This document is mainly meant to just give you a general idea on how our detection code works with the YOLOv5 model, when concerning areas that we redefined from the base **detect.py** code provided by YOLOv5.

Within both our **KABML_server.py** code, and YOLOv5's **detect.py** code, certain values have to be defined prior to the loading of the YOLOv5 model. Within the **KABML_server.py** code, these values are defined here:

```
def parse_opt():
    parser = argparse.ArgumentParser()
    parser.add_argument('--weights', nargs='+', type=str, default=ROOT / 'best415.pt', help='model path(s)')
    parser.add_argument('--source', type=str, default=ROOT / 'data/tcp', help='file/dir/URL/glob, 0 for webcam')
    parser.add_argument('--data', type=str, default=ROOT / 'data/coco128.yaml', help='(optional) dataset.yaml path')
    parser.add_argument('--imgsz', '--img', '--img-size', nargs='+', type=int, default=[640], help='inference size h,w')
    parser.add_argument('--conf-thres', type=float, default=0.35, help='confidence threshold')
    parser.add_argument('--iou-thres', type=float, default=0.45, help='NMS IoU threshold')
    parser.add_argument('--max-det', type=int, default=1000, help='maximum detections per image')
    parser.add_argument('--device', default='cpu', help='cuda device, i.e. 0 or 0,1,2,3 or cpu')
    parser.add_argument('--view-img', action='store_true', help='show results')
    parser.add_argument('--save-txt', action='store_true', help='save results to *.txt')
    parser.add_argument('--save-conf', action='store_true', help='save confidences in --save-txt labels')
    parser.add_argument('--save-crop', action='store_true', help='save cropped prediction boxes')
    parser.add_argument('--nosave', action='store_true', help='do not save images/videos')
    parser.add_argument('--classes', nargs='+', type=int, help='filter by class: --classes 0, or --classes 0 2 3')
    parser.add_argument('--agnostic-nms', action='store_true', help='class-agnostic NMS')
    parser.add_argument('--augment', action='store_true', help='augmented inference')
    parser.add_argument('--visualize', action='store_true', help='visualize features')
    parser.add_argument('--update', action='store_true', help='update all models')
    parser.add_argument('--project', default=ROOT / 'runs/output', help='save results to project/name')
    parser.add_argument('--name', default='', help='save results to project/name')
    parser.add_argument('--exist-ok', action='store_true', help='existing project/name ok, do not increment')
    parser.add_argument('--line-thickness', default=3, type=int, help='bounding box thickness (pixels)')
    parser.add_argument('--hide-labels', default=False, action='store_true', help='hide labels')
    parser.add_argument('--hide-conf', default=False, action='store_true', help='hide confidences')
    parser.add_argument('--half', action='store_true', help='use FP16 half-precision inference')
    parser.add_argument('--dnn', action='store_true', help='use OpenCV DNN for ONNX inference')
```

The main values that we focused on for this project were:

- ***--weights***
- ***--source***
- ***--imgsz, --img, --img-size***
- ***--conf-thres***
- ***--device***
- ***--project***

Due note though, while these values are defined in the code itself, they can also be defined/overwritten within the Command Prompt/Terminal as such:

- **python detect.py --source data/images --weights yolov5s.pt --conf 0.25 --img-size 1280 --device 0**

Why we focused on the ***--weights*** value?

- To simplify it, “weights” are what we use to identify objects within an image. In our case, it’s instances of litter in Google Street images.
- These “weights” are produced after training the YOLOv5 model on a dataset of your choosing. We trained the model on a dataset of Google Street images, which contained only annotations that identified pieces of litter from a street view.
- As such, when we define the ***--weights*** value as our now produced weights, the detection code will now only be able to identify instances of litter in Google Street images.

Why we focused on the ***--source*** value?

- The “source” is the Folder or Image in which the detection code will look at to run an inference on.
- A Folder “source” is defined as such:
    - ‘data/tcp’
    - When a Folder is defined, the detection code will go through ALL IMAGES inside the Folder to run an inference on.
- An Image “source” is defined as such:
    - ‘data/tcp/image.jpg’
    - When an Image is defined, the detection code will go through just that ONE IMAGE to run an inference on.

Why we focused on the ***--imgsz, --img, --img-size*** value?

- Depending on the size/quality of the images we trained with in the YOLOv5 model, the “image size” is needed to reformat our images we want to run inferences on to be as similar as possible, in size/quality, as our training images.
    - If we trained on images that had 640x640 dimensions, the objects on them will have less pixels to help with their quality/definition.
    - So, running an inference on a 1280x720 image will not give back the best results, nor utilize the trained model to the best of its capabilities.
- Additionally, the “image size” value reduces inference time proportionally to the amount of letterboxed area padded onto a square image vs a 32-minimum multiple rectangular image.
    - More insight here: https://github.com/ultralytics/yolov3/issues/232

Why we focused on the ***--conf-thres*** value?

- Depending on the results of one's "weights" produced from training, how confident the YOLOv5 model can detect an object may vary. It's up to the "confidence level" to set a threshold based on those results, so that the model doesn't mistakenly detect an object with a low confidence, and gives you those results in your output.
    - Example: The model detects that a tree is a person with 9% confidence level. If you set the ***--conf-thres*** to .10, or 10%, that tree being detected as a person won't appear in your output.

Why we focused on the ***--device*** value?

- This value chooses on whether the model will use your CPU or GPU(s) to run. Depending on your system, you must choose which "device" value will work best.
    - 'cpu' represents your CPU
    - '0, 1, 2, 3' will represent your GPUs. '0' will represent just one GPU, while '0, 1' will represent two GPUs, and so on.

Why we focused on the ***--project*** value?

- Simply put, this value represents where the output of your inferences will be stored.
- So, if the value is defined as 'runs/output', your results will be stored in the 'output' folder located in the 'runs' folder.

Again, these are the main values that we focused on for this project. If you ever find yourself doing this project over, or starting a new one in a similar fashion, look at changing these values accordingly to your needs.