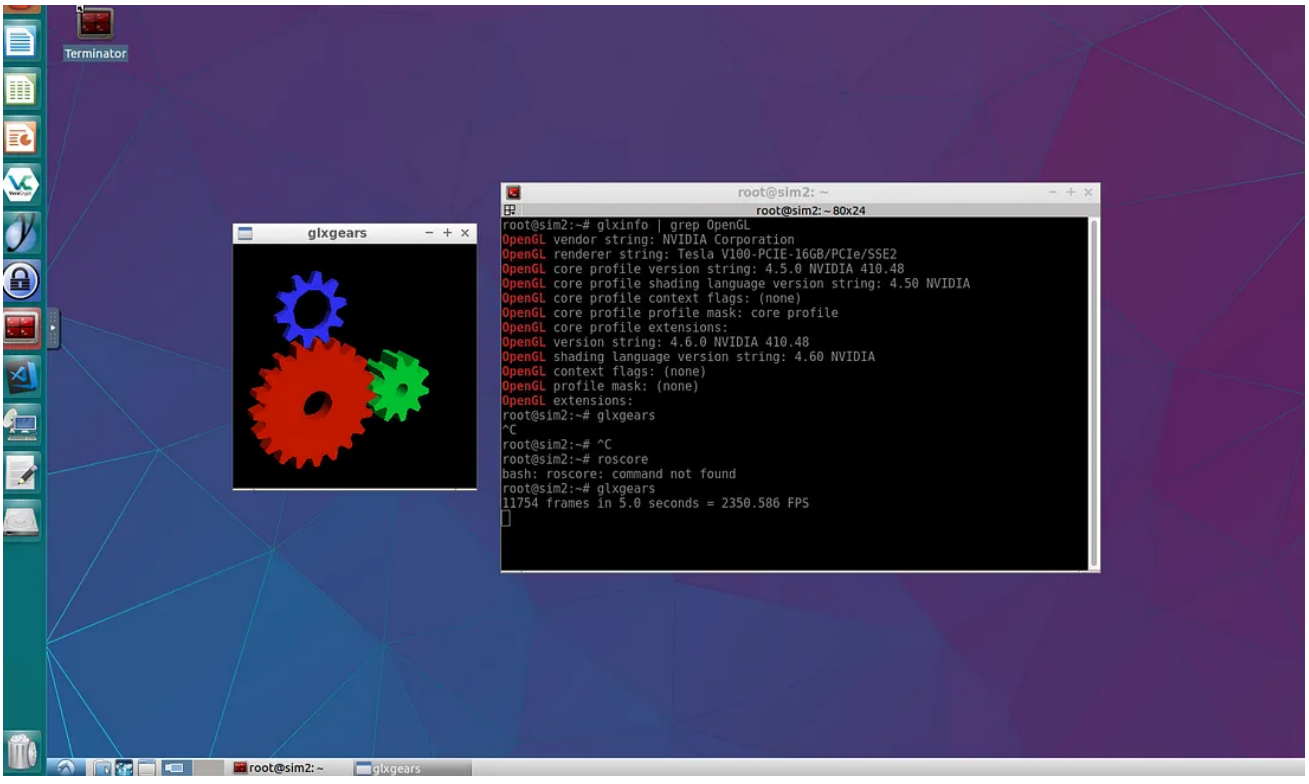
Running OpenGL & CUDA Applications with nvidia-docker2 on a Remote (headless) GPU System

Bugra Turan · Follow
5 min read · Jan 3, 2020

This story tackles the problem of running OpenGL accelerated GUI applications using HW-GPU support in a docker container on a remote headless server system (or cloud).

A possible use case for this is the popular Robot Operating System (ROS) together with its simulator Gazebo. However, all OpenGL applications are in scope (e.g. Unity, OpenAI´s Gym).



Desktop environment within the docker container with glxgears running and output of glxinfo.

For a long time there were multiple issues that hindered the above mentioned use cases. These are for example:

- X11 forwarding using OpenGL with HW support with docker. This works fine as long as you do not need HW acceleration. For more details please check: https://medium.com/@pigiuz/hw-accelerated-gui-apps-on-docker-7fd424fe813e
- Having the need for GLX in docker using NVIDIA graphics card: https://github.com/NVIDIA/nvidia-docker/issues/534. *"OpenGL is not supported at the moment and there is no plan to support GLX is the near future (same as 1.0). OpenGL+EGL however is on the roadmap and will be supported. We will update #11 once we publish it."* I found this pretty frustrating since Gazebo disables certain functionalities if there is no OpenGL available.
- Xorg with HW-accel on a headless server system. Persistent settings to have the advantage of the NVIDIA driver capabilities.

Most issues are caused by lack of knowledge from my side. But now that possible solutions are found I want to give the reader a guide to allow for all possible use-cases. If you want the read more about the background and the tutorials which my guide is based on please refer to:

Setting up a HW accelerated desktop on AWS G2 instances

GPU access is becoming a more and more common requirement in a growing number of use cases. Whether we're doing some...

medium.com

Creating a GPU-Enhanced Virtual Desktop for Udacity

We're thrilled to announce that Udacity has launched one-button access to a full GPU-driven Ubuntu desktop.

engineering.udacity.com

Structure

At the end we will have a docker container that is running on a remote, headless server (or cloud, VM, etc.) using NVIDIA GPU acceleration and OpenGL capabilities. The container accesses the hosts X11 server but the GUI functionalities are enabled by a combination of:

- VirtualGL
- TurboVNC
- noVNC

Since 2018 nvidia-docker2 NVIDIA released a new (vendor specific?) version of the libglvnd library that allows for OpenGL support together with docker (see). We therefore start from these image (often called cudagl).

To allow for multiple use-cases a simple python script was written to generate the final Dockerfile that is used. The following images shows to possibilities and the structure.

Flowchart for the Dockerfile-generation wizard.

The part in the header is always required. Here we select:

- The Ubuntu base version (e.g. 16.04)
- The CUDA version (e.g. 9.0)

1. Afterwards we are asked if we want to include the CuDNN library which is useful for Deep Learning applications.
2. In the main part, all required modules and packages are installed for the GUI abilities (VirtualGL, TurboVNC, noVNC, Lubuntu Desktop)
3. Here we can add a ROS distribution if we want.

4. Next, if we require Gazebo it can be added here.
5. Sometimes it is useful to install common python packages for machine learning and development.
6. For fast inference of neural network the popular TensorRT can be added here. Please note that the .deb needs to be present in the folder since it can not be downloaded automatically (NVIDIA portal credentials required).

In the tail part the entrypoint, command etc of the docker container is defined. The next part describes a step by step instruction for the project.

Step-by-Step Guide

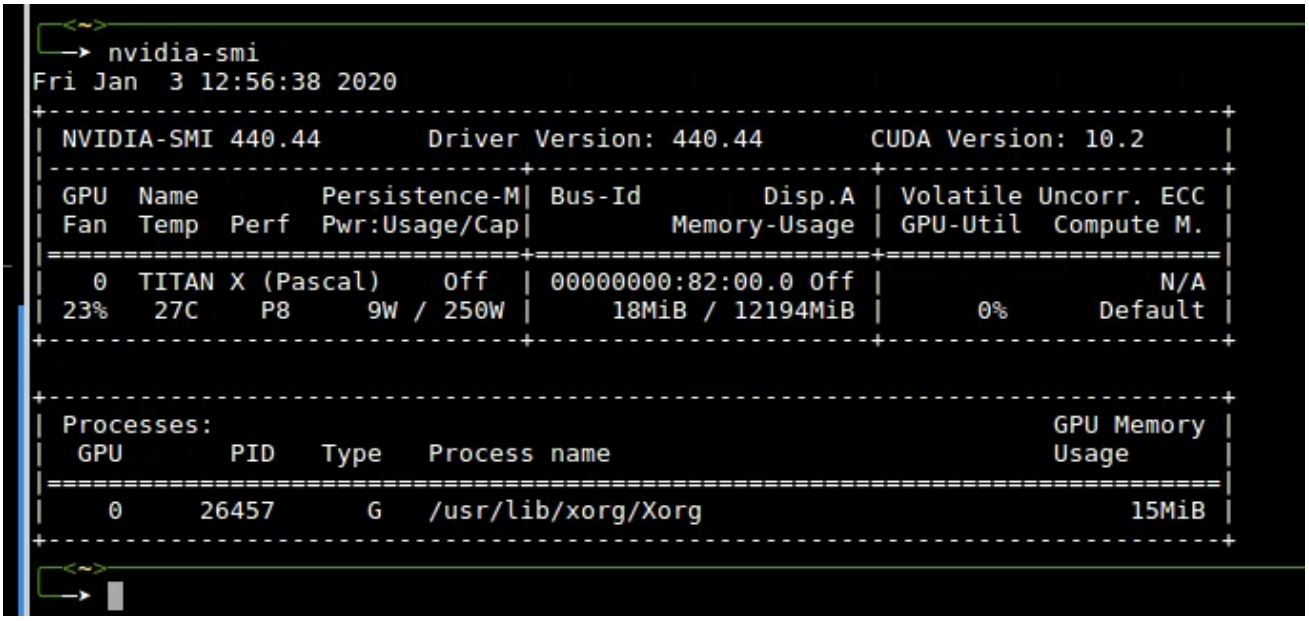
Prerequisites:

- Properly set-up host OS (Ubuntu in our case)
- docker and nvidia-docker2
- NVIDIA Driver (more on this in the following)

Before we can start we need to check if the Xorg process on the remote host is running and uses the dedicated NVIDIA hardware. For this we look at the output of the command:

```
nvidia-smi
```

The output should look like this:



Output of nvidia-smi with HW-accel. Xorg process.

The important part is the HW-accelarated Xorg process. This is very important for proper functioning! **If the process is missing please follow my guide on Github to fix this problem. You can find it here.**

Now we can start the actual guide. First clone the repository with all the required scripts and files: https://github.com/trn84/recipe-wizard

1. Clone the repository:

```
git clone https://github.com/trn84/recipe-wizard
```

2. Execute the wizard:

```
python3 wizard.py
```

Follow the instructions of the script. Please be aware that for the selection of CuDNN the full version is required.

3. Check the generated Dockerfile. Take a look into the build.sh file.

It is useful to use a meaningful image tag name.

4. Create the docker image:

```
./build.sh
```

Now wait, it takes some time. The Dockefile.XXX templates can be optimized. The repo right now is optimized for Robotics and Deep Learning applications.

5. If everything went without errors take a look into the run.sh file. I will go through the arguments and parameters step-by-step:

```
nvidia-docker run -it \
--name=super-container \
--security-opt seccomp=unconfined \
--init \
--net=host \
--privileged=true \
--rm=true \
-e DISPLAY=:1 \
-v /tmp/.X11-unix/X0:/tmp/.X11-unix/X0:ro \
-v /etc/localtime:/etc/localtime:ro \
-v /share-docker:/share-docker \
recipe-wiz
```

- We use nvidia-docker as command. You can also use the — runtime=nvidia flag for the same effect.
- The security-opt and privileged flags are meant to allow for a close coupling of the container to the host OS. Potentially this is not required.
- The flag — net=host allows for native port forwarding. Otherwise you need to specifically map the ports like VNC, ROS, Gazebo etc...
- The other flags are quite standard. We have added a shared volume mount. You should edit the flags to your need.

6. Execute container creation and run:

```
./run.sh
```

This will give you a /bin/bash shell since in our template the CMD was set to /bin/bash. You can change this to your needs. We can start the desktop in the container with the following command:

```
./usr/local/bin/start_desktop.sh
```

Docker Ros Gazebo Cuda Cudnn e note that the IP/Hostname could be wrong and the correct IP needs to be replaced.

7. Start noVNC and connect to the container.

After you are connected to the Lubuntu desktop open the Terminator from the desktop and check the proper GPU features by executing:

```
nfo
rs
re
python -> import tensorflow
```

Follow

Written by Bugra Turan

11 Followers

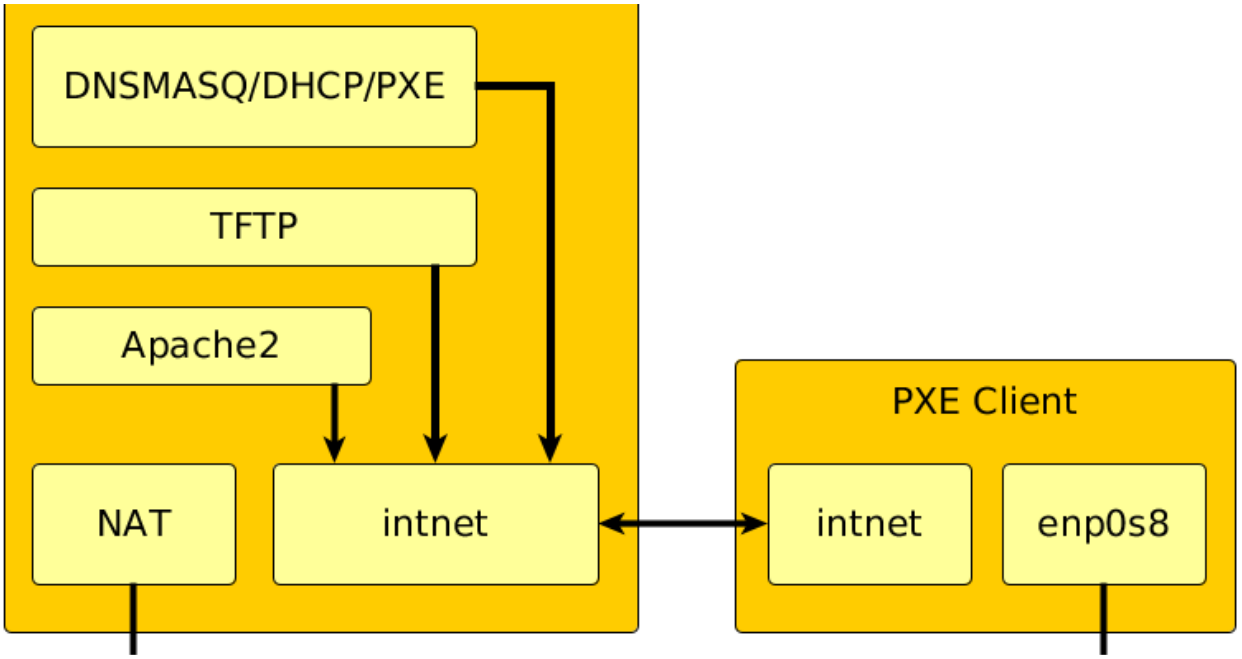
You should be good to go now :-)

Sources

This project is an aggregation and combination of many different source:

More from Bugra Turan

- https://medium.com/@pigiuz/hw-accelerated-gui-apps-on-docker-7fd424fe813e
- https://github.com/plumbee/nvidia-virtualgl
- https://github.com/willkessler/nvidia-docker-novnc
- https://gitlab.com/nvidia/samples/blob/master/opengl/ubuntu16.04/turbovnc-

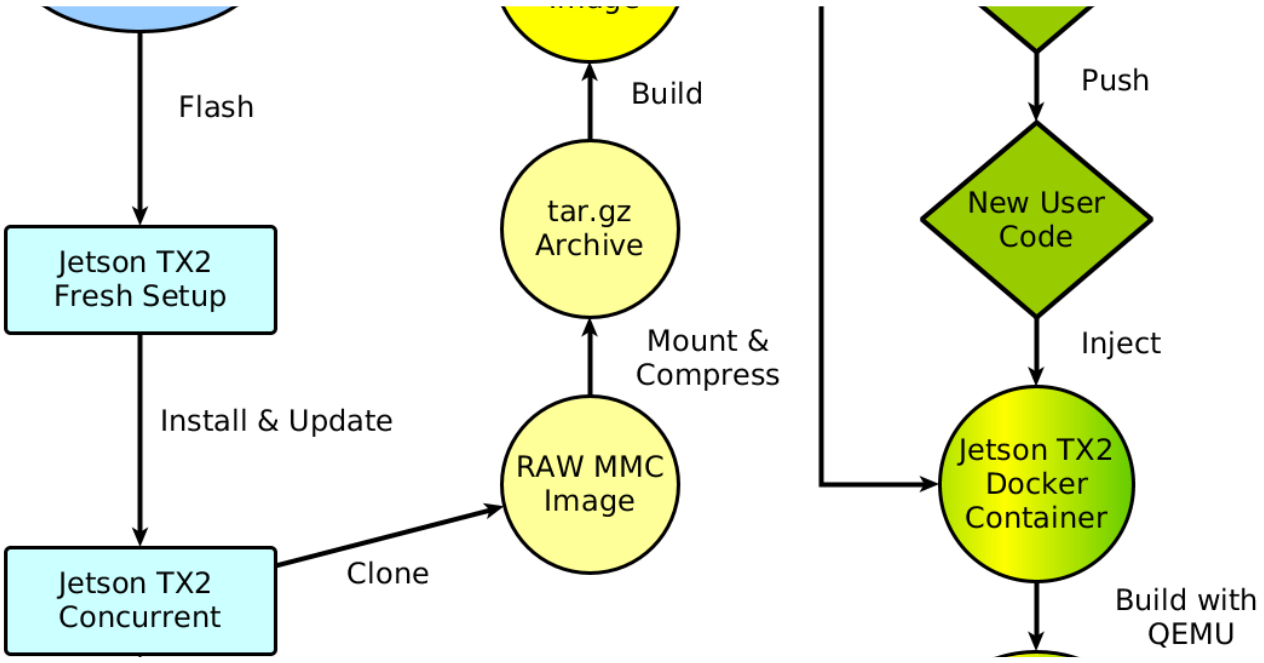


Bugra Turan

Using PXE for the automatic and customized installation of Ubuntu 18.04 Server

This guide is dedicated to install Ubuntu 18.04 Server LTS via PXE.

5 min read · May 18, 2020



Bugra Turan

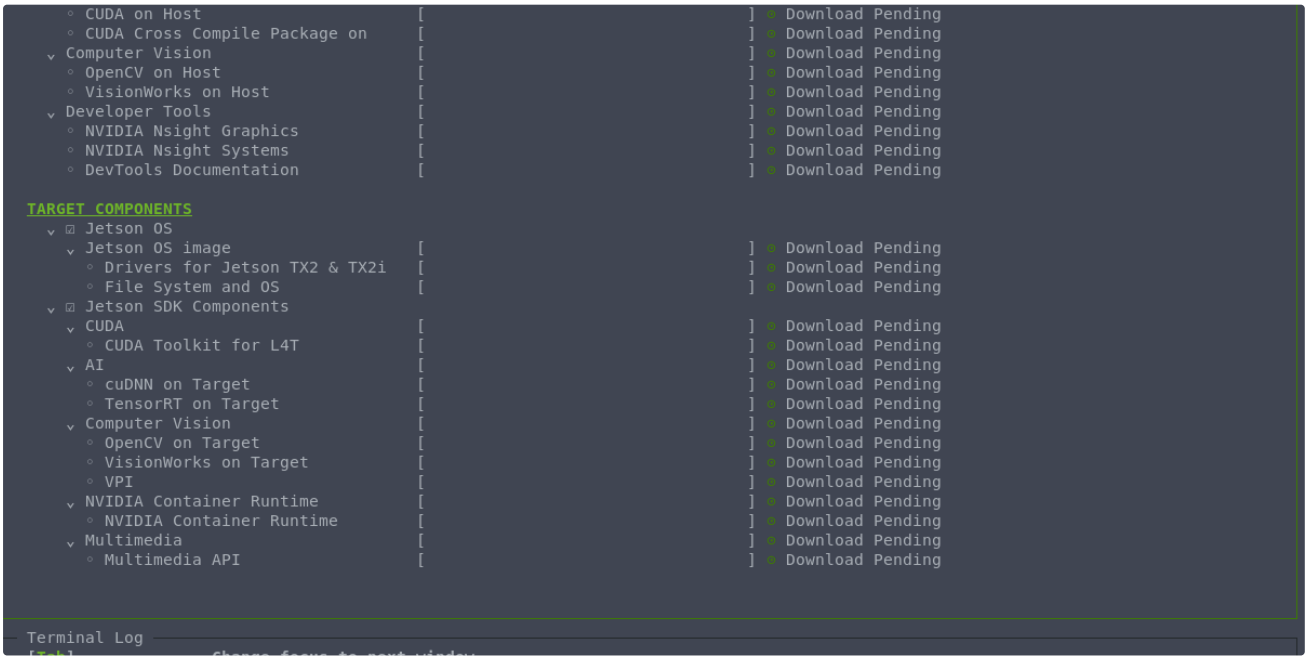
Creating a Cross-Compilation like process for Jetson TX2 (aarch64) using QEMU with Docker

Easily build your applications in your CI/CD on x86 without needing an aarch64 build-farm

6 min read · Apr 21, 2020


28


1



Bugra Turan

JetPack SDK in Docker for simple and clean flashing of a Jetson TX2

The Jetson Platform is nice—the JetPack SDK, not so much...

5 min read · Apr 22, 2020


3


1

Bugra Turan

Setting up a RAID60 on a LSI MegaRAID Controller under Ubuntu 18.04

Simple steps for setting up all software configurations for your RAID controller.

Netra Prasad Neupane

Vatsalya Krishan Maurya

How to Setup GPU for Deep Learning (Windows 11)

I have searched various methods, but I was not able to find the correct way to enable GPU on my Windows machine. But on research and...

6 min read · Jun 16

58

Lists

Coding & Development

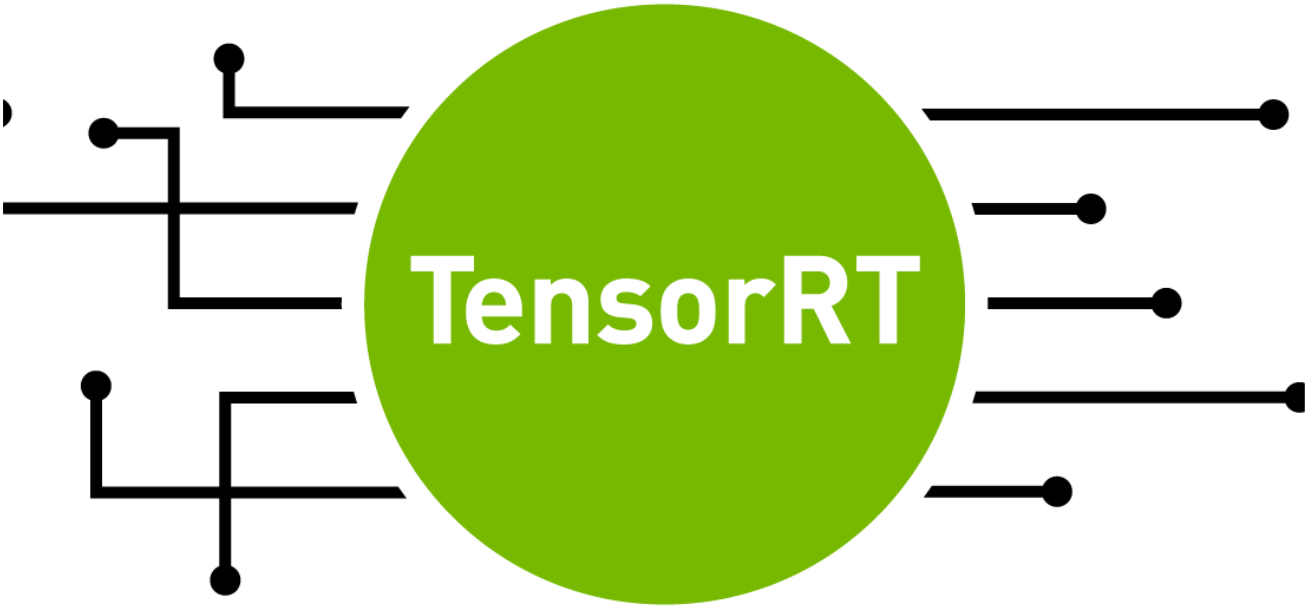
11 stories · 227 saves

New_Reading_List

174 stories · 155 saves

Natural Language Processing

739 stories · 331 saves



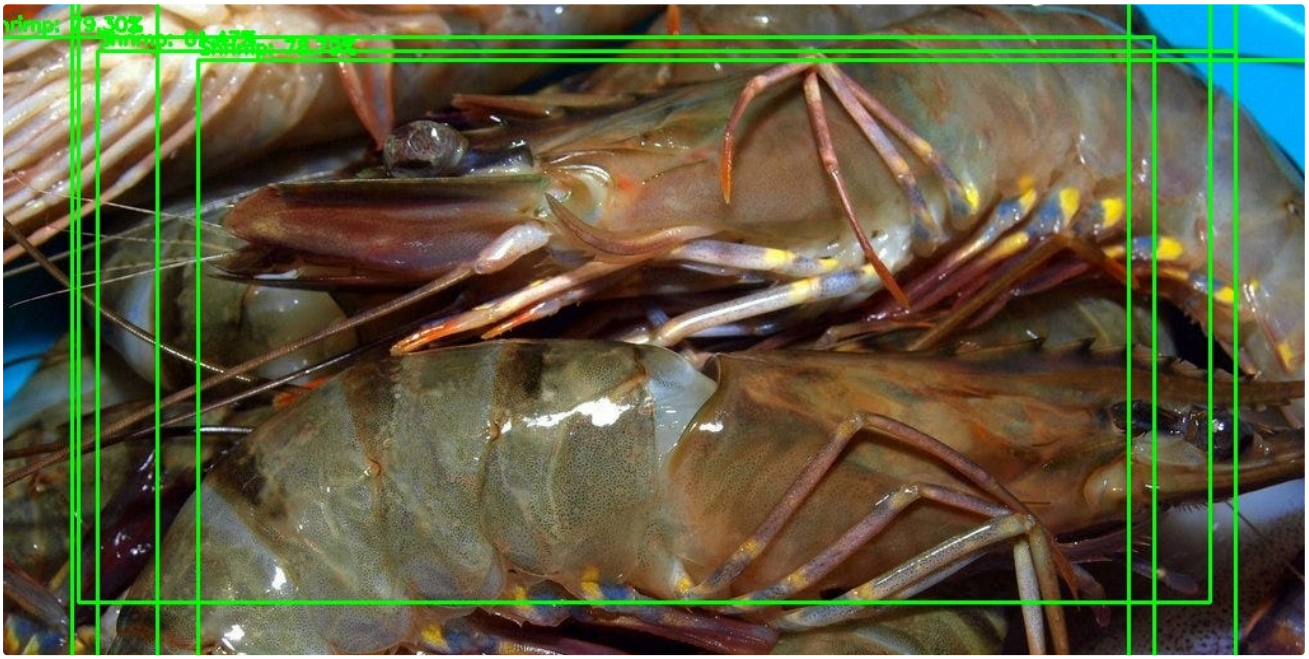
Nawin Raj Kumar S in kgxperience

How to install TensorRT: A comprehensive guide

TensorRT is a high-performance deep-learning inference library developed by NVIDIA. It is specifically designed to optimize and accelerate...

4 min read · Jul 28

51



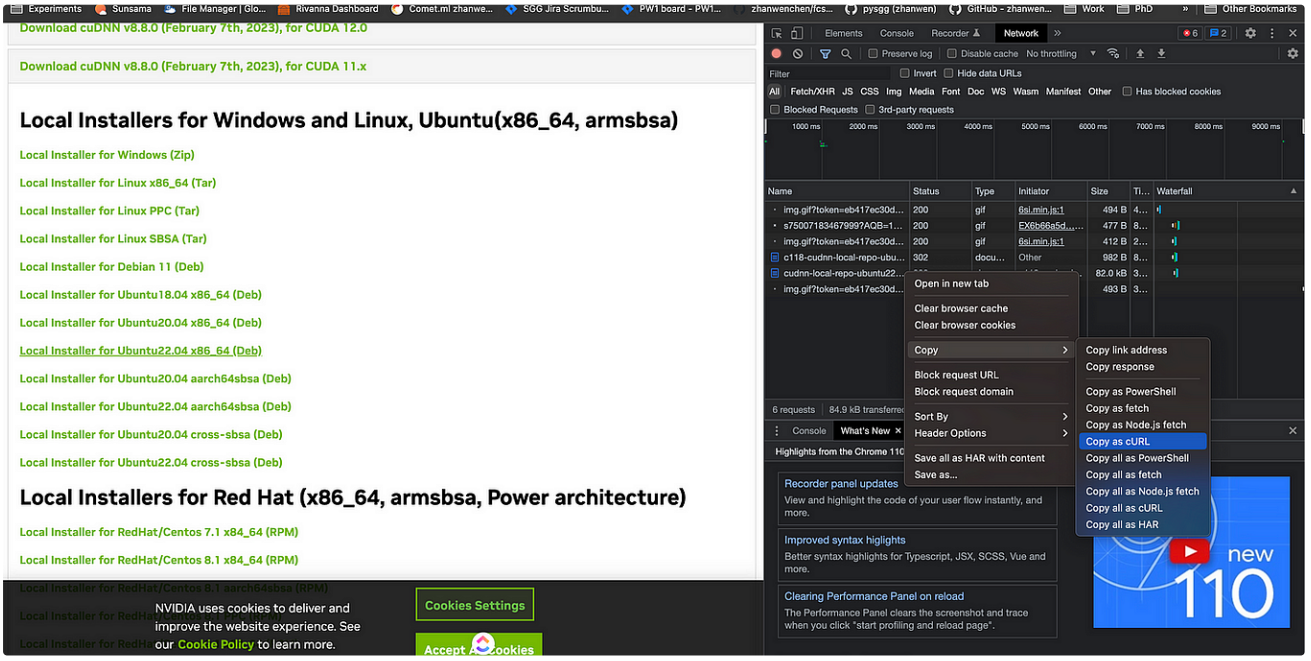
Alejandro Salinas-Medina

Enhancing a YOLO Algorithm for Accurate Prediction of Over 600 Custom Classes

Hello everyone! In this tutorial, we will demonstrate how to train a YOLO (You Only Look Once) algorithm for object detection with custom...

11 min read · May 11

72 1



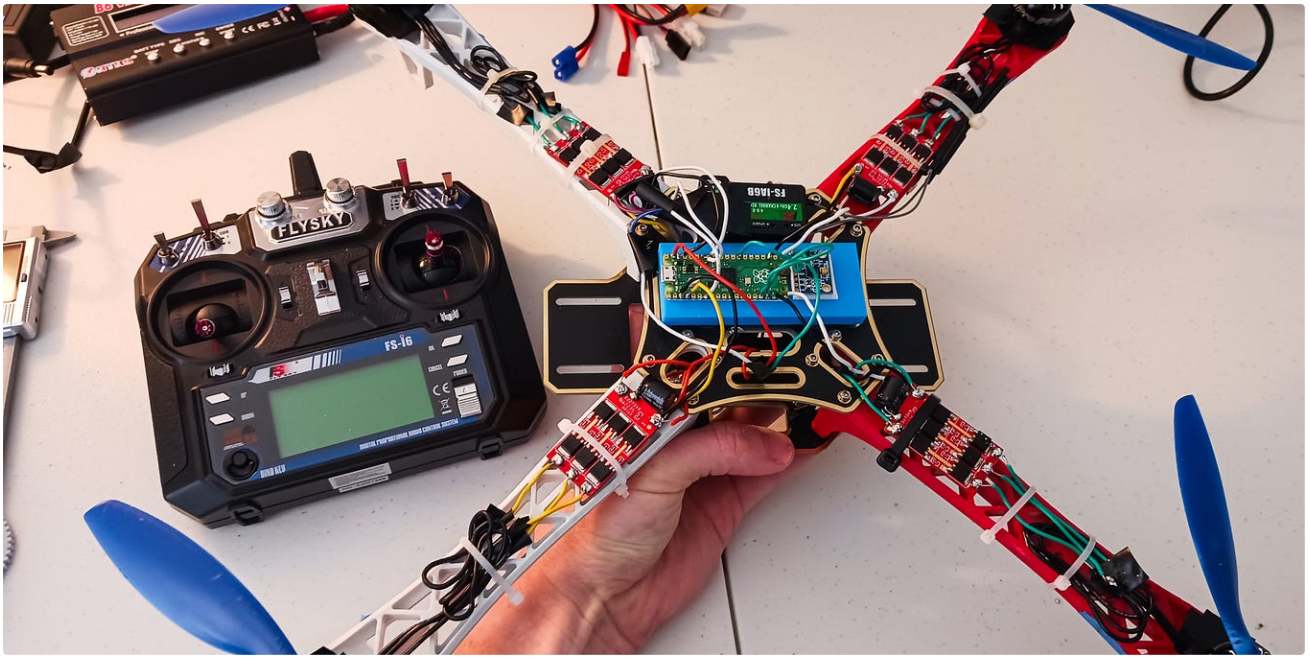
Zhanwen Chen

Build PyTorch from Source with CUDA 12.2.1 with Ubuntu 22.04

The NVIDIA 535 driver provides excellent backward compatibility with CUDA versions. Meanwhile, as of writing, PyTorch does not fully...

20 min read · Aug 10

10 2



Tim Hanewich

My Greatest Engineering Accomplishment: The Scout Flight Controller

Over the course of the past six years, I have dedicated my professional focus to various software development disciplines — back-end...

3 min read · Jul 18

73 1

See more recommendations