Vehicle License Plate Identification Using CNN-Type YOLO Algorithm

Niko Tolentino
Toronto Metropolitan University
Toronto, ON
niko.tolentino@torontomu.ca

Mahir Faisal
Toronto Metropolitan University
Toronto, ON
mahir.faisal@torontomu.ca

Waneeha Samoon
Toronto Metropolitan University
Toronto, ON
waneeha.samoon@torontomu.ca

***Abstract*—This document outlines the development of a modified CNN-Type YOLO Algorithm designed to detect and extract license plate information from vehicles. Developed using Python 3, the program leverages machine learning and computer vision libraries to enhance the capabilities of the YOLO architecture. For access to the projects Python 3 source code and other relevant files, please refer to the Google Drive containing them using the following link: [https://drive.google.com/drive/folders/1newszN0ti1f0dylTvhdbZCd9bnWEt8Ft?usp=sharing]**

***Keywords*—YOLO, algorithm, identification, training, OCR**

I. Introduction

In a time when intelligent transportation systems are becoming more and more prevalent, the requirement for robust and efficient methods for license plate recognition has grown. As urbanization progresses vehicular density will continue to rise, and the need for sophisticated solutions are increasingly apparent. In the context of intelligent transportation systems recognition of license plates is important for managing traffic flow and enhancing road safety. Being able to recognize license plates in a diverse and dynamic real-world environment is a challenging task with many implications for traffic management and other such systems. This paper delves into the application of a Convolutional Neural Network (CNN) - based on a YOLO algorithm, specifically employed for license plate recognition. The utilization of computer vision techniques has seen a significant advancement in recent years, especially in relation to object detection and classification. The YOLO algorithm has the ability to process images in a single forward pass, making it a strong candidate for real world applications. This paper extends the capabilities of the YOLO architecture adapting it to challenges posed by license plate identification. Building on previous studies that have found that the YOLO algorithm has exceptional versatility making it well suited for this problem [1]. Similarly, as demonstrated in [2] the YOLO framework reframes object detection as a regression problem, offering an efficient solution for license plate recognition, as it eliminates the need for separate segmentation and character recognition modules. The proposed methodology in this paper involves a tailored CNN-type YOLO model designed to address the intricacies of license plate recognition. The algorithm is trained on a diverse dataset encompassing various vehicle orientations and lighting conditions. The modification made to the YOLO architecture focuses on enhancing the ability to accurately detect and classify license plates overcoming challenges such as variations in plate design. As intelligent transportation systems continue to evolve, the use of a CNN-type YOLO model offers a robust and efficient solution to the challenge of license plate recognition.

II. Methodology

The methodology for this procedure involves first uploading a comprehensive set of vehicle images for the model. Then, a Google Colab notebook is used to do extensive training on a YOLOv8 model for object detection. A large-scale car dataset for license plate recognition was used in conjunction with the raw YoloV8 model [7].

A. Initializing the YoloV8 Model for Training and Validation

a clean unmodified yolov8 model is loaded into the notebook courtesy of the ultralytics library, then a custom YAML file that contains the paths for the training data, validation data, and label classification information is loaded into the notebook. For this case specifically, only one label of classification is used which is called license plate as that is what will be used for the object detection as stated in the introduction. After the custom YAML file is loaded, the training process for the model is initialized and runs for 10 epochs. Epochs in this context, refers to doing 10 complete passes of the algorithm through the dataset in order to maximize detection accuracy and minimize error rates for the model itself.

```
!unzip -q data.zip

ROOT_DIR = '/content/data'

!pip install ultralytics

import os
from ultralytics import YOLO

#Load a model
model=YOLO("yolov8n.yaml") #build a new
model from scratch

#Use the model
results =
model.train(data=os.path.join(ROOT_DIR,
"data.yaml"), epochs=10) #train the model
```

Fig. 1. Building a new model from scratch. [3][6]

```
path: ''#whatever path here
train: ../data/train/images/
val: ../data/valid/images/

# Classes
names:
   0: license_plate
```

Fig. 2. Custom YAML File for classes and file paths

B. *Testing the Modified and Trained YOLO v8 Model*

After the model has been fully trained using the dataset of license plate images the modified YOLO v8 model is then tested on a sample mp4 video file that contains a bunch of cars moving along a single highway. In this video, first the trained machine learning model is used to detect the license plate objects on the front of each car and then a python 3 optical character recognition library is used to extract the text description of each license plate and display it on the screen. The text extraction code is computed in a file called utils.py (short for utilities) which contains a set of functions that are used to first extract the car object data, then crop the car object data to just get the license plate information, extract the license plate information from the cropped image (the image cropping essentially creates another bounding box only around the license plate), check to make sure if the extracted license plate information complies with correct license plate standards and if it does comply, the results are written to a CSV file and if not, first the license plate information is formatted to the correct government standard and then the final formatted results are written to a CSV file [4][5]. This CSV file also contains the model's confidence score for each license plate extracted. This information can be used to help analyze the accuracy of the trained model which will be discussed in further detail in the results and analysis section of this report.

C. *Main Python Execution File*

Next, there is a main python execution file that makes use of all the python functions from the utils.py python file to properly execute all the necessary steps that are to be used in the vehicle license plate information extraction process [4][5].

D. *Creating a Visualization of the Model's Results*

Lastly, there is an add_missing_data.py file as well as the visualize.py python file which are both used to create a visual representation of how this object detection algorithm would be applied in a real-life scenario such as using it for dash cam footage vehicle-tracking footage for security purposes that can be used the local police department and so on. Starting with the add missing data python file, this file is essentially used to add back in the information from video frames which are missing from the exported CSV file [4][5]. The reason why the data for some video frames are missing is primarily due to how some of the algorithm's logic is programmed. To clarify further in the main.py and util.py python files, both files are used to first draw a bounding box around the various cars on the highway and then to crop the box around the license plate for each detected vehicle on the road and finally OCR is used to extract the license plate text from the bounded license plate and display it on the screen. The issue with this is that the OCR library is not perfect as when the text extraction is performed, it is checking if the license plate text complies with a valid license plate format and if not, the text must be modified in order to meet this valid format criterion. Given these strict guidelines, the OCR technology can be prone to errors and sometimes it is unable to detect certain characters that are part of a correct license plate if a given frame in the video is not clear enough for the OCR to detect it. Also in the CSV output, various car ids are associated with the same license plate which is a problem as each car id must be associated with a single unique license plate id.

TABLE I
TEST.CSV DATA SAMPLE FOR DETECTION

frame	carid	lp_bb_score	plate #	lp score
4	5	0.42639670	HU51KSU	0.40816944
5	5	0.49759772	HU51VSU	0.36735255

The CSV tables contain 8 total columns consisting of frame number, car id car bounding box, license plate bounding box, license plate bounding box confidence score, license plate number, and license plate number confidence score, although only 5 of the 8 are shown for explanation purposes. The frame number refers to a video frame containing a car with its license plate which was captured by the algorithm. The car id refers to a unique identification number that is associated with the vehicle itself. The car bounding box refers to the xy dimensions of the bounding box that the vehicle is encased in and the license plate bounding box column refers to the same description as the former with it specifically referring to the license plate box. The license plate bounding box confidence score refers to the algorithm's confidence level in accurately detecting and creating a box around a car's license plate on the road in the mp4 video sample. The license plate number column refers to the algorithm's output as to what the detected car's license plate number is from the video sample. Lastly, the plate number confidence score refers to the algorithm's confidence level in accurately outputting the correct license plate number for the detected car in the sample video file. To fix this issue, add_missing_data.py takes each car id that has the same associated license plate and then checks which car id has the highest confidence score for that license plate and conclude that that car id is accurately associated with the high-confidence score license plate.

E. *Interpolating Data for Frames without Data*

To solve the issue of frames missing (therefore making the video output not consistent with its display of information as it would not display any information even though the same vehicle is within view), interpolation is performed on the data. Regarding the field computer vision, data interpolation refers to the process of creating new values that lie between known values in an image. This definition makes sense as the primary goal here is to use the interpolation to create estimates based on the averages of the bounding boxes between the frames of the same car for the license plate data that would potentially be associated with those missing vehicle image frames. Thus, it makes for a logical solution for this problem [4][5]. For example, we take the 243rd to 246th frame, which contains the same vehicle according to the car_id, and see that the 244th and 245th frame do not have license plate information that pertains to these frames, even though they clearly indicate that the car is in that frame:

TABLE II
TEST_INTERPOLATED.CSV DATA SAMPLE FOR MISSING FRAMES

frame	carid	lp_bb_score	plate #	lp score
243	1	0.368922681	AP05JEO	0.408169449
244	1	0	0	0
245	1	0	0	0
246	1	0.357486695	AP05JEO	0.471034623

The scores are then averaged in add_missing_data.py to interpolate what the plate would be for those missing frames for a consistent and smooth output video.

III. RESULTS AND ANALYSIS

The investigation of the modified YOLOv8 model's performance for license plate detection and recognition unveiled crucial insights. To evaluate the model's reliability, performance metrics were taken into consideration to quantify the quality of the model.

Below is a detailed view of each of the criteria that the model's performance each epoch in regards to training and validation, which includes: box loss (a loss that measures how "tight" the predicted bounding boxes are to the ground truth object), classification loss (a loss that measures the correctness of the classification of each predicted bounding box: each box may contain an object class, or a "background"), distribution-focal loss (focuses on the distribution of the target rather than the target itself),as well as other metrics such as precision and average precision, recall (the ratio of true positives to the sum of true positives and false negatives), and an extension of the average precision metric; calculated at multiple IoU (intersection over union, which is related to the overlap of bounding boxes) thresholds.

A. Performance Metrics over Time

The following values were recorded and graphed over 10 epochs:

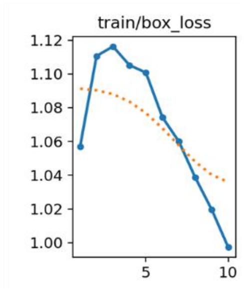

Fig. 3. Box loss in training. This loss was significantly reduced after 10 iterations, which signifies the model's increase in performance in regard to predicting the location of the license plates and their scale.

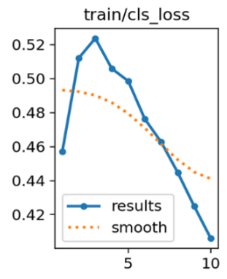

Fig. 4. Classification loss in training. This loss was also significantly reduced, which means the model's correctness in classifying each predicted bounding box.

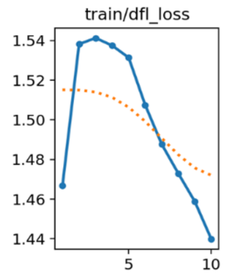

Fig. 5. Distribution-focal loss in training. A decrease in DFL Loss indicates that the model's predictions are getting closer to the actual target distributions.

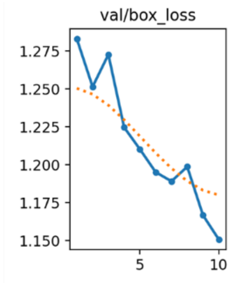

Fig. 6. Box loss in validation. Similar to training box loss, it indicates an increase in performance in regard to predicting location and size of the bounding boxes for objects.

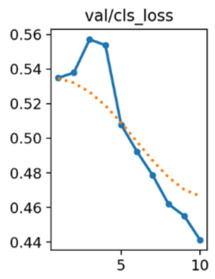

Fig. 7. Classification loss in validation. This indicates that the model is becoming more accurate at classifying the objects it detects.

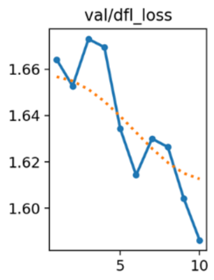

Fig. 8. Distribution-focal loss in validation. This shows that the model is more effectively learning the distribution of the classes in the dataset.

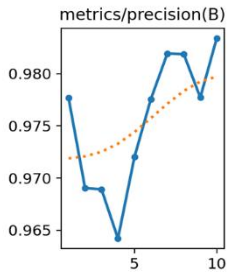

Fig. 9. Precision over time. This suggests that the proportion of true positive detections to total positive detections (true positives + false positives) has improved.

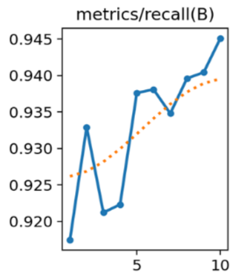

Fig. 10. This implies that the model is correctly identifying a higher proportion of actual positive cases (true positives) out of all actual positives (true positives + false negatives).

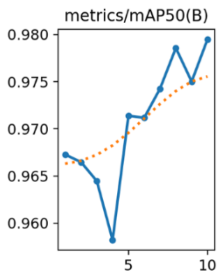

Fig. 11. Average precision over time. The model is more accurate and consistent in detecting objects across different conditions and thresholds.

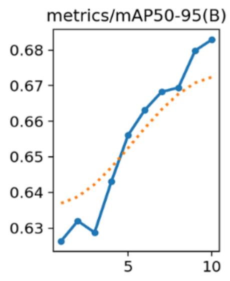

Fig. 12. Average precision calculated at multiple IoU thresholds (50%-95%). This metric gives a more comprehensive picture of an object detection model's performance across a range of IoU thresholds, providing insights into how well the model performs not just at moderate overlaps (like 50%) but also at high precision levels (up to 95%). Here it showcases improvement.

B. General Performance Evaluation

The model also underwent other evaluations to measure its performance. This is necessary to ensure performance and reliability standards are being met by the model.

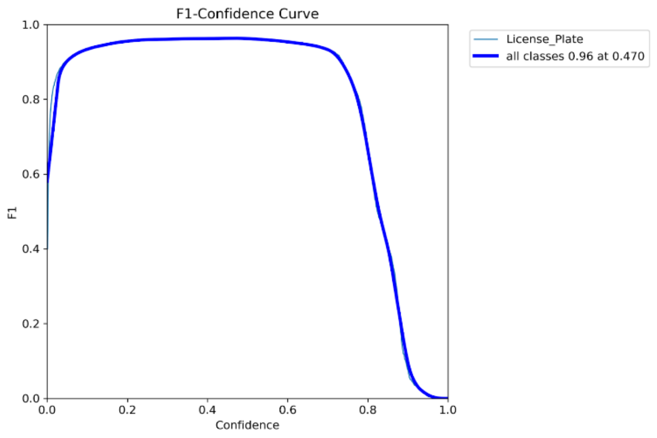

Fig. 13. F1-Confidence curve. A significant peak in the F1 score indicates a good balance between precision and recall at that confidence threshold. The height and sharpness of the peak signifies the model's effectiveness. A broader peak suggests the model is more robust to variations in the threshold.

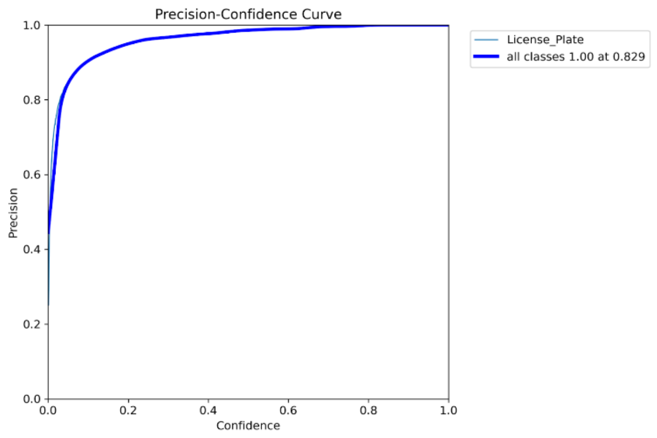

Fig. 14. Precision-Confidence curve. An upward trend as the confidence threshold increases indicates the model is becoming more precise (fewer false positives).

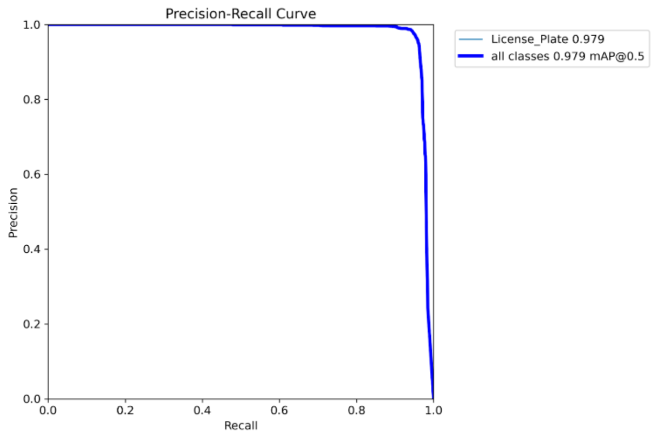

Fig. 15. Precision-Recall curve. A larger area under the Precision-Recall curve signifies better overall performance. The curve stays closer to the top right corner, indicates high precision and high recall.

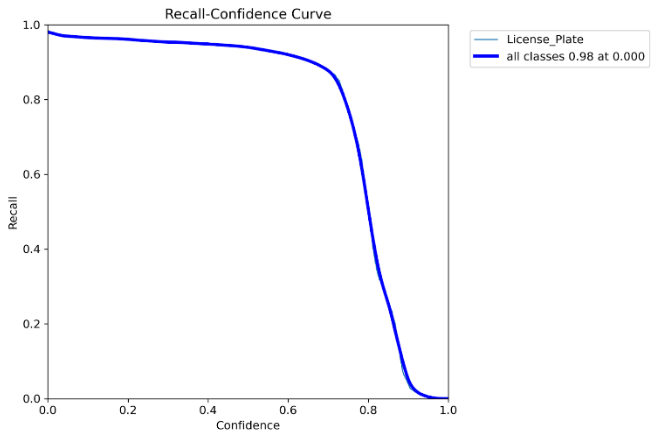

Fig. 16. Recall-Confidence curve. The model is able to detect almost all the relevant instances (true positives) without needing to be highly confident in its predictions. The recall remains consistently high as the confidence threshold increases, which means the model is robust in terms of recall; it does not miss many true positives even as it becomes more stringent in its predictions.

C. Training and Validation Dataset Samples

Below are samples of the dataset provided to the model undergoing training and validation before being tested on a sample video later to verify with the eye-test the performance of the model.

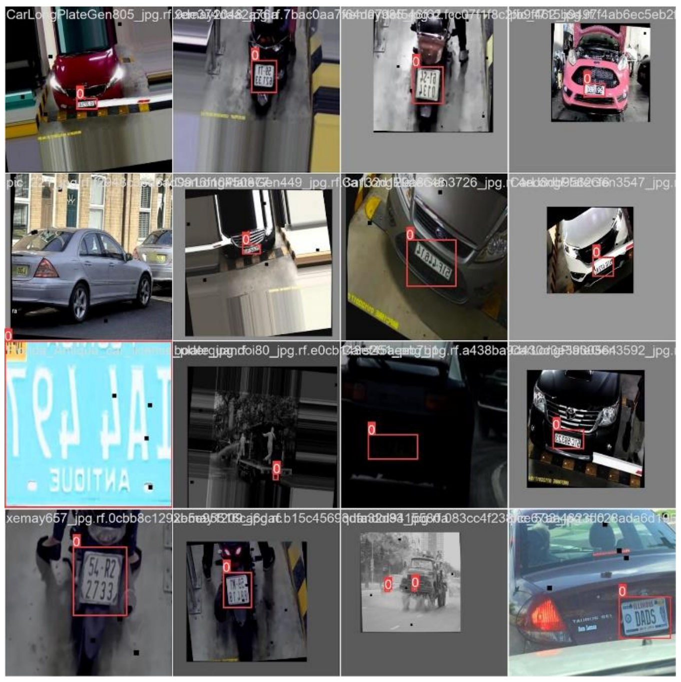

Fig. 17. The images above represent a sample of the training batch for the model visualized.

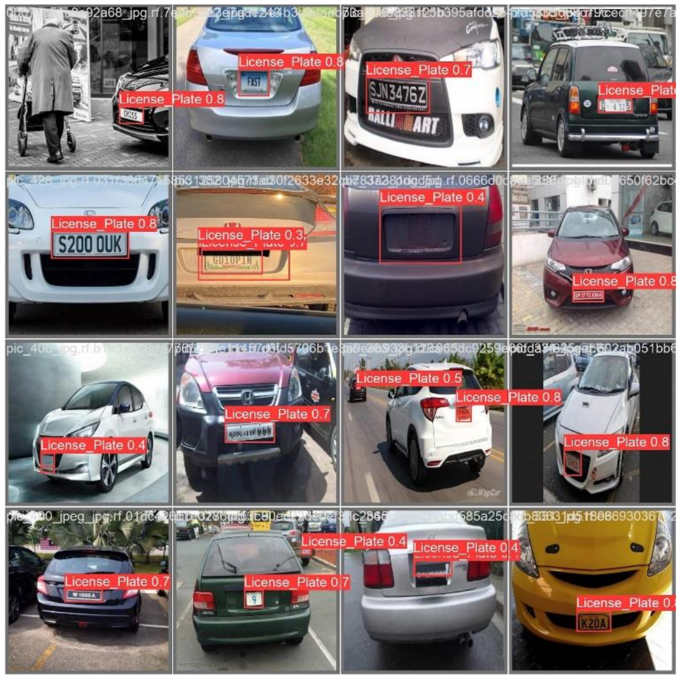

Fig. 18. A sample of the batch of validation, which provides a visualization of the validation the model undergoes.

D. Practical Test

After the model's training phase, the investigation proceeded to a real-world assessment of its practicality through the testing phase using a sample MP4 video. This practical test environment serves as a crucible for evaluating the model's performance under dynamic and complex conditions. The video depicted multiple vehicles traversing a highway, presenting a challenging scenario for license plate detection and recognition to be performed by the model.

The results of the model’s extraction from the video were placed into a .csv file (test.csv) for the application to read and extract from:

TABLE III
TEST.CSV DATA SAMPLE

Values of Identified Vehicles in Sample Video			
frame_nmr	car_id	l_p_bbox_score	license_no
0	3	0.5656788349151611	OA13NRU
1	5	0.5951626300811768	MU51KSU
1	3	0.5656682848930359	OA13NRU

[a] This table is only a snippet of the results and values that within the test.csv file.

The table does not contain the full list of values from the test.csv file such as the bounding box parameters for the car and license plate. However, it does contain information that pertains to the success of the extraction of the license plate for the application to use later. Multiple instances of the same car are extracted as the same car is present in multiple frames of the video.

Some of these instances will be missing a value for the car bounding box and license plate that were interpolated from other values of the same car and averaged to provide a smooth visualization. The interpolated values are in another file (test_interpolated.csv).

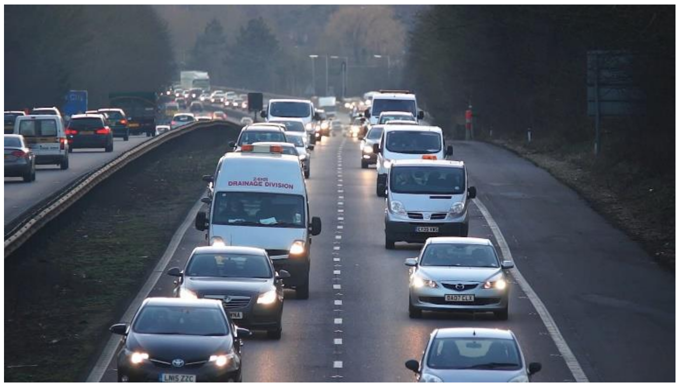

Fig. 19. A still of the sample video given to the model to detect and recognize license plates and extract the information it contains.

The model successfully identifies numerous plates, even multiple in the same instance/frame as the cars are traversing through the highway, as it recognizes, enlarges the visual, and extracts the information of the plate to be displayed in real-time in the video.

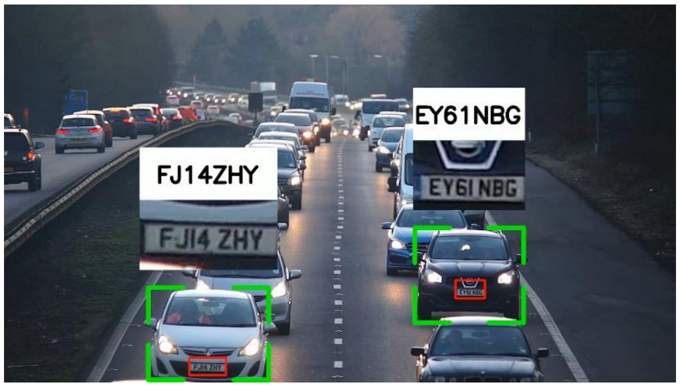

Fig. 20. A still of the output video provides a visualization of the successful performance of the model that was employed.

IV. Conclusion

To conclude, the development and analysis of the modified CNN-type YOLO model for license plate recognition have yielded valuable insights into the potential of computer vision in intelligent transportation systems. The utilization of the YOLO model for license plate detection proved successful, overcoming variations in images. As the algorithm was trained on a large and diverse dataset enhancing its adaptability to real-world scenarios. The YOLOv8 model was extensively trained with a comprehensive set of images contributing to the accuracy of the solution. This is evidenced by the F1 and precision metrics, emphasizing its ability to accurately detect license plates. The use of the Optical Character Recognition (OCR) library enhanced the utility of the system, simplifying the process of extracting and validating text from detected license plates. This solution has the potential for further improvements and some suggestions may be refining the model architecture, diversifying the database further, and fine tuning hyper parameters during the training phase.

Several improvements to the model are also possible that were not feasible due to equipment constraints. With more processing power from a higher-end computer, epochs of 100 or more could be leveraged to finetune the model. The YOLO-CNN algorithm provided also is updated regularly and will continue to see improvements in future updates that could affect the quality of the results emitted from the model after undergoing testing.

The success of the project can be attributed to the efforts of all the team members. Specifically Niko Tolentino was responsible for the modification of the source code and training the YOLOv8 model. The other team members Mahir Faisal and Waneeha Samoon were responsible for research and development.

References

[1] R. Al-Batat, A. Angelopoulou, S. Premkumar, J. Hemanth, and E. Kapetanios, Eds., “An End-to-End Automated License Plate Recognition System Using YOLO Based Vehicle and License Plate Detection with Vehicle Classification,” in Sensors, Dec. 2022. [online]. doi: 10.3390/s22239477

[2] K. Oublal and X. Dai, “An advanced combination of semi-supervised Normalizing Flow & Yolo (YoloNF) to detect and recognize vehicle

license plates,” MoroccanAI Data Challenge Winning Solution, 2021. [Online]. Available: https://arxiv.org/pdf/2207.10777v1.pdf

[3] Computer Vision Engineer, “Train Yolov8 object detection on a custom dataset | Step by step guide | Computer vision tutorial,” www.youtube.com, Jan. 30, 2023. Available: https://www.youtube.com/watch?t=2581&v=m9fH9OWn8YM&feature=youtu.be [accessed Dec. 14, 2023].

[4] Computer Vision Engineer, “Automatic number plate recognition with Python, Yolov8 and EasyOCR | Computer vision tutorial,” www.youtube.com, Jun. 19, 2023. Available: https://www.youtube.com/watch?v=fyJB1t0o0ms [accessed Dec. 14, 2023].

[5] Computer Vision Engineer, “automatic-number-plate-recognition-python-yolov8,” GitHub, Nov. 28, 2023. Available: https://github.com/computervisioneng/automatic-number-plate-recognition-python-yolov8 [accessed Dec. 14, 2023].

[6] Computer Vision Engineer, “train-yolov8-custom-dataset-step-by-step-guide,” GitHub, Dec. 12, 2023. Available: https://github.com/computervisioneng/train-yolov8-custom-dataset-step-by-step-guide [accessed Dec. 14, 2023].

[7] Linjie Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. "A Large-Scale Car Dataset for Fine-Grained Categorization and Verification", In Computer Vision and Pattern Recognition (CVPR), 2015. [online]. Available: https://mmlab.ie.cuhk.edu.hk/datasets/comp_cars/

[8] M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.