6.TensorRT on-board 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.

2.Enter the executable directory

If you are in the **jetson-inference** directory, please execute the following directory.

cd build/aarch64/bin/

```
nano@nano-desktop:~/jetson-inference$ cd build/aarch64/bin/
nano@nano-desktop:~/jetson-inference/build/aarch64/bin$ ls
airplane_0.jpg      drone_0255.png      homography-console  red_apple_0.jpg
banana_0.jpg        drone_0427.png      imagenet-camera     segnet-batch.sh
bird_0.jpg          drone_0428.png      imagenet-console    segnet-camera
black_bear.jpg      drone_0435.png      networks            segnet-console
bottle_0.jpg        drone_0436.png      orange_0.jpg        superres-console
brown_bear.jpg      fontmapA.png        orange_1.jpg        trt-bench
cat_0.jpg           fontmapB.png        output_0.jpg        trt-console
detectnet-camera    gl-display-test     peds-001.jpg        v4l2-console
detectnet-console   granny_smith_0.jpg  peds-002.jpg        v4l2-display
dog_0.jpg           granny_smith_1.jpg  peds-003.jpg
dog_1.jpg           gst-camera          peds-004.jpg
dog_2.jpg           homography-camera   polar_bear.jpg
nano@nano-desktop:~/jetson-inference/build/aarch64/bin$
```

3.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.

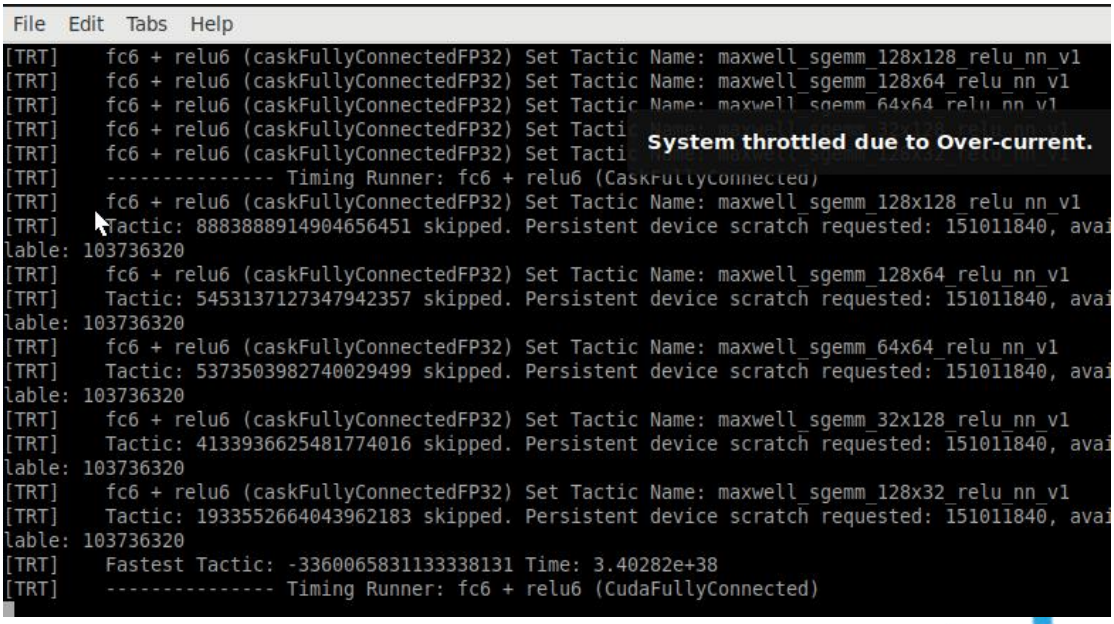
Enter the bin directory:

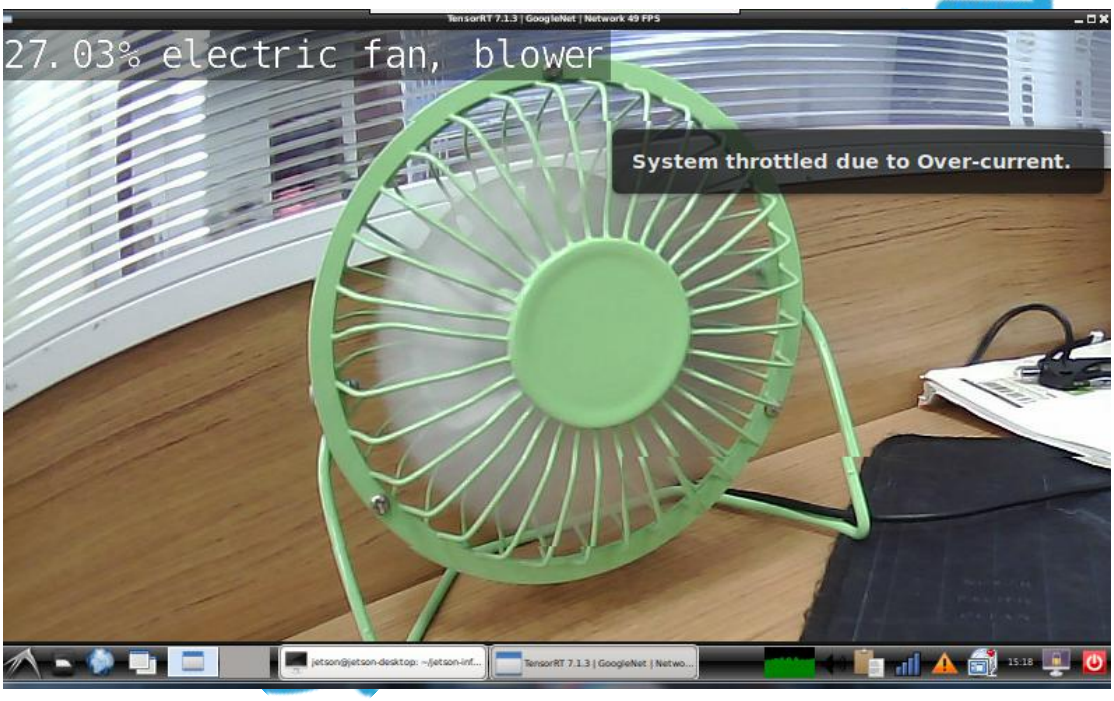
The live image recognition demo is located in **/aarch64/bin** and call **imagenet-camera**. It runs on the live camera stream and loads googlenet or alexnet using TensorRT based on user parameters.

```
$ ./imagenet-camera googlenet          # Run with googlenet
$ ./imagenet-camera alexnet            # 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.

！Note:

After the camera mount is installed, if the video picture is upside down, you can use the following methods to set and modify:

$ cd ~/jetson-inference/utils/camera/

$ gedit gstCamera.cpp

```

- Change the folloing code:
```

```bash
#if NV_TENSORRT_MAJOR > 1 && NV_TENSORRT_MAJOR < 5 // if JetPack 3.1-3.3 (different flip-method)
const int flipMethod = 0; // Xavier (w/TRT5) camera is mounted inverted
#else
const int flipMethod = 2;
#endif
```

We need to change the above code to the code shown below:

```bash
#if NV_TENSORRT_MAJOR > 1 && NV_TENSORRT_MAJOR < 5 // if JetPack 3.1-3.3 (different flip-method)
const int flipMethod = 0; // Xavier (w/TRT5) camera is mounted inverted
#else
const int flipMethod = 0;  // 2 change 0
#endif
```

Build the Code:

```bash
$ cd ~/jetson-inference/build/
$ make
$sudo make install
```