Requirement :

* UNBUNTU 18.04
* CUDA 10.2
* CUDNN 8
* TENSORRT 7.1.3
* OPENCV 4.1
* nano
* yaml-cpp 0.5.2 (apt install libyaml-cpp-dev)
* eigen3 (apt install libeigen3-dev)
* unzip (apt install unzip)
* cmake 3.20

**1-Only If cmake version below 3.1**

apt remove cmake

wget <https://github.com/Kitware/CMake/releases/download/v3.20.3/cmake-3.20.3.tar.gz>

tar -xzf cmake-3.20.3.tar.gz

rm cmake-3.20.3.tar.gz

cd c cmake-3.20.3

./bootstrap

make

make install

#check that installation has been successful

cmake --version

**2-clone this repo**

git clone https:/github.com/jouvencia/tkDNN.git

**3- Copy your Custom file**

cd tkDNN/tests/darknet/cfg/

copy the cfg file of your custom\_yolo in this folder

cd ../names/

copy your custom\_yolo.names file inside this folder

cd ..

mkdir weights

cd weights

copy your custom\_yolo.weights file here

cd ..

here duplicate the yolo4.cpp and rename it with your custom\_yolo name

open this file with nano and edited like this



Exit nano

**4\_compile the repo**

cd ../.. (you should be in tkDNN folder)

mkdir build

cd build

cmake ..

make -j4

**5\_ create all the layers needed to construct the engine**

mkdir a folder and name it as in the yolo\_custom.cpp (the name you first replace in this file)

cd ..

should be again in /tkDNN folder

git clone https://git.hipert.unimore.it/fgatti/darknet.git

cd darknet

make

mkdir layers debug

./darknet export <path-to-cfg-file> <path-to-weights> layers

mv folders layers and debug inside the /tkDNN/build/custom\_name/

**6\_create the rt. Engine file**

go back to /tkDNN/build/

you should see an executable file called “test\_<custom\_name>”

launch it with the command:

./ test\_<custom\_name>

This will create the engine .rt file corresponding to your model. Sometimes the process can take a while but if it’s exceed 10 min I usually abort it (ctrl C) and relaunch ./test\_<custom\_name>

It is a bit strange but a successful process should end with an aborting message. If the .rt file is in the build folder then it’s ok.

By default this will create an engine with FP32 weights

In case you wanted to use FP16 (which will reduce the memory consumption)

First rm all the \*.rt file (or move it elsewhere)

In the build folder:

export TKDNN\_MODE=FP16

./test\_<custom\_name>

To work, you just need the .rt file, the darknet.RT.so library and of course the wrapper python (darknet\_RT.py)