Automated parking tickets

1st Abdulkarim Dawalibi
2nd Adnan Altukleh

I. INTRODUCTION

The problem of parking tickets is a common issue faced by many cities and urban areas. Traditional parking systems often rely on physical tickets that must be purchased and displayed on a car's dashboard. However, this approach can be inefficient and can lead to numerous issues such as lost tickets, expired tickets, and fines for parking violations.

Our project aims to address the parking ticket problem by developing a sophisticated system capable of accurately detecting the plate number of parked vehicles. This will be achieved through the application of optical character recognition (OCR) technology, which will enable the system to accurately read and record the plate numbers in a dataset.

By implementing this system, car owners will no longer be required to purchase a physical parking ticket. Instead, they will receive a digital receipt sent directly to their app or home. Our project will focus on the development of the plate detection and OCR component of the system, ensuring that it is optimized for maximum accuracy and efficiency.

II. METHOD

A. Dataset

The dataset we are using is from Kaggle [1]. The dataset files are divided into training and testing data. There are 433 images with annotation files of car license plates within the images. 80% of the data is designated for training, while the remaining 20% is reserved for testing.

B. Training the model for plate detection from an image

We used the TensorFlow object detection framework to train our model to recognize car license plates from an image. We decided to have a 10 000 step aiming for a good accuracy score from the model.

C. Preparing the model

Initially, we applied the system to a single image instead of performing real-time detection. We began by importing the necessary libraries and setting variable names for the path. To utilize the model, we first created a label map and a TensorFlow record. Before implementing the detection script, we copied the model config from the training folder and updated it for transfer learning. Finally, we loaded the trained model from the checkpoint.

D. Detect license plate from an Image

Initially, we created a category index from the label map and obtained the path for the image on which the model will be applied. Subsequently, we implemented the detection script to apply the model and generate an image that displays the detected license plate number of the car.

E. Apply OCR

To apply OCR to the right part of the image, we will apply a threshold above a certain level. To achieve this, we set a variable to serve as our detection threshold. Next, we initialize our image, scores, boxes, and classes based on our threshold. We then obtain the box coordinates that include the detected license plate number and apply OCR model (easyocr) to this specific area.

F. Filtering the OCR to get just a plate number

In certain cases, we may not want to include the periphery when the license plate number is partially visible. To address this, we apply a filtering script that removes extraneous parts of the image. The function returns the text modified by the OCR model after applying this filtering step.

G. Saving an Image

Finally, we save the image name, license plate number, and the time of detection in a dataset. We also store the image in a designated folder at the same time.

H. Plate detection live

The process for detecting a license plate from a video is similar to the steps we took for an image. However, we wrap all the steps in a while loop since the process needs to be performed continuously. We omit the preparation step since it only needs to be executed once.

I. Model evaluation

For evaluating the object detection model, we used the COCO evaluation metrics. The model is being evaluated on how well it can detect bounding boxes around objects in an image.

III. RESULTS

The image detection model produces an image that displays the detected license plate, which is visually highlighted by a green triangle (as illustrated in Figure 1)

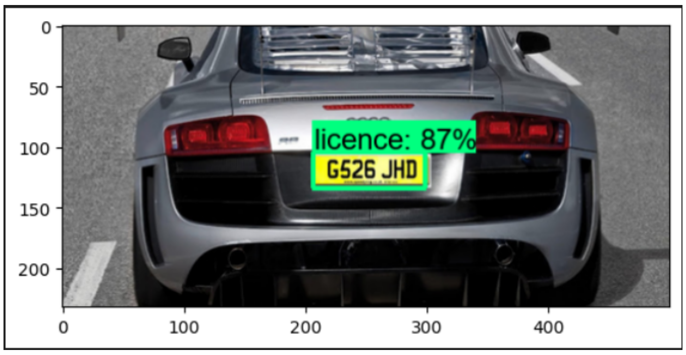


figure 1: license plate detection.

Once the license plate is detected, the image is cropped to focus on the license plate itself (as shown in Figure 2).

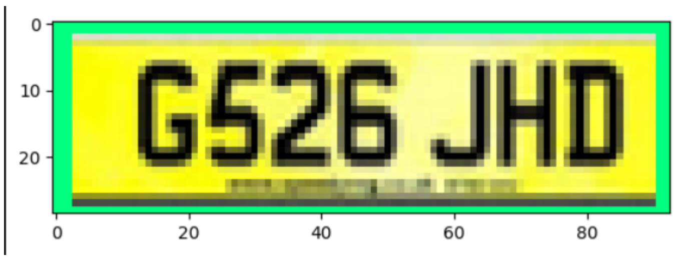
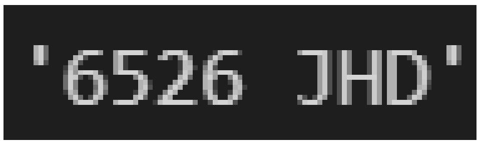


figure 2: license plate in focus.

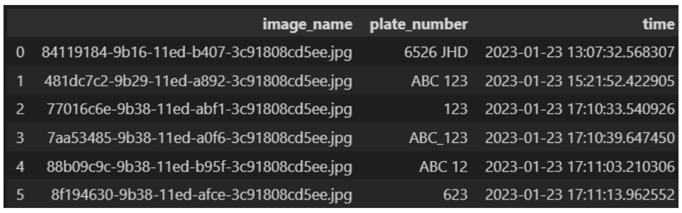
The next step is to apply the OCR model with filtering to extract the license plate number, which is displayed as text (as demonstrated in Figure 3).



```
'6526 JHD'
```

figure 3: OCR model result.

The resulting license plate number is then stored in a dataset along with the corresponding license plate image (as depicted in Figure 4).



	image_name	plate_number	time
0	84119184-9b16-11ed-b407-3c91808cd5ee.jpg	6526 JHD	2023-01-23 13:07:32.568307
1	481dc7c2-9b29-11ed-a892-3c91808cd5ee.jpg	ABC 123	2023-01-23 15:21:52.422905
2	77016c6e-9b38-11ed-abf1-3c91808cd5ee.jpg	123	2023-01-23 17:10:33.540926
3	7aa53485-9b38-11ed-a0f6-3c91808cd5ee.jpg	ABC_123	2023-01-23 17:10:39.647450
4	88b09c9c-9b38-11ed-b95f-3c91808cd5ee.jpg	ABC 12	2023-01-23 17:11:03.210306
5	8f194630-9b38-11ed-afce-3c91808cd5ee.jpg	623	2023-01-23 17:11:13.962552

figure 4: Dataset of the detected license plates.

In the live detection model, the license plate number of a car is detected in real-time from a webcam (as shown in Figure 5). The rest of the process is similar to detecting the license plate from an image

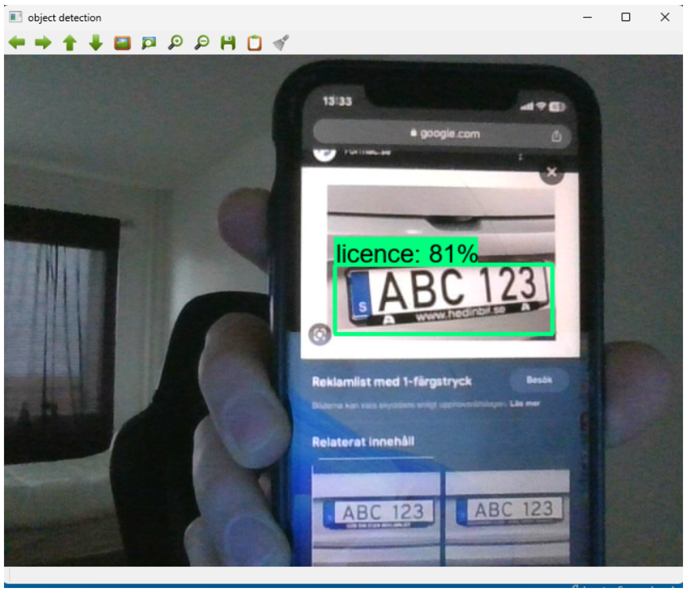
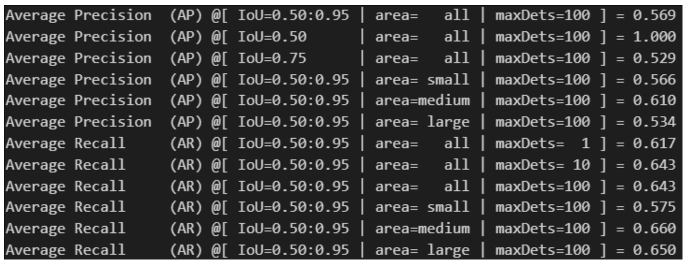


figure 5: license plate detected in real time from a webcam.

The results from evaluating the model, shows the Average Precision (AP) and Average Recall (AR) scores of the model at different Intersection over Union (IoU) thresholds and for different object sizes (small, medium, and large). The output shows that the model has an AP of 0.569 and a maximum recall of 0.643 as shown in figure 6.



```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.569
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 1.000
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.529
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.566
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.610
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.534
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.617
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.643
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.643
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.575
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.660
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.650
```

figure 6: Evaluating results of the model.

IV. DISCUSSION

In object detection tasks, the AP score is a widely used metric to evaluate the model's performance. An AP score of 1 implies that the model has accurately detected all the objects in the image, while a score of 0 indicates that the model has failed to detect any object. However, in most practical scenarios, achieving an AP score of 1 is challenging, and a moderate score is considered acceptable.

In our evaluation, we analyzed the AP and AR scores of the model at different IoU thresholds and for different object sizes. The output revealed that the model had an AP of 0.569, indicating that it had moderately successful performance in detecting objects in the images. The maximum recall achieved by the model was 0.643, which indicates that the model has the potential to detect a majority of the objects in the image.

It is important to note that the evaluation results are influenced by various factors such as the quality and size of the dataset, hyperparameters of the model, and the complexity of the object detection task.

The detection model has an overall decent performance when applied to images to detect license plates. However, there are instances where the model fails to highlight a green box around the license plate. This can be due to various factors such as poor lighting conditions, low quality of the image, or a license plate that is not clearly visible in the image.

To address this issue, the model can be fine-tuned by providing it with more diverse and challenging data to improve its ability to detect license plates accurately in different conditions. Additionally, advanced image processing techniques such as contrast enhancement, image sharpening, and noise reduction can be employed to enhance the quality of the images before they are fed into the model.

In the live detection mode, we encountered some challenges that affected the performance of the output results. Specifically, when we tested the model with a license plate image from a webcam, we observed that the webcam was lagging due to the CPU limitations, which impacted the quality of the image. Consequently, the OCR model did not perform well, resulting in low accuracy of the output results.

To address this issue, we need to optimize the CPU performance to ensure that the webcam can capture high-quality images. Additionally, we can explore other options for improving the OCR model accuracy, such as using deep learning models or applying image enhancement techniques to improve the quality of the images before feeding them to the OCR model. By addressing these challenges, we can enhance the performance of the live detection mode and ensure accurate and reliable results.

We have attempted to address the issue of detecting the same car multiple times, but we have encountered some reliability concerns with our current solution. Specifically, our OCR model has exhibited weak performance, leading to the generation of false license plate numbers. If these false numbers are detected and added to our dataset, there is a risk that customers will be charged twice for the same parking session.

In addition, we have also observed that the model may output false license plate numbers when a car exits the parking lot. This could result in the failure to properly end a parking session, which could cause significant problems for our customers.

Therefore, it is imperative that we address these reliability concerns in order to ensure the accuracy and effectiveness of our parking detection system. This may involve implementing a more robust OCR model, improving the quality of the images captured by our cameras, or exploring alternative solutions to these problems.

V. CONCLUSION

Overall, our evaluation has provided valuable insights into the performance of our object detection and license plate recognition models. While there is room for improvement, our model has demonstrated moderate success in detecting objects and license plates in different conditions. We have also identified several challenges and concerns that need to be addressed to enhance the reliability and accuracy of our parking detection system.

Moving forward, we recommend exploring advanced image processing techniques, optimizing CPU performance, and implementing more robust OCR models to enhance the performance of our system. Additionally, we need to ensure that the dataset used to train the models is diverse and comprehensive enough to capture the variations and complexities of the real-world scenarios.

By addressing these challenges and concerns, we can improve the quality of our parking detection system and provide accurate and reliable results to our customers.

VI. CONTRIBUTION

Adnan and Abdulkarim collaborated to address the issue of parking tickets by devising a plan that involved each team member completing 50% of the work. The team worked together to research the problem and identify suitable algorithms and libraries to solve it. Adnan was responsible for creating the environment and working on live image detection, while Abdulkarim worked on image detection and the README file. Both team members shared responsibility for training and evaluating the models. Additionally, both team members contributed equally to the writing process for each section of the report.

VII. REFERENCES

[1] https://www.kaggle.com/datasets/andrewmvd/car-plate-detection