YOLO TUTORIAL

This tutorial is about installing YOLOv4 on Windows OS without Nvidia GPU (no CUDA).

All requirements for YOLO4 are available on GitHub Yolo4 project in README file: <https://github.com/AlexeyAB/darknet#requirements>

### Requirements

* Windows or Linux
* **CMake >= 3.12**: <https://cmake.org/download/>
* **CUDA >= 10.0**: <https://developer.nvidia.com/cuda-toolkit-archive> (on Linux do [Post-installation Actions](https://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html#post-installation-actions))
* **OpenCV >= 2.4**: use your preferred package manager (brew, apt), build from source using [vcpkg](https://github.com/Microsoft/vcpkg) or download from [OpenCV official site](https://opencv.org/releases.html) (on Windows set system variable OpenCV\_DIR = C:\opencv\build - where are the include and x64 folders [image](https://user-images.githubusercontent.com/4096485/53249516-5130f480-36c9-11e9-8238-a6e82e48c6f2.png))
* **cuDNN >= 7.0** <https://developer.nvidia.com/rdp/cudnn-archive> (on **Linux** copy cudnn.h,libcudnn.so... as desribed here <https://docs.nvidia.com/deeplearning/sdk/cudnn-install/index.html#installlinux-tar> , on **Windows** copy cudnn.h,cudnn64\_7.dll, cudnn64\_7.lib as desribed here <https://docs.nvidia.com/deeplearning/sdk/cudnn-install/index.html#installwindows> )
* **GPU with CC >= 3.0**: <https://en.wikipedia.org/wiki/CUDA#GPUs_supported>
* on Linux **GCC or Clang**, on Windows **MSVC 2017/2019** <https://visualstudio.microsoft.com/thank-you-downloading-visual-studio/?sku=Community>

I installed YOLO and dependencies in one folder called **TeamProjectTutorial** on C drive.

## Create folder TeamProjectTutorial on C drive.

## Install Python.

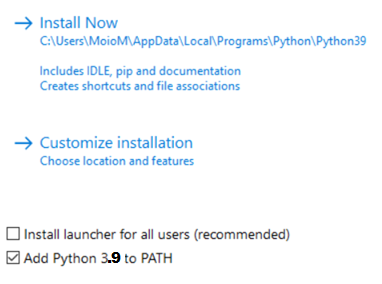
You can install python whenever you want. If you chose ‘TeamProjectTutorial’, add subfolder **python** in ‘TeamProjectTutoria’

Download Python for Windows.

<https://www.python.org/downloads/>

Run installer. Tick **Add Python 3.9 to PATH**. It will add Python to Windows ‘environment variables’. You can leave the rest of options as they are.

You can choose folder for Pyhton installation by choosing **Customize installation**. Just remember the folder name and path. It may be useful if you have multiple Python installation and would like to change PATH to use another version.



Check if Python is installed. Open command prompt and run command:

python --version



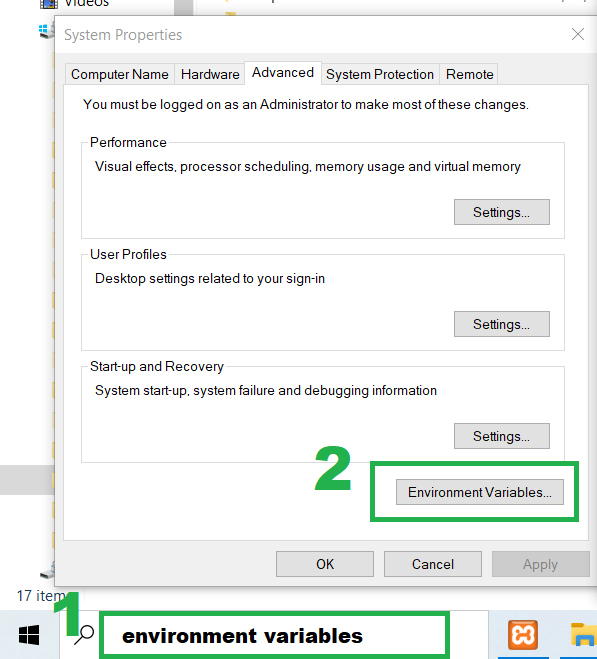
You should see a Python version. If not, check **Environment variables**.

## Environment variables on Windows – just in case you need it.

PATH variables define paths to programs executable, i.e., command-line uses them to find a path to a program you want to run.

Check if PATH to Python is added to User or System PATH variables.

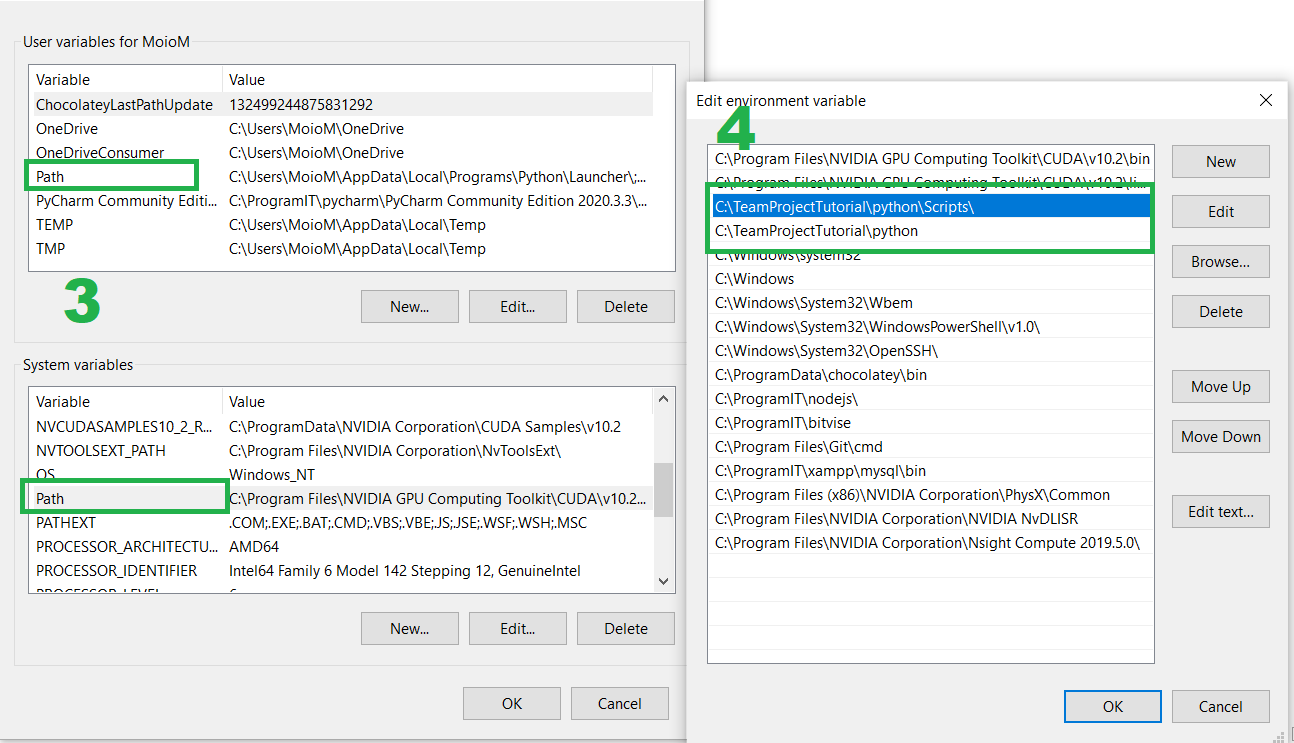
* Write ‘environment variable’ in search bar and chose ‘Edit the system environment variables. They are in control panel.
* Chose ‘Environment Variables’



* Select ‘Path’ in user or system variables. Edit.
* Add path to python executable and python Scripts. In my case:

C:\TeamProjectTutorial\python\Scripts\

C:\TeamProjectTutorial\python



## Numpy install.

Numpy is required by OpenCV.

Open command prompt and run command:

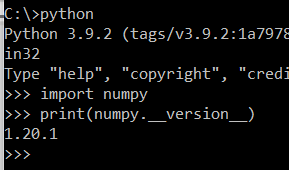
pip install numpy

You can check if Numpy is installed by running:

python

import numpy

print(numpy.\_\_version\_\_)



Physically this package is imported to C:\TeamProjectTutorial\**python\Lib\site-packages**

## Git installation.

Download and install version for Widows. Should not be any issue.

<https://git-scm.com/downloads>

Run command to make sure that it is installed correctly:

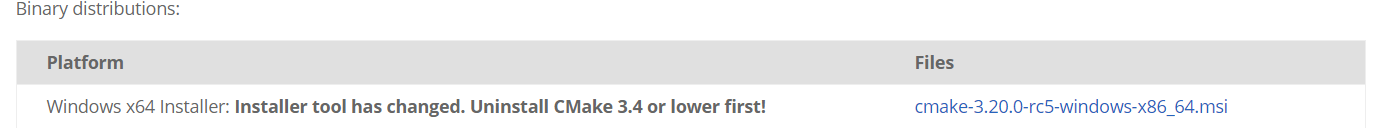
git --version



## CMake installation.

Download Windows version and install: **cmake-3.20.0-rc5-windows-x86\_64.msi**

<https://cmake.org/download/>



We need CMake to install OpenCV from source code. CMake is used to automatically build a project build files, that compiles source code. CMake is designed to be used in conjunction with the native build environment.

CMake uses CMakeList.txt file as a config. Those files are placed in each source directory. They are used to generate standard build files (e.g., projects/workspaces in Windows MSVC) which are used in the usual way.

## Visual Studio 2017/2019 installation.

Download and install **Visual Studio 2019 for Community**, which is free version.

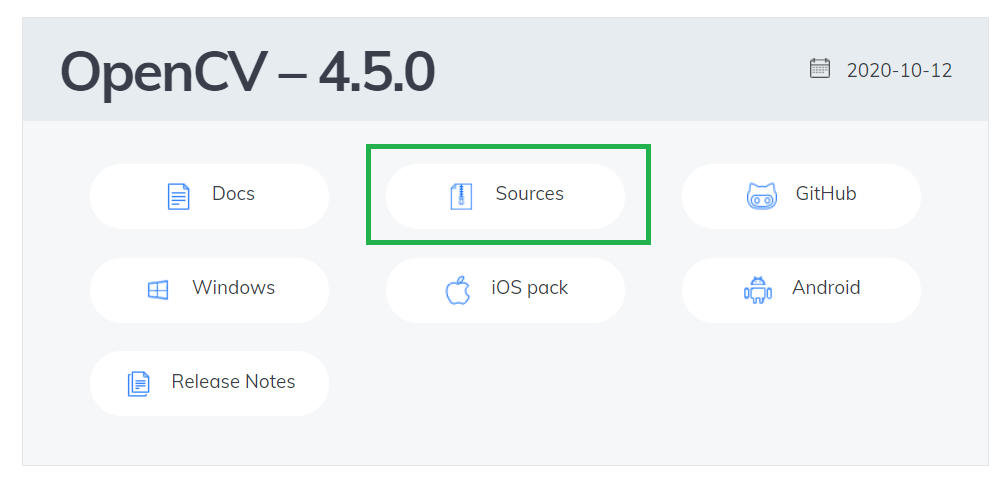
Visual Studio and Visual Studio Code are not the same.

<https://visualstudio.microsoft.com/>

Visual Studio 2019 is IDE. It also has a build-in C compiler, known as MSVC. We will use it to compile OpenCV and Yolo. Both are written in C.

## OpenCV installation.

Download source code, version 4.5.0 is stable: <https://opencv.org/releases/>



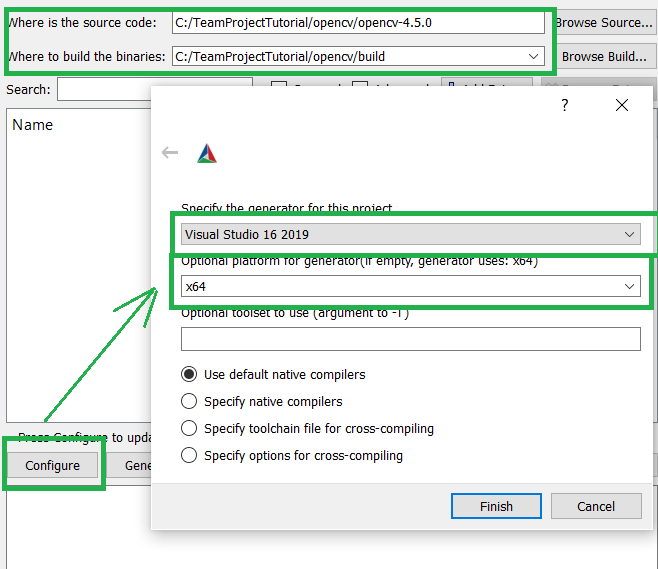
Create subfolder **opencv** in project folder. Move OpenCV zip file to that folder and unzip.

Create subfolder **build** in **opencv** folder.

Open CMake. The CMake icon should be in your Start menu.

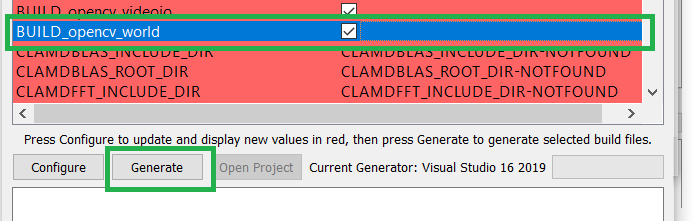
Write path to source and path to build folders. In my case it looks like below. Next press **Configure**. **Select Visual Studio 16 2019** and **x64**. Next click **Finish**.

Btw, you can see that there is ‘CMakeLists.txt’ in OpenCV source folder.



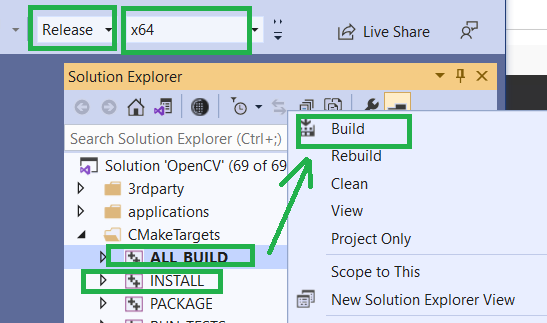
Find and tick **Build\_opencv\_world**. Then **Generate**.

Opencv\_world.lib contains all other modules (like core, highgui, etc). The "world module" is an all-in-one module that has the functionality of all the libraries.



Next go to C:\TeamProjectTutorial\**opencv\build** and run **ALL\_BUILD.vcxproj** file. It will open Visual Studio.

In Visual Studio set up **Release** and **x64**. Right click on **ALL BUILD** in CMakeTarget folder and run **Build**. Next right click on **INSTALL** and run **Build**.



OpenCV is a library for real-time computer vision. Or simply speaking, image manipulation, video processing. OpenCV allows to see object detection in real-time, as the YOLO program runs.

There is also OpenCV Python version. OpenCV-python is unofficial pre-built CPU-only OpenCV packages for Python. That is why it is better to use source code and build OpenCV. <https://pypi.org/project/opencv-python/> I did not try this version so I do not know if it works. I doubt as in following steps you have to copy OpenCV DLL files to YOLO folder.

## Check if OpenCV is installed.

Write command:

python

import cv2

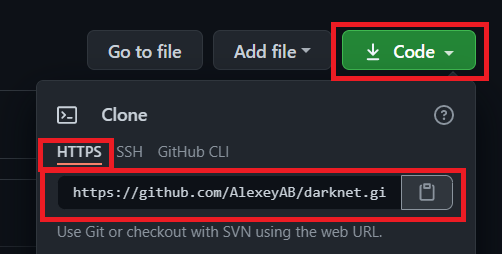
print(cv2.\_\_version\_\_)

## YOLO installation.

We will use git to clone project.

Link to clone Darknet project is on GitHub:

<https://github.com/AlexeyAB/darknet>



Open command prompt and navigate to project folder.

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

## Copy OpenCV DLL files into YOLO build folder.

Go to C:\TeamProjectTutorial\**opencv\build\bin\Release**

Copy two files:

**opencv\_videoio\_ffmpeg450\_64.dll**

**opencv\_world450.dll**

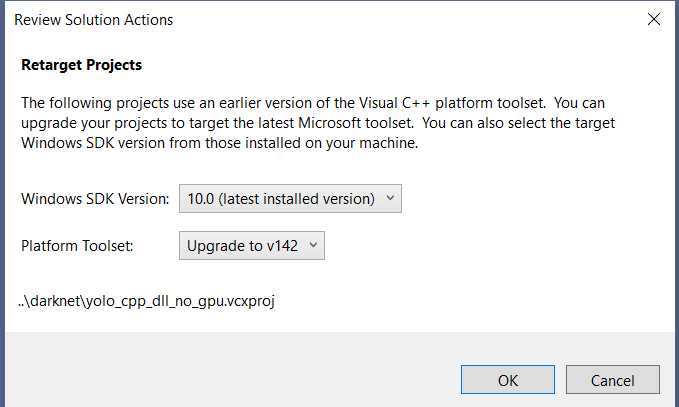
into C:\TeamProjectTutorial\**darknet\build\darknet\x64**

## Build YOLO.

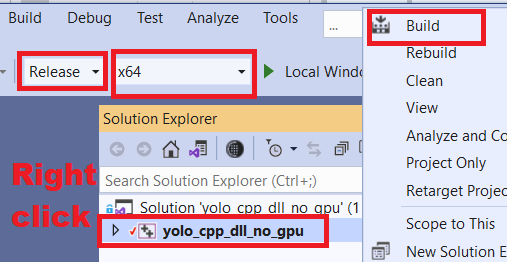
Go to C:\TeamProjectTutorial\**darknet\build\darknet**

Open file **yolo\_cpp\_dll\_no\_gpu.vcxproj**

Click ok if you see the following window:



Select **Release** and **x64**. Next right click on **yolo\_cpp\_dll\_no\_gpu** and click **Build**.



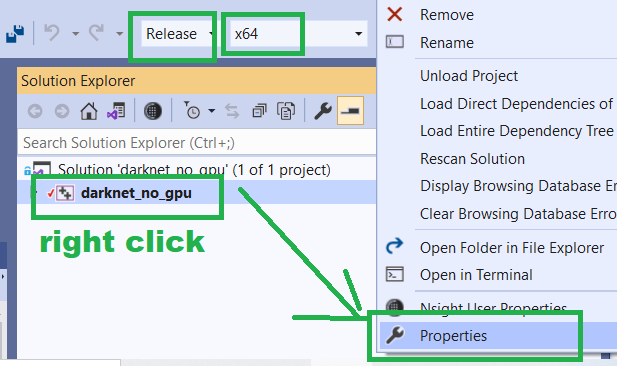
## Build Darknet.

Go to C:\TeamProjectTutorial\**darknet\build\darknet**

Open file **darknet\_no\_gpu.vcxproj**

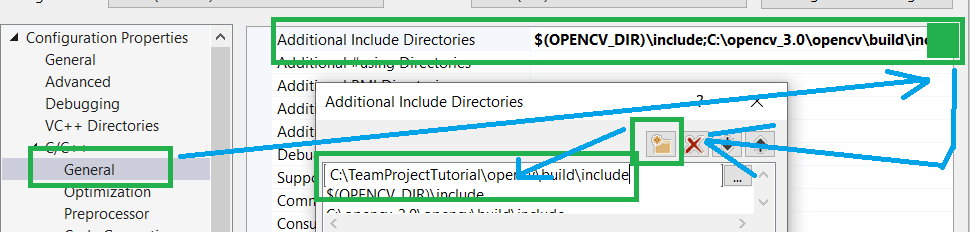
Select **Release** and **x64**. Next right click on **darknet\_no\_gpu** and click **Properties**.

Make sure that ‘Release’ and ‘x64’ are selected.



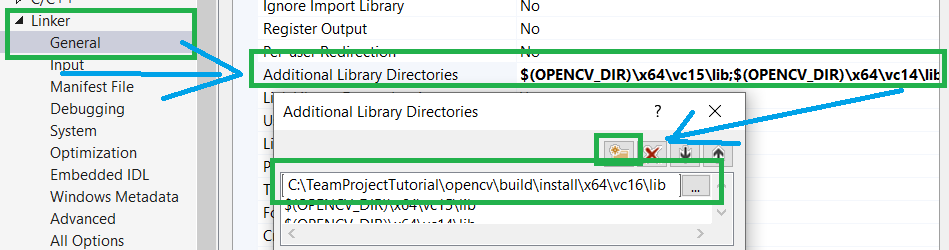
Add path to OpenCV ‘include’ folder into Configuration properties -> **C/C++** -> **General** **-> Additional Include Directories**. There is a small button on the right, highlighted in green colour on image, that you can use to open ‘Additional Include Directories’ window. Next you can click on Folder icon to open new line where you can add path.

In my case path is: C:\TeamProjectTutorial\**opencv\build\install\include**

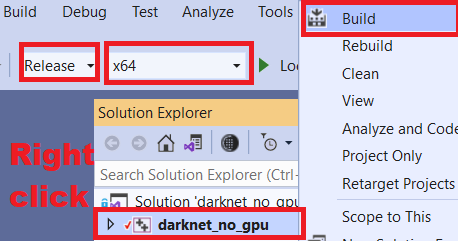


Add path to OpenCV ‘lib’ folder into Configuration properties -> **Linker** -> **General** -> **Additional Library Directories**.

In my case Path is: C:\TeamProjectTutorial\**opencv\build\install\x64\vc16\lib**

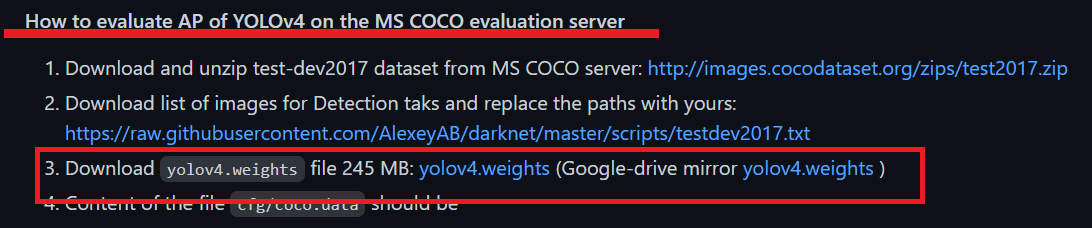


Next again right click on **darknet\_no\_gpu** and select **Build**.



## Download yolo weights from GitHub.

Use the link below. If it does not work, you can find weights on darknet GitHub in README file under ‘How to evaluate …’ topic.



<https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.weights>

Move weight to this folder:

C:\TeamProjectTutorial\**darknet\build\darknet\x64**

## Run YOLO.

Open command prompt in C:\TeamProjectTutorial\**darknet\build\darknet\x64** folder.

Run the below command:

darknet\_no\_gpu.exe detector test cfg/coco.data cfg/yolov4.cfg yolov4.weights

It should run darknet. It will load weights at first. Next it will ask for a path to image. There are some test images in the data folder, so you can use them. Here is an example path:

C:\TeamProjectTutorial\**darknet\build\darknet\x64\data\dog.jpg**

## Run YOLO with your camera.

Open command prompt in C:\TeamProjectTutorial\**darknet\build\darknet\x64** folder.

Run the below command:

python darknet\_video.py

## Run our program.

Go to C:\TeamProjectTutorial\**darknet\build\darknet\x64\results** and create folder **output** (no capital letters).

Download movie to **data** folder: C:\TeamProjectTutorial\**darknet\build\darknet\x64\data**

Download script from GitHub: darknet\_video\_team.py

Move script to C:\TeamProjectTutorial\**darknet\build\darknet\x64**

Open command prompt in C:\TeamProjectTutorial\**darknet\build\darknet\x64**

Run command:

python darknet\_video\_team.py

You can find result, which are images, in folder: C:\TeamProjectTutorial\**darknet\build\darknet\x64\results\output**

Enjoy 😊