Image Classification Inference

1. ImageNet Classification

There are many types of deep learning networks available, including recognition, detection/localization, and semantic segmentation. The first deep learning function we highlight in this tutorial is image recognition, which uses a classification network trained on a large dataset to identify scenes and objects.

The imageNet object takes an input image and outputs the probability of each class. The GoogleNet and ResNet-18 models are automatically downloaded during the build step after being trained on the ImageNet ILSVRC dataset of 1000 objects. For other classification models that can be downloaded and used, we provide example programs in C++ and Python

Using the ImageNet program on Jetson:

First, let's try to test imagenet recognition on some example images using the imagenet program. It loads one or more images, performs inference using TensorRT and the imageNet class, and then overwrites the classification results and saves the output image. The project provides example images for you to use under the images/ directory.

After building the project, make sure your terminal is in the aarch64/bin directory:

```
cd jetson-inference/build/aarch64/bin
```

Next, let's classify a sample image using the imagenet program, using either the C++ or Python variant. If you are using a Docker container, it is recommended to save the classified output images to a directory mounted at images/test. These images will then be easily viewable from the host device's jetson-inference/data/images/test directory

```
# C++
$ ./imagenet images/orange_0.jpg images/test/output_0.jpg # (default network is
googlenet)

# Python
$ ./imagenet.py images/orange_0.jpg images/test/output_0.jpg # (default network
is googlenet)
```

Note: The first time you run each model, TensorRT will take several minutes to optimize the network. This optimized network file is then cached to disk, so future runs using the model will load faster

In addition to loading a single image, you can also load a directory of additional images or a video file.

2. Download other classification models

The following pre-trained image classification models are available for use and will be automatically downloaded:

Network	CLI argument	NetworkType enum
AlexNet	`alexnet`	`ALEXNET`
GoogleNet	`googlenet`	`GOOGLENET`
GoogleNet-12	`googlenet-12`	`GOOGLENET_12`
ResNet-18	`resnet-18`	`RESNET_18`
ResNet-50	`resnet-50`	`RESNET_50`
ResNet-101	`resnet-101`	`RESNET_101`
ResNet-152	`resnet-152`	`RESNET_152`
VGG-16	`vgg-16`	`VGG-16`
VGG-19	`vgg-19`	`VGG-19`
Inception-v4	`inception-v4`	`INCEPTION_V4`

Generally, more complex networks can have higher classification accuracy and increase runtime.

Using a different classification model:

You can specify the model to load by setting the --network flag on the command line to one of the corresponding CLI parameters in the table above. By default, GoogleNet is loaded if the optional --network flag is not specified.

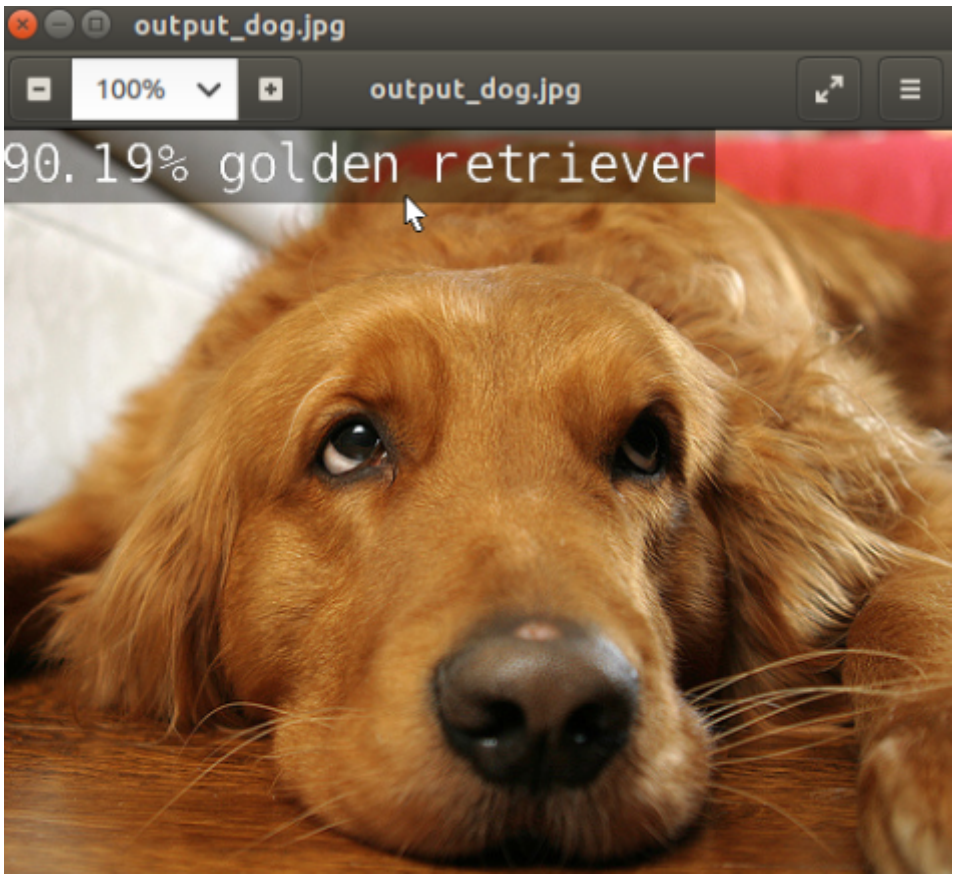
After building the project, make sure your terminal is in the aarch64/bin directory:

```
cd jetson-inference/build/aarch64/bin
```

Here are some examples using the ResNet-18 model:

```
# C++
$ ./imagenet --network=resnet-18 images/dog_2.jpg images/test/output_dog.jpg

# Python
$ ./imagenet.py --network=resnet-18 images/dog_2.jpg images/test/output_dog.jpg
```



Note: If you build your own environment, you must download the resnet-18.tar.gz model file to the network folder and unzip it before you can run the above program. You can directly enter the above program using the image we provide

3. Run the real-time camera recognition demo

The imagenet.cpp/imagenet.py sample we used before can also be used for real-time camera streams. Supported camera types include:

- MIPI CSI cameras (`csi://0`)
- V4L2 cameras (`/dev/video0`)
- RTP/RTSP streams (`rtsp://username:password@ip:port`)

Below are some typical scenarios for launching programs on camera feeds.

C++

```
$ ./imagenet csi://0 # MIPI CSI camera
$ ./imagenet /dev/video0 # V4L2 camera
$ ./imagenet /dev/video0 output.mp4 # save to video file
```

python

```
$ ./imagenet.py csi://0 # MIPI CSI camera
$ ./imagenet.py /dev/video0 # V4L2 camera
$ ./imagenet.py /dev/video0 output.mp4 # save to video file
```

The OpenGL window shows the live camera stream, the classified object name, the confidence of the classified object, and the frame rate of the network. On Jetson Nano, the frame rate of GoogleNet and ResNet-18 should be up to about 75 FPS (faster than other Jetson).

The application can recognize up to 1000 different types of objects because the classification model is trained on the ILSVRC ImageNet dataset, which contains 1000 classes of objects. The name mappings for the 1000 types of objects can be found in the repo under data/networks/ilsvrc12_synset_words.txt

This concludes this section of the Hello AI World tutorial on image classification. Next, we will start using the object detection network, which provides us with bounding box coordinates for multiple objects per frame.