Pedestrian Detection

Jenny Rong, Xuweiyi Chen

CHEMICAL ENGINEERING
UNIVERSITY *of* WASHINGTON

Background

Pedestrian detection is a key problem in computer vision, with several applications including robotics, surveillance, and automotive safety. Some challenges include detecting various styles of clothing in appearance, different possible articulations, the presence of occluding accessories. Some existing approaches including Holistic detection, Part-based detection, and Patch-based detection. However, as we are mainly focusing on the application of this technique to self-driving cars, a key challenge is to perform the detection instantaneously through the camera. Thus, to approach the solution, we worked on applying the **YOLO v3** (you only look once) object detection technique in solving the problem of detecting pedestrians through a self-driving car camera in an urban environment.

Main Goal: train a CNN model that inputs a pedestrian image and outputs an image with bounding boxes around a person or groups of people.

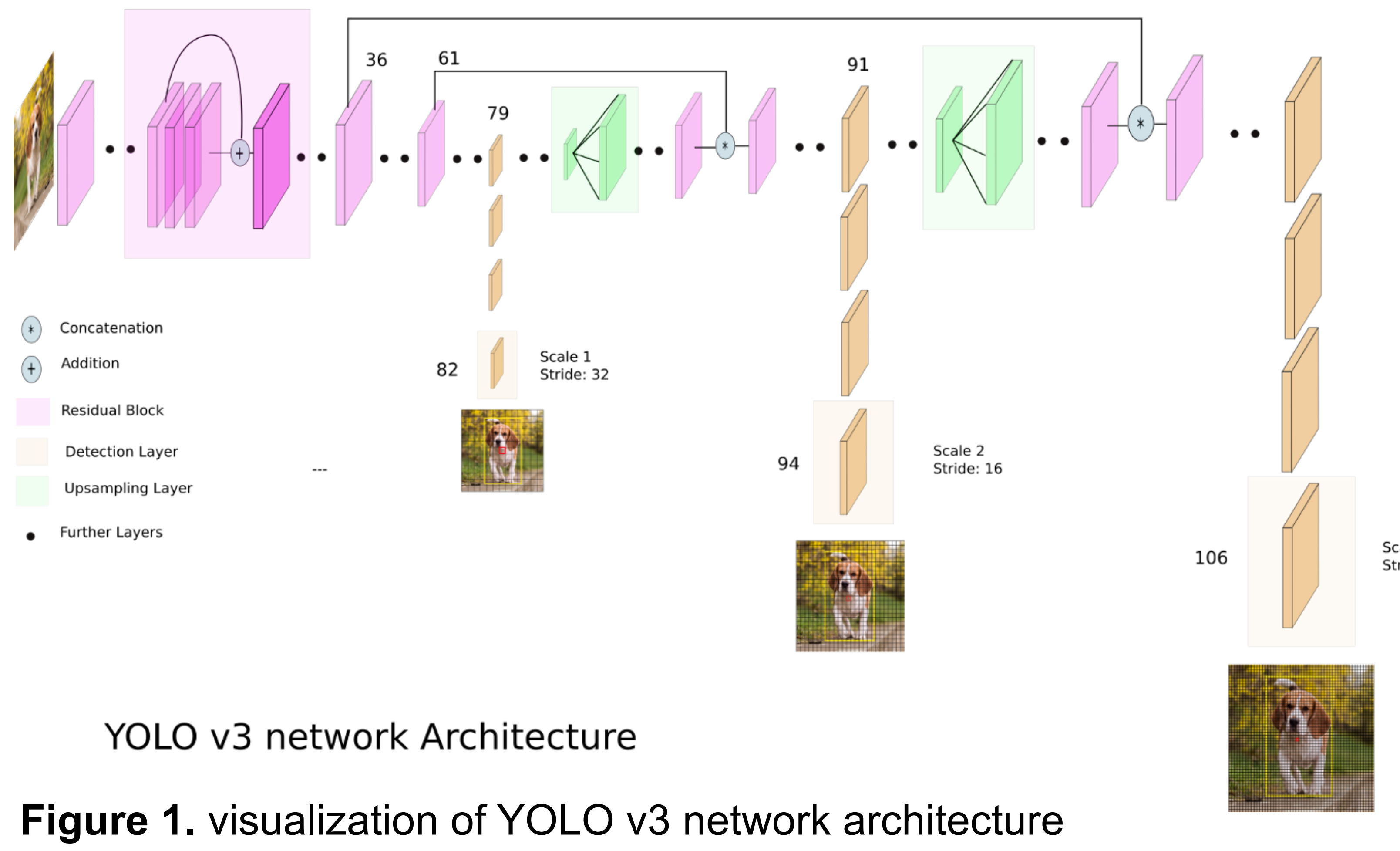

Figure 1. visualization of YOLO v3 network architecture

YOLO v3 is the state of the Art of object detection. It‘s able to instantaneously detect multiple objects in an image. At 320 × 320 YOLOv3 runs in 22 ms at 28.2 mAP, as accurate as SSD but three times faster. The main techniques that YOLO uses include Bounding Box Prediction, Class Prediction, Predictions Across Scales, Intersection Over Union, Non-max Suppression, Anchor Boxes, and it used darknet-53 as the feature extractor. Comparing to previous versions of the YOLO algorithm, YOLO v3 is using more convolutional layers to detect images with different scales. It’s now able to perform performs multilabel classification at a faster speed.

Methods

We started by looking for the existing pedestrian datasets. The first dataset we found was the Caltech pedestrian data set. It consists of 10-hour 640x480 video taken from a vehicle driving through regular traffic in an urban environment. It contains about 250,000 frames with a total of 350,000 bounding boxes were annotated. The annotation includes temporal correspondence between bounding boxes and detailed occlusion labels. And we first converted the video into frames and images that can be used for training in the model.

We implemented the neural network and trained the model using PyTorch in Google Colab. Initially, using the Caltech pedestrian data set did not result in a good performance. Only a limited number of frames in the video have pedestrians. Ultimately, we moved on using the Pascal VOC Object Classes Challenge dataset to train the model. It contains twenty object classes including a class for the person. With a greater population for the dataset, we were able to train the model and get good training and testing accuracies.

Results and Discussions

Using pre-trained weights in our project, we will be able to detect the pedestrians and objects on road with 69.4% accuracy for object detection and the bounding boxes. After we implemented our own YOLO v3 algorithm using PyTorch, we performed experiments based on different datasets. When we utilize Cross-Entropy instead of Binary Cross-Entropy, the results for this algorithm based on the dataset of Pascal VOC are

Class accuracy is: 38.888889%
No obj accuracy is: 6.138319%
Obj accuracy is: 72.682182%

Clearly, this is not the optimal result which people should expect from the CHEME599_FINAL_GPU.ipynb. I want to discuss a little bit about optimizations and the result from No obj accuracy here.

For the optimizations, we could use parallelism in this model, which means we can detect different scales of images at the same time. For the Darknet-53, we could utilize the pre-trained weights and then adopt it using our own dataset, which should save a lot of time. Based on our implementation, we could utilize pre-trained weights by turning on the LOAD in the config file.

I believe the No obj accuracy part is indeed reflecting false positives, which is the hardest part to handle. In other words, there are only backgrounds for some images. However, our model outputs some bounding boxes and probabilities for objects, which they recognize some features on only background images. We could reduce false positives by increasing thresholds for Intersection over union and confidence probability. We could also optimize the model by training the same images but adjust their sizes.

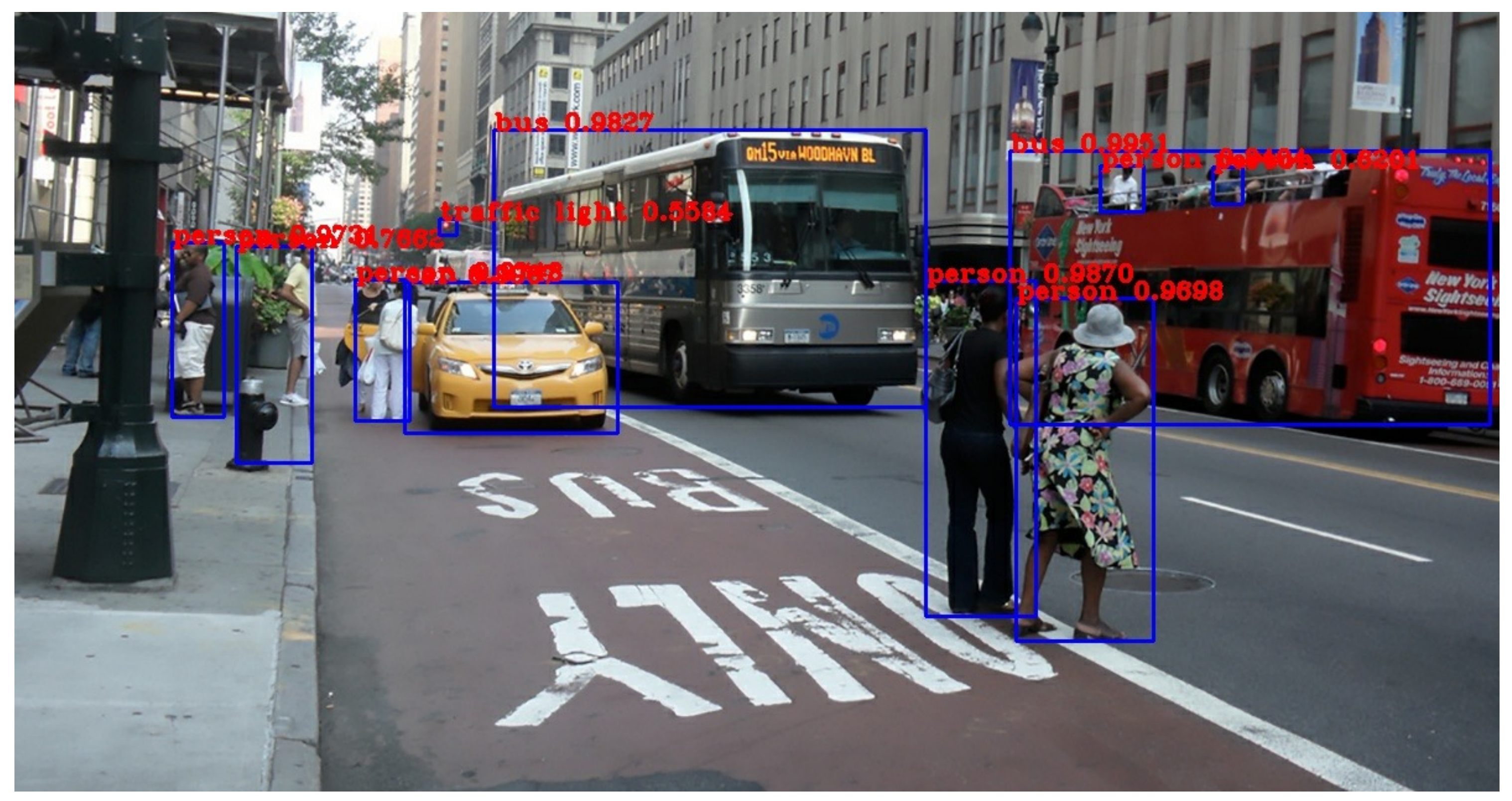

Figure 2. The result image of Yolo V3 in an urban environment

I would also want to discuss the most important characteristic of Yolo v3: efficiency. The time we detect an image depends on the size of the images because we are working with a more complicated matrix for a larger image. The average time to approximate an image of size 1200*630 is about 1.32 seconds. However, there are still some errors we want to optimize in the future.

In the future, even YOLO v3 had already utilized some features from R-CNN (Region-based Convolutional Neural Networks). We could train the model to color different areas with our training sets and categorize different regions. We might be able to apply this idea further in the YOLO algorithm in order to solve the issue of false positives.

References and Acknowledgements

[1] Redmon et al., 2015, You Only Look Once: Unified real-time object detection
[2] Redmon et al., 2018, YOLOv3: An Incremental Improvement
[3] Sanna Persson., 2018, YOLOv3 Implementation with Training setup from Scratch