

Week1: Getting Started with Jetson Nano Developer Kit

Edge Computing

C. García garsanca@ucm.es

May 4, 2022

- “NVIDIA-Jetson: Hello AI World”,
<https://github.com/dusty-nv/jetson-inference/blob/master/docs/aux-docker.md>



Outline

- 1 Intro
- 2 Setup Container
- 3 Installing PyTorch
- 4 Setup ML Container



Setup Jetson-Nano

- In theory SD Card has been flashed with a installation (see Slides **Week1-Setup_JetsonNano-install**)

Credentials

- User name: **nano**
- User password: **nano_pass**



Using Camera (CSI Camera)

- CSI camera is the Rpi Camera connected through [ribbon cable](#)
 - More info could be consulted in [Jetson Nano 2GB Developer Kit User Guide](#)
- To check CSI camera, you can run `nvgstcapture-1.0`, which will start capture and preview display it on the screen:

Terminal #1

```
nano@jetson-nano:~$ nvgstcapture-1.0
Encoder null, cannot set bitrate!
Encoder Profile = High
Supported resolutions in case of ARGUS Camera
(2) : 640x480
(3) : 1280x720
(4) : 1920x1080
.....
```



Using Camera (CSI Camera)

- Check rotation
- This example command will rotate the image 180 degrees (vertical flip)

```
Terminal #1
```

```
nano@jetson-nano:~$ nvgstcapture-1.0 --orientation 2  
.....
```



Take a picture and save to disk

- 1 Connect CSI camera
- 2 Execute in a shell the command `nvgstcapture-1.0 -automate -capture-auto`
- 3 Open File with `eog nvcamtest_XX.jpg`



Capture a video and save to disk

- 1 Connect CSI camera
- 2 Execute in a shell the command `nvgstcapture-1.0 -mode=2 -automate -capture-auto`
- 3 Application will record 10 seconds of video
- 4 Play File recorded with `totem nvcamtest_XX.mp4`



Intro

- There are several pre-configured containers to be able to use the Jetson-Nano board
- The most common are related to their use for artificial intelligence and machine learning



Machine Learning Containers for Jetson and JetPack

- Extracted from <https://github.com/dusty-nv/jetson-containers>
- Hosted on NVIDIA GPU Cloud (NGC) are the following Docker container images for machine learning on Jetson:
 - l4t-ml
 - l4t-pytorch
 - l4t-tensorflow



Machine Learning Container

- The **14t-ml** docker image contains *TensorFlow*, *PyTorch*, *JupyterLab*, and other popular ML and data science frameworks such as scikit-learn, scipy, and Pandas pre-installed in a Python 3.6 environment
 - Latest 14t-ml:r32.6.1-py3
 - TensorFlow 1.15.5
 - PyTorch v1.9.0
 - torchvision v0.10.0
 - torchaudio v0.9.0
 - onnx 1.8.0
 - CuPy 9.2.0
 - numpy 1.19.5
 - numba 0.53.1
 - OpenCV 4.5.0 (with CUDA)
 - pandas 1.1.5
 - scipy 1.5.4



Pytorch Container

- The **l4t-pytorch** docker image contains *PyTorch* and *torchvision* pre-installed in a Python 3.6 environment
 - Latest `l4t-pytorch:r32.6.1-pt1.9-py3`
 - PyTorch v1.9.0
 - torchvision v0.10.0
 - torchaudio v0.9.0



Running Docker Container

- Pre-built Docker container images for this project are hosted on [DockerHub](#)
- These containers use the [l4t-pytorch](#) base container, so support for transfer learning / re-training is already included



Inference instructions

- Follow the github <https://github.com/dusty-nv/jetson-inference/blob/master/docs/aux-docker.md>



Launching the Container

- It's recommended to use the script `docker/run.sh` script to run the container
 - `docker/run.sh` will automatically pull the correct container tag from DockerHub based on your currently-installed version of JetPack-L4T
 - It also mount the appropriate data directories and devices so that you can use cameras/display/etc from within the container
 - More DNN models available in <https://github.com/dusty-nv/jetson-inference/blob/master/docs/building-repo-2.md#downloading-models>



Launching the Container

- **IMPORTANT:** if you are using CSI (Rpi Camera): **-volume /tmp/argus_socket:/tmp/argus_socket**

Terminal #1

```
nano@jetson-nano:~$ git clone --recursive https://github.com/dusty-nv/jetson-inference
Cloning into jetson-inference...
remote: Enumerating objects: 20861, done.
....

nano@jetson-nano:~$ cd jetson-inference/
nano@jetson-nano:~/jetson-inference$ docker/run.sh --volume /tmp/argus_socket:/tmp/argus_socket
reading L4T version from /etc/nv_tegra_release
L4T BSP Version: L4T R32.6.1
[sudo] password for nano:
size of data/networks: 79397 bytes
.....
```



Mount data volumes

- For reference, the following paths automatically get mounted from your host device into the container:
 - *jetson-inference/data* (stores the network models, serialized TensorRT engines, and test images)
 - *jetson-inference/python/training/classification /data* (stores classification training datasets)
 - *jetson-inference/python/training/classification/models* (stores classification models trained by PyTorch)
 - *jetson-inference/python/training/detection/ssd/data* (stores detection training datasets)
 - *jetson-inference/python/training/detection/ssd/models* (stores detection models trained by PyTorch)



Running applications

- Once the container is up and running, you can then run example programs from the tutorial like normal inside the container:

Terminal #1

```
root@jetson-nano:/jetson-inference# cd build/aarch64/bin
root@jetson-nano:/jetson-inference/build/aarch64/bin# ./video-viewer
root@jetson-nano:/jetson-inference/build/aarch64/bin# ./imagenet images/jellyfish.jpg images/test/
root@jetson-nano:/jetson-inference/build/aarch64/bin# ./detectnet images/peds_0.jpg images/test/
# (press Ctrl+D to exit the container)
```



Running applications

- Note that video-viewer catches the image from webcam



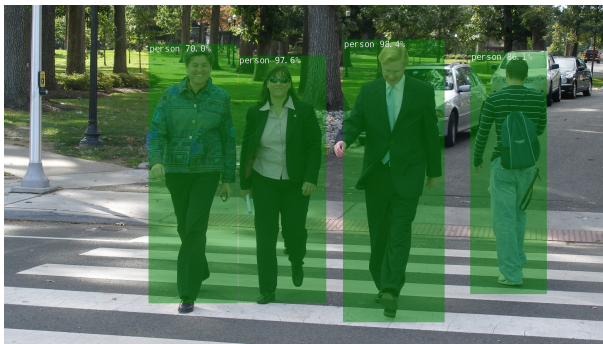
Running applications

- Note that imagenet app **classifies** the image *jellyfish.jpg* as a **jellyfish** and store the image solution in the path *data/images/test* with a confidence of 99.85%

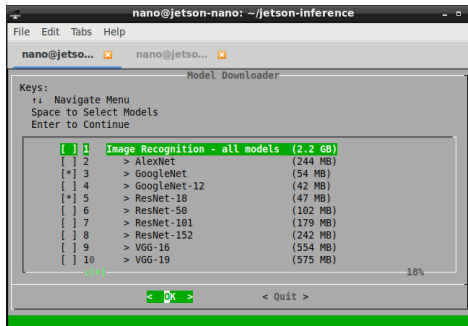


Running applications

- Note that detectnet app detects four persons with a confidence of 70.0%, 97.6%, 98.4% and 86.1% and store the image solution in path `data/images/test`



Download other models



Terminal #1

```
nano@jetson-nano:~$ cd jetson-inference/tools
nano@jetson-nano:~/jetson-inference$ ./download-models.sh
.....
```



Installing PyTorch

- If you are running the Docker Container, it should already be installed on your Jetson
- Otherwise you can install it

Terminal #1

```
nano@jetson-nano:~$ cd jetson-inference/build  
nano@jetson-nano~/jetson-inference/build$ ./install-pytorch.sh
```



Verifying PyTorch

- Test the success PyTorch installation by executing the next commands from an interactive Python shell:

testing.py

```
import torch
print(torch.__version__)
print('CUDA available: ' + str(torch.cuda.is_available()))
a = torch.cuda.FloatTensor(2).zero_()
print('Tensor a = ' + str(a))
b = torch.randn(2).cuda()
print('Tensor b = ' + str(b))
c = a + b
print('Tensor c = ' + str(c))

import torchvision
print(torchvision.__version__)
```



Machine Learning Container

- The **l4t-ml** docker image contains *TensorFlow*, *PyTorch*, *JupyterLab*, and other popular ML and data science frameworks such as scikit-learn, scipy, and Pandas pre-installed in a Python 3.6 environment
 - Latest l4t-ml:r32.6.1-py3
 - TensorFlow 1.15.5
 - PyTorch v1.9.0
 - torchvision v0.10.0
 - torchaudio v0.9.0
 - onnx 1.8.0
 - CuPy 9.2.0
 - numpy 1.19.5
 - numba 0.53.1
 - OpenCV 4.5.0 (with CUDA)
 - pandas 1.1.5
 - scipy 1.5.4



Running the Container

- First pull one of the **l4t-ml** container:

Terminal #1

```
nano@jetson-nano:~$ sudo docker pull nvcr.io/nvidia/l4t-ml:r32.6.1-py3
....
```

- Then to start an interactive session in the container, run the following command:

Terminal #1

```
nano@jetson-nano:~$ sudo docker run -it --gpus all -e DISPLAY=:0 -v /tmp/.X11-unix:/tmp/.X11-unix --network host nvcr.io/nvidia/l4t-ml:r32.6.1-py3
```



Mounting Directories

- To mount scripts, data, ect. from your Jetson's filesystem to run inside the container, use Docker's `-v` flag when starting your Docker instance:

Terminal #1

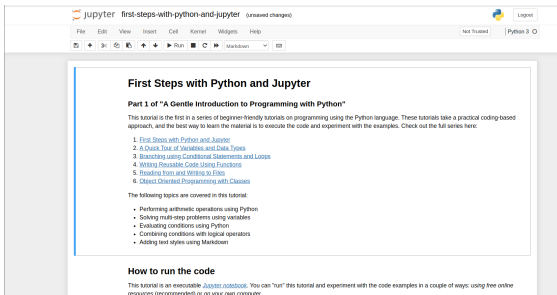
```
nano@jetson-nano:~$ sudo docker run -it --rm --runtime nvidia --network host -v /home/user/project:/location/in/container nvcr.io/nvidia/14t-ml:r32.6.1-py3
```

- You should then be able to start a Python3 interpreter



Connecting to JupyterLab Server

- A JupyterLab server instance is automatically started along with the container.
- You can connect <http://localhost:8888> (or substitute the IP address of your Jetson device)
- Password: **nvidia**



The screenshot shows a JupyterLab notebook titled "first-steps-with-python-and-jupyter (unsaved changes)". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help), a toolbar with navigation and execution icons, and a status bar showing "next trusted" and "Python 3.0". The main content area displays the "First Steps with Python and Jupyter" tutorial, which is the first part of a series of beginner-friendly tutorials on programming with Python. The tutorial includes a list of topics to be covered and a section on how to run the code.

First Steps with Python and Jupyter

Part 1 of "A Gentle Introduction to Programming with Python"

This tutorial is the first in a series of beginner-friendly tutorials on programming using the Python language. These tutorials take a practical coding-based approach, and the best way to learn the material is to execute the code and experiment with the examples. Check out the full series here:

1. [First Steps with Python and Jupyter](#)
2. [A Quick Tour of Variables and Data Types](#)
3. [Branching using Conditional Statements and Loops](#)
4. [Writing Reusable Code Using Functions](#)
5. [Reading from and Writing to Files](#)
6. [Object Oriented Programming with Classes](#)

The following topics are covered in this tutorial:

- Performing arithmetic operations using Python
- Solving multi-step problems using variables
- Evaluating conditions using Python
- Controlling conditions with logical operators
- Adding text styles using Markdown

How to run the code

This tutorial is an executable [Jupyter notebook](#). You can "run" this tutorial and experiment with the code examples in a couple of ways: using free online resources (recommended) or on your own computer.

