License Plate Detection System for Stolen Vehicles

Arush Kumar
akumar67@buffalo.edu

Atharva Aarya
aarya2@buffalo.edu

***Abstract*— This report explains our project which is a license plate detection system which utilizes the UFPR ALPR dataset. Our system integrates YOLOv5 for vehicle and license plate detection along with our own optical character recognition model built from scratch to extract text from license plates. We use different computer vision techniques in image pre-processing to address challenges of real-word scenarios which have varied lighting conditions, plate sizes and occlusions. We have used certain metrics to find our system's accuracy.**

***Index Terms*—** computer vision, detection, ocr, license plate, object detection

I. PROJECT OVERVIEW

Application: Our project is a license plate detection system where we aim to get the extract text which is present on the license plates. The application has many use cases such as in law enforcement, parking management and toll collection systems. For our application, we take an image for input where a car is in the picture with a license plate. Then the pipeline code is run which first detects the vehicle and the license plate. Then that license plate is extracted and pre-processing is done. Then that image is sent to our custom ocr model for text recognition. Once the text is identified, an output image is generated where there is a bounding box over the license plate. The identified characters are printed on top of the license plate. We are also calculating the character accuracy, character error rate and Intersection over Union for our analysis and storing it in a csv file and json file. We are also creating graphs for this and comparing it with easyocr, which is an existing OCR model.

State of the Art: License plate detection and recognition systems have seen advancements with time. There are systems like OpenALPR (Automatic License Plate Recognition) which use YOLO for object detection and OCR models for extracting texts. These systems rely on pre-trained models which are mostly designed as black-box systems which have limited opportunities in terms of fine-tuning or customization. We have developed a custom pipeline where we have created our own OCR model from scratch giving us more control over the model architecture and training process. This approach deepens our understanding of object detection and character recognition techniques along with analysing the performance on real world datasets.

Contributions:

1. Custom Pipeline development – We have designed a complete pipeline which involves plate detection, pre-processing and text recognition.

2. Custom OCR Model- We have developed our own OCR model which is tailored specifically for license plate recognition.

3. Evaluation - We have calculated various performance metrics to evaluate or system's performance. We have also benchmarked our OCR model against EasyOCR for comparison. We have ran our code on the entire UFPR ALPR dataset consisting of 4500 images.

II. APPROACH

Algorithms: We have utilized YOLOv5 object detection algorithm for detecting vehicles and license plates. We went for a pretrained model as we had limited computation power. YOLO is efficient in detecting multiple objects simultaneously in real-world scenarios. We trained the YOLOv5 model on the UFPR-ALPR dataset. The dataset contains 4500 images which was split into training , testing and validation folders. We converted the bounding box coordinates for license plates from the dataset format into YOLO format for training. We choose version 5 of YOLO as it has a balance between speed and accuracy and works well with systems with less computation power. We also tried version 7 but noticed that the detection was not up to the mark. We defined two classes, 0 for the vehicle and 1 for the license plate. This algorithm had many pros such as fast and efficient detection along with limited computational

power. In terms of cons, fine tuning was still required for optimal performance. It struggled with occlusions in some cases.

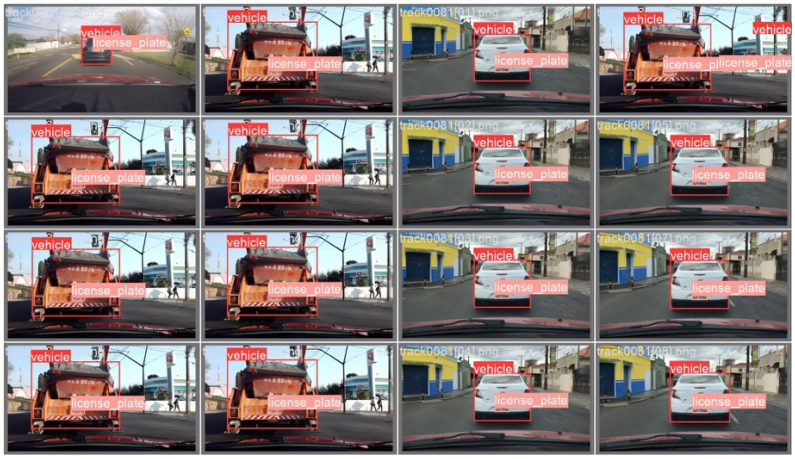

Fig. 1 ***Object Detection Results After Fine-Tuning YOLOv5:*** *The grid showcases the model's predictions, including detected vehicles and license plates, with bounding boxes drawn on test images.*

For text recognition, we developed a custom convolutional neural network. It has 3 convolutional layers followed by pooling layers, fully connected layers and a softmax output layer to predict the alphanumeric sequence of the license plate. We trained our model on the chars74K dataset. We also fine-tuned our model on alphanumeric sequences present in license plates from License plates, a dataset which has over 120k photos of vehicles and number-plates**.** It is a paid dataset but we found the Brazil version for free. [1] As the UFPR-ALPR dataset also comprises of vehicles from Brazil, this was beneficial for our training. The CNN takes 32x32 pixel image of a license plate, processes it through each layer and gives an output sequence of character probabilities which are then decoded into the final text recognised. Pros for this were that as it was our own model, we had the freedom to fully customize it for recognizing alphanumeric characters. We could fine tune it based on our own need. Cons for this were that the model depended heavily on the variety of training data.

For pre-processing, We first tried preprocessing using segmentation but did not get good results. Multiple characters were detected as one in one window as seen in Fig 2.1, In our approach we enhanced text clarity with techniques like grayscale conversion, sharpening the image with a unsharp mask and resizing to the input size for our OCR model.

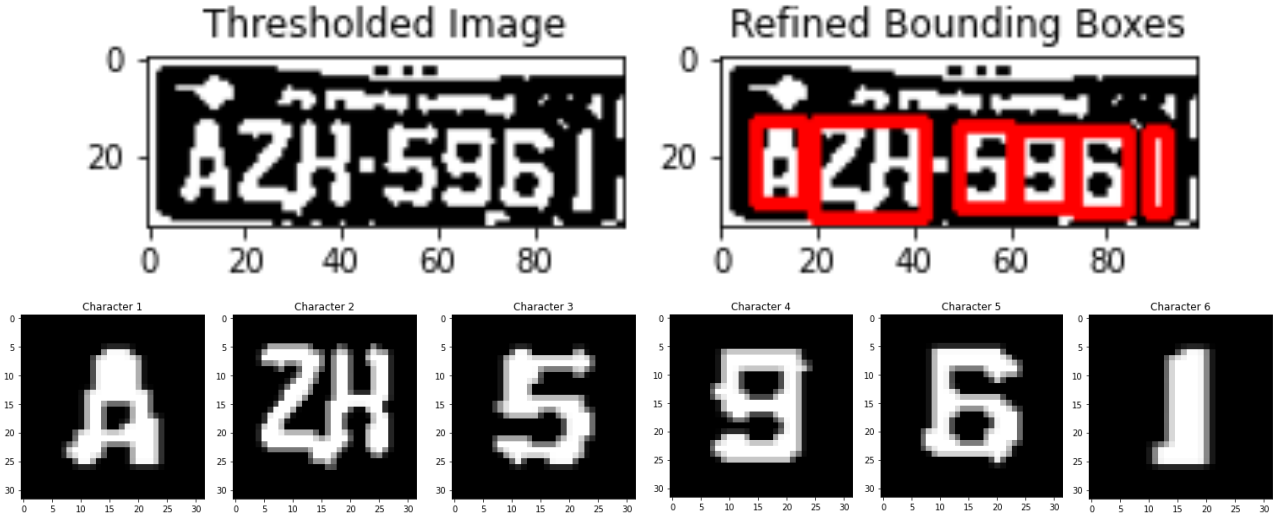

Fig. 2.1 ***Character Segmentation for License Plate Detection:*** *The left image shows the pre-processed license plate, while the right image illustrates the segmented characters with bounding boxes, enabling OCR to recognize each individual character, but there is a error in character-level segmentation, seen by recognizing 2 characters as 1.*

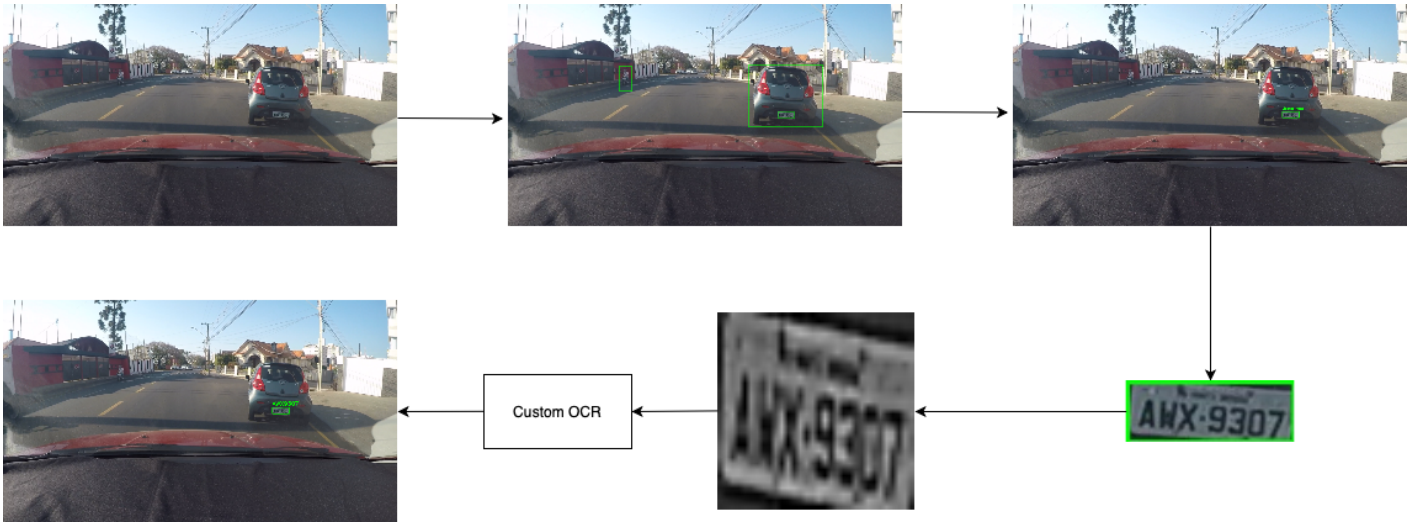

Fig. 2.2 In our pipeline, we first take an image and run our trained YOLO model. It returns two bounding boxes, one for the vehicle and one for the license plate. Then the cropped license plate is taken and preprocessing is done on that plate. Once the image is pre-processed, it is sent to our custom OCR model and each character is predicted.

When the code is run, as seen in Fig. 2.2, we are storing the generated predictions in a separate folder where bounding boxes are created with the predicted text on top of the license plate. Results are stored in csv, json and graphs are generated for analysing accuracy.

Aspects coded from scratch: We wrote a custom code to convert the dataset annotations present for each picture into YOLO format and then trained the model on the UFPR-ALPR dataset. For OCR, we have created the entire CNN architecture and done its training process from scratch. The pre-processing module has also been done from scratch. Then, we have created a pipeline where each file has its purpose such as detection, ocr and evaluations. The final calculations for character accuracy and IoU have also been done by us.

Aspects used from Online Resources: We have used YOLOv5, a pre-trained model for object detection. We still had to fine tune it for our application and dataset. [2]

We have also utilized EasyOCR to compare results with our OCR implementation.[3]

III. Experimental Protocol

We have used the UFPR-ALPR dataset for our project which has 4500 annotated images from 150 vehicles which are present in real-world scenarios. There are various types of vehicles including cars, trucks, buses and motorcycles. The dataset is released for only academic research so we had to request for it from its creator [4]. Each image is paired with a ground truth file which had the plate coordinates and the actual license plate text. This dataset was essential for assessing the performance of both detection and recognition. For OCR, we used the chars74k dataset which contains over 74000 images representing characters and sequence of characters. It served as a crucial resource and set the base for our model. We also fine-tuned our model using another dataset which had license plates from Brazil.

Evaluation of Success: We have used various metrics both qualitative and quantitative. We calculated the IoU (Intersection over Union) which finds the accuracy of the predicted bounding boxes against the ground truth boxes. Character accuracy and error rates are calculated which find out the accuracy of OCR at the character level and the errors found in the recognized text when compared to the ground truth. We have also done a comparison of results by running the pipeline with our custom ocr and then followed by easyOCR. We have different visualizations generated to better understand the model's performance. Full plate accuracy is also calculated which is all characters were correctly predicted in the license plate.

Compute Resources Needed: The vehicle and license plate detection uses YOLO which requires a GPU for real time processing of the entire dataset. For processing each image, YOLO uses 15.8 GFLOPs. For training our OCR model, it

took a lot of CPU resources as there were a total of 157 layers and 7015519 parameters. To run our pipeline on the entire dataset, it takes around an hour.

IV. Results

We got decent results Fig. 3.1, when we ran our pipeline code on the entire dataset. For our custom ocr, we got a average character accuracy as 78.94%. We got a full plate accuracy of 49.16% which means our system was able to correctly identify all the characters of a license plate in almost 2250 images. Mean IoU of 0.88 shows an excellent license plate detection accuracy.

```
Total images processed: 4500

Custom OCR Metrics:
--------------------------------------------------
Mean IoU: 0.88
Full Plate Accuracy: 49.16%
Character Accuracy: 78.94%
```

Fig. 3.1 Out of 4500 images processed, our system achieved a Mean IoU of 0.88, a Full Plate Accuracy of 49.16%, and a Character Accuracy of 78.94%.

Based on the IoU graph in the Fig. 3.2, we can see that majority of detections fall between 0.8 and 1 which shows strong localization of license plate. Our model is accurately able to locate license plates in most images.

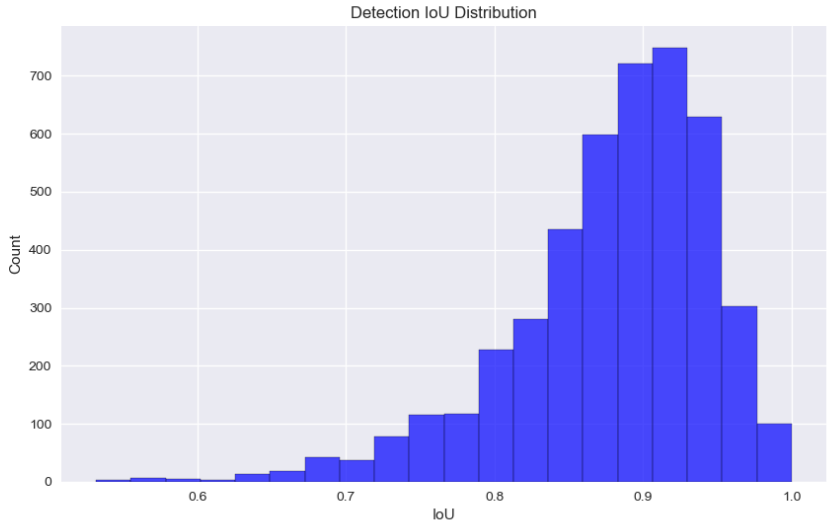

Fig. 3.2 The histogram shows the distribution of IoU scores for our model's predictions. Most detections achieve high IoU values (0.85–0.95).

In terms of state of the art OCR easyOCR, it got a character accuracy as 27.12% with the full plate accuracy of 1.33%. Our custom ocr performed much better than easyOCR. This could be due to our fine tuning of the model on two datasets. EasyOCR is a general purpose OCR tool which is more suitable to detect text in various scenarios. Its performance on specialized tasks like license plate character recognition might not be optimal. We would need to adapt its underlying neural network for fine tuning which would require more resources.

```
EasyOCR Metrics:
--------------------------------------------------
Mean IoU: 0.88
Full Plate Accuracy: 1.33%
Character Accuracy: 27.12%
```

Fig. 3.3 Out of 4500 images processed, the EasyOCR system achieved a Full Plate Accuracy of 1.33%, and a Character Accuracy of 27.12%.

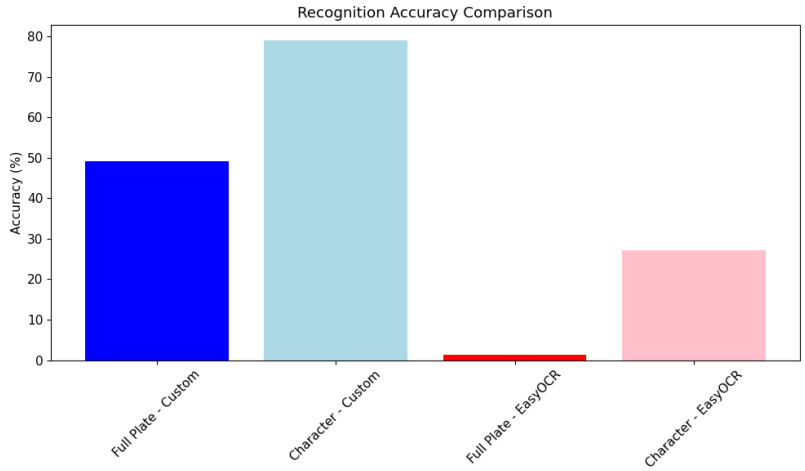

Fig. 3.4 ***Recognition Accuracy Comparison Between Custom Model and EasyOCR:*** *The custom OCR model significantly outperforms EasyOCR, achieving ~45% full plate accuracy and ~75% character-level accuracy.*

The following graph in Fig. 3.5, shows the levenshtein distance distribution between our OCR and EasyOCR. The first one which is our OCR, shows a strong left skewed distribution suggesting that for more than 2000 cases, 0 edit distance was the highest meaning all characters were accurate. Most of the errors require only 1-2 character corrections.

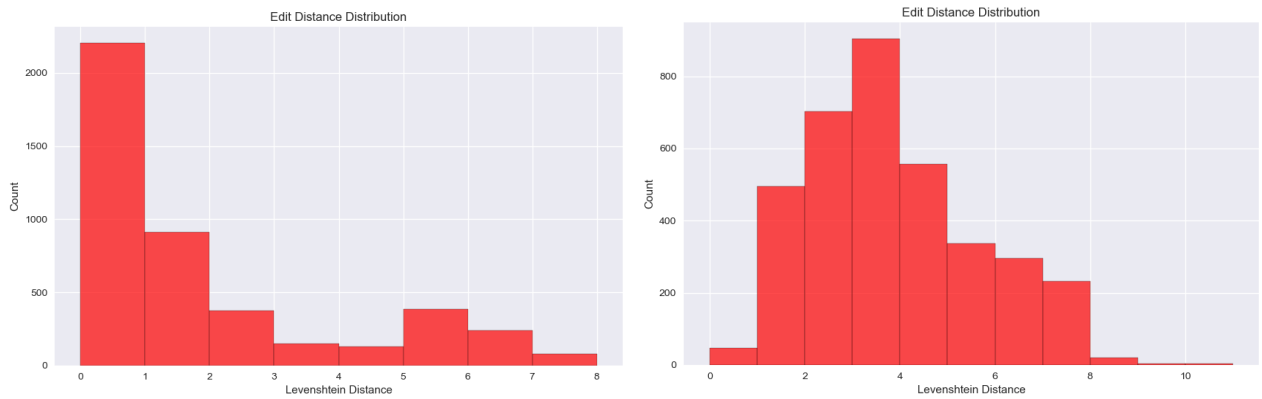

Fig. 3.5 ***Edit Distance Comparison: Custom OCR vs. EasyOCR***
The first histogram (Custom OCR) shows a majority of predictions with lower Levenshtein Distances, with over 2000 predictions requiring 0 or 1 edits, highlighting the high accuracy of the custom OCR model. The second histogram (EasyOCR) exhibits a more spread-out distribution, with most edit distances clustering around 3-5, indicating lower overall accuracy.

For easyocr, peak frequency is at distance 3-4 which means that predictions will require 3-4 character corrections.

For our pipeline, we also calculated the confusion matrix seen in Fig. 3.6 which showed binary classification of the license plate. There were only 17 false positives which aligns with the high IoU value, the second graph shows that with easyOCR case, there were more false positives.

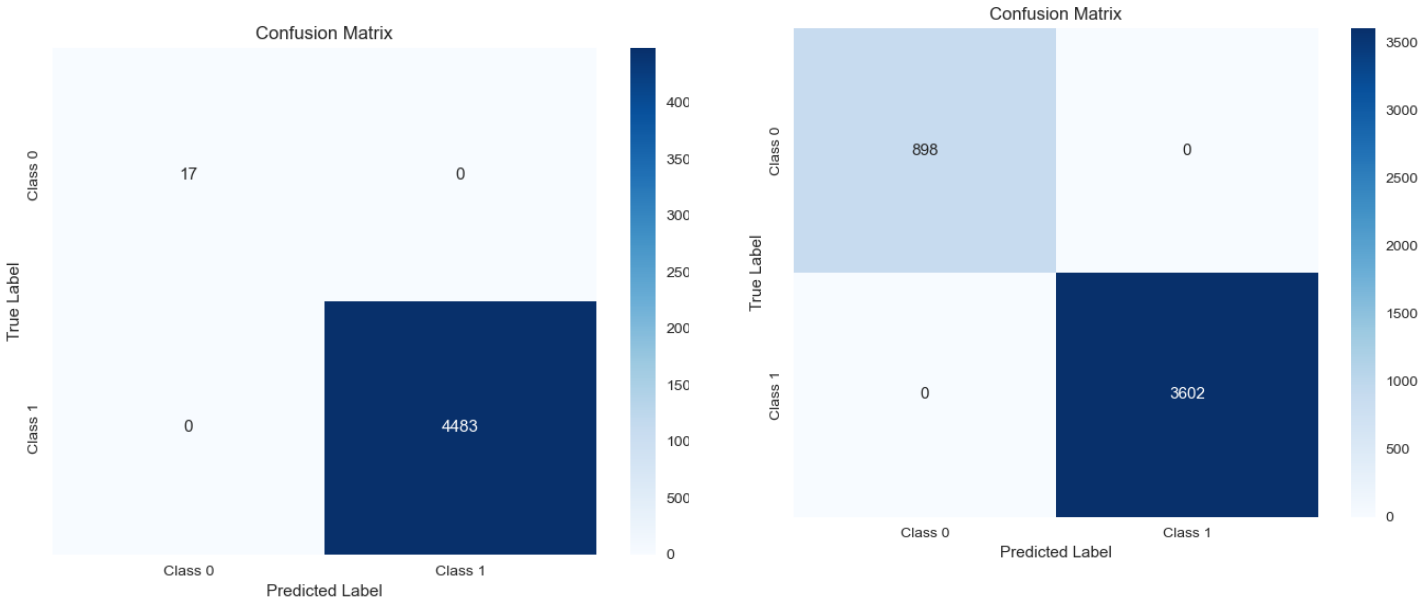

Fig. 3.6 ***Confusion Matrix Comparison: Custom OCR vs. EasyOCR***
The Custom OCR (left) achieves perfect classification with no misclassifications. EasyOCR (right) also performs well, with 898 correct Class 0 and 3602 correct Class 1 predictions but demonstrates a less balanced dataset.

The following graph shows the character accuracy distribution between our OCR and EasyOCR. The first one Fig. 3.8, which is our ocr shows that the most predictions achieve high character accuracy indicating reliable ocr performance. However, a small tail of lower accuracy scores (below 70%) suggesting occasional recognition challenges.

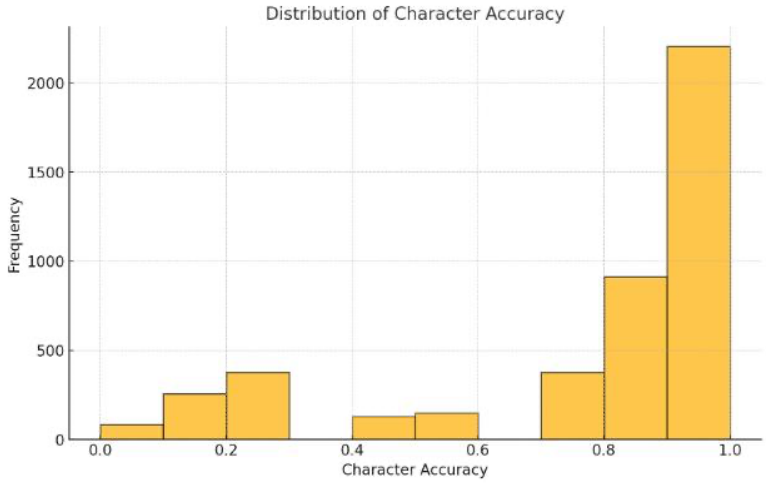

Fig. 3.7 ***Distribution of Character Accuracy:***
This histogram displays the frequency of character-level accuracy for the OCR model. The majority of predictions achieve high accuracy (0.8–1.0).

For EasyOCR, seen in Fig. 3.8 most samples have low character accuracy values i.e. near 0 indicating frequent mispredictions, whereas few samples achieve higher accuracy.

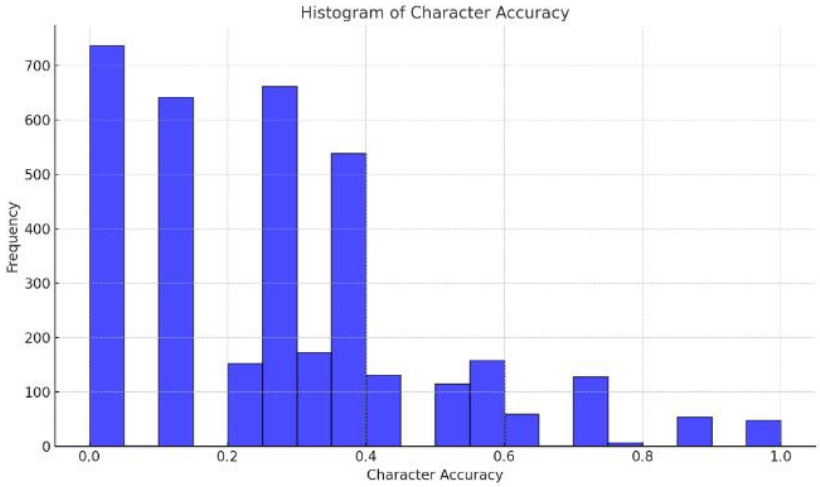

Fig. 3.8 ***Distribution of Character Accuracy:***
This histogram displays the frequency of character-level accuracy for the EasyOCR model with many predictions clustering around 0.0–0.4.

As a whole, our current system gave a full plate accuracy of 49.16% which includes cars, bikes and trucks. Other state of the art systems involving both detection and character recognition as a whole like Sighthound have overall accuracy of 47.4% and OpenALPR's 50.9%. So we were able to get similar results when compared to these systems. Although with time, there have been improvements in methods and Laroca can get an overall accuracy of 90%.

The following images are some of the examples of the final result. We can see that our system was able to detect license plates of motorbikes or buses as well. We were able to get information despite shadows/occlusions present using pre-processing techniques.

Fig. 3.9. ***Final Output: License Plate Detection in Real-World Scenarios***
This collage demonstrates the final output of our model, accurately detecting and highlighting license plates in diverse real-world driving conditions

V. Analysis

Limitations: The YOLOv5 model was trained on the UFPR-ALPR dataset, which has Brazilian license plates. Our system would struggle to generalize different license plate styles present from other countries. To improve, we will have to train our model with new data. We also noticed that for smaller license plates in bikes, the model struggled to correctly identify license plate. Our custom OCR depends on high quality training data which required training it over multiple datasets. We could include sequence modelling where we could add bidirectional long short-term memory (LSTM) to get better accuracy.

Advantages: With a mean IoU of 0.88, YOLO demonstrated a very high accuracy of localizing license plate in real world scenario. We were easily able to train it on our dataset. Using a custom cnn was beneficial as we could specifically design to for license plate character recognition. It gave more control over the architecture where we could experiment different configurations.

VI. Discussion and Lessons Learned

From this project, we learned about building a complete computer vision pipeline which integrated object detection and optical character recognition. We were able to understand the ins and outs of how such a system works in real-world. We tried various techniques in terms of pre-processing. Hit and trial methods were used as we experimented different techniques like sharpening. Using CLAHE method decreased the final accuracy. We learned that a custom model requires high quality training data to get optimal performance. Certain aspects were quite challenging like fine tuning models for real-world scenarios. We had to do a lot of research on how YOLO takes its inputs and had to convert the annotations present in the dataset to the right format. Additionally, by comparing our OCR with EasyOCR, we learned about the trade-offs between flexibility and ease of use.

There are various potential future applications from these lessons learned. We customized existing models and did a lot of fine tuning. We can apply these principles to various other computer vision challenges like identifying street signs for autonomous vehicles. The experience of creating a custom model along with using pre-trained models would help us in future projects which require quick prototyping. The pipeline we created can be altered to adapt to other types of application like text recognition or document scanning.

VII. Bibliography

[1] https://www.kaggle.com/datasets/trainingdatapro/license-plates-1-209-438-ocr-plates

[2] https://github.com/ultralytics/yolov5

[3] https://github.com/JaidedAI/EasyOCR

[4] https://github.com/raysonlaroca/ufpr-alpr-dataset

5. R. Laroca, E. Severo, L. A. Zanlorensi, L. S. Oliveira, G. R. Gonçalves, W. R. Schwartz, and D. Menotti, "*A Robust Real-Time Automatic License Plate Recognition Based on the YOLO Detector*" in International Joint Conference on Neural Networks (IJCNN), pp. 1–10, July 2018. [IEEE Xplore]

6. S. Du, M. Ibrahim, M. Shehata and W. Badawy, "Automatic License Plate Recognition (ALPR): A State-of-the-Art Review," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 23, no. 2, pp. 311-325, Feb. 2013, doi: 10.1109/TCSVT.2012.2203741.

7. L. Lakshmanan, Y. Vora and R. Ghate, "Deep Learning Based Vehicle Tracking System Using License Plate Detection And Recognition", https://arxiv.org/abs/2005.086

8. X. Wang, S. Hong, J. Li, Y. Zhao, C. Xiang and X. Chen, "License plate detection and recognition based on deep learning," *2024 2nd International Conference on Mechatronics, IoT and Industrial Informatics (ICMIII)*, Melbourne, Australia, 2024, pp. 86-90, doi: 10.1109/ICMIII62623.2024.00022. [link]

9. D. Sarma, A. Bora and A. Bhagat, "Automatic License Plate Detection and Recognition System for Security Purposes," *2023 IEEE Guwahati Subsection Conference (GCON)*, Guwahati, India, 2023, pp. 1-5, doi: 10.1109/GCON58516.2023.10183638. [link]