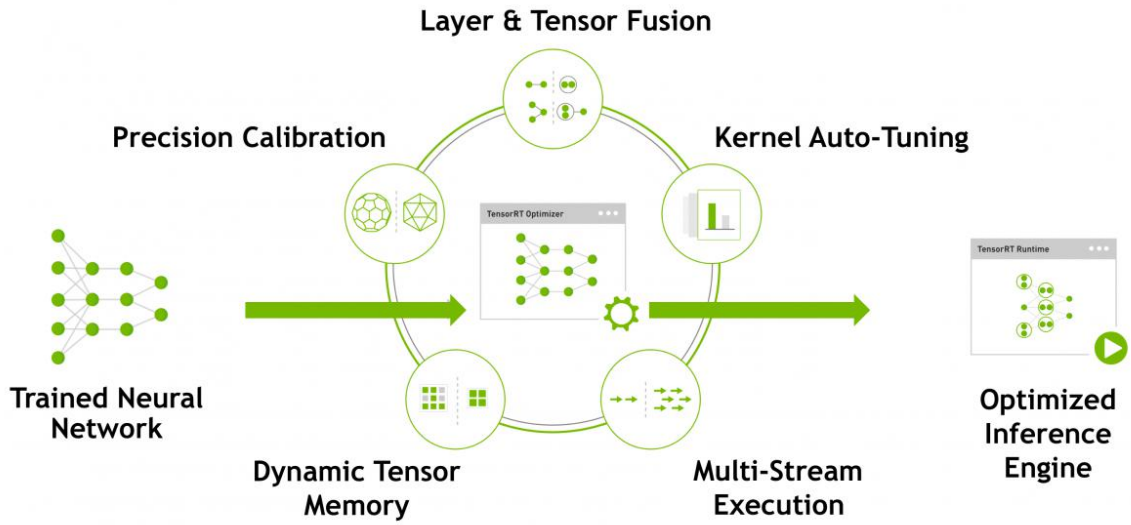
5、Jetson Xavier NX 之 TensorRT 环境搭建(jetson-inference)

NVIDIA TensorRT™是一个高性能深度学习推理平台。它包括深度学习推理优化器和运行时，可为深度学习推理应用程序提供低延迟和高吞吐量。在推理期间，基于 TensorRT 的应用程序比仅 CPU 平台的执行速度快 40 倍。使用 TensorRT，您可以优化在所有主要框架中培训的神经网络模型，以高精度校准低精度，最后部署到超大规模数据中心，嵌入式或汽车产品平台。

TensorRT 构建于 NVIDIA 的并行编程模型 CUDA 之上，使您能够利用 CUDA-X AI 中的库，开发工具和技术，为人工智能，自动机器，高性能计算和图形优化所有深度学习框架的推理。

TensorRT 为深度学习推理应用的生产部署提供 INT8 和 FP16 优化，例如视频流，语音识别，推荐和自然语言处理。降低精度推断可显着减少应用程序延迟，这是许多实时服务，自动和嵌入式应用程序的要求。



您可以将训练有素的模型从每个深度学习框架导入 TensorRT。应用优化后，TensorRT 选择特定于平台的内核，以最大限度地提高数据中心，Jetson 嵌入式平台和 NVIDIA DRIVE 自动驾驶平台中 Tesla GPU 的性能。

为了在数据中心生产中使用 AI 模型，TensorRT 推理服务器是一种容器化微服务，可最大化 GPU 利用率，并在节点上同时运行来自不同框架的多个模型。它利用 Docker 和 Kubernetes 无缝集成到 DevOps 架构中。

使用 TensorRT，开发人员可以专注于创建新颖的 AI 驱动的应用程序，而不是用于推理部署的性能调整。

（一）准备工作：

这个例子需要的模型大概 1G 以上，所以这个例子的大部分并没有放到 SD 卡上（SD 卡上只有运行这个模型所需要的 TensorRT）。悲剧的是存放这些模型的服务器被墙了，所以只

能将之前下载好的包远程传输到对应的下载目录下。
首先如果您没有安装 git 和 cmake，先安装它们
sudo apt-get install libpython3-dev python3-numpy

sudo apt-get install git cmake

接着从 git 上克隆 jetson-inference 库

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

注意:可能会出现 error: RPC failed; curl 56 GnuTLS recv error (-54): Error in the pull function.这个原因是由于 git 默认缓存大小不足导致的，使用下面的命令增加缓存大小
git config --global http.postBuffer 5242880000
如果还是不行可能是网速慢，配置 git 的最低速度和最低速度时间
git config --global http.lowSpeedLimit 0
git config --global http.lowSpeedTime 999999
如果还是不行，以本人经历来说是网络问题，本人是在夜间网络中使用人数较少的时候成功的，建议在网络情况较好时进行。

cd jetson-inference

git submodule update --init （都需要执行）

配置 cmake，如果您科学上网没问题的话，会自动下载许多模型。

mkdir build #创建 build 文件夹
cd build #进入 build
cmake ../ #运行 cmake，它会自动执行上一级目录下面的 CMakePrebuild.sh（这里如果没能科学上网，请手动下载然后放到上面提到的路径 data/networks
注意:可能会出现个弹框出现说时候安装 pytorch，这个根据个人需要选择。

步骤如下：
编辑 jetson-inference/CMakePrebuild.sh。把./download-models.sh 注释掉,（前面加个#注释）如下图 1）。教程目录下“5、常用的库和模型文件” 下 “jetson-inference 需要的包”然后通过 WinSCP 或者 SSH 传输到 data/networks 目录,如下图 2 拖动到右侧目录下。

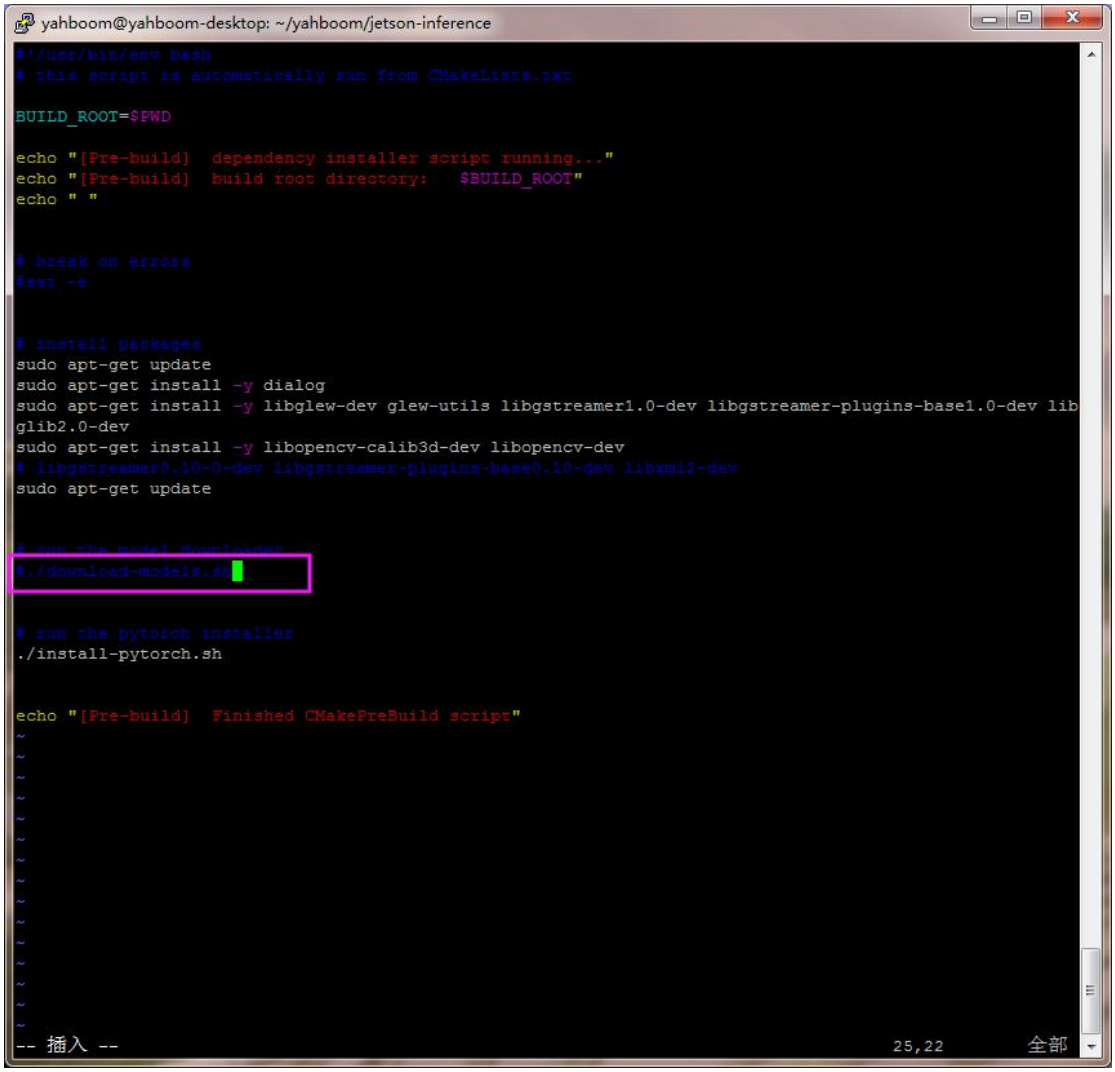
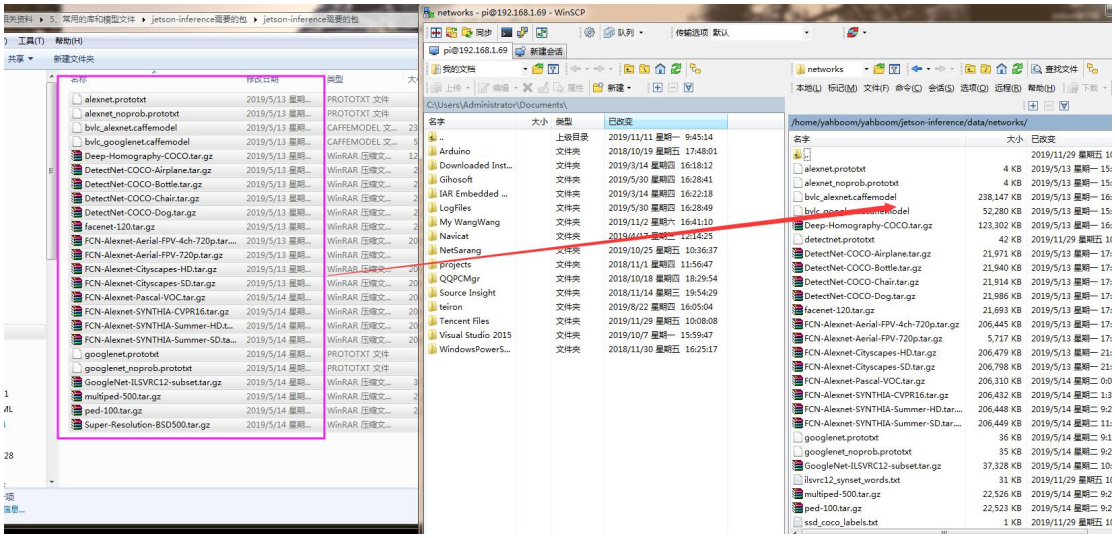


图 1



然后在此目录执行解压：

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$
```

注释：

对于解压多个.gz 文件的，用此命令：

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

对于解压多个.tar.gz 文件的，用下面命令：

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

cmake 成功后，就需要编译了

cd jetson-inference/build

这里的 make 不用 sudo

后面 -j4 使用 4 个 CPU 核同时编译，缩短时间

make (或者 make -j4)　　注意：（在 build 目录下）

sudo make install　　注意：（在 build 目录下）

如果编译成功，会生成下列文件夹结构

|-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

（三）测试

进入测试文件夹，运行

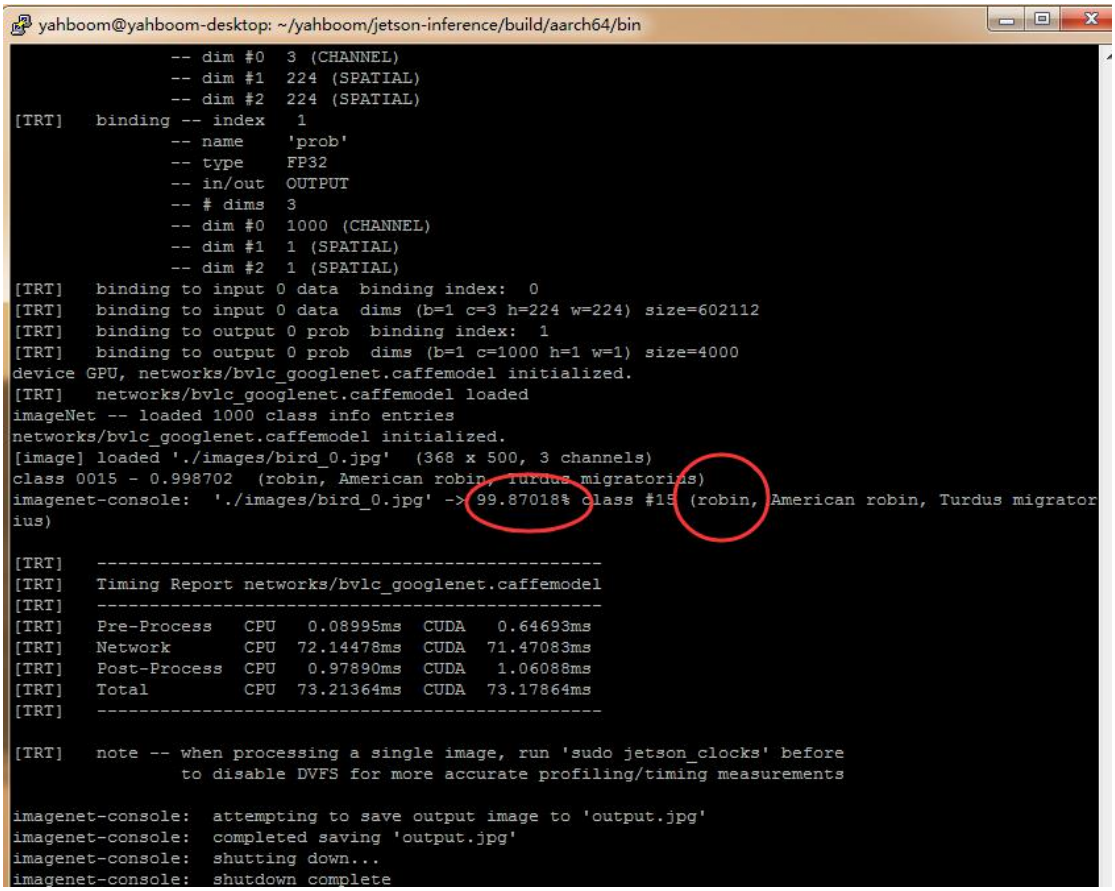
cd jetson-inference/build/aarch64/bin

./imagenet-console ./images/bird_0.jpg output.jpg

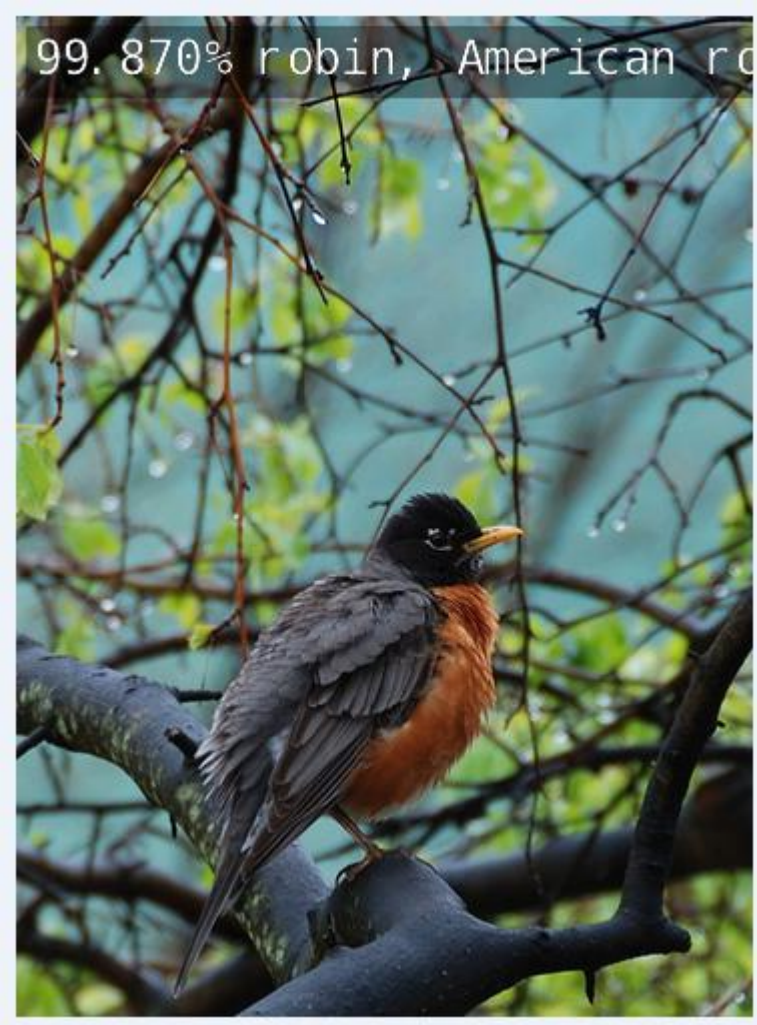
执行等待许久后出现如下（第一次需要很长时间，后面执行就会很快）：

```
yahboom@yahboom-desktop:~/yahboom/jetson-inference/build/aarch64/bin$ ./imagenet-console ./images/bird_
0.jpg output.jpg

imageNet -- loading classification network model from:
         -- prototxt     networks/googlenet.prototxt
         -- model        networks/bvlc_googlenet.caffemodel
         -- class_labels networks/ilsvrc12_synset_words.txt
         -- input_blob   'data'
         -- output_blob  'prob'
         -- batch_size   1
```



找到对应目录下查看 output.jpg 如下，会在图片上端显示识别结果。



更多学习请看官方文档：
官方 Demo：
https://developer.nvidia.com/embedded/twodaystoademo
官方 TensorRT 教程：
https://docs.nvidia.com/deeplearning/sdk/tensorrt-sample-support-guide/index.html