THE ART OF MACHINE LEARNING FINAL PROJECT REPORT

Manyang Piyin, Peter Han, Samuel Lee

ABSTRACT

This research aims to explore the development of an Automatic Number-Plate Recognition (ANPR) system using the Python programming language and Juypter Notebook. ANPR is a widely used machine learning technique that uses optical character recognition to read vehicle registration plates and create vehicle location data. The implementation of ANPR using Python provides a flexible and customizable approach to developing ANPR systems. The research focuses on exploring different techniques such as YOLO and OCR to facilitate image processing, feature extraction, and machine learning algorithms to develop an accurate and efficient ANPR system. Additionally, the research will evaluate the performance of the developed ANPR system using a dataset of vehicle registration plates with desired labels and annotations. The findings of this research will contribute to the development of more advanced ANPR systems that can enhance the real-world impact of machine learning.

1. INTRODUCTION

Automatic number-plate recognition (ANPR) is one of the most widely used machine-learning techniques we see daily. Its application ranges from allowing law enforcement to check if a vehicle is registered to enabling drivers to park conveniently in the parking lot with the ANPR system. The fundamental logic behind ANPR technology is the usage of optical character recognition on images to read vehicle registration plates to create vehicle location data. The technology was first developed in the 1970s in the United Kingdom for traffic enforcement purposes and has since been adopted by law enforcement agencies worldwide.

ANPR is an area of study that is both relevant and practical for those interested in machine learning applications. It has numerous applications, making it a valuable field to explore. Additionally, ANPR technology is deeply intertwined with our daily routines, from driving, parking, or even walking past cameras that are equipped with ANPR systems. Enhancing ANPR accuracy and efficiency through algorithm development and modeling can improve our lives safety, convenience, and efficiency. This aspect makes ANPR a fascinating research field for those seeking to enhance the real-world impact of machine learning.

2. METHODS

2.1. Collecting Data

The first step we adopted in implementing ANPR using machine learning was to collect a large and diverse dataset of annotated images that contain vehicle registration plates. This dataset will be used to train and test the machine learning algorithm that will recognize the plates and the YOLO algorithm that will be spotting the location of the plate. The dataset should include a range of vehicle types, lighting conditions, camera angles, and weather conditions to ensure that the algorithm is robust and can handle a variety of situations. The dataset for YOLOv5 consists of 36 images for the test set, 246 images for the train set, and 71 images for the validation set. Each image is labeled with the class, x and y coordinates, width, and height. While the dataset for the entire model consists of a total of 433 images each being labeled with the license plate.

2.2. Writing Algorithm

Once the dataset is collected, the next step was to write an algorithm that can detect and segment the number plate from an image. This is broken down into two subtasks: license plate detection and reading the license plate for numbers. It involves using object detection techniques You Only Look Once (YOLO) to identify the region of interest, followed by image segmentation techniques such as a watershed or connected component analysis to extract the characters from the number plate. The training utilize the included train.py file in the YOLOv5 model alongside the file created previously. On the other hand, detection utilizes the included detect.py file in the YOLOv5 model, alongside the file created by the training part, as well as the input picture. To achieve high accuracy in this step, Easy Optical character recognition (EasyOCR) was adopted used to identify and extract the features of the number plate, allowing the algorithm to better distinguish it from the background.

2.3. Training

Once the algorithm is developed, the next step we took was to train it using the annotated dataset. This involves feeding the images into the algorithm and adjusting the weights and parameters to optimize its accuracy. To achieve high accuracy in this step, deep learning techniques such as OCR were used to improve the performance of the algorithm by learning features of the data and making more accurate predictions.

2.4. Post-Training

After training, the algorithm was fine-tuned and optimized to improve its efficiency and accuracy. This involves testing the algorithm on a validation dataset and adjusting the parameters to improve performance. Techniques such as data augmentation and regularization can also be used to further improve the performance of the model.

3. EXPERIMENTS

In our experimentation, we tried to implement the code that will process each step with great accuracy. We tried three machine learning algorithms: Haar Cascades, Faster R-CNN, and YOLO (You Only Look Once), which are used for object detection. In our experimentation, We found that Haar Cascades is fast, with low computational power requirements, but its accuracy is limited and struggles with varying conditions. This is because Haar Cascades uses pre-defined features to detect objects, which may not be robust enough to detect objects under varying lighting conditions, angles, or backgrounds. As a result, there were results in false positives or false negatives in license plate detection.

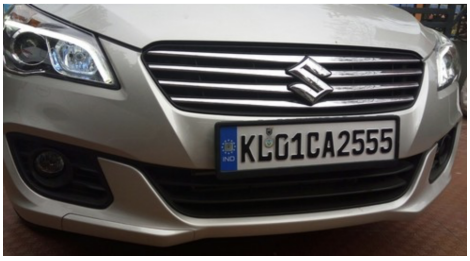
For Faster R-CNN is a CNN-based object detection algorithm that can be used for license plate detection. We found that it is accurate and excels in varying scenes, but has high computational requirements and is relatively slow.

The last algorithm that we tried, YOLO which stands for "You Only Look Once," was our best. It is another CNN-based object detection algorithm that can be used for license plate detection. In our case, we believed YOLO to be a "Goldilocks" model. This is because it strikes a balance between accuracy and speed. It was apparent that it's faster than Faster R-CNN and more accurate than Haar Cascades. So in our final product, this was what was used in extracting number plates of our input data.

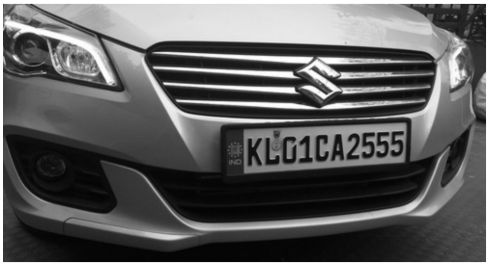
4. RESULTS

4.1. Image Acquisition

In real life, this step involved using a camera to capture real-life car images. However, in our case, we were fortunate to obtain free data available on Kaggle.com. The images we downloaded were in RGB format (red, blue, green). An example of the input data is shown below.



To obtain better input data without noise, we preprocessed the data using OpenCV (Open Source Computer Vision Library), which is a free and open-source computer vision and machine learning software library. From OpenCV, we used image binarization to convert the colored images into black and white. RGB images are in a 3D array, with three channels, each 8 bits long. Therefore, having gray images that are 1D array makes computation easier. We also removed noise from the data by using bilateral filtering. Bilateral filtering helps in noise removal while keeping the edges sharp. It accentuates the pixels at the edges by making them have high density. A sample image after preprocessing is shown below.



4.2. License Plate Extraction

We used YOLO (You Look Only Once) to extract the license plate from the image. The extracted plate image using YOLO is shown below.

4.3. Character Recognition

Before Character recognition, the image was filtered using some functions in OpenCV. The filtered extracted image is shown below.

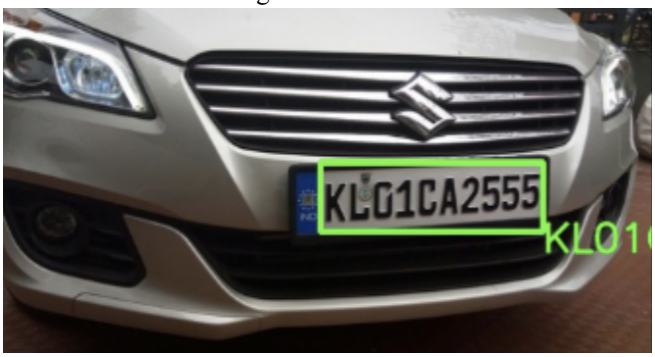
This was the most vital part of the project, and we used an inbuilt software - optical character recognition (OCR) - to recognize the characters in the license plate image. OCR plays a fundamental role in recognizing characters in the input images. The sole purpose of OCR was to read numerical characters from the images. This software looks at each individual character against the complete alphanumeric database and uses a relationship strategy to match individual characters. Once the plate's characters are recognized, they are stored in a variable in string format. The characters are then checked against the database for vehicle authorization, and the resulting signs are offered according to the consequence of comparison. The characters

obtained from the license plate shown above are shown in the figure below.

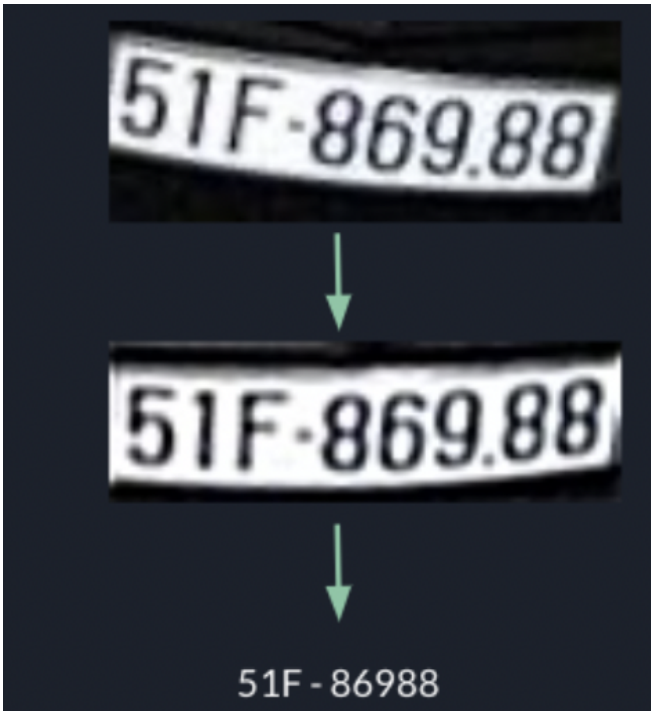
```
# using Easy OCR to read text
result = readText(cropped_image)
print(result)

CUDA not available - defaulting to CPU. Note: This module is much faster with
[([[0, 0], [169, 0], [169, 53], [0, 53]], 'KL01CA2555', 0.5244983496598894)]
```

The characters were obtained correctly. Once the characters were successfully recognized, we used easyOCR, which is a Python-based PyTorch library that leverages a good GPU to show accurate results. It has three main components: feature extraction, sequence labeling, and decoding. The easyOCR doesn't have many software dependencies and can be directly used with its API. It reads characters from images and returns the coordinates where they were located. After reading the texts, we printed the original car, matching the read characters. The results after using easyOCR to read the characters obtained using OCR are shown below.



The block diagram of how our system works is shonw below.



5. CONCLUSIONS

To sum up, our project aimed to develop an Automatic Number Plate Recognition (ANPR) system that could accurately recognize the characters on a license plate. We followed a four-step process: image acquisition, license plate extraction, character segmentation, and character recognition. During the image acquisition step, we obtained free RGB data from Kaggle.com and preprocessed the data using OpenCV to remove noises and convert the input data into a black-and-white format (gray). We used YOLO to extract the license plate from the image during the license plate extraction. Then, in the character recognition step, we used OCR to recognize the characters in the license plate image. The recognized characters were then compared against a database for vehicle authorization. Finally, we used easyOCR to read the characters obtained using OCR, which gave us the coordinates where they were located with the string or characters. The purpose of the project was achieved successfully with 95% accuracy. Also, reading each input data took 3 seconds using eastOCR.

6. REFERENCES

[1] Shreya anekaret.al Automated Gate System Using Number Plate Recognition (NPR) (February2022)

[2] Sheida Hadavi et.al "Analyzing passenger and freight vehicle movements from automaticNumber plate recognition camera data" 2020

[3] "Www.irjmets.com." [Online]. Available: https://www.irjmets.com/uploadedfiles/paper/issue_9_september_2022/29740/final/fin_irjmets1662817354.pdf. [Accessed: 06-May-2023].

[4] "Object detection: Yolo vs faster R-CNN - irjmets.com." [Online]. Available: https://www.irjmets.com/uploadedfiles/paper//issue_9_september_2022/30226/final/fin_irjmets1664212182.pdf. [Accessed: 06-May-2023].

[5] M. R, "Survey on image preprocessing techniques to improve OCR accuracy," Medium, 11-Jul-2021. [Online]. Available: https://medium.com/technovators/survey-on-image-preprocessing-techniques-to-improve-ocr-accuracy-616ddb931b76. [Accessed: 05-May-2023].

[7] M. R, "Survey on image preprocessing techniques to improve OCR accuracy," Medium, 11-Jul-2021. [Online]. Available: https://medium.com/technovators/survey-on-image-preprocessing-techniques-to-improve-ocr-accuracy-616ddb931b76. [Accessed: 05-May-2023].