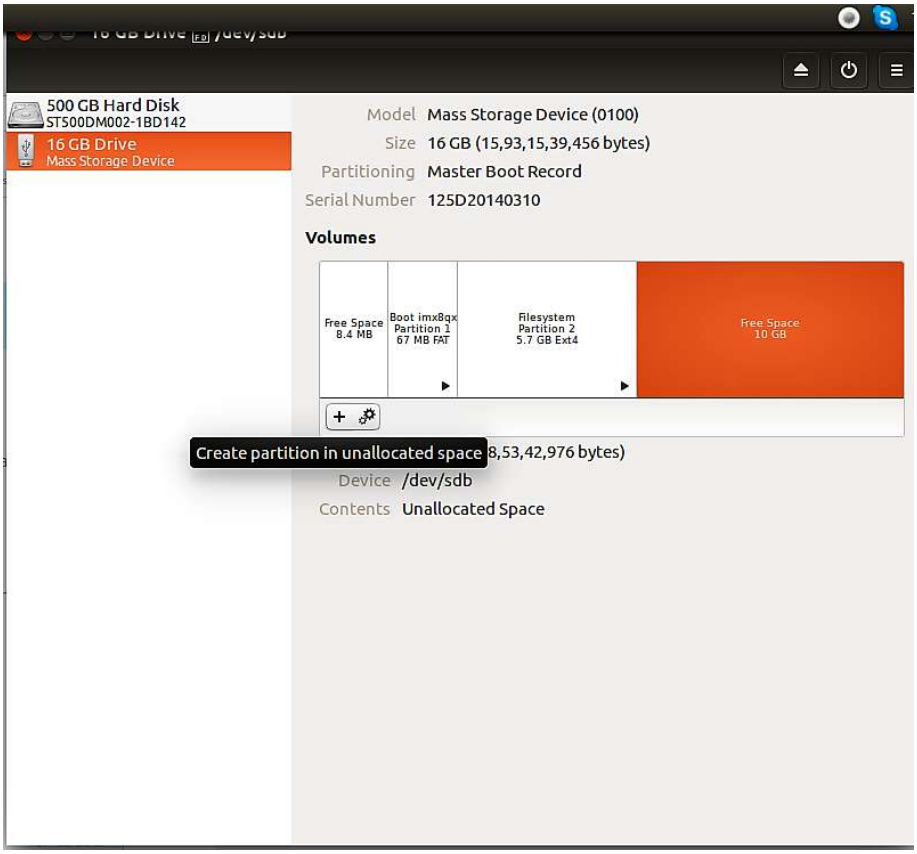

Figure 1: SD card partitions overview after flashing firmware

- As shown in above figure, inside SDcard partitions, one can see unused partition (10 GB) at end. One can utilize it.
- Now Click on “+” sign to create new partition.
- Please select file system ext4 and name it as shown below.

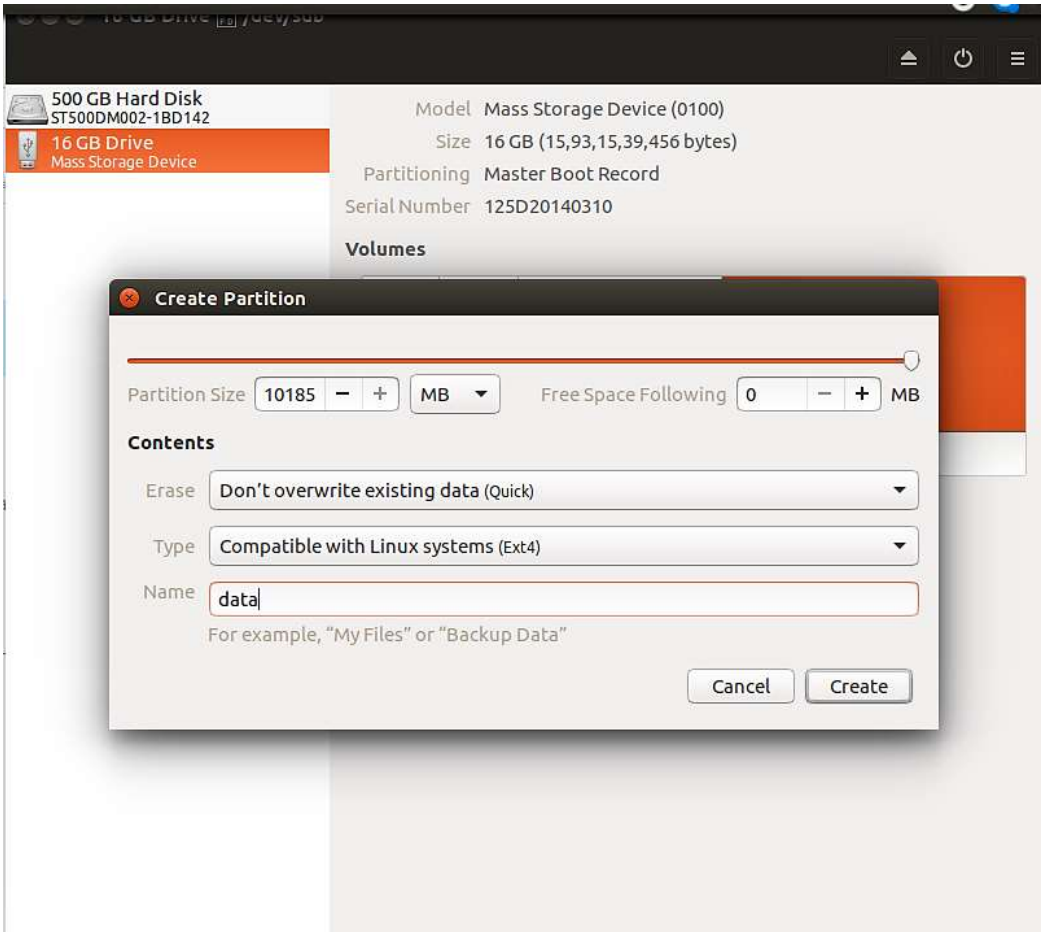

Figure 2: Create New EXT4 partition

- It will take some time to create partition and one should be able to mount that partition. (See below Figure for reference.)

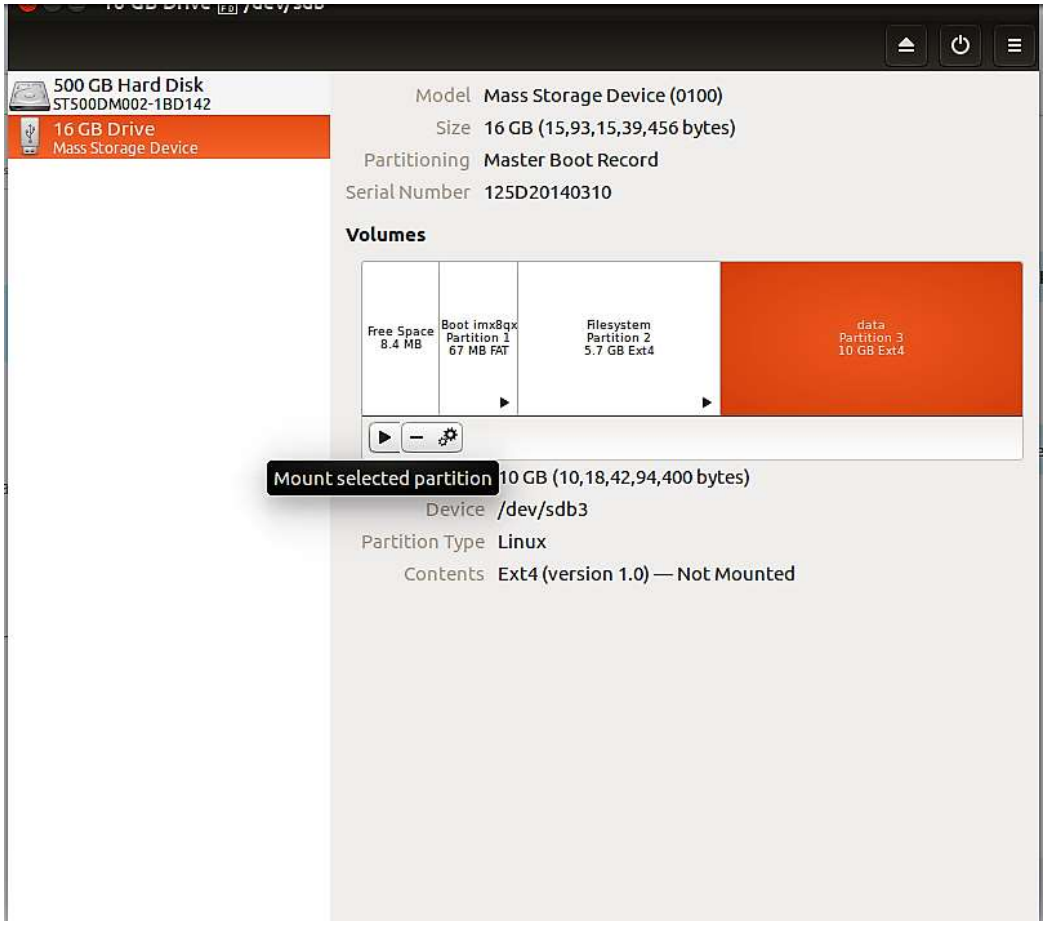

Figure 3: SD card partition after creating new one

- Copy ARROW DEMOS on this partition. In case one is not able to do, then unmount and mount again.
- After successful copy, boot the AIML /Thor96 board with this SDcard.
- After boot up one can see Demos at below location:

```
# ls -la /run/media/mmcblk1p3/ARROW_DEMOS/
total 40
drwxrwxrwx 7 1000 tracing 4096 Apr 9 14:22 .
drwx------ 5 1000 tracing 4096 Apr 9 14:22 ..
drwxrwxrwx 4 1000 tracing 4096 Apr 1 13:07 ai-crowd_count
drwxrwxrwx 6 1000 tracing 4096 Apr 9 14:40 face_recognition
drwxrwxrwx 4 1000 tracing 4096 Apr 9 12:24 real-time-object-detection
-rwxrwxrwx 1 1000 tracing 9322 Apr 9 14:18 run_ml_demos.sh
drwxrwxrwx 2 1000 tracing 4096 Apr 12 11:12 speech_recognition_tensorflow
```

2.2 Run Setup

All the needed Python packages for ML demos are present in AIML/THOR96 firmware image. The demos run using Python3, hence Python3 package is added. There is support for Python and Python3 PIP Package. Using these packages, one can add or remove any python package and remove dependencies for rebuilding firmware image each time.

To install any python3 or python package use command:

<pip3 or pip> install <PACKAGE NAME OR PACKAGE WHEEL NAME>

To remove any python3 or python package use command:

<pip3 or pip> uninstall <PACKAGE NAME>

Some python packages, viz. tensorflow has no standard python wheel package for ARM AARCH64 platform so it has to be cross compiled from source and need to create one (wheel package) for board. The wheel packages is available inside home folder to setup python module on the board. To do so, user need to run setup_ml_demo.sh script using below commands:

sh ~/setup_ml_demo.sh

This script takes approximately 15-20 minutes to install all required python3 packages for ML demos so it has to be executed atleast once before running all the ML demos. The wheel package is provided with all packages inside home folder so this script does not required any internet connectivity for installing packages. However, apart from script if you want to install any package as describe above then you required clientless (without any firewall) internet connectivity.

ML Demos does not come with default firmware images. This is needed to limit the size of original firmware image as it takes more time to flash SDcard. In addition, with separate release of ML demos one can remove dependencies of firmware image release. This helps to improve the demos without affecting firmware packages if there is no dependencies on software or packages.

3 ML DEMOS BACKGROUND

To run ML Demos - use run_ml_demo.sh shell script. This script requests for user preferences like demo type, camera types, camera node entry, desired MIC etc., one can select option to run ML demos.

In this section, below is the step to run each demo.
Detailed description of Demo is available in the next section.

Note: If the user is validating the Machine Learning Demos with AI_ML board, the Environment dtb has to be updated from uboot

For aiml,
 # setenv fdt_file imx8qxp-aiml-ei-ov5640.dtb
 # saveenv
 # boot

3.1 Crowd Counting Demo

This is a demo application using Python, QT, and Tensorflow to run on embedded devices for Crowd counting. In this demo, the heads/persons in the crowd are counted. This is useful in human flow monitoring or traffic control.
This demo runs on either pre-captured image mode or in Live-Camera mode. In pre-captured image mode, few sample images are taken and find head counts in those images. In live-camera mode live frames are captured through webcam or D3 mezzanine camera and try to find head count from it. User can select any mode by clicking on GUI.

Pre-requisite:

- Webcam or D3 Mezzanine camera
- USB mouse
- HDMI Display having minimum 1080p resolution

Steps to run Demo:

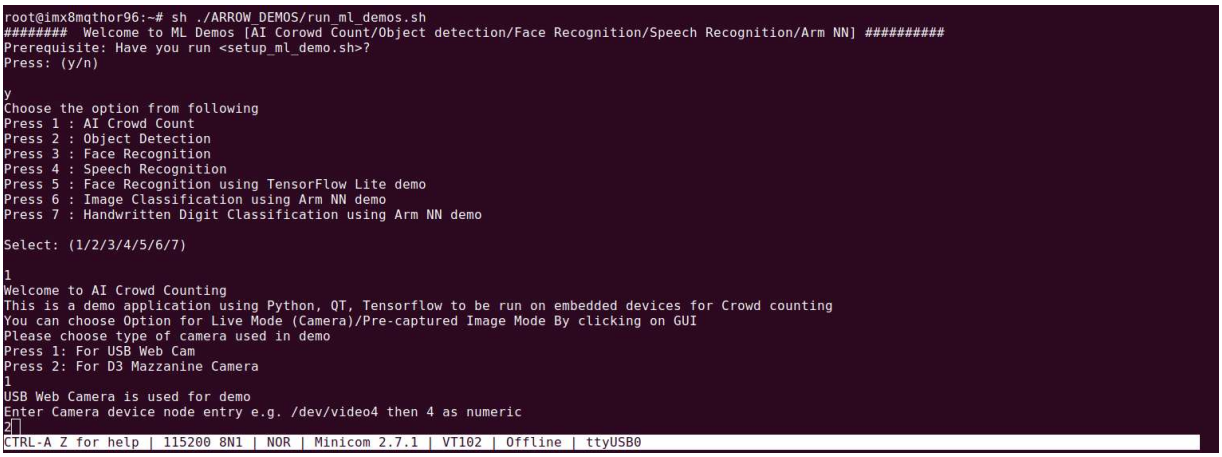

Figure 4: Run Crowd Count Demo

- Run “sh ./ARROW_DEMOS/run_ml_demos.sh” script and select option 1.
- See below full log to run demo, where user input is in BOLD RED fonts.

```
sh ./ARROW_DEMOS/run_ml_demos.sh
######### Welcome to ML Demos [AI Corowd Count/Object detection/Face Recognition/Speech Recognition/Arm NN] ##########
Prerequisite: Have you run <setup_ml_demo.sh>?
Press: (y/n)
y
Choose the option from following
Press 1 : AI Crowd Count
Press 2 : Object Detection
Press 3 : Face Recognition
Press 4 : Speech Recognition
Press 5 : Face Recognition using TensorFlow Lite demo
```

Press 6 : Image Classification using Arm NN demo
Press 7 : Handwritten Digit Classification using Arm NN demo
Select: (1/2/3/4/5/6/7)
1
Welcome to AI Crowd Counting
This is a demo application using Python, QT, Tensorflow to be run on embedded devices for Crowd counting
You can choose Option for Live Mode (Camera)/Pre-captured Image Mode By clicking on GUI
Please choose type of camera used in demo
Press 1: For USB Web Cam
Press 2: For D3 Mazzanine Camera
1
USB Web Camera is used for demo
Enter Camera device node entry e.g. /dev/video4 then 4 as numeric
2

- Here "2" i.e /dev/video2 is webcam camera node using which frame are captured. User can check his/her node entry by plugging/unplugging webcam and see which /dev/ node entry is removed/showed.
- By default, demo runs in pre-capture mode, and see output of headcount with inference time and date. Inference time is time taken to process one frame and finding headcount from it. Inference time is in milli-seconds (ms). In pre-capture mode, Inference time is around 2000 to 3000 ms (2 to 3 sec) while in live-mode, inference time is 200 to 300 ms. That's because we use smaller input image in live-mode.

- As shown in below picture, on the right side the "Density Maps" are shown for the input image on the left side.

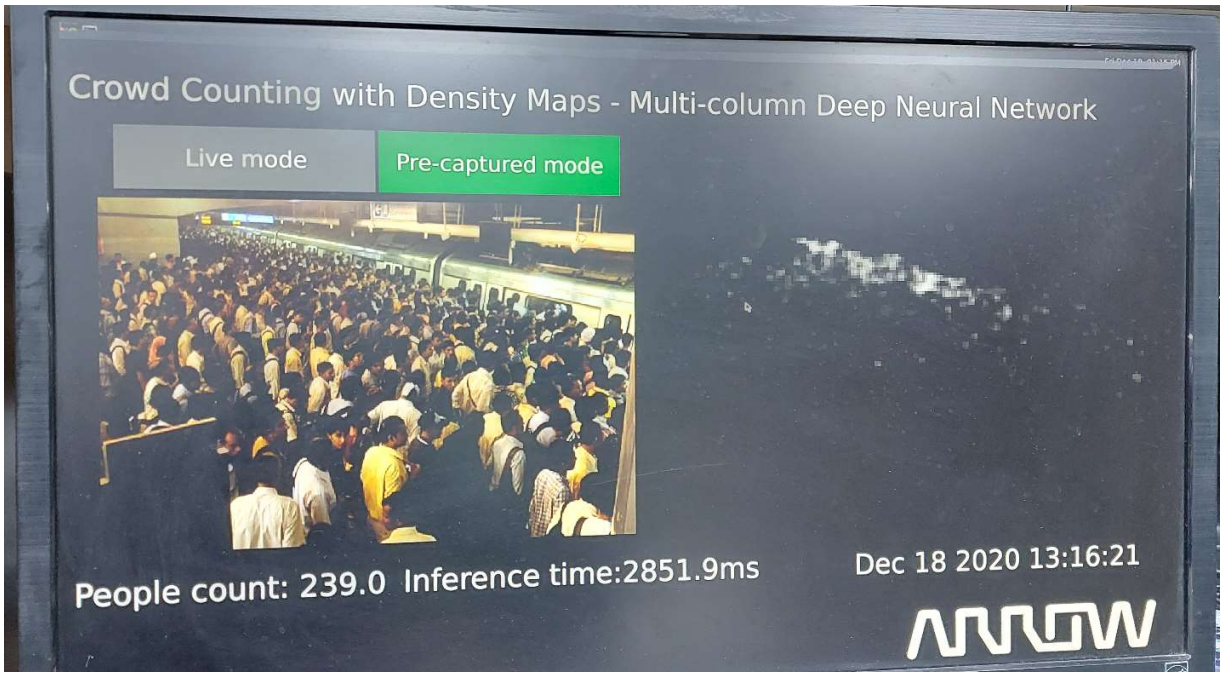

Figure 5: Crowd Count Pre-Captured Mode

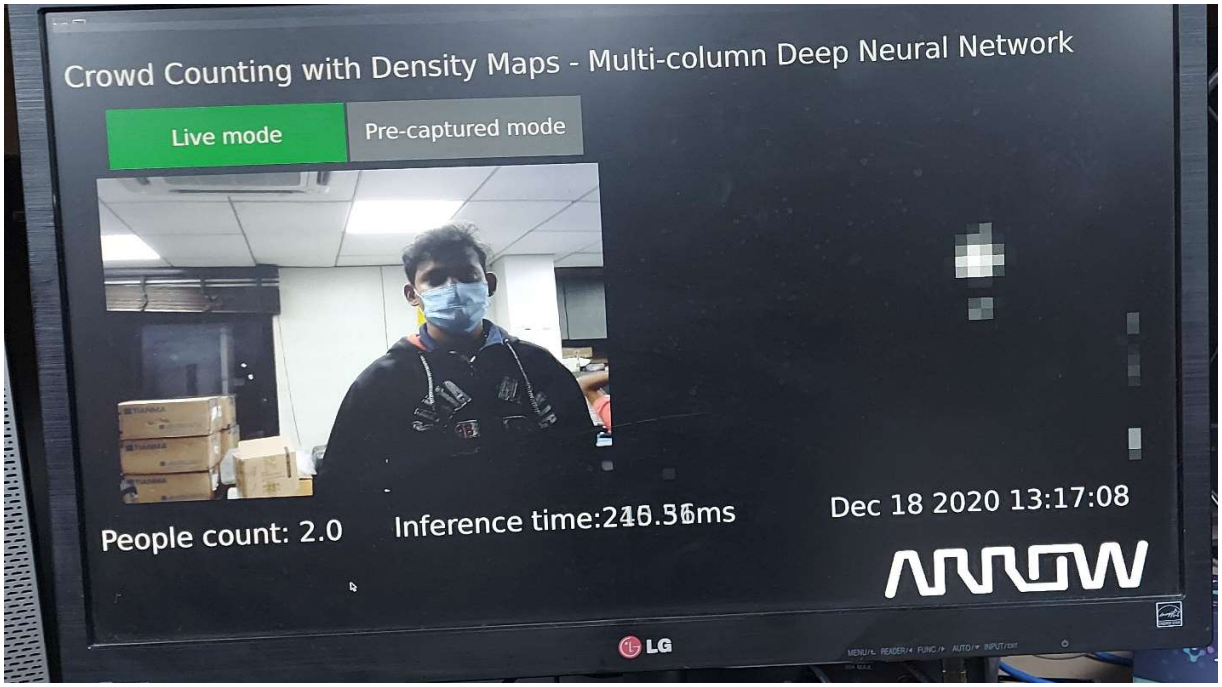

Figure 6: Crowd Count Live Mode

3.2 Object Detection Demo

In this Demo, few objects are detected like aero-plane, bicycle, bus, car, cat, cow, dog, horse, motorbike, person, sheep, train (objects necessary for self-driving cars).
There are two versions of Object detection. Both demo uses same caffe based object detection model so accuracy remains same for both the demos. The only difference is in video output.
In Fast Object detection, there is smooth video. Here two python processes are created. In one python process, a frame is sampled at a time for object detection on that frame. Another python process will use this object detection credentials and apply it on the entire frame that was read from camera. Thus, camera output is smooth but object detection takes 2 to 3 secs to give actual real-time output.
In Slow Object detection, we have single python process, which reads camera frame first and does object detection on it. Object detection is done on each frame and thus video output is choppy. However, object detection output is real time that is without delay. this demo is thus perfect to identify board's capabilities.

Pre-requisite:

- Webcam or D3 Mezzanine camera
- USB mouse
- HDMI Display having minimum 1080p resolution
- Objects which we want to detect
- Object Images and another PC or laptop (in case of no real objects)

Steps to run Demo:

Run "sh ./ARROW_DEMOS/run_ml_demos.sh" **script** and select **option 2.**

```
root@imx8mqthor96:~# sh ./ARROW_DEMOS/run_ml_demos.sh
########  Welcome to ML Demos [AI Corowd Count/Object detection/Face Recognition/Speech Recognition/Arm NN] ##########
Prerequisite: Have you run <setup_ml_demo.sh>?
Press: (y/n)

y
Choose the option from following
Press 1 : AI Crowd Count
Press 2 : Object Detection
Press 3 : Face Recognition
Press 4 : Speech Recognition
Press 5 : Face Recognition using TensorFlow Lite demo
Press 6 : Image Classification using Arm NN demo
Press 7 : Handwritten Digit Classification using Arm NN demo

Select: (1/2/3/4/5/6/7)

2
Welcome to object Detection
This model detect aeroplane, bicycle, bus, car, cat, cow, dog, horse, motorbike, person, sheep, train (objects necessary for self-driving)

Please choose type of camera used in demo
Press 1: For USB Web Cam
Press 2: For D3 Mazzanine Camera
1
USB Web Camera is used for demo
Enter Camera device node entry e.g. /dev/video4 then 4 as numeric
2
Which Object detection Demo you want to run:
Press 1: For Fast Object Detection. Here Video Output is smooth.
Because we randomly sample only few frames from camera and applied same object detections on the rest of frames.

Press 2: For Slow Object detection. Here we applied object detections on each camera frame and display output.
So video output is very choppy. But get real-time detection here.

Please Select: (1/2)

2
```

Figure 7: Run Object Detection Demo

See below full log to run demo, where user input is in **BOLD RED** fonts.

```
sh ./ARROW_DEMOS/run_ml_demos.sh
```

```
######### Welcome to ML Demos [AI Corowd Count/Object detection/Face Recognition/Speech
Recognition/Arm NN] ##########
Prerequisite: Have you run <setup_ml_demo.sh>?
Press: (y/n)
y
Choose the option from following
Press 1 : AI Crowd Count
Press 2 : Object Detection
Press 3 : Face Recognition
Press 4 : Speech Recognition
Press 5 : Face Recognition using TensorFlow Lite demo
Press 6 : Image Classification using Arm NN demo
Press 7 : Handwritten Digit Classification using Arm NN demo
Select: (1/2/3/4/5/6/7)
2
Welcome to object Detection
This model detect aeroplane, bicycle, bus, car, cat, cow, dog, horse, motorbike, person, sheep,
train (objects necessary for self-driving)
Please choose type of camera used in demo
Press 1: For USB Web Cam
Press 2: For D3 Mazzanine Camera
1
USB Web Camera is used for demo
Enter Camera device node entry e.g. /dev/video4 then 4 as numeric
2
Which Object detection Demo you want to run:
Press 1: For Fast Object Detection. Here Video Output is smooth.
Because we randomly sample only few frames from camera and applied same object detections
on the rest of frames.
Press 2: For Slow Object detection. Here we applied object detections on each camera frame
and display output.
So video output is very choppy. But get real-time detection here.
Please Select: (1/2)
2

Slow Object Detection demo
Loading model...
Starting video stream...

(python3:3911): GStreamer-CRITICAL **: gst_element_get_state: assertion 'GST_IS_ELEMENT
(element)' failed
Using Wayland-EGL
Using the 'xdg-shell-v6' shell integration
Total Elapsed time: 15.52
Approx. FPS: 0.90
Exiting Demo...
```

Now in this demo, object is provided in front of camera to be detected. Person is the best real-time object for detection. Most of the object is easily available outside the environment. However, to test model actual object is not needed. Some good image of object can be provided instead of real object to verify the model. Input image must be provided with correct angle and exposure of light to detect objects. Place input image inside PC or Laptop and set camera in front of it. Please refer below figure for setup. Also provided sample input image and its outputs.

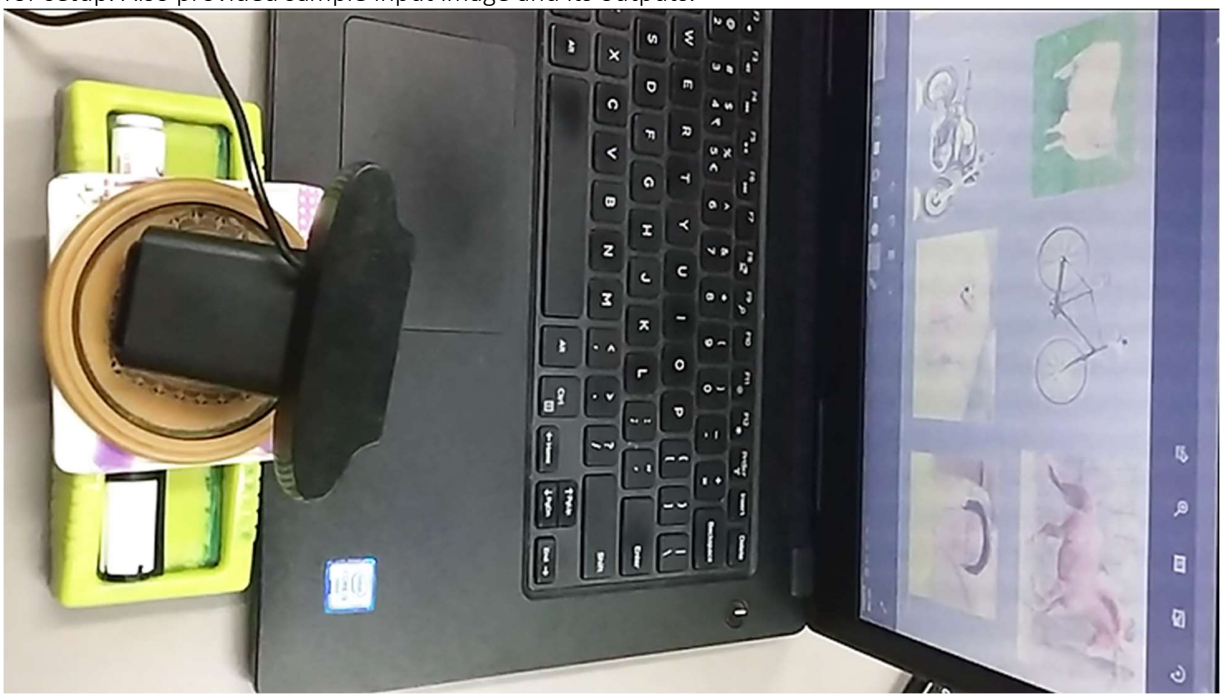

Figure 8: Setup for object detection

Figure 9: Sample Input Image for Object Detection

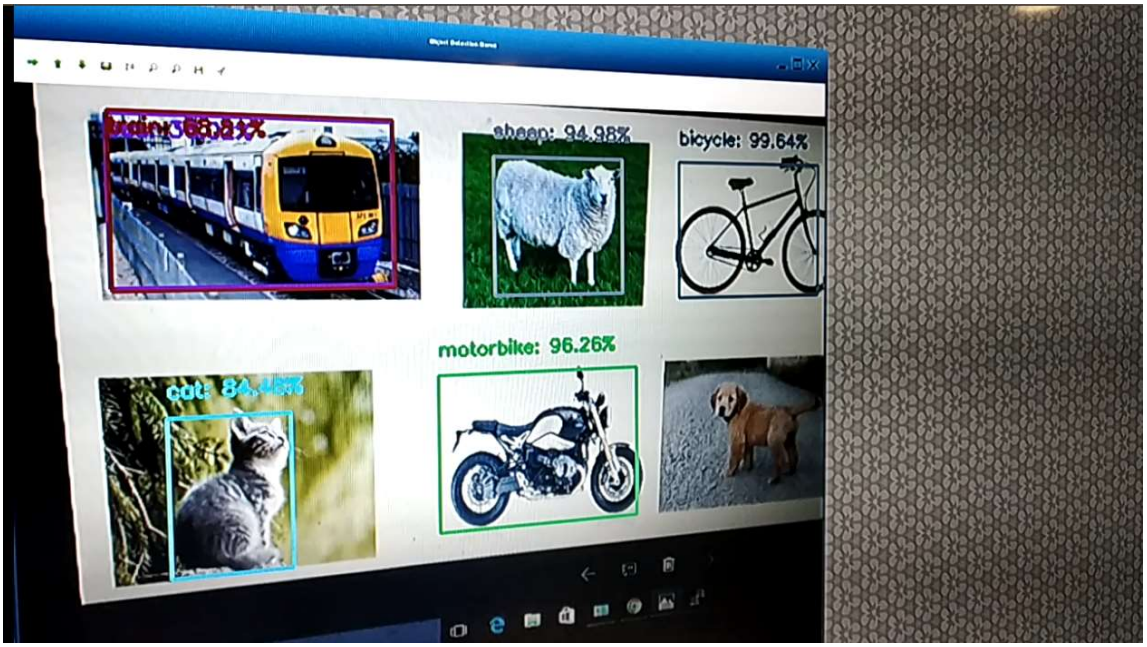

Figure 10: Sample Object detection output

It can be seen in above figure that in the output image the dog is not detected, as it is not in good angle and exposure. The model detects it with very low percentage and it is ignored due to low confidence. Here the model is confused on the train image having train and bus. If the model is retrained to train more images then there is a better chance to achieve better performance.

3.3 Face Recognition Demo

In face recognition demo, the face is detected from the given image or video frame. After detecting the face, the pre-trained model is used to recognize the face.
In this demo, few ML techniques and OpenCV face recognition models has been used along with the "face recognition" python module. Based on speed and accuracy it can be concluded that:

- **FACE DETECTION:** OpenCV is very basic model. It is slow in face detection as compared to ML module but fast compared to Python module.
- **FACE RECOGNITION:** Here OpenCV is winner. ML module is slow though but faster than python module in terms of recognition time.
- **ACCURACY:** Here Python module is winner. ML has more accuracy than OpenCV model.

Based on above conclusion, there are two face recognition demos.
In **Fast face recognition demo**, python face recognition module is used, which is more accurate than other two, but very slow. To make it fast, two python processes are created. In one python process, one frame at a time is sampled and face recognition is done on it. Another python process will use this face recognition data and apply it on the entire frame it reads from camera. Thus, camera output is smooth but real face detection take 3 to 4 secs to provide real-time output.
In **real-time face recognition demo**, there is single python process, which reads camera frame first and does face recognition on it. Therefore, here face recognition is done on each frame and video output is choppy. However, there is real-time output without any delay. this demo thus helps to identify board's capabilities.

Both the demo have **capability to retrain model** run-time on board. For that, new label (Person Name) has to be provided. After that, new training data (photos) is captured, where the person's face is labeled. If camera frame has more than one face, then it is considered as a malfunction or wrong frame and won't consider that frame as valid for training data. It is user's responsibility to provide correct face data. If user provides different face images for training under same name then the model will be confused as it finds separate face vector data for different faces but found same name (label). In addition, if user creates multiple label with same faces data, then also model will easily get confused and will give output with mix of those labels.

Provide **correct data and label** for training data. This training data is same for both the demos. Therefore, no need to create separate training data for both demos for example "anil_fast" and "anil_slow". Also user need to provide data with different posture like smiling face, sad face, neutral face, angry face, face with eyeglasses etc. In addition, create training data with small (far) and big (near) faces.

There are approximately 50 photos of user. User can interrupt this capturing process by **pressing "q"** button from keyboard attached to board. Same interrupt process is working for testing as well. Means user can cancel testing any time by pressing "q" button.
If user want to re-train model again without capturing new training data, then label is not provided and but run training process.

Pre-requisite:

- Webcam or D3 Mezzanine camera
- USB Mouse

- USB Keyboard
- Use USB Hub (if have USB webcam) because we have only two USB ports
- HDMI Display having minimum 1080p resolution

Steps to run Demo:

Run "sh ./ARROW_DEMOS/run_ml_demos.sh" **script** and select **option 3.**

Figure 11: Face Recognition Demo Testing

See below full log to run demo, where user input is in **BOLD RED** fonts.

```
# sh ./ARROW_DEMOS/run_ml_demos.sh
######## Welcome to ML Demos [AI Corowd Count/Object detection/Face Recognition/Speech Recognition/Arm NN] ##########
Prerequisite: Have you run <setup_ml_demo.sh>?
Press: (y/n)
y
Choose the option from following
Press 1 : AI Crowd Count
Press 2 : Object Detection
Press 3 : Face Recognition
Press 4 : Speech Recognition
Press 5 : Face Recognition using TensorFlow Lite demo
Press 6 : Image Classification using Arm NN demo
Press 7 : Handwritten Digit Classification using Arm NN demo
Select: (1/2/3/4/5/6/7)
3
Welcome to Face Recognition Demo
This is a demo application using Python modules to be run on embedded devices for recognition of
```

```
faces.
We already train model using given images. But can retrain model with new images and can increase accuracy of model.
Please choose type of camera used in demo
Press 1: For USB Web Cam
Press 2: For D3 Mazzanine Camera
1
USB Web Camera is used for demo
Enter Camera device node entry e.g. /dev/video4 then 4 as numeric
2
Please choose which face recognition demo you want to run
Press 1: For Fast Face recognition. Here Video Output is smooth.
Because we randomly sample only few frames from camera and applied same face recognition on the rest of frames.
As we only sample few frames, here output is slow. You can get correct result around after 2-3 sec.
Press 2: For Slow Face Recognition. Here we applied face recognition on each camera frame and display output.
So video output is very choppy. But get real-time detection here.
Please Select: (1/2)
2
Real-time face recognition demo
Please choose mode of operation for demo
Press 1: Test Model
Press 2: Train Model
1
Face recognition Testing ...
Initializing Face recognition model...
Starting video stream...

(python3:3946): GStreamer-CRITICAL **: gst_element_get_state: assertion 'GST_IS_ELEMENT (element)' failed
Using Wayland-EGL
Using the 'xdg-shell-v6' shell integration
`q` pressed, Exiting...
Total Elapsed time: 12.07 sec
Approx. FPS: 0.99
Exiting Demo...
```

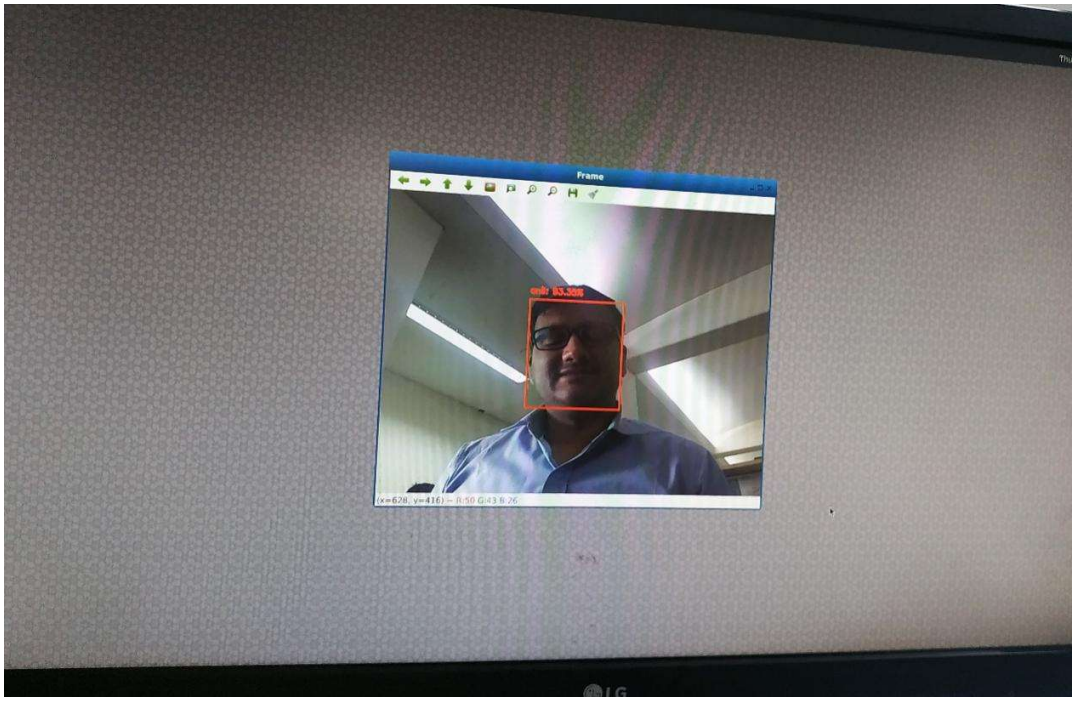

Figure 12: Face Recognition Output

If user want to re-train model with new person/label then he/she need to select train model with label as shown in below logs:

```
Please choose mode of operation for demo
Press 1: Test Model
Press 2: Train Model
2
Face recognition Training ...
Please provide new label... e.g. Joshua
With this label, we will create new training dataset and will our retrain model.
If you don't want to create new training dataset but simply retrain model with existing images,
then press enter with
out giving any label name
anil
Directory anil created
Starting video stream to capture new training dataset...

(python3:3993): GStreamer-CRITICAL **: gst_element_get_state: assertion 'GST_IS_ELEMENT
(element)' failed
Using Wayland-EGL
Using the 'xdg-shell-v6' shell integration
[]
detected faces -  0
[]
detected faces -  0
[]
detected faces -  0
```

```
[(125, 440, 254, 311)]
detected faces -  1
new training image anil_0_20190409-143240.png saved!
[]
detected faces -  0
[]
detected faces -  0
[(36, 366, 222, 180)]
detected faces -  1
new training image anil_1_20190409-143240.png saved!
[]
detected faces -  0
[(23, 438, 290, 171)]
detected faces -  1
new training image anil_2_20190409-143240.png saved!
[]
detected faces -  0
[]
detected faces -  0
[]
detected faces -  0
`q` pressed, Exiting...          <---User Press "q" here from keyboard attached to board
Total Elapsed time: 31.82 sec
Directory anil removed
Quantifying faces from training dataset...
Processing Image :  training_data/unknown/00000022.JPG
Processing Image :  training_data/unknown/00000034.JPG
Processing Image :  training_data/unknown/ellie_sattler.jpg
Processing Image :  training_data/unknown/00000009.jpg
Processing Image :  training_data/unknown/00000004.jpg
Encoding done for image = training_data/unknown/00000022.JPG
Encoding done for image = training_data/unknown/00000034.JPG
Processing Image :  training_data/unknown/ds3.jpg
Processing Image :  training_data/unknown/00000016.jpg
```

This will re-train model with new label **"anil"** training images along with other training images which are already provided inside **training_data** folder.

Create new training directory anil or copy new images inside existing directory if directory with same name already exists. In case of low confidence for any face, re-train the model with that face and that will increase confidence for that image.

Training takes few minutes (around 10-15 minutes) depending on number of training images. In training, four cores of CPU are utilized and performs training on images using four concurrent python process, thus speeding up the execution.

If User simply want to re-train model without providing any new input images or labels then press "enter" and **leave blank when script asks for new label**. This scenario is only useful if your existing trained model is corrupted or deleted by mistake. We can do training on our HOST Linux machine and can use trained model for testing but python module version must be same or compatible with version of board's package.

3.4 Speech Recognition Demo

This audio demo is use case of on-board DMIC. There are two demos for testing MIC.
In First Demo, a custom trained model is used for few selected keywords like **"yes no up down left right on off stop go".** In this demo an accuracy of 80-85% is achieved, as it is bit hard for audio to identify correct keyword compared to image with accuracy more than 90% by CNN (convolutional neural network).

Steps to run Demo:

Run "sh ./ARROW_DEMOS/run_ml_demos.sh" **script** and select **option 4.**

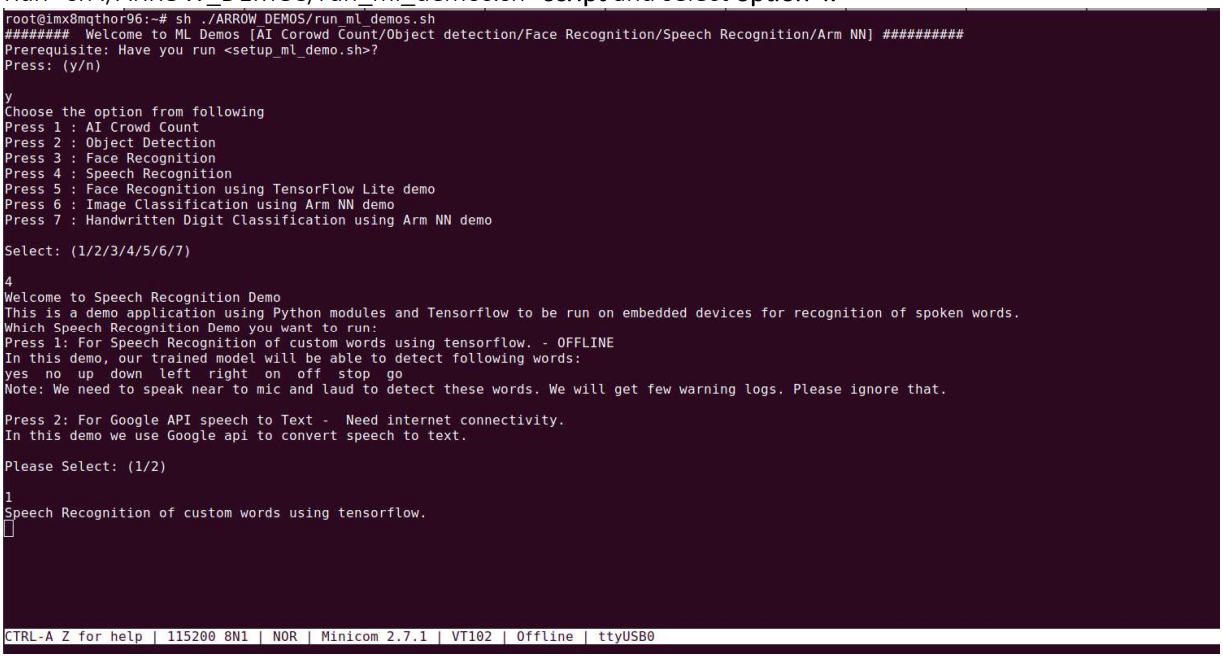

Figure 13: Run Speech Recognition Demo

```
# sh ./ARROW_DEMOS/run_ml_demos.sh
######### Welcome to ML Demos [AI Corowd Count/Object detection/Face Recognition/Speech
Recognition/Arm NN] ###########
Prerequisite: Have you run <setup_ml_demo.sh>?
Press: (y/n)
y
Choose the option from following
Press 1 : AI Crowd Count
Press 2 : Object Detection
Press 3 : Face Recognition
Press 4 : Speech Recognition
Press 5 : Face Recognition using TensorFlow Lite demo
Press 6 : Image Classification using Arm NN demo
Press 7 : Handwritten Digit Classification using Arm NN demo
Select: (1/2/3/4/5/6/7)
4
Welcome to Speech Recognition Demo
This is a demo application using Python modules and Tensorflow to be run on embedded
```

```
devices for recognition of spoken words.
Which Speech Recognition Demo you want to run:
Press 1: For Speech Recognition of custom words using tensorflow. - OFFLINE
In this demo, our trained model will be able to detect following words:
yes no up down left right on off stop go
Note: We need to speak near to mic and laud to detect these words. We will get few warning logs. Please ignore that.
Press 2: For Google API speech to Text - Need internet connectivity.
In this demo we use Google api to convert speech to text.
Please Select: (1/2)
1
Speech Recognition of custom words using tensorflow.
.
.
.
ALSA lib ../../alsa-lib-1.1.9/src/confmisc.c:1281:(snd_func_refer) Unable to find definition 'cards.imx-spdif.pcm.surround51.0:CARD=0'
ALSA lib ../../alsa-lib-1.1.9/src/conf.c:4568:(_snd_config_evaluate) function snd_func_refer returned error: No such file or directory
ALSA lib ../../alsa-lib-1.1.9/src/conf.c:5047:(snd_config_expand) Evaluate error: No such file or directory
ALSA lib ../../../alsa-lib-1.1.9/src/pcm/pcm.c:2564:(snd_pcm_open_noupdate) Unknown PCM surround21
ALSA lib ../../alsa-lib-1.1.9/src/confmisc.c:1281:(snd_func_refer) Unable to find definition 'cards.imx-spdif.pcm.surround51.0:CARD=0'
.
.
.
.
Available Audio Devices : Index
imx-spdif: S/PDIF PCM snd-soc-dummy-dai-0 (hw:0,0) : 0
imx-hdmi-arc: S/PDIF PCM snd-soc-dummy-dai-0 (hw:1,0) : 1
adau1361-audio: - (hw:2,0) : 2
ad24xx-a2b-bus: - (hw:3,0) : 3
imx-audio-hdmi: - (hw:4,0) : 4
USB Device 0x46d:0x81b: Audio (hw:5,0) : 5
sysdefault : 6
pulse : 7
dmix_48000 : 8
dmix_32000 : 9
dmix_16000 : 10
dmix_8000 : 11
asymed : 12
dsp0 : 13
dmix : 14
default : 15
Which input (audio) device you want? Please provide index value (in number) : 5
You selected audio device : 5
ALSA lib ../../alsa-lib-1.1.9/src/confmisc.c:1281:(snd_func_refer) Unable to find definition 'cards.imx-spdif.pcm.front.0:CARD=0
.
```

```
.
.
JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
Say Something Now!. It will continuously convert speech to text and Wait till user pressed
CTRL+C ...

_unknown_ (prediction score = 44.53)

_unknown_ (prediction score = 22.55)

stop (prediction score = 22.94)

go (prediction score = 48.45)

left (prediction score = 44.40)
yes (prediction score = 26.72)

right (prediction score = 97.46)

down (prediction score = 27.49)
stop (prediction score = 20.01)

yes (prediction score = 93.53)

no (prediction score = 50.74)

go (prediction score = 51.61)

up (prediction score = 51.55)

no (prediction score = 50.00)
go (prediction score = 26.50)
down (prediction score = 20.24)

down (prediction score = 46.60)
no (prediction score = 30.87)

no (prediction score = 20.47)
^CExiting Demo...
Exiting Demo...
```

Here as seen in the above output, lots of error and warning is observed regarding ALSA Lib. This is because ALSA lib is trying to configure, capture only device to playback and vice versa. These errors won't affect the actual behavior so it can be ignored.
Select Audio device from the list of available audio devices as shown below:

```
imx-spdif: S/PDIF PCM snd-soc-dummy-dai-0 (hw:0,0) : 0
imx-hdmi-arc: S/PDIF PCM snd-soc-dummy-dai-0 (hw:1,0) : 1
adau1361-audio: - (hw:2,0) : 2
ad24xx-a2b-bus: - (hw:3,0) : 3
imx-audio-hdmi: - (hw:4,0) : 4
USB Device 0x46d:0x81b: Audio (hw:5,0) : 5
```

which is "USB Device 0x46d:0x81b: Audio (hw:5,0)" USB Cam MIC. Speak loud and clear near to board to capture audio perfectly and to achieve higher confidence. If confidence is lower than ignore that spoken word. Top three predictions for spoken words are displayed, if their confidence is at least

greater than 20%. Some keyword like "**go**" and "**left**" have lower confidence due to their echo is similar to other spoken words like "**no**".
There is no mechanism for re-training but in reality for speech recognition there has to be a continuous retrain of the model with spoken words to increase accuracy and confidence. This can be observed from Google and other company like Amazon, Apple who has better voice recognition system. (They have lots of spoken data and increasing it by each day.)
If there is good use of USB MIC instead of other MIC sources available, there will be better performance. As in external USB MIC, there is a good feature like noise and echo cancellation. Due to that audio data input is much accurate and valid. To test demo with USB MIC then there is a need to provide appropriate hardware index entry.

In second demo, **Google Speech to Text** API is used to validate the MIC. To work on this demo internet connection is needed. This model has more accuracy than the custom model. This Demo listens for 3 to 5 seconds and converts those audio data to text. Here keywords can be defined to perform basic operations. For example, if browser support is provided in the firmware release then, keywords can be identified to perform action on browser like "**Anil open wikipedia**".
In this demo ANIL, WIKIPEDIA, YOUTUBE, GOOGLE are keywords.
The latest firmware does not support browser (The only reason - browser is resource consuming) but if that is supported then one can perform some task as mentioned below:

Say Something Now!. It will continuously convert speech to text and Wait till 'ENTER' pressed...
Speech Recognition thinks you said ***: Anil open Google***
Main Keyword detected...
Opening Google in browser...
Speech Recognition thinks you said : ***Anil open Wikipedia***
Main Keyword detected...
Opening WikiPedia in browser...
Speech Recognition thinks you said ***: Anil***
Main Keyword detected...
Speech Recognition thinks you said : ***Anil open YouTube***
Main Keyword detected...
Opening YouTube in browser...
Speech Recognition thinks you said : ***Anil open YouTube search latest song***
Main Keyword detected...
Opening YouTube in browser...
Speech Recognition thinks you said : ***Anil open Wikipedia search today's history***
Main Keyword detected...
Opening WikiPedia in browser...

3.5 Face Recognition using Tensorflow Lite demo

This application demo uses Haar Feature-based Cascade Classifiers for real time face detection. The pre-trained Haar Feature-based Cascade Classifiers for face, named as XML. TensorFlow Lite implementation for MobileFaceNets.
The MobileFaceNets is re-trained with a smaller batch size and input size to get a higher performance on a host PC. The trained model is loaded as a source file in this demo.

Steps to run Tf_based Face Recognition Demo:

Run "sh ./ARROW_DEMOS/run_ml_demos.sh" **script and select appropriate option** ,then select Node Entry e.g **/0/1/2/3/4**.

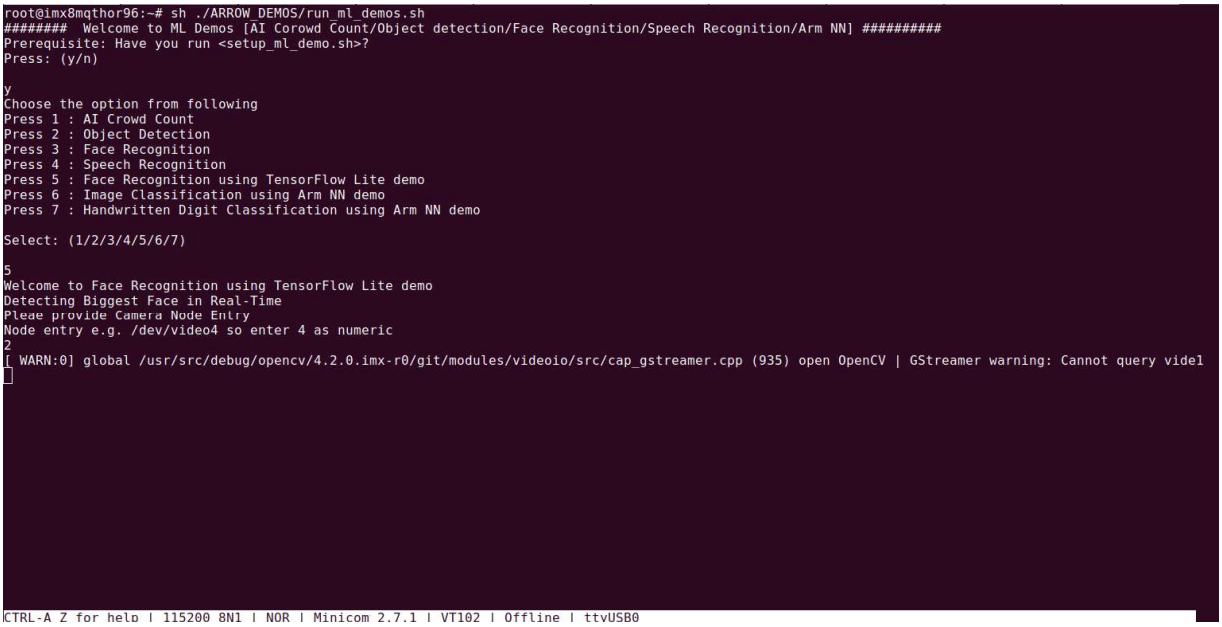

Figure 14: Tensorflow based Face Recognition demo run screen

```
# sh ./ARROW_DEMOS/run_ml_demos.sh
######## Welcome to ML Demos [AI Corowd Count/Object detection/Face Recognition/Speech
Recognition/Arm NN] ##########
Prerequisite: Have you run <setup_ml_demo.sh>?
Press: (y/n)
y
Choose the option from following
Press 1 : AI Crowd Count
Press 2 : Object Detection
Press 3 : Face Recognition
Press 4 : Speech Recognition
Press 5 : Face Recognition using TensorFlow Lite demo
Press 6 : Image Classification using Arm NN demo
Press 7 : Handwritten Digit Classification using Arm NN demo
Select: (1/2/3/4/5/6/7)
```

```
5
Welcome to Face Recognition using TensorFlow Lite demo
Detecting Biggest Face in Real-Time
Pleae provide Camera Node Entry
Node entry e.g. /dev/video4 so enter 4 as numeric
2
```

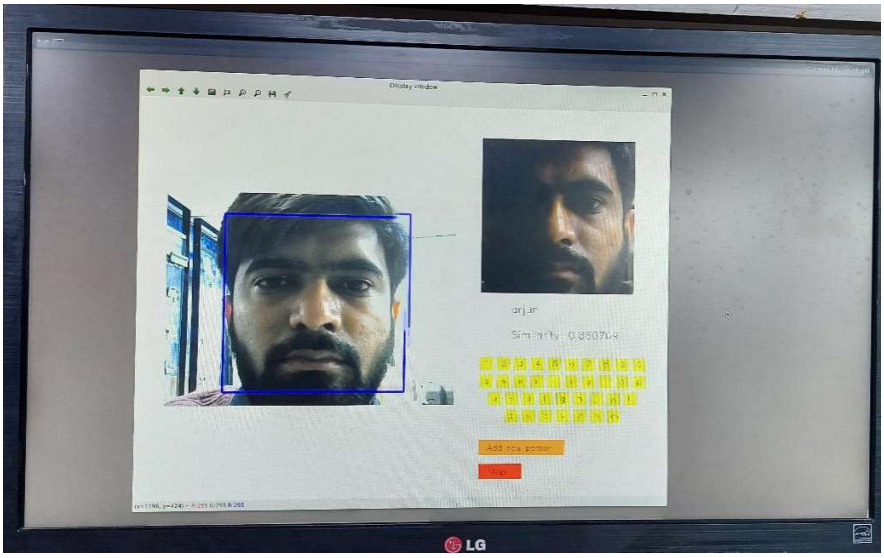

Figure 15: Tensorflow based Face Recognition demo output screen

When the demo is running, it will detect the biggest face at real time. Once the face is detected and Blue colored rectangular box is shown, one can click keyboard on the right of GUI to input the new person's name. Then, click 'Add new person' to "Enter image label:" into the console, add the face to data set.

In brief,

1. Detect face.
2. Click 'Add new person' with mouse.
3. Input new person's name on the Console to "Enter image label:".
4. On the side screen snippet, person's live image feed captured is shown with amount of similarity probability(0-1)percent.

Note: Once new faces are added, it will create a folder named 'data' in current directory. If you want to remove the new face from the data set, just delete it in 'data'.

3.6 Image Classification using Arm NN demo

Let's take a look at an image classification example and how it can take advantage of NNAPI. We will need to do the following to use a TensorFlow Lite model with NNAPI.

1. Load the labels for the TensorFlow Lite Model
2. Create a TensorFlow interpreter options object and add an NNAPI delegate to it
3. Create the TensorFlow interpreter object
4. Pre-process image to be recognized
5. Run the interpreter against the image
6. Examine the results

A delegate object can be used to accelerate a model on a device's GPU, Digital Signal Processor (DSP), or Neural Processing Unit (NPU). The NNAPI delegate will determine which of these is the best for running a specific model. If none of these are available, then NNAPI will fall back to evaluating the model on the CPU.

While this example has been focused on image recognition, potential applications are not limited to recognition. It could be applied to a range of applications, including ML Super Resolution for providing a sharp image from a lower resolution image or for pose recognition for detecting where and how someone is positioned within an image

Steps to run File_Based Demo:

Run "sh ./ARROW_DEMOS/run_ml_demos.sh" script and select option 6 ,then select option 1.

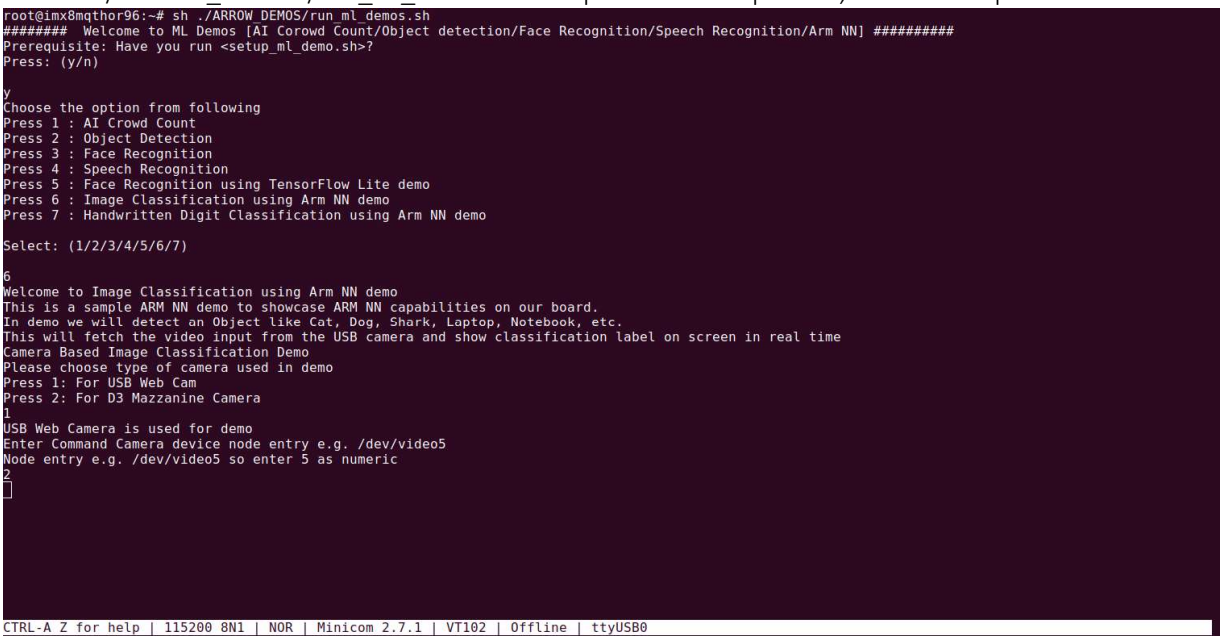

Figure 16: Arm NN Image Classification run screen

```
# sh ./ARROW_DEMOS/run_ml_demos.sh
```

######## Welcome to ML Demos [AI Corowd Count/Object detection/Face Recognition/Speech Recognition/Arm NN] ##########
Prerequisite: Have you run <setup_ml_demo.sh>?
Press: (y/n)
y
Choose the option from following
Press 1 : AI Crowd Count
Press 2 : Object Detection
Press 3 : Face Recognition
Press 4 : Speech Recognition
Press 5 : Face Recognition using TensorFlow Lite demo
Press 6 : Image Classification using Arm NN demo
Press 7 : Handwritten Digit Classification using Arm NN demo
Select: (1/2/3/4/5/6/7)
6
Welcome to Image Classification using Arm NN demo
This is a sample ARM NN demo to showcase ARM NN capabilities on our board.
In demo we will detect an Object like Cat, Dog, Shark, Laptop, Notebook, etc.
This will fetch the video input from the USB camera and show classification label on screen in real time
Camera Based Image Classification Demo
Please choose type of camera used in demo
Press 1: For USB Web Cam
Press 2: For D3 Mazzanine Camera
1
USB Web Camera is used for demo
Enter Command Camera device node entry e.g. /dev/video5
Node entry e.g. /dev/video5 so enter 5 as numeric
2

Figure 17: Arm NN Image Classification output screen

This is a demo output shows how much accuracy in detecting an Object, output display on the screen with “class = computer | keyboard”.

3.7 Handwritten Digit Classification using Arm NN

The MNIST is a large database of handwritten digits commonly used for training various image processing systems. This section provides a TensorFlow models for Handwritten Digit Recognition. The data set used for these applications is from Yann Lecun. This is an MNIST data set sample:

Steps to run File_Based Demo:

Run “sh ./ARROW_DEMOS/run_ml_demos.sh” script and select option 7.

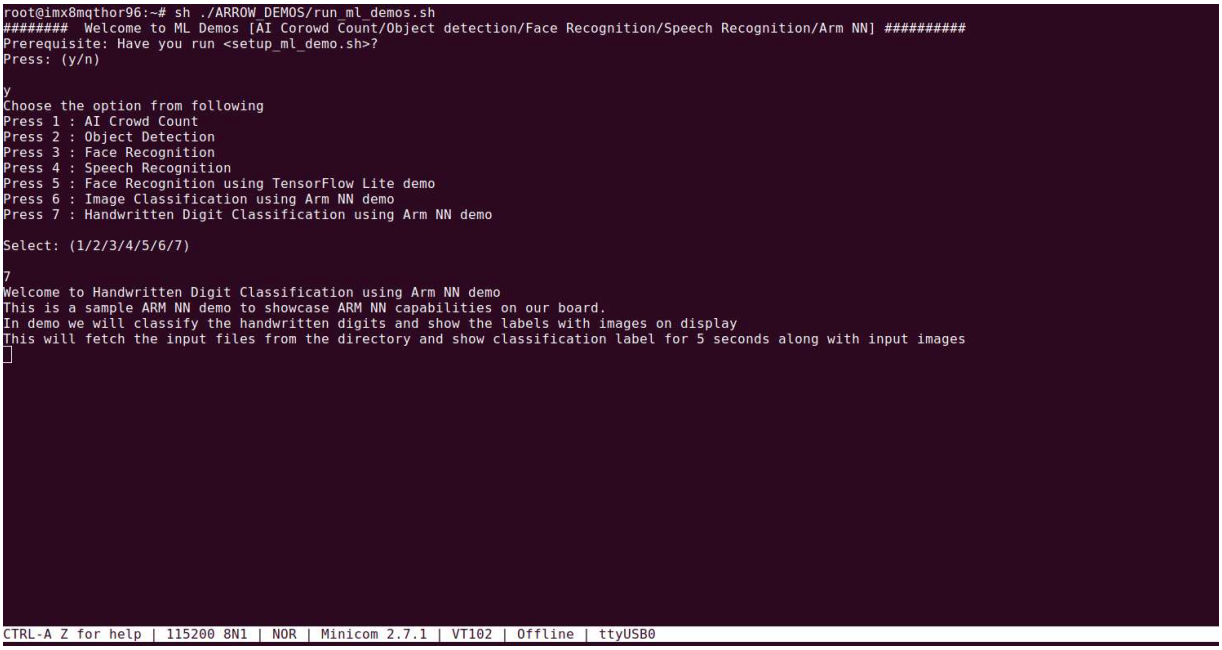

Figure 18: Arm NN Handwritten Digit Classification run screen

```
# sh ./ARROW_DEMOS/run_ml_demos.sh
######## Welcome to ML Demos [AI Corowd Count/Object detection/Face Recognition/Speech
Recognition/Arm NN] ##########
Prerequisite: Have you run <setup_ml_demo.sh>?
Press: (y/n)
y
Choose the option from following
Press 1 : AI Crowd Count
Press 2 : Object Detection
Press 3 : Face Recognition
Press 4 : Speech Recognition
Press 5 : Face Recognition using TensorFlow Lite demo
```

```
Press 6 : Image Classification using Arm NN demo
Press 7 : Handwritten Digit Classification using Arm NN demo
Select: (1/2/3/4/5/6/7)
7
Welcome to Handwritten Digit Classification using Arm NN demo
This is a sample ARM NN demo to showcase ARM NN capabilities on our board.
In demo we will classify the handwritten digits and show the labels with images on display
This will fetch the input files from the directory and show classification label for 5 seconds along with input images
Working with ARMNN 20200200
Graph Id: 0
Input Names: ('flatten_input',)
tensor id: 2560,
tensor info: TensorInfo{DataType: 1, IsQuantized: 0, QuantizationScale: 0.000000, QuantizationOffset: 0, NumDimensions: 3, NumElements: 784}
Loaded network, id=0
Actual Image : ./images/7.jpg
Workload tensor 0 shape: TensorShape{Shape(1, 10), NumDimensions: 2, NumElements: 10}
=#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#=
[RESULT] Actual Character : 7 | Predicted Value : 7
=#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#=
Actual Image : ./images/2.jpg
Workload tensor 0 shape: TensorShape{Shape(1, 10), NumDimensions: 2, NumElements: 10}
=#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#=
[RESULT] Actual Character : 2 | Predicted Value : 2
=#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#=
Actual Image : ./images/3.jpg
Workload tensor 0 shape: TensorShape{Shape(1, 10), NumDimensions: 2, NumElements: 10}
=#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#=
[RESULT] Actual Character : 3 | Predicted Value : 3
=#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#=
Actual Image : ./images/9.jpg
Workload tensor 0 shape: TensorShape{Shape(1, 10), NumDimensions: 2, NumElements: 10}
=#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#=
[RESULT] Actual Character : 9 | Predicted Value : 9
=#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#=
Exiting Demo...
```

```
Press 7 : Handwritten Digit Classification using Arm NN demo

Select: (1/2/3/4/5/6/7)

7
Welcome to Handwritten Digit Classification using Arm NN demo
This is a sample ARM NN demo to showcase ARM NN capabilities on our board.
In demo we will classify the handwritten digits and show the labels with images on display
This will fetch the input files from the directory and show classification label for 5 seconds along with input images
Working with ARMNN 20200200
Graph Id: 0
Input Names:  ('flatten_input',)

tensor id: 2560,
tensor info: TensorInfo{DataType: 1, IsQuantized: 0, QuantizationScale: 0.000000, QuantizationOffset: 0, NumDimensions: 3, NumElements: 784}

Loaded network, id=0
Actual Image : ./images/7.jpg
Workload tensor 0 shape: TensorShape{Shape(1, 10), NumDimensions: 2, NumElements: 10}
=#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#=
[RESULT] Actual Character : 7 | Predicted Value : 7
=#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#=
Actual Image : ./images/2.jpg
Workload tensor 0 shape: TensorShape{Shape(1, 10), NumDimensions: 2, NumElements: 10}
=#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#=
[RESULT] Actual Character : 2 | Predicted Value : 2
=#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#=
Actual Image : ./images/3.jpg
Workload tensor 0 shape: TensorShape{Shape(1, 10), NumDimensions: 2, NumElements: 10}
=#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#=
[RESULT] Actual Character : 3 | Predicted Value : 3
=#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#=
Actual Image : ./images/9.jpg
Workload tensor 0 shape: TensorShape{Shape(1, 10), NumDimensions: 2, NumElements: 10}
=#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#=
[RESULT] Actual Character : 9 | Predicted Value : 9
=#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#==#=
Exiting Demo...
root@imx8mqthor96:~#
CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.7.1 | VT102 | Offline | ttyUSB0
```

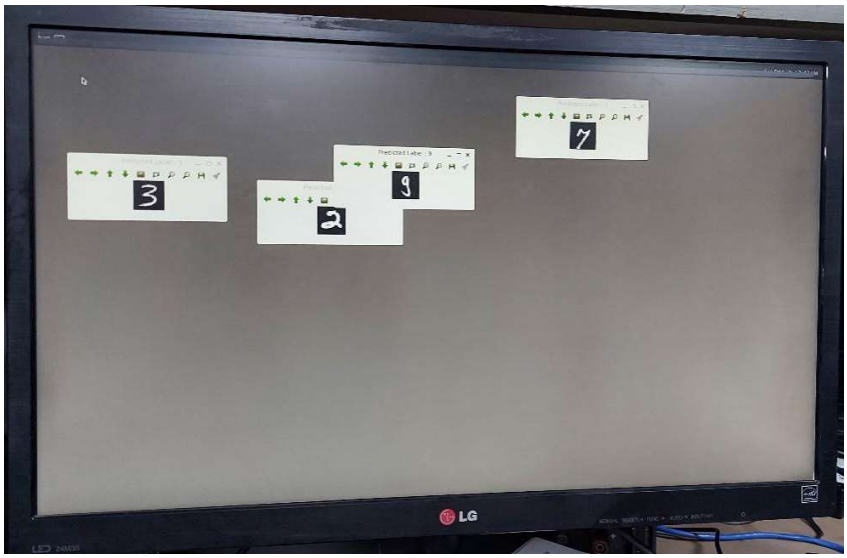

Figure 19: Arm NN Handwritten Digit Classification Output screen

In above demo, hard written number detected and output logs are seen on the serial console with prediction value like 2,3,9,7

NOTE: The argument 10 refers to the number of predictions for each test.

These tests run the inference on the input MNIST dataset images (Actual), showing the inference results (Predict) and how long it took to complete the prediction. The input images for this test are in the binary form and can be found at the t10k-images-idx3-ubyte.gz package from Yann Lecun.

4 TROUBLESHOOTING

4.1 HDMI

- Although HDMI hot plug detection feature is provided, connect HDMI before booting up the board. Because it has been observed that if HDMI is not connected before board boots up, hot plug feature does not work and even with HDMI connected there is no output on HDMI display. A reboot is needed to get HDMI working.
- In case If HDMI is not connected and demo is executed then below error will be seen.

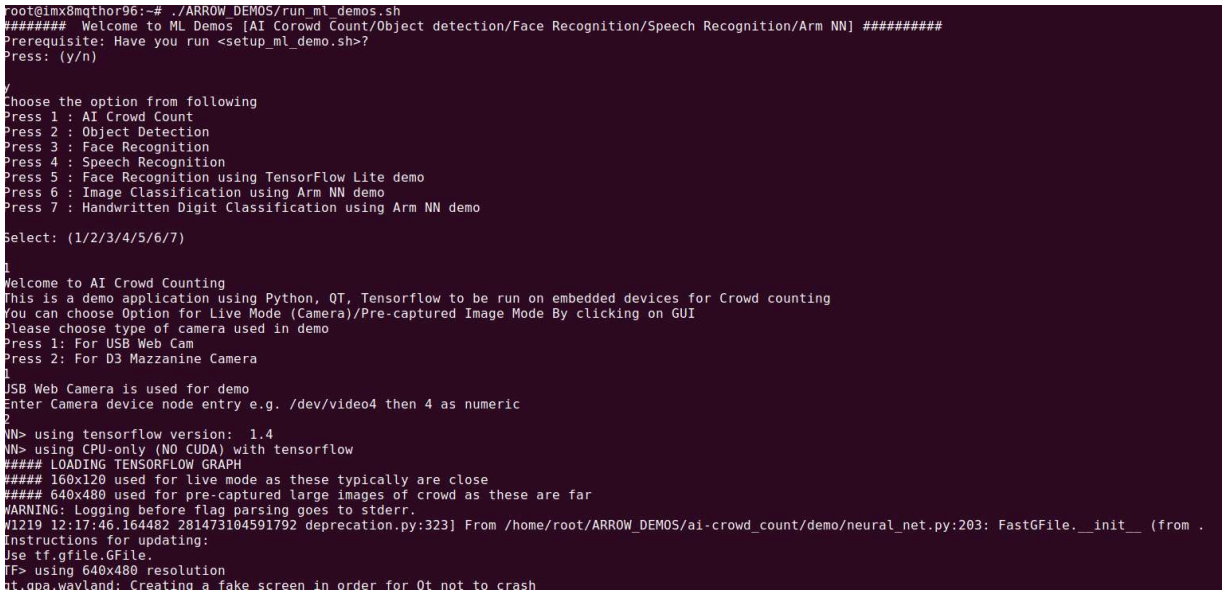

Figure 20: No HDMI Connected Error

- In case of such error, connect HDMI and reboot.

4.2 Camera

If camera is not connected and demo is started to capture frame then one might see error as shown in below attached figure.

To resolve camera error, one can do following checks:

- Check correct node entry i.e. **"/dev/video7"** is provided while running demo in case of USB web cam.
- Please check that appropriate dtb file is set in uboot environment i.e. "**fsl-imx8qxp-aiml-mipi-ov5640.dtb**" in case of D3 camera.
- Please verify D3 camera or USB camera is working fine before running Demos. We can check that by gstreamer pipeline:
  gst-launch-1.0 v4l2src device=/dev/video0 ! video/x-raw,width=1280,height=720 ! glimagesink

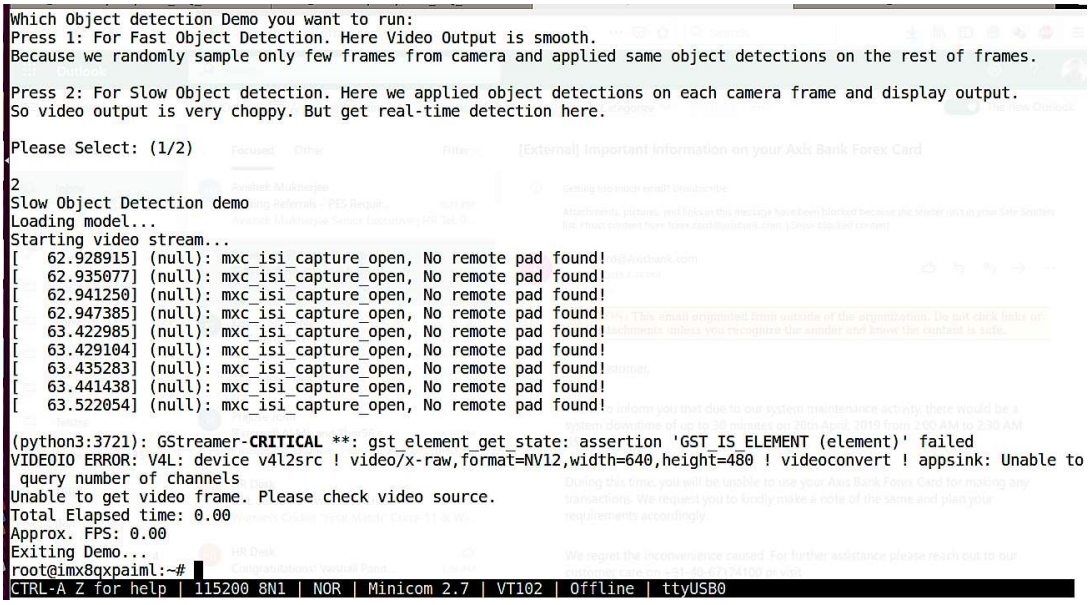

Figure 21: Camera not connected error

5 REFERENCES

1. https://github.com/ageitgey/face_recognition
2. https://github.com/davisking/dlib
3. https://cmusatyalab.github.io/openface/
4. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_objdetect/py_face_detection/py_face_detection.html#face-detection
5. https://github.com/tensorflow/models/tree/master/research/object_detection
6. https://www.tensorflow.org/tutorials/sequences/audio_recognition
7. https://github.com/tensorflow/models
8. https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md
9. https://github.com/jrosebr1/imutils
10. https://www.pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/
11. https://github.com/chuanqi305/MobileNet-SSD
12. https://github.com/Uberi/speech_recognition#readme
13. https://medium.com/@ageitgey/quick-tip-speed-up-your-python-data-processing-scripts-with-process-pools-cf275350163a
14. https://github.com/spmallick/learnopencv
15. https://github.com/ARM-software/armnn
16. https://www.pyimagesearch.com/2017/09/18/real-time-object-detection-with-deep-learning-and-opencv/
17. https://github.com/opencv/opencv/tree/master/modules/dnn
18. https://github.com/opencv/opencv/tree/master/samples/dnn
19. https://heartbeat.fritz.ai/real-time-object-detection-on-raspberry-pi-using-opencv-dnn-98827255fa60
20. https://github.com/ARM-software/ML-examples