Assignment 2

MD SAKIB RAHMAN
Student ID: 120033990004
School of Electronics, Information and Electrical Engineering
Shanghai Jiao Tong University

I. Introduction

A machine vision system is a set of technologies that enables a computer to study, analyse, and recognize static or moving objects. A classic application of computer vision is license plate identification and recognition and its algorithms are separated into traditional methods and deep learning methods. Traditional approaches for detecting and recognizing license plates rely on morphological and color characteristics which may give poor performance in certain cases. On the other hand, deep learning methods need a huge number of data sets and computational resources for training in order to produce superior outcomes. In this assignment, I tried to implement both methods for detecting the license plate from images and recognizing the license number from the plate.

Fig. 1 LPR of the cars in an access control system

A. Task description

The main purpose of the task is to build a license plates detector and get the license number as a text. To do this, I've chosen to divide the job into four sub-tasks. These are the tasks:

1) Preprocessing of the dataset images.
2) Recognizing the license plate area correctly.
3) Extracting each letter and number from the plate.
4) Classifying each letter and number.

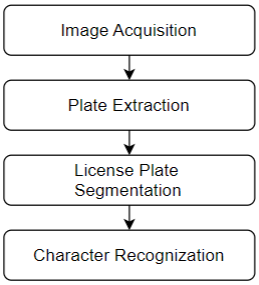


Fig. 2 Steps to recognize the license plate of a car

B. Dataset

For the traditional-based experiment, I utilize the dataset entitled "license plates.zip" supplied by our course instructor, which comprises 956 photos of automobiles coupled with Chinese license plates.

For deep learning-based approaches, I required a large number of datasets to train the model for better accuracy. The datasets must also be labeled. I trained a convolutional neural network (CNN) to recognise license plate characters using the Chinese City Parking Dataset (CCPD) [1]. This sequence of images was captured by parking charge collectors in a Chinese provincial capital. There are over 250k photos in the collection. Data labels were utilized to build bounding boxes for characters that were supplied into the network one at a time. OpenCV was used for preprocessing.

C. Experimental requirements

To retrieve the license plate image, the initial objective is to segment and assess the image based on the morphological and color aspects of the license plate. Object detection methods such as YOLO and SSD, as well as different image segmentation algorithm can be utilized for detection and segmentation. The second objective is to determine the four vertices and inclination angles of the license plate based on the segmented license plate images, so that it may be adjusted using the perspective transformation function given by OpenCV. Finally, each character on the license plate must be separated. For recognition, template matching have employed, or a CNN convolution neural network can be used to train recognition.

II. License plate recognition using traditional methods

A. Convert to binary image based on license plate color features

First, I need to convert the image from BGR to HSV color format where H stands for hue, S for saturation, and V stands for value. Its more straightforward color model enables adjusting the range of license plate color attributes easier. The primary color design for the blue license plate has an HSV color range of [100, 115, 115] to [124, 255, 255]. For other license plate HSV color range of [35, 10, 160] to [70, 100, 200].

The inRange function is then handed a binary picture through the HSV color range through OpenCV, with the pixel value in the color range being 255. The remaining values are 0, and Fig. 3 depicts images before and after getting the binary image. After obtaining the binary picture in this manner, the region that adheres to the color characteristics of the license plate can be retrieved, and the disturbance area may be avoided to a considerable extent.

Fig. 3 A binary image based on HSV color range

B. License plate location based on contour and morphological features

First perform morphological opening and closing operations on the binary image to remove some small noise points and partially integrate the license plate as a whole, as shown in Fig. 4, which is convenient for edge detection to obtain the outline of the entire license plate.

Then, following the opening and closing operations, discover all the contours in the binary image. The binary map is discovered using OpenCV's findContours function. Then, using the smallest rectangle, frame these outlines with a rectangle that conforms to the aspect ratio of the license plate. To transform all of the contours in the figure, utilize

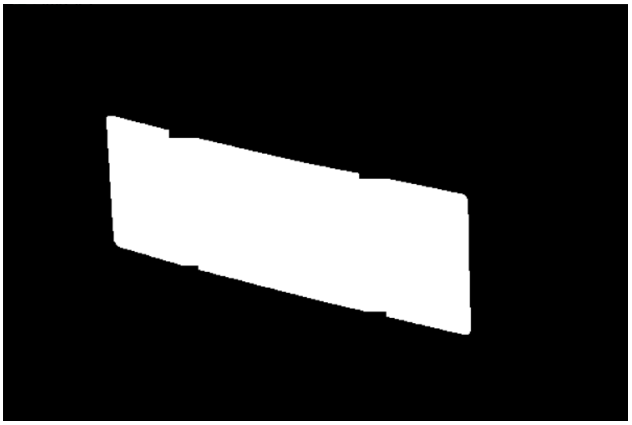


Fig. 4 Binary image after opening and closing operation

OpenCV's boundingRect and minAreaRect functions. The license plate is located in the rectangle-selected area. It is cropped the original picture.

Next To obtain the front view license plate, I did perspective modification on the segmented license plate. I utilized the smallest regular rectangle and oblique rectangle to frame the image one by one while also obtaining the angular position of the license plate in the photo. According to the license plate's length/width ratio Whether the broken rectangle satisfies the specifications. If it fits the requirements, the license plate region can be cropped from both the original and binary images, as illustrated in Fig. 5.

Fig. 5 Cropped License plate area

In this manner, the license plate region is effectively reduced, and the license plate image of the front view must be generated by perspective transformation sheet, which is useful for the final text recognition.

C. Perspective transformation based on basic image transformation

The cropped binary picture may be used to calculate the lowest value is x_1 and maximum value x_2 of the license plate on the x-axis as well as the difference $dx = x_2 - x_1$. At the same time, the y-axis minimum and maximum values y_1 and y_2 respectively. However, the difference dy is not equal to $y_2 - y_1$. Therefore, by dividing the area of the approximate parallelogram in the binary picture by the height dx, geometric methods may be utilized to calculate the base length dy. The resulting inclination angle may be used to determine if the license plate in the image is inclined to the left or right and various inclination directions have distinct representation ways for the license plate's apex. The coordinates of the license plate's four vertices may be stated using dx, dy, and inclination angle.

If the license plate is inclined to the right, then its four vertices can be approximated as $(x1, y2 - dy)$, $(x1, y2)$, $(x2, y1 + dy)$, $(x2, y1)$. If the license plate is inclined to the left, then its four vertices can be approximated as (x1, y1), (x1, y1 + dy), (x2, y2 - dy), (x2, y2). According to the four vertices of the license plate, through OpenCV's perspective transformation function getPerspectiveTransform function and The warpPerspective function obtains the perspective transformation matrix and performs perspective transformation on the cropped license plate area image, thus obtaining the front view of the license plate picture as shown in Fig. 6.

Fig. 6 Front view of license plate

D. Characters segmentation and template matching

As showing in Fig. 7, license plate now converting into a grayscale image and apply adaptive binary threshold. It should be noted that the text color and background color of the license plate varies for the blue license plate and other color license plates such as yellow, green and white. The binary picture of the blue license plate produces the opposite effect when compared to the other color. As a result, the image color of the other color license plate Flip must be the binary image should guarantee that the text half is white(255) and the background section is black(0).

Fig. 7 Process figure 1

To improve the robustness of the traditional technique, I created a function called backgroudChecking that determines whether the license plate background of license plate is blue or other color. This function adds the values of channel 0 (blue) and channel 1 (other color) of the complete license plate picture. If the blue value is larger, the license plate is considered blue. otherwise, it is considered other color.

Then, as illustrated in Fig. 8, the morphological opening operation is done on the binary image of the license plate to eliminate tiny noise spots, followed by the expansion operation to make each letter Expand as a whole. Following that, edge detection is applied to the whole character, and a rectangle is utilized to frame the discovered. A sequence of erosion's (to remove tiny regions) and dilation's (to lengthen larger regions) were applied in an attempt to smooth out the picture and remove small pockets of white space.

Fig. 8 Process figure 2

According to the aspect ratio of the character, the rectangle frame that meets the requirements can be obtained. The "·" symbol in the license plate will be ignored and it will be Automatic padding during plate matching. The crop the license plate binary image after the opening operation through a rectangular frame, and divide the characters on the license plate one by one cut out, as shown in Fig. 9.

Fig. 9 Separated Characters

Finally, go through each prepared character template one by one, first resizing the template to the same width as the

separated character, and then using OpenCV. The matchTemplate function computes the matching correlation coefficient and picks as the matching result the text corresponding to the template with the highest correlation coefficient. After matching each character picture, output the final recognition result.

III. License plate recognition using deep learning methods

A. Prepossessing

As previously mentioned, I trained and tested the deep learning algorithm using the CCPD dataset. The image filenames contain information such as bounding box positions and license plate numbers. A filename which contains label information. The information kinds are separated by hyphens, which are then separated further by underscores and ampersands. Area, tilt degree, bounding box coordinates, four vertices, and license plate number are the data kinds. Fig. 10 depicts these filenames. The files includes the license plate label. The first number corresponds to an index in the list of "provinces," the second number corresponds to an index in the list of "alphabets," and the final five numbers belong to indices in the list of "ads."

```
0255699233717-90_82-162&507_461&603-479&600_184&598_153&510_448&512-0_0_16_31_22_31_25-82-66.jpg
```

Fig. 10 Filename structure of CCPD dataset

The initial strategy was to crop the photos at the bounding box regions and feed the results to the neural network. Fig. 11 depicts the generated pictures. The model performed poorly when these photos were used. The photos are slanted at various degrees depending on where the photo taker was positioned at moment of the shot was taken.

Fig. 11 Cropped License plate area at bounding box location

Following the first technique, an attempt was made to create bounding boxes around the individual characters in order for the model to be trained on individual characters rather than the entire license plate picture in order to improve performance. To do this, The four vertices information in the picture filenames was utilized instead of the bounding box coordinates. This was accomplished through the use of the OpenCV function cv2.getPerspectiveTransform method. Two numpy arrays were passed to the function. The first represents the locations of the four vertices, while the second represents the intended output coordinates of those points.

cv2.getPerspectiveTransform creates a transformation matrix. This transformation matrix is passed to cv2.warpPerspective along with the input image. This creates the transformed image shown in Fig. 12 which is the license plate blown up. The image is then resized to 24 x 120.

After changing the preprocessing approach, bounding box locations could be made the same for all images. The resulting images are shown in Fig. 13.

The dataset was then split 80% for training/validation and 20% for the test set. The training/validation was further split into 75% for training and 25% for validation. All the datasets were one hot coded.

B. Convolution Neural Network

After adjusting the preprocessing to crop out individual characters for the training approach, the images were passed to the neural network at size 20 X 20. A network with three hidden layers and max pooling was used. Adding batch normalization at the end of each layer lead to significantly improved performance. The model summary for this CNN can be seen in figure 8 and the diagram can be seen in Fig. 14 . Output channels for the third hidden layer was 200.

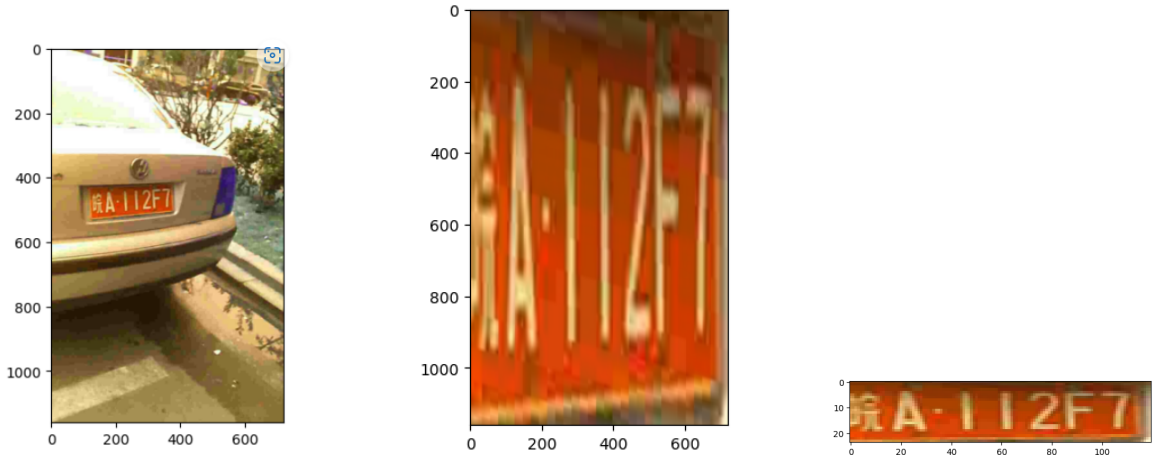


Fig. 12 Image preprocessing steps using OpenCV

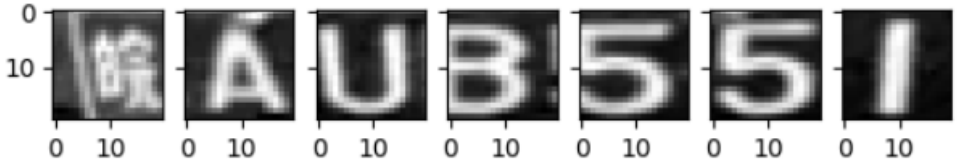


Fig. 13 Individual character images to be fed to the neural network

C. Evaluation Results

For the model, individual character images were passed in, and the accuracy significantly improved. First of all, CNN was trained on only 2000 images picked at random. Out of 2000 test license plates, 1995 were correctly recognized and an accuracy of 99.75% was achieved. The loss and accuracy on the training and validation sets per epoch of training shown in Fig. 15a and Fig. 16a.

In order to boost performance even more, the model was trained on 60,000 photos chosen at random. Fig. 16b shows that validation accuracy declined somewhat throughout training. When the model was tested on the test set, however, 59,965 out of 60,000 license plates were successfully detected, resulting in an accuracy of 99.94%. The loss and accuracy on the training and validation sets per epoch of training shown in Fig. 15b and Fig. 16b.

An example of a correctly recognized license plates areS shown in Fig. 17.

IV. Conclusion

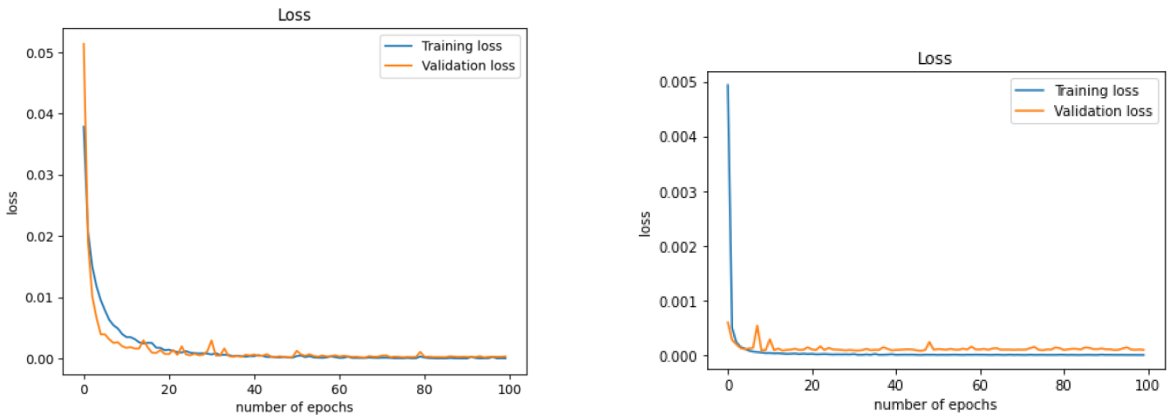
I attempted to incorporate both traditional and CNN-based solutions in this project. This work might be enhanced in a variety of ways. Traditional methods only work fine or blue plates, but because there is less data in other colors, it need to be more tested on multiple colors in a more particular fashion. To increase its capacity to recognize hazy or obstructed photos, CNN models might be trained using augmented data or photographs captured during unfavorable weather conditions. Additionally, the collection might be enhanced by include more photos of license plates from various Chinese regions.

References

[1] Xu, Z., Yang, W., Meng, A., Lu, N., and Huang, H., "Towards End-to-End License Plate Detection and Recognition: A Large Dataset and Baseline," *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 255–271.

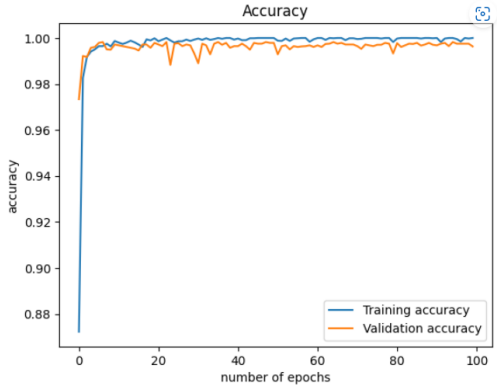
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 20, 20, 10)        260

batch_normalization (BatchN  (None, 20, 20, 10)       40
ormalization)

conv2d_1 (Conv2D)            (None, 20, 20, 40)        10040

max_pooling2d (MaxPooling2D  (None, 10, 10, 40)       0
)

batch_normalization_1 (Batc  (None, 10, 10, 40)       160
hNormalization)

conv2d_2 (Conv2D)            (None, 10, 10, 100)       100100

max_pooling2d_1 (MaxPooling  (None, 5, 5, 100)        0
2D)

batch_normalization_2 (Batc  (None, 5, 5, 100)        400
hNormalization)

conv2d_3 (Conv2D)            (None, 2, 2, 200)         180200

batch_normalization_3 (Batc  (None, 2, 2, 200)        800
hNormalization)

max_pooling2d_2 (MaxPooling  (None, 1, 1, 200)        0
2D)

conv2d_4 (Conv2D)            (None, 1, 1, 62)          12462

batch_normalization_4 (Batc  (None, 1, 1, 62)         248
hNormalization)

reshape (Reshape)            (None, 1, 62)             0

softmax (Softmax)            (None, 1, 62)             0

=================================================================
Total params: 304,710
Trainable params: 303,886
Non-trainable params: 824
```

Fig. 14 Model summary for CNN

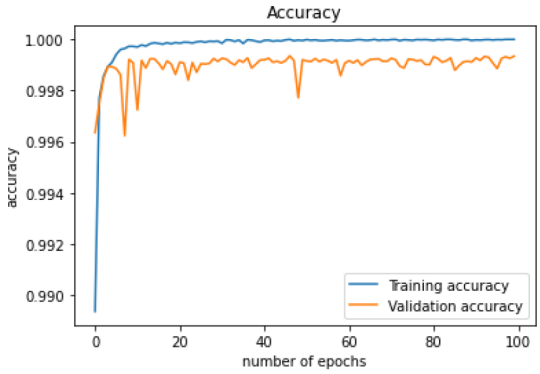


(a) For 2000 images **(b) For 60000 images**

Fig. 15 Loss from model during training with dataset

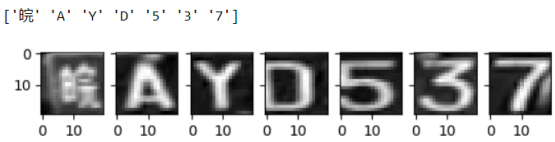


(a) For 2000 images

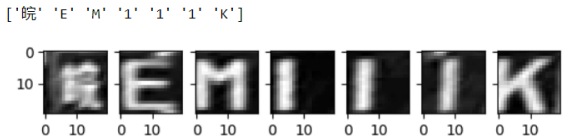


(b) For 60000 images

Fig. 16 Accuracy of model on the training and validation sets



(a) For 2000 images



(b) For 60000 images

Fig. 17 Correct Output