**NUMBER PLATE DETECTION AND TEXT EXTRACTION**

**MODELS USED:**

1. EasyOCR
2. YOLOv7 : standard for vehicle detection (available on “https://github.com/WongKinYiu/yolov7”), may require you to set-up a different virtual environment depending on the version of python installed(python 3.9.10 is compatible with yolov7 for sure).
3. YOLOv7 : pretrained for number plate detection

**REQUIREMENTS:**

1. Pytorch (version depends on whether GPU is available or not)
2. Opencv
3. Ultralytics
4. Python file of code
5. Input: video file(mp4)
6. Outputs:
   1. Video file(mp4)
   2. Csv file
   3. Vehicle images(only if you create an empty folder to store vehicle images)
   4. Number plate images(only if you create an empty folder to store the images)

**THEORY:**

The current version of code uses vehicle detection model to detect vehicles and number plate detection(pretrained) to detect number plates. Coordinates of number plate bounding box are extracted and these coordinates are passed into a function which extracts and stores the text of the number plate using EasyOCR in a csv file.The csv file is further filtered to store only the relevant results.

**ALGORTIHM:**

1. Use standard yolov7 model to detect vehicles, specify the classes which are to be detected, current code detects:
   1. Car : 2
   2. Motorcycle : 3
   3. Bus : 5
   4. Truck : 7
   5. Train : 6
2. A unique ID is maintained for each vehicle across the entire video. The following logic is used to achieve this:
   1. Create a dictionary to store the unique ID and coordinates of the centroids for each vehicle in the current and previous frames.
   2. For each frame in the video:
      1. **Calculate Centroids:**
         1. Detect all vehicles in the frame.
         2. Calculate the centroid of each bounding box of the detected vehicles.
      2. **Assign Unique IDs:**
         1. If it is the first frame:
            1. Assign a unique ID to each vehicle and store the coordinates of the centroids in the dictionary.
         2. If it is not the first frame:
            1. For each centroid in the current frame, find the closest centroid from the previous frame.
            2. Assign the ID of the closest centroid from the previous frame to the current centroid.
            3. If a new vehicle appears (i.e., no unused close centroid is found in the previous frame), assign a new unique ID to it.
      3. **Update Centroid Coordinates:**
         1. Update the dictionary with the coordinates of the centroids and their associated unique IDs.
         2. Remove centroids that are no longer detected in the current frame.
3. Use pretrained number plate detection model to detect number plates in the current frame. If the centroid of the number plate bounding box is inside the bounding box of a vehicle, number plate is assigned to that vehicle. Else it is rejected.
4. The detected detected number plates are preprocessed before we extract text from them, this increases the accuracy of EasyOCR. The steps taken are:

Number plate image -> Gray scale -> Binary -> Skew angle rotation ->processed image.

1. Apply EasyOCR on the processed image and store the ‘vehicle ID’, ’time stamp’, ‘ vehicle image’, ‘plate image’, ‘plate text’ and ‘confidence score’ in a csv file.
2. After the above steps have been applied to the entire video, the the following steps are performed to get a processed csv file:
   1. Group the csv file by 'Vehicle ID' and 'Plate Text' to count the occurrences of each plate text for each vehicle.
   2. For each 'Vehicle ID', select the 'Plate Text' with the highest occurrence.
   3. From these, select the row with the highest confidence score and save this row in a new csv file.
3. The new csv file now stores ‘vehicle id’ , ‘time stamp’, ‘plate text’, ‘plate confidence’, ‘vehicle image’ and ‘plate image’ for each vehicle that appeared in the video.

**NOTE:** Before the number plate text is stored in a csv file as in step 4, it is passed through some additional steps to store only the relevant instances:

1. Remove all characters from the text that are not letters or digits
2. Convert the entire text to uppercase.
3. If the length of the text is not 10 characters,discard the text (Indian number plates have 10 characters).
4. Maintain two dictionaries which include some common misinterpretations made by OCR between letters and digits.
5. The first dictionary contains letters as keys and the digits they are commonly misinterpreted as values (e.g., {'O': '0', 'I': '1', 'J': '3', 'L': '4', 'G': '6', 'S': '5', 'Q': '0'}). The second dictionary contains digits as keys and the letters they are commonly misinterpreted as values (e.g., {'0': 'O', '1': 'I', '3': 'J', '4': 'L', '6': 'G', '5': 'S'}).
6. Define the positions where letters and numbers are expected in the text. For Indian number plates letters are expected at index positions: 0,1,4,5 and digits are expected at index positions: 2,3,6,7,8,9.
7. For each character in the expected letter positions, if it is a digit, replace it using the character-to-digit dictionary.
8. For each character in the expected number positions, if it is a letter, replace it using the digit-to-character dictionary.
9. Return the corrected text.

**FOLDER STRUCTURE:**

License\_plate\_detections:

-code.py(code to be run)

-output\_videos

-test\_videos(input videos)

Create folders for rotated number plate images and vehicles images and give there path in code if you want to save images.