Python framework for object detection and classification:

We will be calling your Python code from a C++ program, so your code needs to conform to certain standards.

1) Libraries
    a. Use ONLY the following library versions
        i. Python: python 3.6
        ii. CUDA and cudnn: cuda 10.1 cudnn 7.6.5
        iii. ML platform: PyTorch 1.7.0 (IF AND ONLY IF PyTorch is not supported by a particular model, use a TensorFlow version that can work with CUDA 10.1 AND Python 3.6)

2) Prepare your Python main script (see example "detector.py" which was used with YOLOv5):
    a. Define your class using iClass: Note that "iClass" is a keyword in our C++ framework, so do not change it. Package all your code in iClass.

    b. Define your "initialize" model: Initialize your model, device, and input image size in the "initialize" function. Our C++ program will run your "initialize" at the beginning. The "initialize" is also a keyword in our C++ framework, so do not change it.

    c. Define your "detect" function: The "detect" function is the primary function to process frame/image. The input to this function should be the frame and any necessary parameters (e.g., confidence, IOU threshold for non-max suppression). The output should be the bounding box, confidence, category, and mask (if supported). The "detect" is also a keyword in our C++ framework, so do not change it.

3) To ensure that your Python code works within our C++ program
    a. Provide your python main script file

    b. Document the required dependencies (name and version)

    c. Document the input and output specifications of "initialize" and "detect" functions
        i. Input format:
            1. Channel: BGR (if you need a different channel, let's discuss)
            2. Size: Original input size. You need to resize the image to fit the input size of your model.
        ii. Output format:
            1. Coordiante: Transform/resize all resulting coordinates to match the size of original inpu image.
            2. Format: Use a list to output multiple detection results. For each detection, provide (bounding box, confidence, class/category).

3. Bounding box format: Provide (x left-top corner, y left-top corner, x right-bottom corner, y right-bottom corner).

d. Include your pre-trained model and any documentation/reference

e. Provide all your Python codes to run

f. Provide a README.txt file to document your test environment
   i. Python version
   ii. Linux/Windows
   iii. CUDA version
   iv. The cudnn version

g. Provide the Github link if you used any code from online open-source libraries. DO NOT USE ANY LIBRARY/MODEL WITH GPL OR A SIMILARLY RESTRICTIVE LICENSE.