



Yolov4 Dependences

Steps to install Cuda and cuDNN and run yolov4 on linux

1. Install Anaconda in your linux machine to a create a virtual environment

- Create Conda virtual environment

```
syntax: conda create -n <env_name> python=version>  
cmd:    conda create -n yolo python=3.7
```

2. Git Clone the darknet repository required to run yolov4

```
cmd: git clone https://github.com/AlexeyAB/darknet.git
```

3. Installing CUDA

3.1 Download and install CUDA Toolkit

- Firstly, run the following commands to update and upgrade all the packages on your ubuntu.

```
sudo apt-get update  
sudo apt-get upgrade
```

- Next, install all the necessary dependencies

```
sudo apt-get install build-essential cmake unzip pkg-config  
sudo apt-get install gcc-6 g++-6  
sudo apt-get install libxmu-dev libxi-dev libglu1-mesa libglu1-mesa-dev  
sudo apt-get install libjpeg-dev libpng-dev libtiff-dev  
sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev  
sudo apt-get install libxvidcore-dev libx264-dev  
sudo apt-get install libopenblas-dev libatlas-base-dev liblapack-dev gfortran  
sudo apt-get install libhdf5-serial-dev  
sudo apt-get install python3-dev python3-tk python-imaging-tk  
sudo apt-get install libgtk-3-dev
```

- Next, add the ppa:graphics-drivers/ppa repository into your system:

```
sudo add-apt-repository ppa:graphics-drivers/ppa  
sudo apt update
```

- Next, install the NVidia driver compatible with your CUDA version. For CUDA 11.0 the minimum driver version for Linux is $\geq 450.36.06$.

```
sudo apt-get install nvidia-driver-450
```

- (You can read about the minimum driver versions required here on the links given below.)

```
https://docs.nvidia.com/deploy/cuda-compatibility/index.html
```


```
https://docs.nvidia.com/cuda/cuda-toolkit-release-notes/index.html
```

- Next, create a folder and run the commands from the installation instructions for the base installer.

Get a suitable CUDA version from the below mentioned link

CUDA Toolkit 11.5 Downloads

[Resources](#) [CUDA Documentation/Release Notes](#) [MacOS Tools](#) [Training Sample Code](#) [Forums](#) [Archive of Previous CUDA Releases](#) [FAQ](#)
[Open Source Packages](#) [Submit a Bug](#)

 <https://developer.nvidia.com/cuda-downloads>

```
cd ~
mkdir installers
cd installers/

# Get your cuda version from the above mentioned link this command uses cuda-11.0
wget https://developer.download.nvidia.com/compute/cuda/11.0.3/local_installers/cuda_11.0.3_450.51.06_linux.run
sudo sh cuda_11.0.3_450.51.06_linux.run
```

- When the setup starts accept the EULA agreement, then on the next screen uncheck the NVidia driver option since we have already installed it manually earlier and finally press Enter to install CUDA.

```
ziz@ubuntu: ~/Installers
File Edit View Search Terminal Help

CUDA Installer
- [ ] Driver
  [ ] 450.51.06
+ [X] CUDA Toolkit 11.0
  [X] CUDA Samples 11.0
  [X] CUDA Demo Suite 11.0
  [X] CUDA Documentation 11.0
Options
Install

Up/Down: Move | Left/Right: Expand | 'Enter': Select | 'A': Advanced options
```

- You'll see a summary at the end of CUDA installation as shown below.

```
ziz@ubuntu: ~/Installers
File Edit View Search Terminal Help

=====
= Summary =
=====

Driver:    Not Selected
Toolkit:   Installed in /usr/local/cuda-11.0/
Samples:   Installed in /home/ziz/, but missing recommended libraries

Please make sure that
- PATH includes /usr/local/cuda-11.0/bin
- LD_LIBRARY_PATH includes /usr/local/cuda-11.0/lib64, or, add /usr/local/cuda-11.0/lib64 to /etc/ld.so.conf and run ldconfig as root

To uninstall the CUDA Toolkit, run cuda-uninstaller in /usr/local/cuda-11.0/bin
***WARNING: Incomplete installation! This installation did not install the CUDA Driver. A driver of version at least .00 is required for CUDA 11.0 functionality to work.
To install the driver using this installer, run the following command, replacing <CudaInstaller> with the name of this run file:
    sudo <CudaInstaller>.run --silent --driver

Logfile is /var/log/cuda-installer.log
```

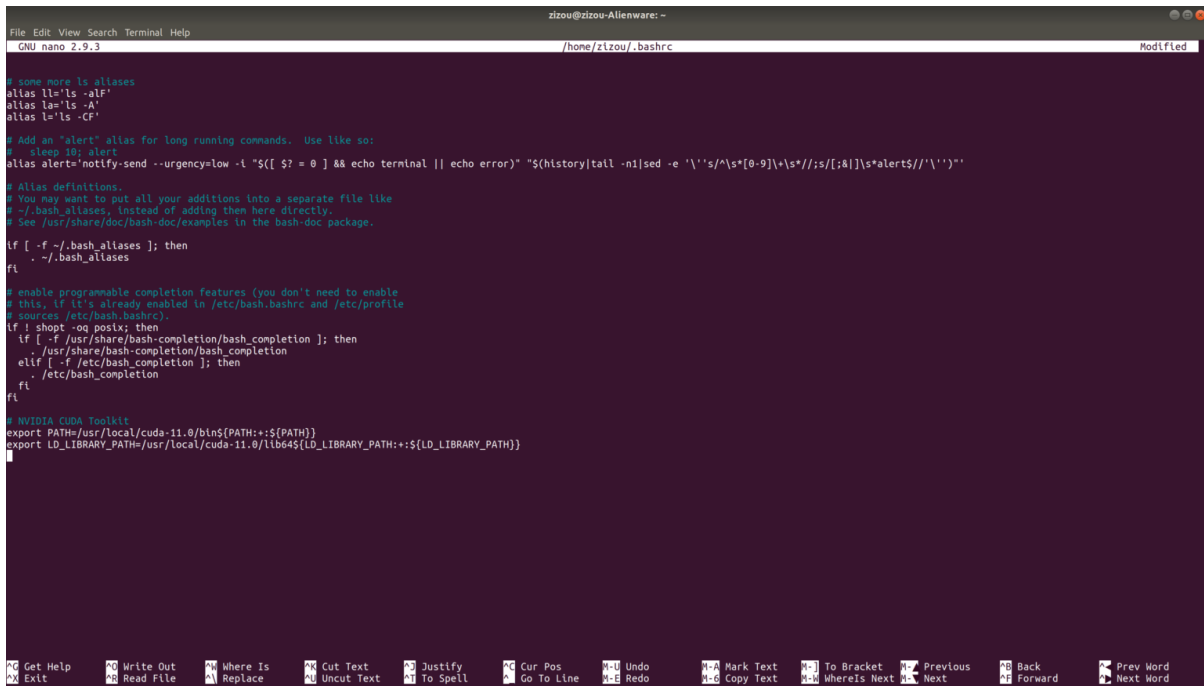
- Next, open the bashrc script file using the following command.

```
nano ~/.bashrc
```

- Add the following lines at the end of the bashrc file.

```
# NVIDIA CUDA TOOLKIT
export PATH=/usr/local/cuda-11.0/bin${PATH:+:${PATH}}
export LD_LIBRARY_PATH=/usr/local/cuda-11.0/lib64${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}
```

- The bashrc file looks like below:



```
zizou@zizou-Allienware: ~
File Edit View Search Terminal Help
GNU nano 2.9.3 /home/zizou/.bashrc Modified

# some more ls aliases
alias ll='ls -alF'
alias la='ls -A'
alias l='ls -CF'

# Add an "alert" alias for long running commands. Use like so:
# sleep 10; alert
alias alert='notify-send --urgency=low -i "${?} = 0" && echo terminal || echo error' "${history|tail -n1|sed -e '\s/[0-9]\+\s*//;s/[;:&]\s*alert$//'\''}'

# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
    if [ -f /usr/share/bash-completion/bash_completion ]; then
        . /usr/share/bash-completion/bash_completion
    elif [ -f /etc/bash_completion ]; then
        . /etc/bash_completion
    fi
fi

# NVIDIA CUDA Toolkit
export PATH=/usr/local/cuda-11.0/bin${PATH:+:${PATH}}
export LD_LIBRARY_PATH=/usr/local/cuda-11.0/lib64${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}
```

- Press Ctrl + x, y and Enter to save changes.
- Next, run the source command update your current shell environment.

```
source ~/.bashrc
```

- That's it! We have successfully installed CUDA on our system. Run sudo reboot to restart your system for the new changes to take effect.

```
sudo reboot
```

```
sudo reboot
```

- After ubuntu restarts, you can confirm your CUDA installation by running nvcc -V and nvidia-smi commands.

```
nvcc -V
nvidia-smi
```

```
zizou@zizou-Alienware:~$ nvcc -V
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2020 NVIDIA Corporation
Built on Wed_Jul_22_19:09:09_PDT_2020
Cuda compilation tools, release 11.0, V11.0.221
Build cuda_11.0_bu.TC445_37.28845127_0
zizou@zizou-Alienware:~$
```

4. Download and setup CUDNN

- Go to <https://developer.nvidia.com/cudnn> to download the latest version of CUDNN for the latest CUDA toolkit version OR go to <https://developer.nvidia.com/rdp/cudnn-archive> to download a previous version of CUDNN that is compatible with your CUDA toolkit.

NOTE: You have to be signed in using your Nvidia account to download CUDNN. If you do not have an account, create one.

Since I have CUDA 11.0.1, I will download cuDNN 8.0.5 for CUDA 11.0

- Copy the downloaded cuDNN zip file to the installers folder.
- Unzip the cuDNN zip file using the following command. You will see a folder named cuda with include and lib64 sub-folders in it after unzipping this.

```
tar -zxvf cudnn-11.0-linux-x64-v8.0.5.39.tgz
```

- We need the contents of the include & lib64 folders from cuDNN to be inside the include and lib64 folders of CUDA directory (where we installed CUDA shown in step 1 above). Run the following commands.

```
cd cuda
sudo cp -P lib64/* /usr/local/cuda/lib64/
sudo cp -P include/* /usr/local/cuda/include/
```

5. Install OpenCV from Ubuntu Repository.

- The following command using apt install will install an older version of OpenCV. Right now the available version using this is at 3.2 while the official OpenCV release is at 4.5.3.

```
sudo apt update
sudo apt install python-opencv
sudo apt install libopencv-dev
```

- OR use pip install. This will install the latest OpenCV release on your system. You need to have pip installed on your system. (Use pip3 if you have python3 and pip3 separately installed along with python2.)

```
pip install opencv-python
OR
pip3 install opencv-python
```

6 . Make changes in the Makefile

- Make changes in the Makefile according to your requirements. There are various changes you can make in the Makefile as mentioned

```
GPU=1
CUDNN=1
CUDNN_HALF=1
OPENCV=1
```

- Next, set the ARCH to your GPU Architecture. You can find your GPU architecture(compute capability) [here](#). For eg: The CC for my Nvidia GPU is 6.1 so I will use the following line:

```
ARCH= -gencode arch=compute_61,code=[sm_61,compute_61]
```

For more on installing CUDA and cuDNN Information refer the below Blog:

YOLOv4-darknet installation and usage on your system (Windows & Linux) - TECHZIZOU

CONTENTS (But first ✓ Subscribe to my YouTube channel 📌 <https://bit.ly/3Ap3sdi> 😊🙏)

Download Darknet zip-archive with the latest commit and uncompress it: master.zip Go to MSVC:
<https://visualstudio.microsoft.com/thank-you-downloading-visual-studio/?sku=Community>. This will

🔗 https://techzizou.com/yolo-installation-on-windows-and-linux/#install_linux

INSTALL YOLOV4-DARKNET ON
WINDOWS & UBUNTU



Install and run YOLOv4-Darknet on Linux(Ubuntu)

This video shows step by step tutorial on how to install and run yolov4-darknet on your local Linux system. ① ⚡ Website Blog post on this ⚡📌 <https://tec...>

📺 https://www.youtube.com/watch?v=5jmxjl-Pm6Q&ab_channel=techzizou



For custom train yolov4 follow the below Blog:

TRAIN A CUSTOM YOLOv4 OBJECT DETECTOR (Using Google Colab)

✓ Subscribe to my YouTube channel 🖱️ <https://bit.ly/3Ap3sdi> 😊 Open my Colab notebook on your browser. Click on File in the menu bar and click on Save a copy in drive. This will open a copy of my Colab notebook on your browser which you can now use.

📄 <https://medium.com/analytics-vidhya/train-a-custom-yolov4-object-detector-using-google-colab-61a659d4868>

