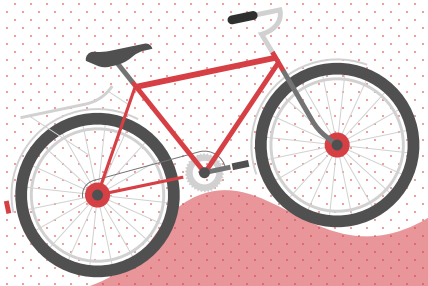
PHYSICS 301

Advanced Signal and Image Processing

ACTIVITY 06

finding objects & scenes in video



Rene L. Principe Jr.
2015-04622

Dr. Maricor N. Soriano

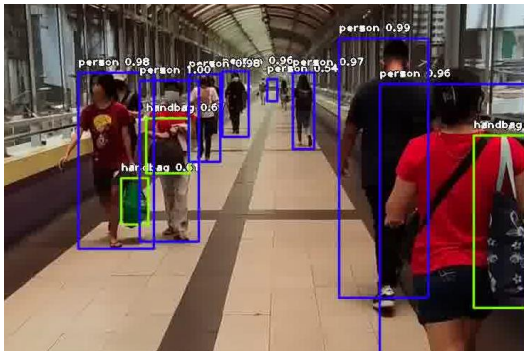
objectives

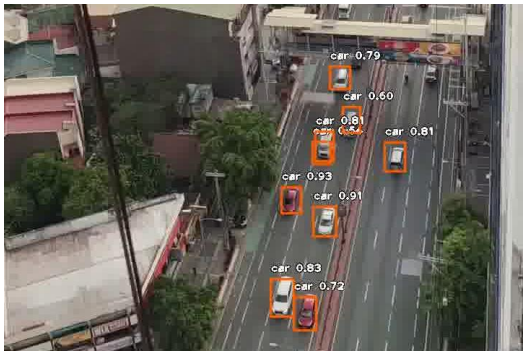
Open and write videos using Matlab/Python

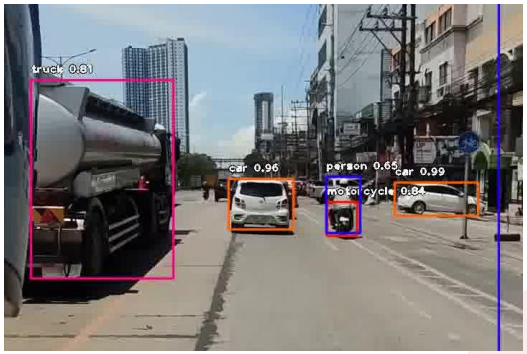
Use pretrained models to identify objects in a streaming video

key results

- Detecting and classifying objects in a video using YOLO





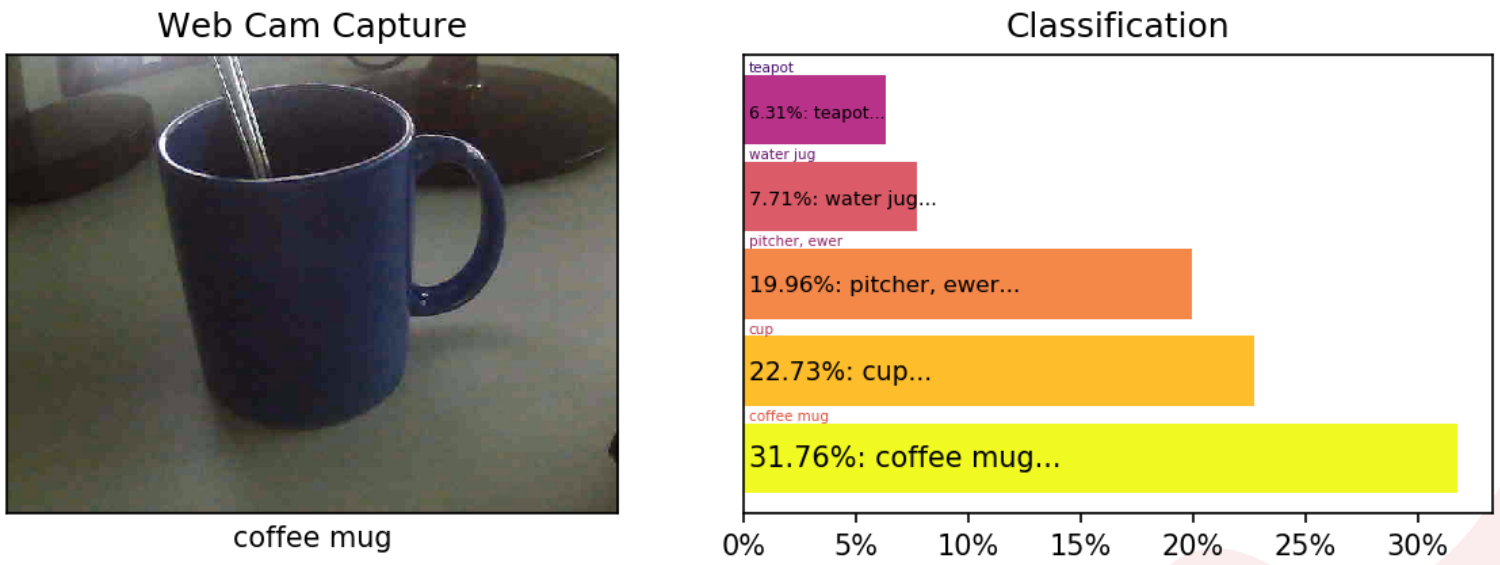


SOURCE CODE

- Physics-301/Activity 6 - Finding Objects and Scenes in Video.ipynb at main · reneprincipejr/Physics-301 (github.com)
- https://drive.google.com/file/d/13Iea5CgVMoUqUAX9ISZNh7PAHcre43vS/view?usp=sharing

Classification using NNs

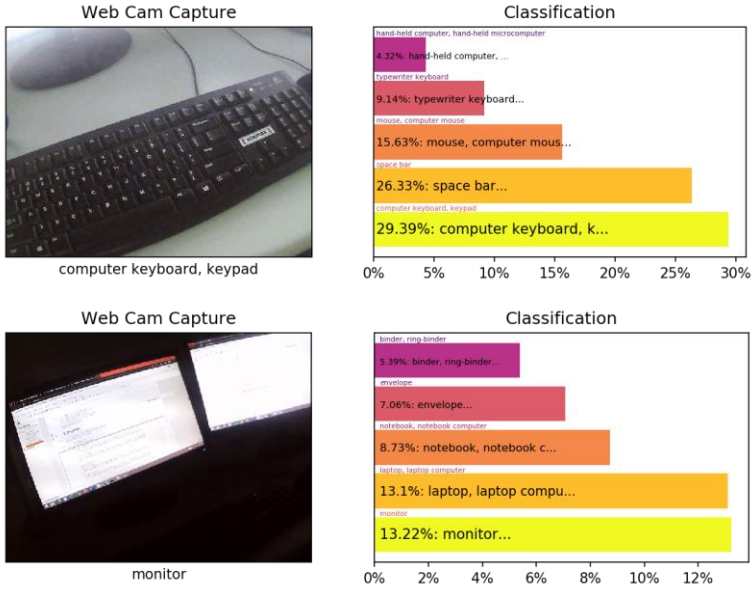
The rationale of this activity is to **classify objects using pre-trained deep learning models**. The implementation on MATLAB was relatively easier and much more straight-forward as these trained classifier models can be loaded in one line of code [1]. Instead, we use the equivalent tools in Python, namely **Torchvision to classify any input image** [2]. We used the webcam's input feed and then displayed the top 5 classifications along with their respective probabilities. Here's a sample result:

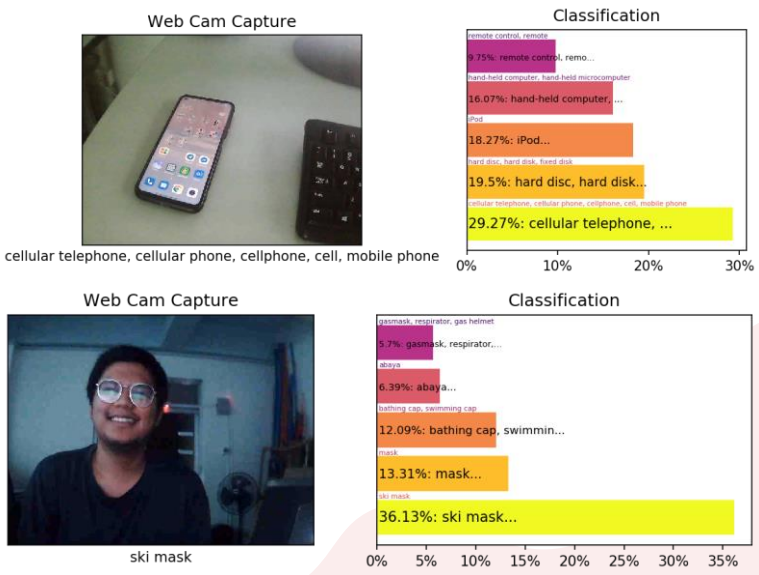


AlexNet

More results displayed below show that **AlexNet had a decent prediction** but in general, the probabilities of the top five classes had small difference. In fact, slight camera movements sometimes change the prediction.

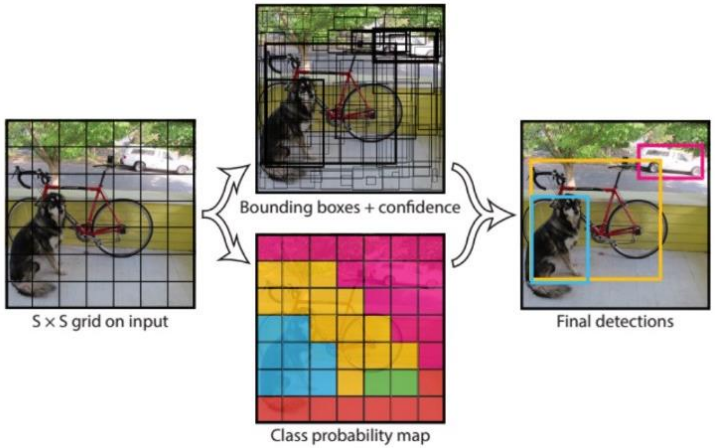
Together with GoogleNet and MobileNetV2, they were mostly trained on the ImageNet dataset of around 15 million images of 22,000 classes and so, we expect a relatively similar performance [1]. Next, **we now explore other contributed deep learning NNs for object detection**.

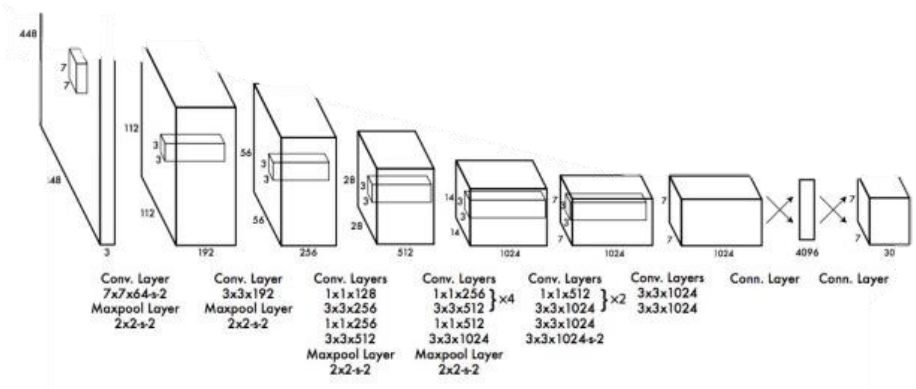




Contributed Deep NNs

In this work, we used **YOLO (You Only Look Once)**, a state-of the-art detection algorithm which automatically bounds boxes on the objects of interest and classify them in real-time using a Full-CNN [3].



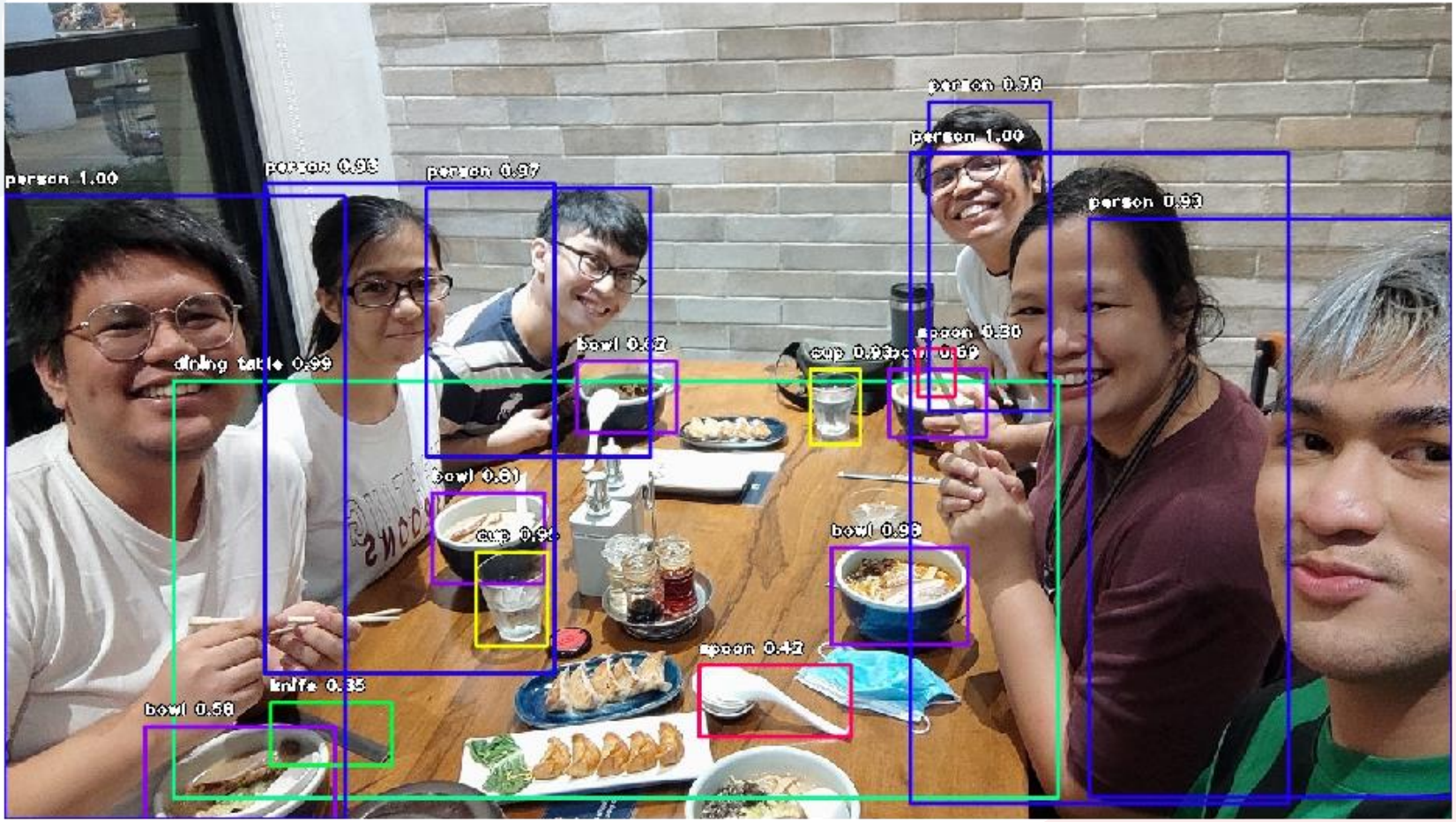


Images from: [3] [1506.02640] You Only Look Once: Unified, Real-Time Object Detection (arxiv.org)

We implemented the algorithms using the imageai.detection package in Python [4]. Pre-recorded video files were fed on the the pre-trained YOLO and Tiny-YOLO; both of which can classify 80 common objects [4]. For sanity check, we visualized the object detection on still images first as shown in the next page.

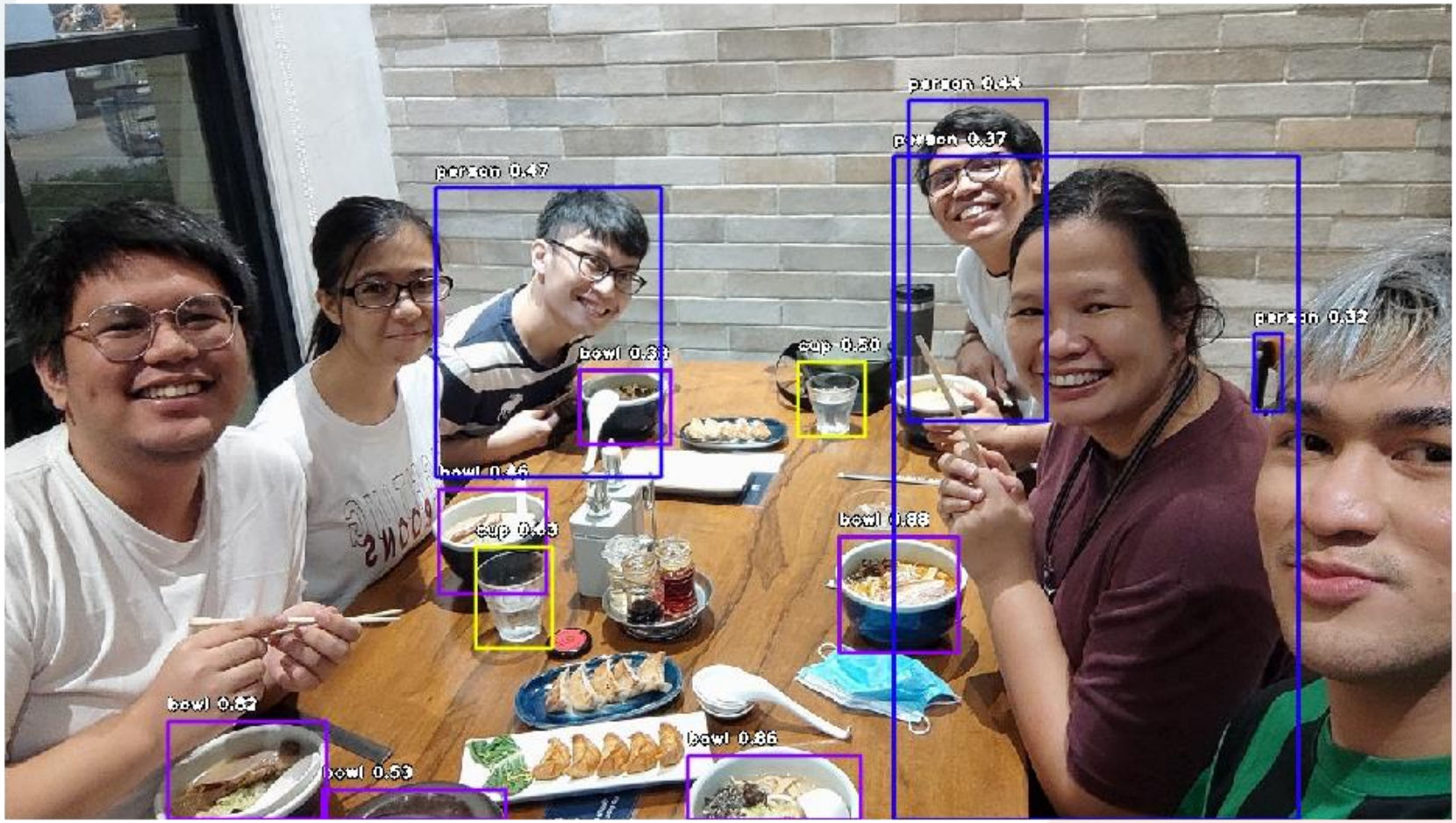
YOLO

YOLO was able to accurately detect and classify objects even with obstructions in less than 38 seconds.



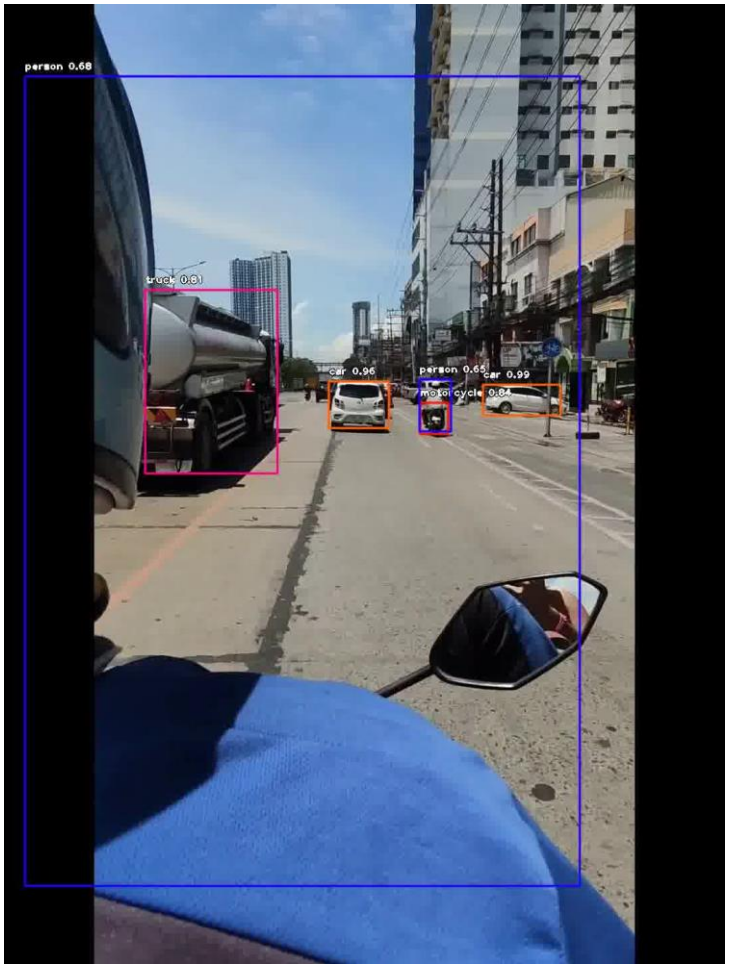
Tiny-YOLO

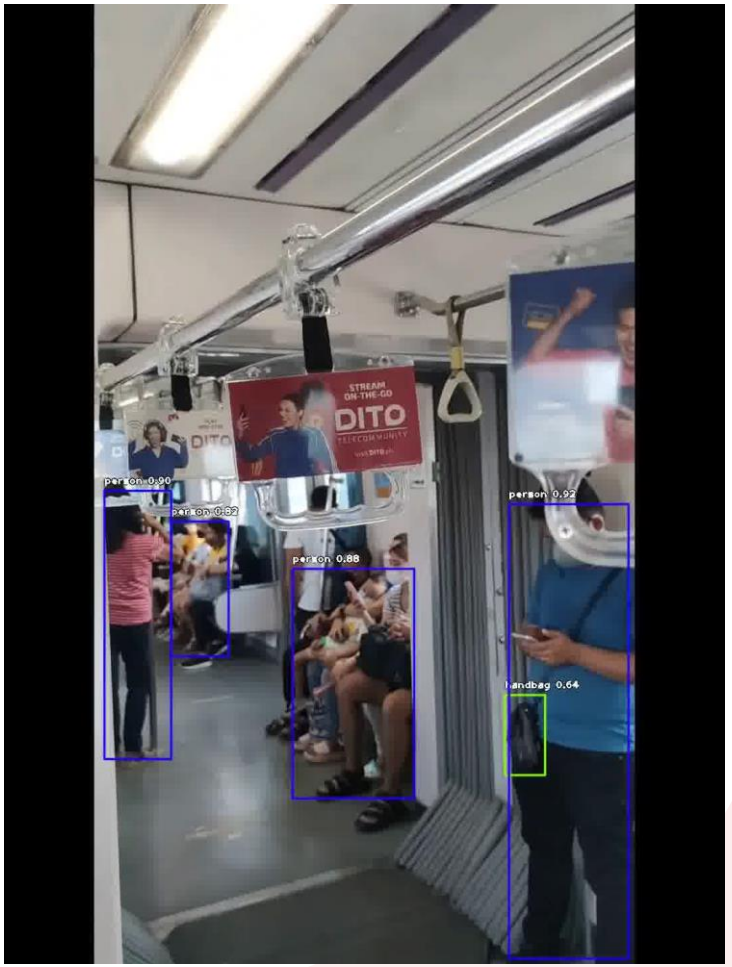
Tiny-YOLO is a smaller but faster version of YOLO. However, the trade-off for the detection speed (21.8s) is the lower detection accuracy.



Video Object Detection

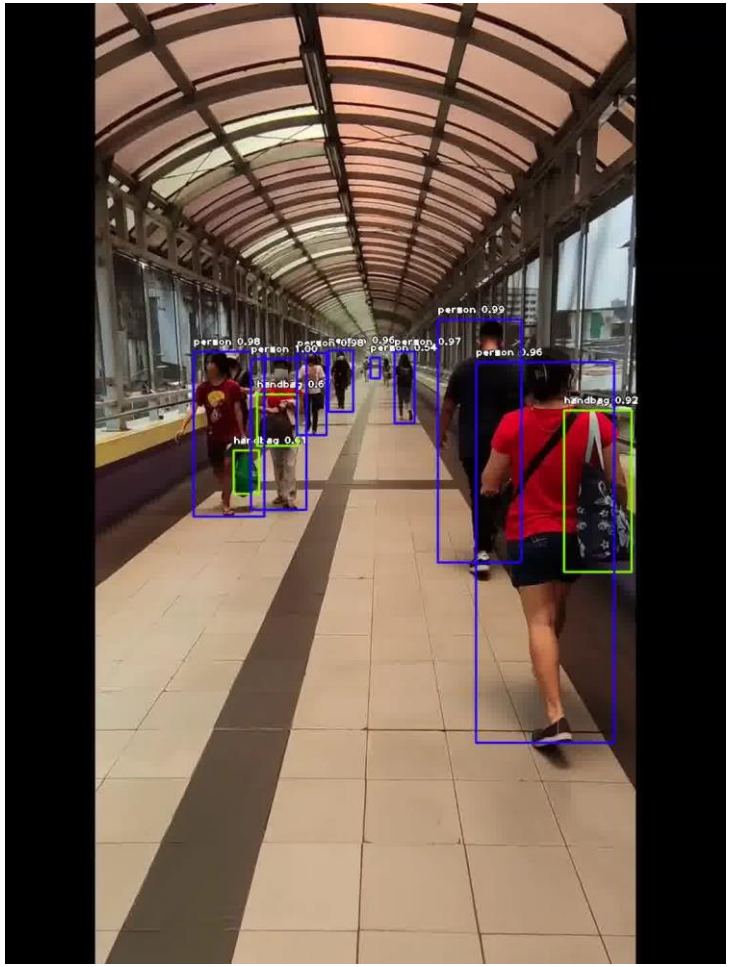
Shown below are some snapshots of the object detection results using outdoor footages that I took recently using my phone.

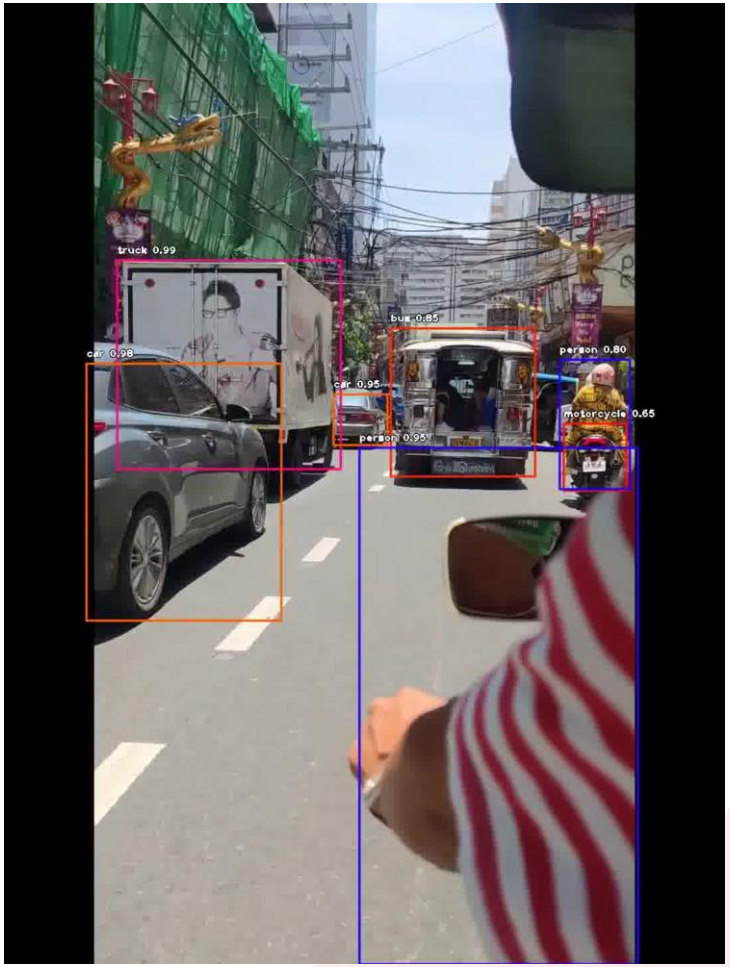




Video Object Detection

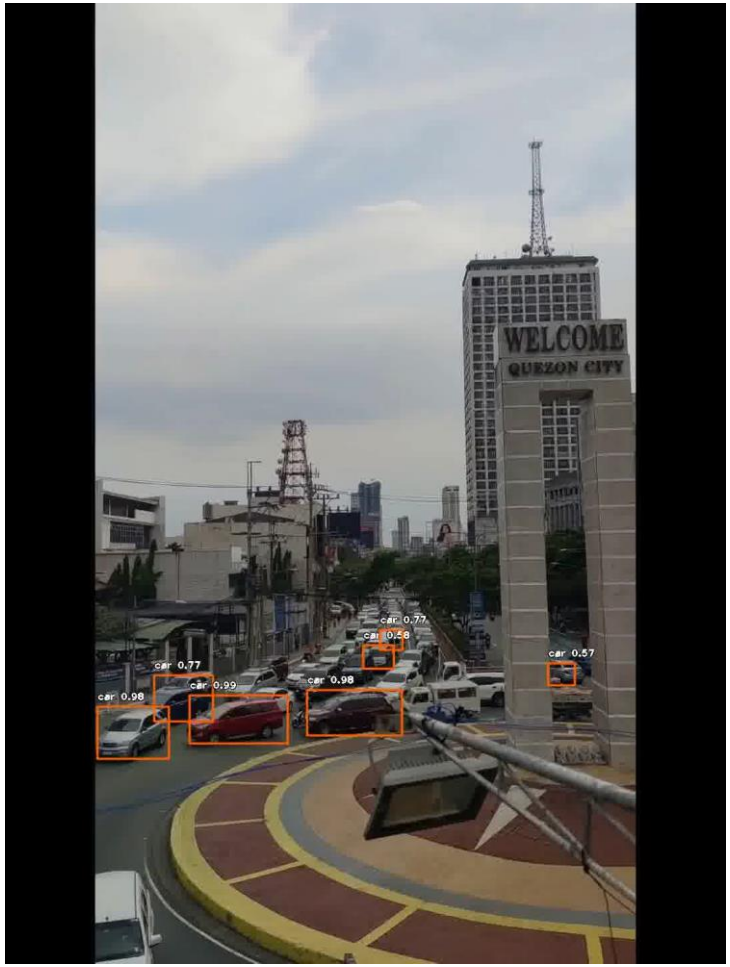
The probabilities are quite high. Small handbags, persons at different distances, and moving motorcycles were successfully detected.

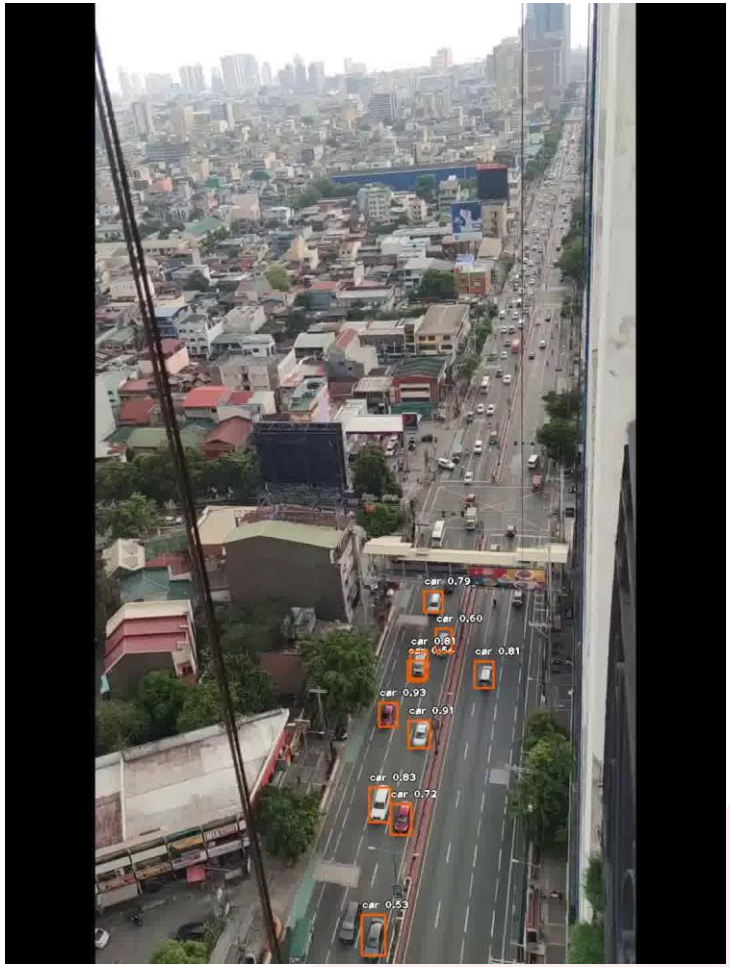




Video Object Detection

Interestingly, it can still detect and classify the cars regardless if the videos are taken on a lateral view or on a bird's eye view from a building.





reflection

Before doing this activity, I was already familiar with YOLO, and I was aware of its capacity to detect and classify objects with high accuracies. I implemented AlexNet first on Matlab, figured out its equivalent on Python, then went straight onto using YOLO and the results did not disappoint. Comparing the methods here to the previous activity which tracks objects based on color, it's a thrilling experience to exploit the power of deep learning to facilitate object detection. To sum up the take-aways from lessons that I learned in the past six activities so far, it's all about pattern recognition. By recognizing and calculating for patterns in our data, we were able to carry out signal recovery, compression by PCA, and video analysis through deep learning. There's so much to do with the tools that I used in this activity, and I honestly can't wait to check them out soon.

With that said, I'd give myself a score of **95/100.**

references

- [1] M. Soriano, Physics 301 – Video Processing – Finding Objects and Scenes, (2022).
- [2] torchvision — Torchvision 0.12 documentation (pytorch.org)
- [3] [1506.02640] You Only Look Once: Unified, Real-Time Object Detection (arxiv.org)
- [4] Detection Classes — ImageAI 2.1.6 documentation