EECS 504 Project Report
License Plate Recognition

Ziru Wang, Zhiran Wang, Lin Jia, Tianyang Jiang

November 30,2023

1 Executive Overview

In order to get a better parking experience in the parking lot, we are trying to develop a license plate recognition technology so that people can enter and exit the parking lot in the winter without opening the window. To recognize a license plate, there are two essential steps: to locate the plate, to recognize the characters. For the first step, to locate the plate, find all contours in the image and consider conditions such as width, height, aspect ratio, and size of the contours which helps to find the potential characters. Then we group potential characters together by considering characters that are in close proximity to each other and have similar sizes and aspect ratios. Then possible plates can be found. The next step is to separate the characters in the plate and then recognize them. And within each possible plate, suppose the longest list of potential matching chars is the actual list of chars. We use LeNet to do deep learning and recognize the chars and suppose the plate with the most recognized chars is the actual plate. The final step is just output the chars detected. With this text, we can easily link to the parking lot and provide a better parking experience.

2 Background and Impact

This plate recognition system could be applied on public parking lots to help reduce material waste and improve the efficiency and raise the automation level. Our system is a software-based, modular system that is easy to be upgraded and modified according to distinct needs of different parking lots.

Compared to the widely used RFID (Radio-Frequency Identification) system, using computer vision to recognize number plates could save the trouble of assigning RFID cards to the customer and thus save the material cost of the parking lot. Also, using a software-based recognition system is more flexible and could be open to future update. With this easily-updated property, the system could have a longer life span. For example, if in the future the parking lot wants to identify the make of the car to ensure that no one is using fabricated number plates to get into the parking lot, this kind of feature could be added to the system without the necessity to purchasing new product, that could make the lifecycle of our system longer than the hardware-based system like RFID or so. For those parking lots that need higher security levels, more functions such as facial recognition and motion detection could be added.

3 Method

For the preprocessing part, we first increase image contrast by subtracting top hat image from gray image and adding black hat image, This meticulous process contributes to accentuating relevant features within the images, providing a foundation for subsequent stages of analysis. Following the contrast enhancement, we use a GaussianBlur function to suppress noise and smooth the image. By reducing the impact of irrelevant details and imperfections, the GaussianBlur step optimizes the images for more accurate and efficient processing in subsequent stages of the computer vision pipeline. The third step involves utilizing the adaptiveThreshold function, a critical component in our approach. This function facilitates the creation of binarized images, where pixel values are determined based on local neighborhoods. This adaptive thresholding technique enhances the segmentation of objects in varying lighting conditions, contributing to the robustness of our computer vision model.

These three interconnected steps, namely contrast enhancement, noise suppression, and adaptive thresholding, align closely with the core concepts covered in our course. By strategically integrating these preprocessing techniques, we ensure that the subsequent stages of our computer vision methodology operate on refined and optimized image data, ultimately enhancing the accuracy and reliability of our analysis.

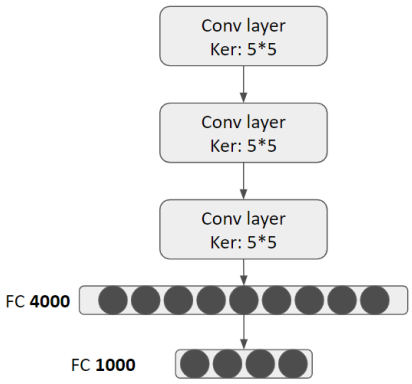

Figure 1: Model Structure

For the plate recognition part, we used the backbone of Lenet as the baseline model and we adjusted the width of the model to fit better for our classification task. Lenet is a traditional light weighted model designed to recognize handwritten and printed digits. It has overall 5 layers with learnable parameters and hence named lenet-5. The architecture of the model has 3 sets of convolution layers with a combination of average pooling. Following these, there are 2 fully connected layers.

However the ability of the architecture failed to predict the printed numbers and characters very well because the model parameters aren't enough to make a decent classification. To enhance the expressive ability of the model, we decided to add more neurons in the fully connected layer so that we get better classification results.

Another problem occurred in the training process is that we found our data isn't robust enough. In the real task, we will have the following images either cropped or noisy.

Figure 2: Character segmentation on real data(left) training data(right)

We need to enrich the dataset by randomly erasing and randomly adding noise. From 13163 images we applied two functions on it and we eventually got the final version dataset of 39163 images. With $epochs = 30$ and $lr = 0.001$, finally we get the training loss and test accuracy = 99.68%.

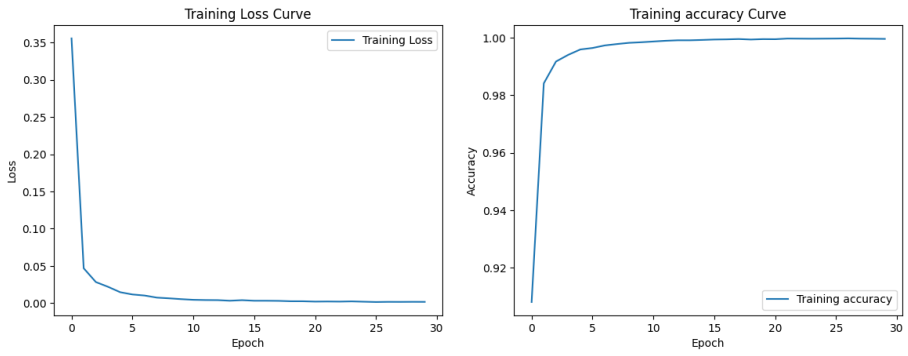

Figure 3: Training loss curve(left) training accuracy curve(right)

4 Prototype

The implementation includes First, in the preprocess part, we use top-hat operation and black-hat operation to enhance the contrast, use gaussian blur to suppress the noise and smooth the image, and use adaptive threshold to obtain binarized images. The next part is to locate all possible plates. We use the findContours function in OpenCV to find connected contours. Subsequently, we employ length range, width range, and aspect ratio to determine if the identified contours represent

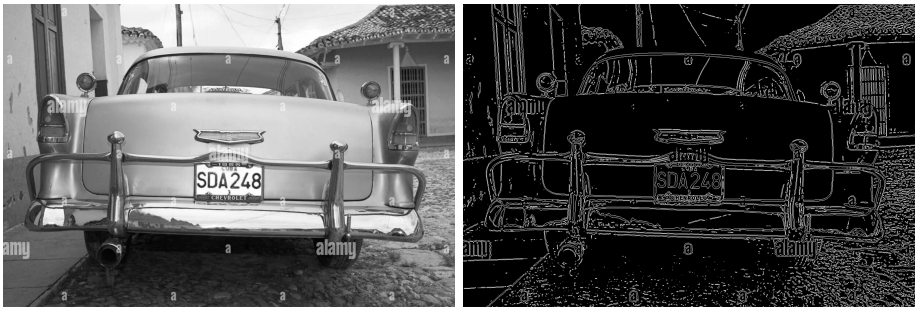

Figure 4: The left image is the original image and the right is its contours.

letters in a license plate, thereby obtaining potential license plate candidates. For all the identified potential license plates, we store information regarding their length, width, inclination angle, and coordinates for precise localization. Then, we proceed with the license plate segmentation. For

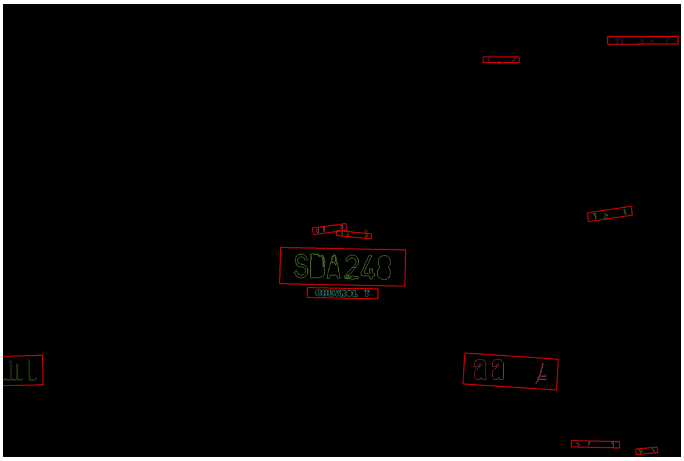

Figure 5: All possible plates

each located section of a license plate in the image, we perform segmentation and character extraction. We compare different characters within a license plate and discard smaller characters in case of overlap. Then, we use complete edges to segment characters, dividing a complete license plate image into multiple images, each containing a single letter, consistent with the size of our training dataset. Finally, the character recognition phase takes place. We utilize a custom-trained

Figure 6: Character segmentation

LeNet model to identify each character in the license plate. Subsequently, by comparing the lengths of recognized characters in each potential license plate, we select the longest string as the final recognized license plate.

5 Results

Our project prototype has been successfully implemented and tested on real data.The overall performance of our project is acceptable. Although for some of the test images our output is not perfect, we think this can be improved with more time and further learning. Our license plate

recognition system not only addresses the primary goal of providing a better parking experience by eliminating the need for window opening in winter but also offers potential applications in public parking lots. The software-based nature of the system makes it easily upgradable and adaptable to varying requirements, distinguishing it from traditional hardware-based systems like RFID.

In conclusion, our project successfully combines computer vision methods, data augmentation techniques, and deep learning models to create a robust license plate recognition system. The web interface enhances user accessibility, making the technology practical and user-friendly for real-world applications.

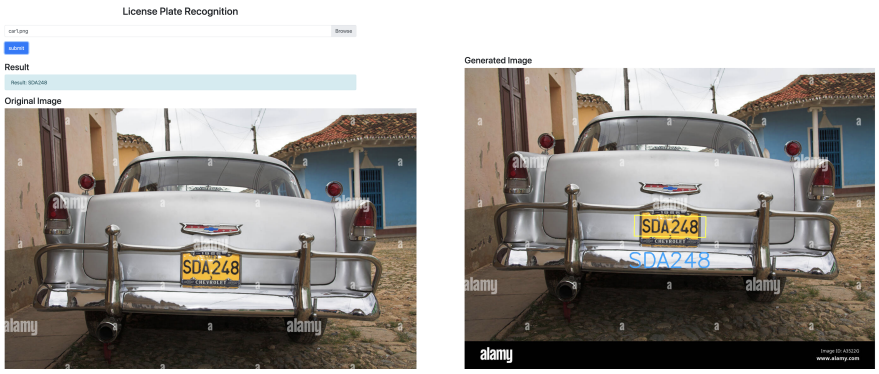

Figure 7: User Website Page

Here is the output of our project. To make it easier to run the entire code and display the final result, we created a web interface for license plate recognition. The interface uses the flask framework in python to complete the operation of the whole system locally, and carries out image input and result display through HTML and Javascript. Users only need to upload one image and click the submit button.

6 Github Link

https://github.com/LinJ002/EECS-504-Project.git