How to Run NVIDIA® Jetson™ Inference Example on Forecr Products (Installation + ImageNet)

Jetson Nano

02 August 2021

WHAT YOU WILL LEARN?

1- How to setup the system for image recognition program using jetson-inference?
2- How to recognize and classify an image using imageNet form pictures and videos?
3- How to write your own image recognition program using C++ or Python?

Hardware: DSBOX-N2

OS: Jetpack 4.5

In this blog post, we are going to explain how to classify images on Jetson™ Nano™ using jetson-inference. The GitHub post from dusty-nv will be taken as reference for the whole process.

How to Run the Docker Container

In this project, we will use the pre-built docker container from

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

but it will take more time.

Primero, debemos clonar los archivos en el proyecto.

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

Al ingresar al directorio jetson-inference que se creó, debemos ejecutar el contenedor.

```
cd jetson-inference
docker/run.sh
```

El contenedor Docker se ejecutará automáticamente y extraerá todos los archivos, tomará unos minutos dependiendo de la red. Esta es la primera configuración y solo se realizará una vez.

Luego, debemos construir el contenedor.

Entonces estamos listos para continuar con el siguiente paso.

Cómo construir el proyecto desde la fuente

Si decide no usar el contenedor Docker, puede instalar y compilar el proyecto directamente con los siguientes comandos.

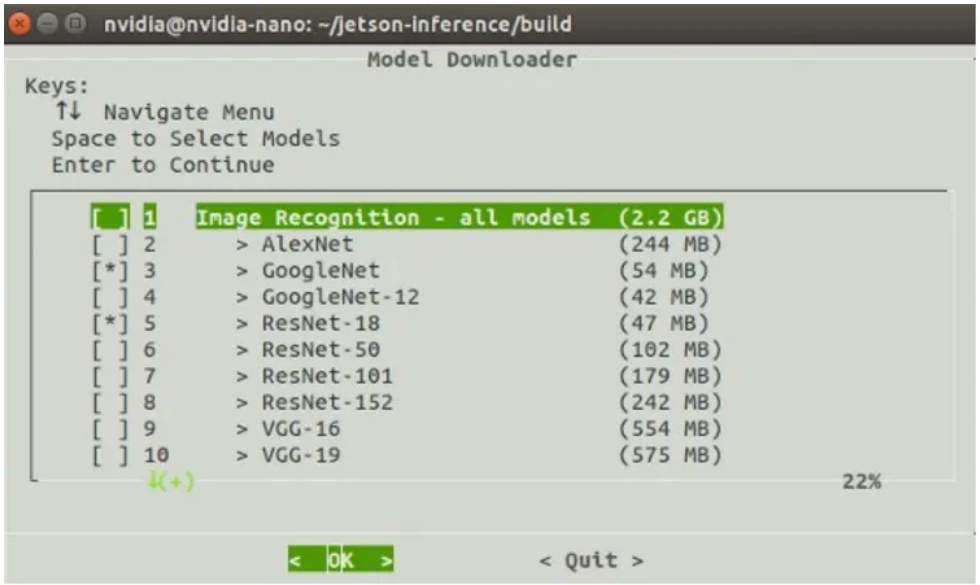
Primero, clone el proyecto Jetson-inference similar al Docker Container.

```
git clone https://github.com/dusty-nv/jetson-inference
cd jetson-inference
git submodule update –init
```

A continuación, para descargar todos los archivos necesarios y compilar el proyecto, cree una carpeta llamada build y ejecute cmake.

```
cmake ../
```

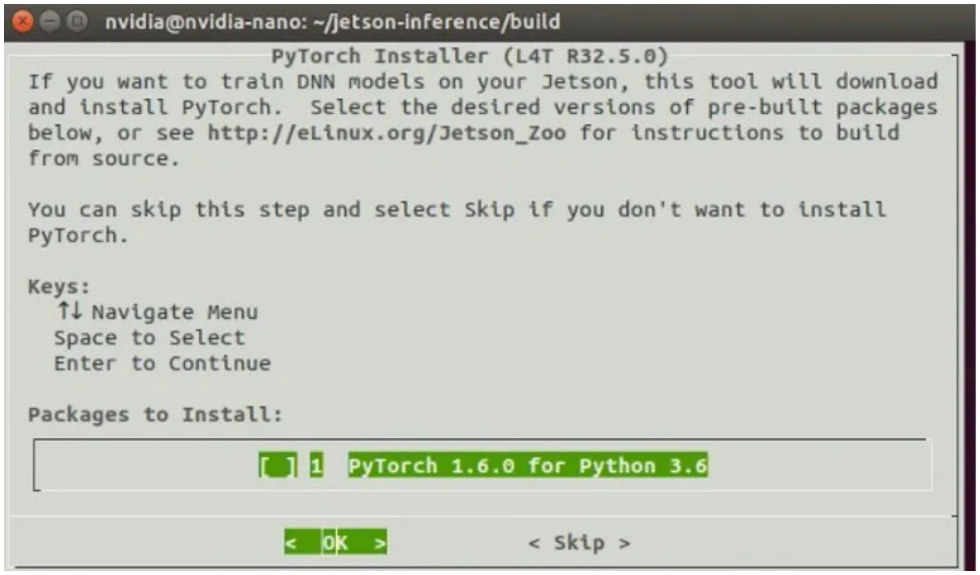
Luego, la herramienta Model-Downloader se ejecutará automáticamente en la pantalla. Este proyecto viene con varios modelos de red pre-entrenados, puede elegir cuál(es) descargar.



También puede volver a ejecutar la herramienta Model-Downloader más tarde con el siguiente

```
cd jetson-inference/tools
./download-models.sh
```

Luego, PyTorch Installer aparecerá en la pantalla. PyTorch se usa para volver a entrenar redes y no lo necesitaremos en este proyecto, por lo que puede omitir esta parte.



directorio de compilación.

```
make
sudo make install
sudo ldconfig
```

Cómo clasificar imágenes con imagen

Ahora, probaremos el programa ImageNet en imágenes de muestra para clasificar.

Primera opción:

Si usó el contenedor Docker, vaya al siguiente directorio.

```
cd jetson-inference
```

Luego, ejecute el archivo docker.

```
docker/run.sh
```

```
cd build/aarch64/bin
```

Segunda opción:

Si creó el proyecto desde la fuente, vaya directamente al siguiente directorio.

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

Next step is same for both options. Using ImageNet program written in both C++ and Python codes inside the docker container, classify sample images. You can also add your images to the data/images directory under jetson-inference. Supported image formats are JPG, PNG, TGA, BMP, GIF, PSD, HDR, PIC, and PNM (PPM/PGM binary).

directory following codes can be used:

```
./imagenet images/dog_2.jpg images/test/dog_2.jpg
```

OR

```
# Python
./imagenet.py images/dog_2.jpg images/test/dog_2.jpg
```

To test multiple images:

```
./imagenet "images/object_*.jpg"
"images/test/object_%i.jpg"
```

OR

```
# Python

./imagenet.py "images/object_*.jpg"
"images/test/object_%i.jpg"
```

We can also classify images from a video. Supported video formats are MP4, MKV, AVI, and FLV.

You can either download a sample video to jetson-inference directory by hand or by the following code:

```
wget
https://nvidia.box.com/shared/static/tlswont1jnyu3ix2t
bf7utaekpzcx4rc.mkv -O jellyfish.mkv
```

Then using the same code, you can run the imagenet program and save it on the test directory.

```
./imagenet jellyfish.mkv images/test/jellyfish.mkv
```

Imagenet uses GoogleNet as the default training program. You can also use different networks as in the

```
./imagenet --network=resnet-18 images/jellyfish.jpg
images/test/output_jellyfish.jpg
```

How to Code Your Own Image Recognition Program with C++

You can also create your own image recognition program similar to the ones come with the Jetson-inference project.
First, create a directory to the location you want in the terminal. In this example, we will create a directory named my-recognition in the home. Create C++ codes named my-recognition.cpp and CMakeLists.txt to make compiling easier.

```
mkdir ~/my-recognition
cd ~/my-recognition
touch my-recognition.cpp
touch CMakeLists.txt
```

```
wget https://github.com/dusty-nv/jetson-
inference/raw/master/data/images/black_bear.jpg
wget https://github.com/dusty-nv/jetson-
inference/raw/master/data/images/brown_bear.jpg
wget https://github.com/dusty-nv/jetson-
inference/raw/master/data/images/polar_bear.jpg
```

You can now open my-recognition.cpp with a text editor.
Include headers in jetson-inference for image recognition and loading images.

```
#include
#include
```

In the main, make sure you take the image filename and store it.

```
printf("my-recognition:  expected image filename as
argument\n");
printf("example usage:   ./my-recognition
my_image.jpg\n");
return 0;
}
// store the image filename
const char* imgFilename = argv[1];
```

To load the image from the disk use loadImage() function by storing image data pointer, image width and height as well.

```
uchar3* imgPtr = NULL;   //stores as RGB
int imgWidth   = 0;      // (in pixels)
int imgHeight  = 0;      // (in pixels)
// load the image from disk as uchar3 RGB (24 bits per
pixel)
if( !loadImage(imgFilename, &imgPtr, &imgWidth,
&imgHeight) )
{
printf("failed to load image '%s'\n", imgFilename);
return 0;
}
```

To load the image recognition network, use imageNet::Create() function . You can load different

Do not forget to check whether the network loaded properly.

```
imageNet* net = imageNet::Create(imageNet::GOOGLENET);
if( !net )
{
printf("failed to load image recognition network\n");
return 0;
}
```

Now it is time to classify the images using imageNet::Classify() function by using image filename, sata pointer, width, height and confidence of classification which is between 0 and 1.

```
float confidence = 0.0;
const int classIndex = net->Classify(imgPtr, imgWidth,
imgHeight, &confidence);
```

imageNet::GetClassDesc() function.

Finally, free the network's resource before shutting down the program.

```
delete net;
return 0;
}
```

Next, create CMakeList.txt file to compile the code easier. You can simply copy and paste the following code.

```
# import jetson-inference and jetson-utils packages.
# note that if you didn't do "sudo make install"
# while building jetson-inference, this will error.
find_package(jetson-utils)
find_package(jetson-inference)

# CUDA and Qt4 are required
find_package(CUDA)

# compile the my-recognition program
cuda_add_executable(my-recognition my-recognition.cpp)

# link my-recognition to jetson-inference library
target_link_libraries(my-recognition jetson-inference)
```

We must build our program before running. To do this, go into jetson-inference directory in the terminal and run docker/run.sh by allowing the docker access my-recognition directory as well.

```
cd ~/jetson-inference
docker/run.sh --volume ~/my-recognition:/my-
recognition
```

While running, go to my-recognition directory.
Now you can run the project from the terminal by writing the following example code.

How to Code Your Own Image Recognition Program with Python

You can either use the previous directory that you have created for example using C++, or create a new directory and repeat the following steps.

```
mkdir ~/my-recognition
cd ~/my-recognition
touch my-recognition.py
chmod +x my-recognition.pytouch CMakeLists.txt
```

We added chmod +x command to make the file executable later.
Then you can download and add the pictures you want. Again, we used the same sample pictures.
Now, we can start editing our code. First, add the libraries needed to recognize and load the images and parsing the command line.

Next, add the following code to parse the image filename and network name if a different network than GoogleNet would like to be use.

```
parser = argparse.ArgumentParser()
parser.add_argument("filename", type=str,
help="filename of the image to process")
parser.add_argument("--network", type=str,
default="googlenet", help="model to use, can be:
googlenet, resnet-18, ect. (see --help for others)")
opt = parser.parse_args()
```

To load images from the disk, you can use loadImage() function.

```
img = jetson.utils.loadImage(opt.filename)
```

The loaded image will be in the cudaImage format which contains memory address, size shape etc. You can find more information about cudaImage at

capsules-in-python

```
.ptr      # memory address (not typically used)
.size     # size in bytes
.shape    # (height,width,channels) tuple
.width    # width in pixels
.height   # height in pixels
.channels # number of color channels
.format   # format string
.mapped   # true if ZeroCopy
```

To classify the image, we will first need to load imageNet object and the network.

```
net = jetson.inference.imageNet(opt.network)
```

Then, we can use the recognition network by using imageNet.Classify() function.

```
class_idx, confidence = net.Classify(img)
```

Finally, we need to interpret the results that describes the class of the image at the terminal. Use

```
print("image is recognized as '{:s}' (class #{:d})
with {:f}% confidence".format(class_desc, class_idx,
confidence * 100))
```

Ahora, ve a la terminal para ejecutar nuestro código. Si creó el proyecto desde la fuente sin usar el contenedor docker, ejecute la siguiente línea de comando directamente mientras se encuentra en el directorio my-recognition.

```
cd my-recognition
my-recognition.py polar_bear.jpg
```

Si usó el contenedor docker, ejecute el docker permitiendo que el contenedor acceda también al directorio my-recognition. Use --volume para este propósito

```
cd jetson-inference
docker/run.sh --volume ~/my-recognition:/my-
recognition
```

```
cd my-recognition
my-recognition.py polar_bear.jpg
```

Gracias por leer nuestra publicación de blog.

Paginas

Hogar

Sobre nosotros

personalizar

Blog

Contáctenos

Envío y devolución

Presentacion de producto

productos

Contrato de venta a distancia