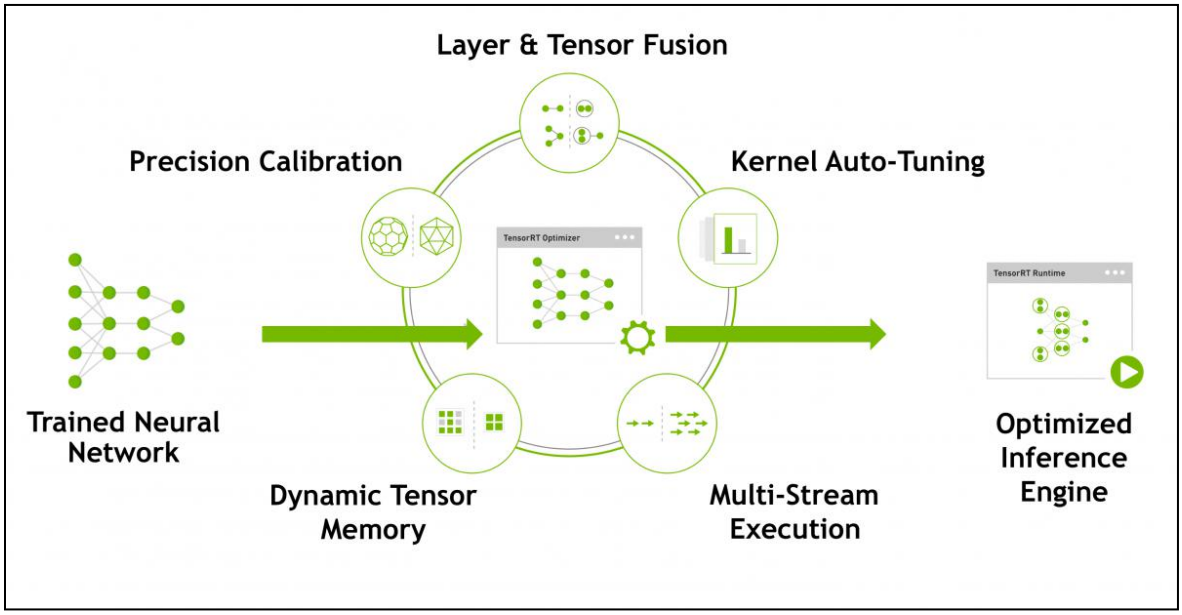
5.TensorRT Environmental construction（jetson-inference）



1. Preparation:

1) Input following command to install git and cmake.

```
sudo apt-get install git cmake
```

Then, clone the jetson-inference library from git

```
git clone https://github.com/dusty-nv/jetson-inference
cd jetson-inference
```

```
git submodule update --init
```

2) Configure cmake.

If you can surf the Internet scientifically, many models will be automatically downloaded.

```
mkdir build      #Create build folder
cd build         #Enter build
cmake ../        #Run cmake, it will automatically execute CMakePrebuild.sh in the upper-level directory
```

Then, we need to extract the .rar file:

for tar in *.tar.gz; do tar xvf $tar; done

```
nano@nano-desktop:~/jetson-inference/data/networks$ ls
alexnet_noprob.prototxt                 FCN-Alexnet-Cityscapes-HD.tar.gz
alexnet.prototxt                        FCN-Alexnet-Cityscapes-SD
bvlc_alexnet.caffemodel                 FCN-Alexnet-Cityscapes-SD.tar.gz
bvlc_googlenet.caffemodel               FCN-Alexnet-Pascal-VOC
Deep-Homography-COCO                    FCN-Alexnet-Pascal-VOC.tar.gz
Deep-Homography-COCO.tar.gz             FCN-Alexnet-SYNTHIA-CVPR16
DetectNet-COCO-Airplane                 FCN-Alexnet-SYNTHIA-CVPR16.tar.gz
DetectNet-COCO-Airplane.tar.gz          FCN-Alexnet-SYNTHIA-Summer-HD
DetectNet-COCO-Bottle                   FCN-Alexnet-SYNTHIA-Summer-HD.tar.gz
DetectNet-COCO-Bottle.tar.gz            FCN-Alexnet-SYNTHIA-Summer-SD
DetectNet-COCO-Chair                    FCN-Alexnet-SYNTHIA-Summer-SD.tar.gz
DetectNet-COCO-Chair.tar.gz             GoogleNet-ILSVRC12-subset
DetectNet-COCO-Dog                      GoogleNet-ILSVRC12-subset.tar.gz
DetectNet-COCO-Dog.tar.gz               googlenet_noprob.prototxt
detectnet.prototxt                      googlenet.prototxt
facenet-120                             ilsvrc12_synset_words.txt
facenet-120.tar.gz                      multiped-500
FCN-Alexnet-Aerial-FPV-4ch-720p         multiped-500.tar.gz
FCN-Alexnet-Aerial-FPV-4ch-720p.tar.gz  ped-100
FCN-Alexnet-Aerial-FPV-720p             ped-100.tar.gz
FCN-Alexnet-Aerial-FPV-720p.tar.gz      Super-Resolution-BSD500
FCN-Alexnet-Cityscapes-HD               Super-Resolution-BSD500.tar.gz
nano@nano-desktop:~/jetson-inference/data/networks$
```

Note:

For extracting multiple .gz files, we can use this command:

for gz in *.gz; do gunzip $gz; done

For extracting multiple tar.gz files, we can use this command:

for tar in *.tar.gz; do tar xvf $tar; done

After cmake is completed, we need to compile

cd jetson-inference/build

make (or make -j4) Note: (under the build directory)

sudo make install Note: (under the build directory)

If the compilation is successful, the following folder structure will be generated

|-build

\aarch64 (64-bit)

\bin where the sample binaries are built to

\include where the headers reside

\lib where the libraries are build to

\armhf (32-bit)

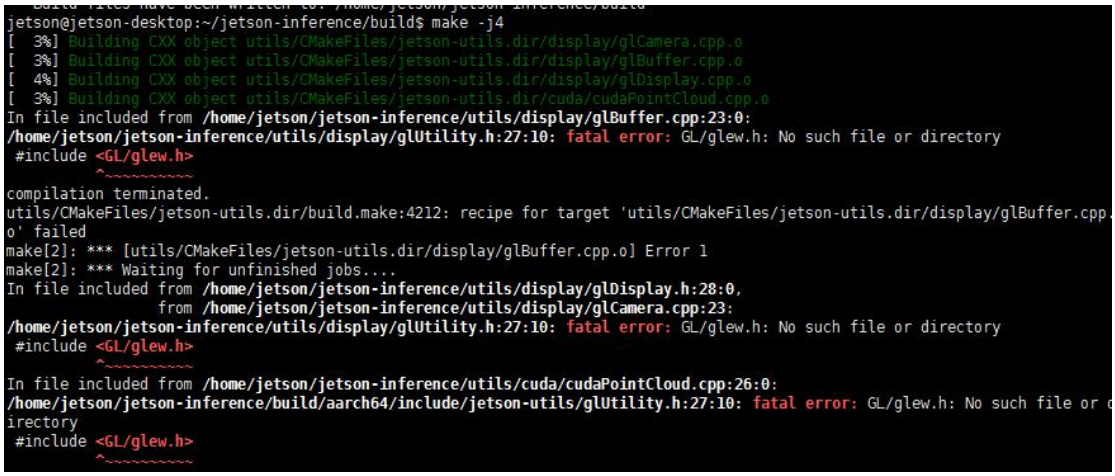
\bin where the sample binaries are built to

\include where the headers reside

\lib where the libraries are build to

If the system prompts an error as shown below during compilation.

We can solve this error by installing **opengl**



Install opengl by following command:
sudo apt-get install build-essential
sudo apt-get install build-essential libgl1-mesa-dev
sudo apt-get install libglew-dev libsdl2-dev libsdl2-image-dev libglm-dev libfreetype6-dev
sudo apt-get install libglfw3-dev libglfw3

Install OpenGL Library:
sudo apt-get install libgl1-mesa-dev

Install OpenGL Utilities:
sudo apt-get install libglu1-mesa-dev

Install OpenGL Utility Toolkit:
sudo apt-get install libglut-dev

Note: At this step, if you can following shell prompts:
Reading package lists… Done
Building dependency tree
Reading state information… Done
E: Unable to locate package libglut-dev

You can solve this problem by following command:
sudo apt-get install freeglut3-dev

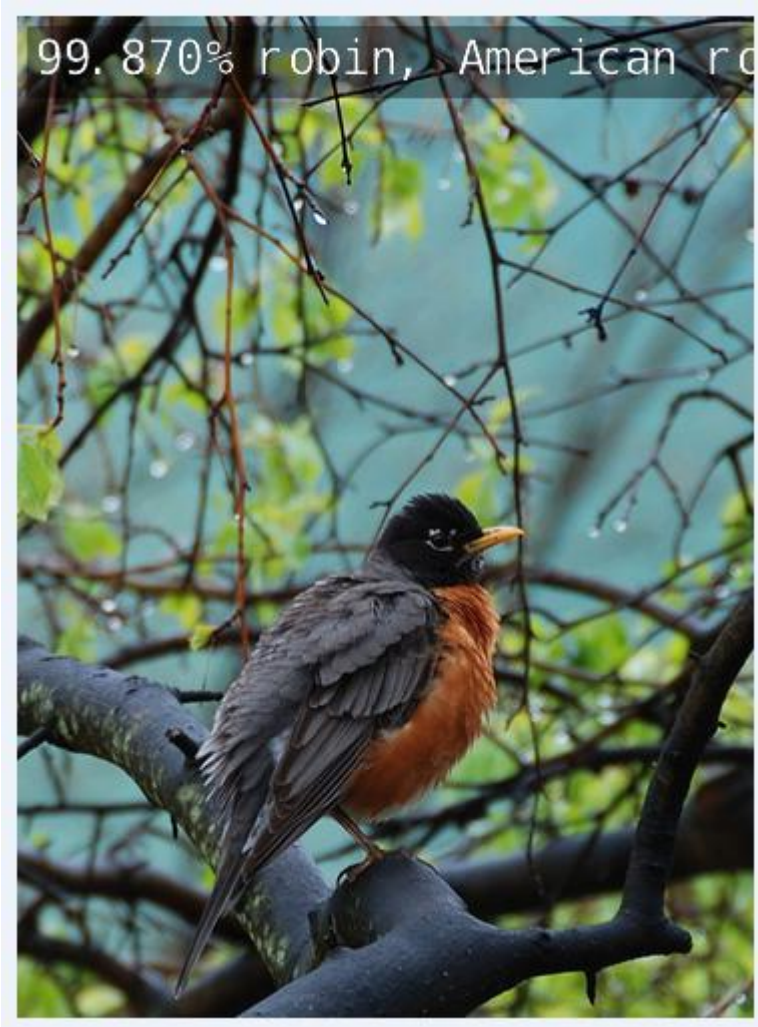
3) Test
We need to input the following command:
cd jetson-inference/build/aarch64/bin
./imagenet-console orange_0.jpg output_0.jpg

After waiting patiently, we will see the interface shown below.

```
[TRT]   layer inception_4e/pool_proj + inception_4e/relu_pool_proj - 0.363490 ms
[TRT]   layer inception_4e/1x1 copy - 0.026979 ms
[TRT]   layer pool4/3x3_s2 - 0.151042 ms
[TRT]   layer inception_5a/1x1 + inception_5a/relu_1x1 || inception_5a/3x3_reduce
 + inception_5a/relu_3x3_reduce || inception_5a/5x5_reduce + inception_5a/relu_5
x5_reduce - 0.522812 ms
[TRT]   layer inception_5a/3x3 + inception_5a/relu_3x3 - 0.550833 ms
[TRT]   layer inception_5a/5x5 + inception_5a/relu_5x5 - 0.207032 ms
[TRT]   layer inception_5a/pool - 0.085729 ms
[TRT]   layer inception_5a/pool_proj + inception_5a/relu_pool_proj - 0.207187 ms
[TRT]   layer inception_5a/1x1 copy - 0.011771 ms
[TRT]   layer inception_5b/1x1 + inception_5b/relu_1x1 || inception_5b/3x3_reduce
 + inception_5b/relu_3x3_reduce || inception_5b/5x5_reduce + inception_5b/relu_5
x5_reduce - 0.894167 ms
[TRT]   layer inception_5b/3x3 + inception_5b/relu_3x3 - 1.069375 ms
[TRT]   layer inception_5b/5x5 + inception_5b/relu_5x5 - 0.271042 ms
[TRT]   layer inception_5b/pool - 0.086979 ms
[TRT]   layer inception_5b/pool_proj + inception_5b/relu_pool_proj - 0.208177 ms
[TRT]   layer inception_5b/1x1 copy - 0.016146 ms
[TRT]   layer pool5/7x7_s1 - 0.058073 ms
[TRT]   layer loss3/classifier input reformatter 0 - 0.008854 ms
[TRT]   layer loss3/classifier - 0.283698 ms
[TRT]   layer prob - 0.023229 ms
[TRT]   layer prob output reformatter 0 - 0.013125 ms
[TRT]   layer network time - 132.269409 ms
class 0950 - 0.979004   (orange)
class 0951 - 0.020645   (lemon)
imagenet-console:  'orange_0.jpg' -> 97.90039% class #950 (orange)
loaded image  fontmapA.png  (256 x 512)  2097152 bytes
[cuda]  cudaAllocMapped 2097152 bytes, CPU 0x1048a0000 GPU 0x1048a0000
[cuda]  cudaAllocMapped 8192 bytes, CPU 0x100f62000 GPU 0x100f62000
imagenet-console:  attempting to save output image to 'output_0.jpg'
imagenet-console:  completed saving 'output_0.jpg'

shutting down...
nano@nano-desktop:~/jetson-inference/build/aarch64/bin$
```

We need to find the corresponding directory to view **output_0.jpg**, the recognition result will be displayed at the top of the picture, as shown below.



For more detail, please see the official documentation:
Official Demo:
https://developer.nvidia.com/embedded/twodaystoademo
Official TensorRT tutorial:
https://docs.nvidia.com/deeplearning/sdk/tensorrt-sample-support-guide/index.html