Vehicle Detection and Classification

The Dataset:

A variation of COCO dataset I downloaded from here

It consists of:

- Train instances 13300 - Valid instances: 3798 - Test instances 1900
- A folder for each subset with images and annotations.csv file that contains the name of the files of these instances and their associated data and metadata like the localization data of the vehicle & its name, and the image size,.

The models I used:

First I was going to choose FAST-RCNN, but chose YOLO because it compromises between the accuracy and the speed making it more suitable for real time applications.

I have chosen YOLOV5X eventually because it was what performed better.

For the classification, I ended up with InceptionV3, the justification is explained in the notebook section of this report.

The notebook contains some of my trials in working in this project:

- First given the structure previously mentioned, I had to restructure the dataset to make it suitable for the YOLOv5 expected format, so I went through the annotations and created a text file for every image containing all the information about all the vehicles in the image while filtering out any vehicle that's not a car, a truck, or a bus. Then I included all the image instances of these three in the format of the yolo "class_id x_center y_center width height" per line.
- Generated YAML file
- First I made the class IDs 2, 5, 7 to keep the same order as the coco.yaml which is the default yaml in YOLO prediction, then changed them to 0, 1, 2 during training and other parts.
- I tried multiple YOLOv5 models, namely small, large, and extra large, both for inference and training, but the training actually performed worse in the small model, and was too exhausting for my resources for the extra large model, so I ended up with using the pre-trained YOLOv5x for detection.
- I measured the performance based on both the accuracy of predicting the label and the intersection between the actual bounding box and the predicted bounding box.
- After that comes classification; for pre-processing I tried three cases:

  1- Masking all the area outside the bounding box per predicted vehicle

  2- averaging this area with the mask to give context while giving focus on the area inside the box since it's not bleneded.

  3- just cropping the bounding box, and after trials with different DL models, I ended up with this one.

  In order not to reinvent the wheel, I went for transfer learning.

I added some layers to the top, and unfreezed some layers just before those, I experimented with a different number of unfreezed layers as well.

And I tried different models including ResNet50, Xception, NASLarge, MobileNetV2

Some of these models were too big for my gpu to handle, and it takes a very long time when I decrease the batch size.

Others were underfitting no matter how hard I fine-tuned them.

The best among them was InceptionV3

I fine tuned the model by steps including but not limited to: increasing #epochs, adding Image Augmentation, scheduling learning rate, adding early stopping, changing a bit in the added layers, like number of nodes, nature of the layer, dropout percentage etc, and tuning the input size

Finally I ended up with mAP 88 which is good but needs improvement.

The Script:

consists of a the pipeline that:

- starts with localization using the yolov5x model, we use this to predict and save the bounding box(es) indices
- Followed by classification pre-processing of the image and parsing of the bounding box(es) to be sent to the IncerptionV3 partially trained model for this task
- After which it goes through a post processing where the original image is viewed with the bounding box(es) and the predicted class(es).
- It also handles all flask-related coding

- The way to run it is mentioned in the README file

What to Improve?

- Hypertuner would be really significant for this task.
- Checking other yolo versions and trying to train some of its layers on the three classes in hand
- A better GUI.
- Pre-processing based on the cropped image’s size.