7.TensorRT USB camera real-time image recognition tutorial

1.Preparation

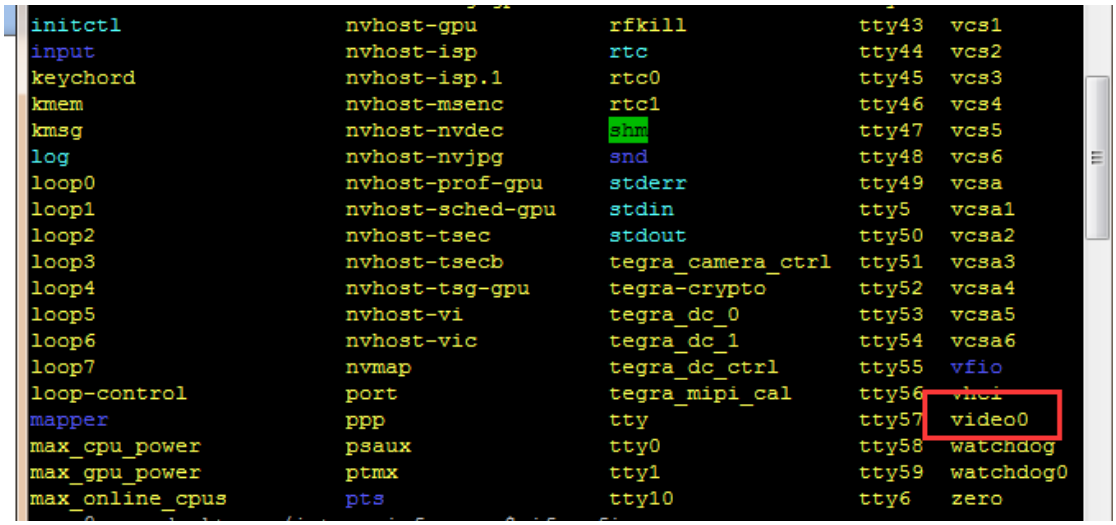
Before we start this step, we need to make sure that we have completed all the steps in Tutorials 4 and 5 and can test simple examples.

It is best to unplug the on-board camera, plug in the USB camera then open the power.

2.Check camera service

If you are in **jeston-inference** directory, you can input the following command:

ls /dev



We need to determine if there is video0 in here, it is possible to have multiple cameras. They have different numbers behind.

3.Parameter introduction

Similar to the previous imagenet-console example, the camera application is built in this **/aarch64/bin** directory. They run on a live camera stream with OpenGL rendering and accept 4 optional command line arguments:

- - - network flag sets the classification model (default is GoogleNet)
- See Download other classification models for available networks.
- - - camera flag sets the camera device to be used
- Use MIPI CSI cameras by specifying the sensor index (0 or 1 etc.)
- The V4L2 USB camera is used by specifying its /dev/video node (/dev/video0, /dev/video1, etc.).
- Default is to use MIPI CSI sensor 0 (--camera = 0)
- - - width and - - height flags set the camera resolution (default is 1280x720)
- Resolution should be set to a format supported by the camera.
- Query the available formats using:

  sudo apt-get install v4l-utils

  v4l2-ctl --list-formats-ext

You can combine these flags as needed, and there are other command line parameters available for loading custom models. Launch the application with the --help flag for more information, or see the Examples readme.

Here are some typical scenarios for start programs:

C ++

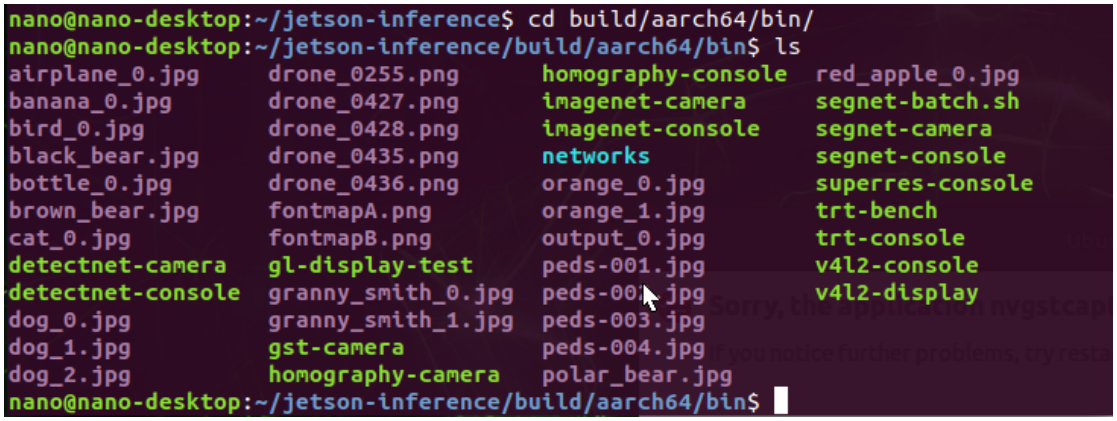
```
$ ./imagenet-camera      # Use GoogleNet, default MIPI CSI camera (1280 × 720)
$ ./imagenet-camera - - network = RESNET-18   # Use RESNET-18, default MIPI CSI camera (1280 × 720)
$ ./imagenet-camera - - camera = /dev /video0   # Use GoogleNet, V4L2 camera / dev / video0 (1280x720)
$ ./imagenet-camera - - width = 640 - - height = 480   # Use GoogleNet, default is MIPI CSI camera (640x480)
```

Python

```
$ ./imagenet-camera.py   # Using GoogleNet, the default MIPI CSI camera (1280x720)
$ ./imagenet-camera.py - - network = RESNET-18   # Use RESNET-18, the default MIPI CSI camera (1280x720)
$ ./ imagenet-camera.py - - camera = /dev /video0    # Use GoogleNet, V4L2 camera /dev/video0 (1280x720)
$ ./imagenet-camera.py - - width = 640 - - height = 480   # Use GoogleNet, default is MIPI CSI camera (640x480)
```

4.Execute image recognition command

At this point we are better able to execute by the remote desktop, otherwise you may not see the camera interface, or connect by VNC remote desktop.

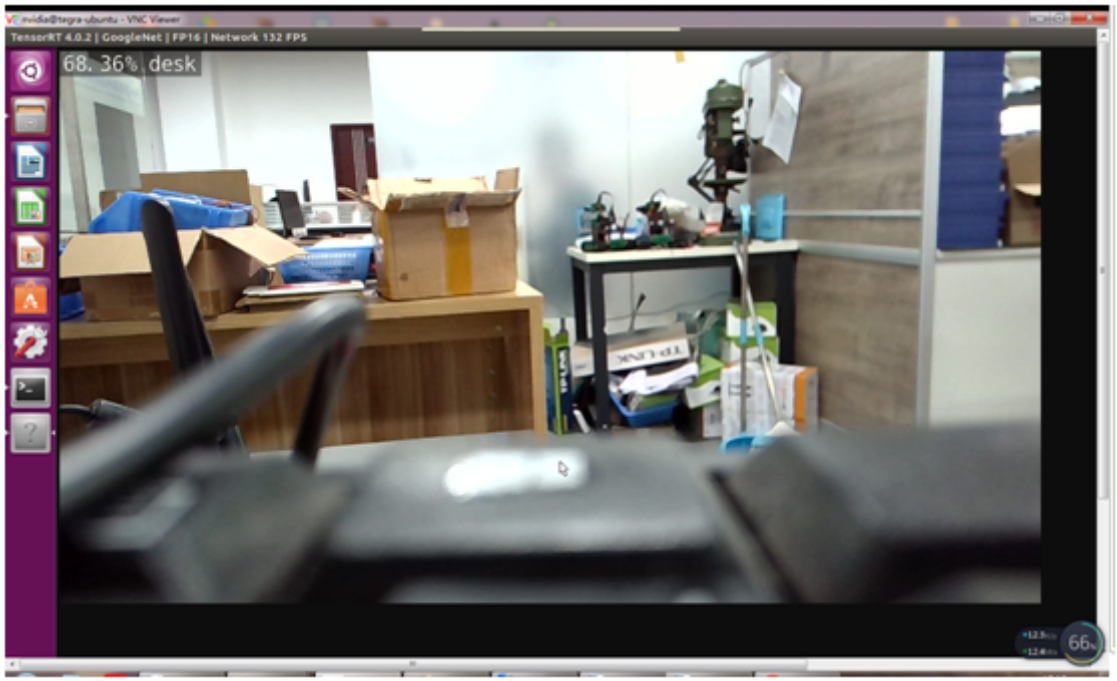


Enter the bin directory:

The live image recognition demo is located in **/aarch64/bin** and call **imagenet-camera**. It runs on the live camera stream and loads googlenet or alexnet using TensorRT based on user parameters.

```
$ ./imagenet-camera - - network=googlenet - - camera=/dev/video1
# Run with googlenet USB camera
$ ./imagenet-camera - - network=axlenet - - camera=/dev/video1
# Run with alexnet
```

Frames per second (FPS), the classification object name from the video and the confidence of the classification object are printed to the OpenGL window title bar. By default, the application can recognize up to 1000 different types of objects, name mappings for 1000 types of objects, which can be found under repo:
data/networks/ilsvrc12_synset_words.txt



When an object is recognized, the English name of the object is displayed on the interface, and the percentage is the matching percentage.