Automatic Car Plate Recognition

Mahima Sunny and Kenneth Collins

EE P 595

DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING, UNIVERSITY OF WASHINGTON, SEATTLE, WA, 98195

***Abstract* — Automatic license plate recognition is highly sought after technology for law enforcement agencies around the world. Here we use the Chinese City Parking Dataset (CCPD) to train a convolutional neural network (CNN) to detect the license plate characters. Data labels were used to create bounding boxes for characters which were individually fed to the network. Preprocessing was done using OpenCV. A license plate detection accuracy of 99.9% was achieved.**

I. INTRODUCTION

Automatic license plate detection is now being widely used by law enforcement to search for and detect stolen vehicles, for parking management, and to enforce traffic rules. According to the Insurance Information Institute, $6.4 Billion was lost to auto theft in the US in 2019 ("Facts + Statistics: Auto Theft", 2021).

To train the model, the Chinese City Parking Dataset was used ("CCPD (Chinese City Parking Dataset, ECCV)", 2021). This dataset consists of images taken by parking fee collectors in a Chinese provincial capital. These workers operate from morning until late at night regardless of weather conditions. When a parking bill is issued, the parking fee collector takes a picture of the car with an android device and records the license plate number. The result is that the images are from different angles, some are blurred, and some are taken in adverse weather (Xu et al., 2018). Figure 1 shows an example of one of the images in the CCPD dataset. The dataset contains over 250k images. The filenames of the images correspond to data about the image.

The structure of the model was a convolutional neural network designed to take in an image with one channel and output a probability for each character of the license plate with a list whose length is equal to the total number of classes. With limited success as far as accuracy detecting the test set, the initial approach was adjusted. The preprocessing strategy was adjusted so that the CNN could take in individual plate characters. This resulted in a significant increase in accuracy.

Figure 1: Sample image from the CCPD dataset.

II. PREPROCESSING

As previously mentioned, the filenames of the images contain info including bounding box locations and license plate number. These filenames are shown in figures 2 and 3. The initial approach was to crop the images at the bounding box locations and pass this to the neural network. The resulting images are shown in figure 4. Using these images resulted in poor model performance. The images are slanted at different angles depending on where the parking fee collector was standing when taking the photo.

Figure 2: Example filename containing label information. The information types are divided by hyphens, then further subdivided by underscores, and then ampersands. The data types are area, tilt degree, bounding box coordinates, four vertices, and license plate number.

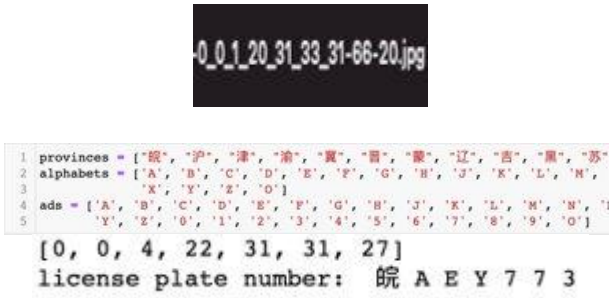

Figure 3: License plate label included in filename. The first number corresponds to an index in the "provinces" list, the second number corresponds to an index in the "alphabets" list, and the last five numbers correspond to indices in the "ads" list.

Figure 4: Images cropped at bounding box location.

After the initial approach, in order to increase performance, an attempt to put bounding boxes around the individual characters was made so that the model could be trained on individual characters instead of the whole license plate image. In order to do this, the preprocessing implementation was changed. Instead of using the bounding box coordinates, the four vertices information in the image filenames was used. This was done using the OpenCV function "cv2.getPerspectiveTransform". Two numpy arrays were passed to the function. One representing the four vertices locations, the second representing the desired output positions of those points. This code is shown in figure 6. "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 figure 5 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 figure 7. The data 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.

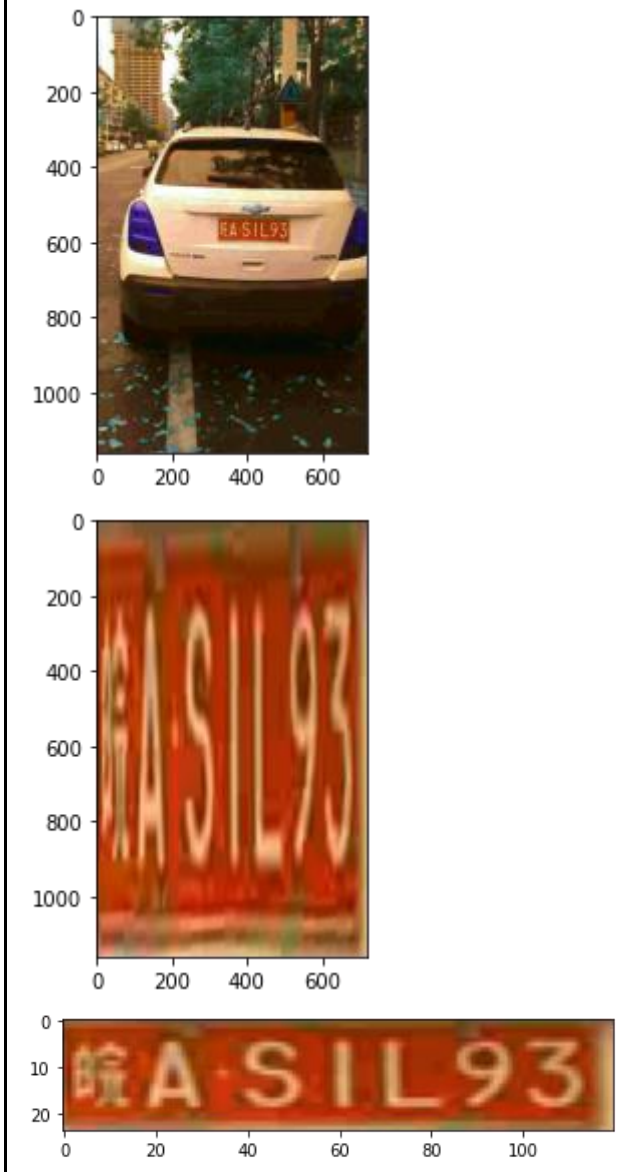

Figure 5: Image preprocessing steps using OpenCV.

```
    pts1 = np.float32([[coord[4], coord[5]],\
                       [coord[6], coord[7]],\
                       [coord[2], coord[3]],\
                       [coord[0], coord[1]]])
    pts2 = np.float32([[0, 0], [img.shape[1], 0],\
                       [0, img.shape[0] ],\
                       [img.shape[1] , img.shape[0] ]])
    M = cv2.getPerspectiveTransform(pts1, pts2)
    outimg = cv2.warpPerspective(img, M, img.shape[1::-1])
```

Figure 6: Preprocessing code used to get uniform character locations.

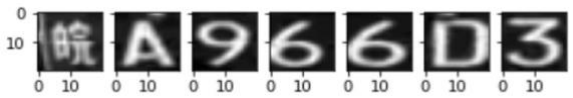

Figure 7: Individual character images to be fed to the neural network.

III. Convolution Neural Network

For the initial approach, the images as shown in figure 4 were passed to the neural network at size 24 X 120. A network with three hidden layers and max pooling was initially 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 figure 9. However, this approach gave an accuracy of 52% only.

For the second approach, after adjusting the preprocessing to crop out individual characters, the model was adjusted to take in an input of 20 x 20. Also the number of output channels for the third hidden layer was reduced from 800 to 200. Again max pooling and batch normalization is used as shown in figure 10.

Layer (type)	Output Shape	Param #
conv2d_92 (Conv2D)	(None, 24, 120, 10)	260
max_pooling2d_56 (MaxPooling	(None, 12, 60, 10)	0
batch_normalization (BatchNo	(None, 12, 60, 10)	40
conv2d_93 (Conv2D)	(None, 12, 60, 40)	10040
max_pooling2d_57 (MaxPooling	(None, 6, 30, 40)	0
batch_normalization_1 (Batch	(None, 6, 30, 40)	160
conv2d_94 (Conv2D)	(None, 6, 30, 100)	100100
max_pooling2d_58 (MaxPooling	(None, 3, 15, 100)	0
batch_normalization_2 (Batch	(None, 3, 15, 100)	400
conv2d_95 (Conv2D)	(None, 1, 7, 800)	720800
batch_normalization_3 (Batch	(None, 1, 7, 800)	3200
conv2d_96 (Conv2D)	(None, 1, 7, 49)	39249
batch_normalization_4 (Batch	(None, 1, 7, 49)	196
reshape_18 (Reshape)	(None, 7, 49)	0
softmax_17 (Softmax)	(None, 7, 49)	0

Total params: 874,445
Trainable params: 872,447
Non-trainable params: 1,998

None

Figure 8: Model summary for initial approach

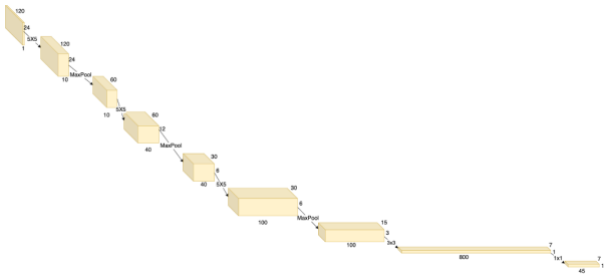

Figure 9: Initial approach CNN diagram

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 20, 20, 10)	260
batch_normalization (BatchNo	(None, 20, 20, 10)	40
conv2d_1 (Conv2D)	(None, 20, 20, 40)	10040
max_pooling2d (MaxPooling2D)	(None, 10, 10, 40)	0
batch_normalization_1 (Batch	(None, 10, 10, 40)	160
conv2d_2 (Conv2D)	(None, 10, 10, 100)	100100
max_pooling2d_1 (MaxPooling2	(None, 5, 5, 100)	0
batch_normalization_2 (Batch	(None, 5, 5, 100)	400
conv2d_3 (Conv2D)	(None, 2, 2, 200)	180200
batch_normalization_3 (Batch	(None, 2, 2, 200)	800
max_pooling2d_2 (MaxPooling2	(None, 1, 1, 200)	0
conv2d_4 (Conv2D)	(None, 1, 1, 48)	9648
batch_normalization_4 (Batch	(None, 1, 1, 48)	192
reshape (Reshape)	(None, 1, 48)	0
softmax (Softmax)	(None, 1, 48)	0

Figure 10: Improved approach CNN diagram

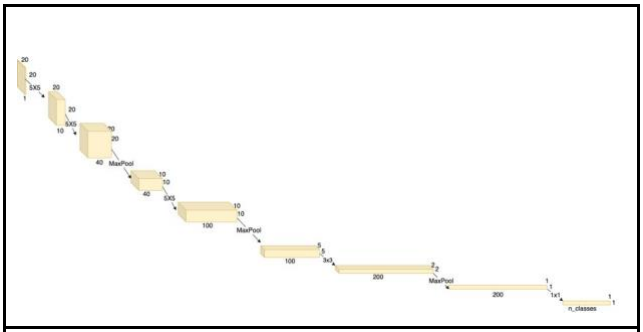

Figure 11: Improved CNN diagram

IV. Evaluation and Results

The accuracy of the initial model which took in the full license plate images of size 24 x 120 was not ideal. The accuracy on the training set was extremely high while the accuracy on the validation set was around 50%. This is most likely due to the low quality of the images and the different angles the pictures were taken from. Often the model had a hard time recognizing the Chinese province or mistaking G for B.

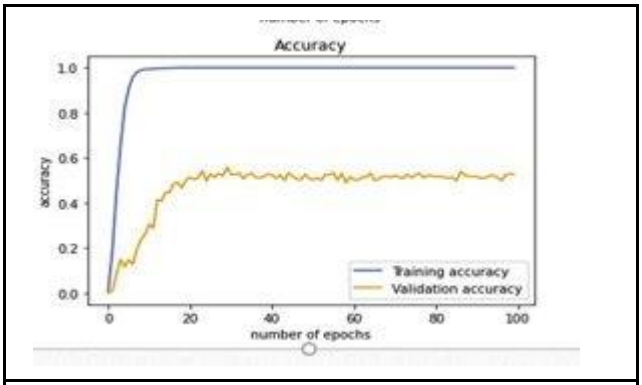

Figure 12: Accuracy of initial approach

For the second model, individual character images were passed in, and the accuracy significantly improved. The loss and accuracy on the training and validation sets per epoch of training shown in figures 13 and 14. This was with training 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. An example of a correctly recognized license plate is shown in figure 14.

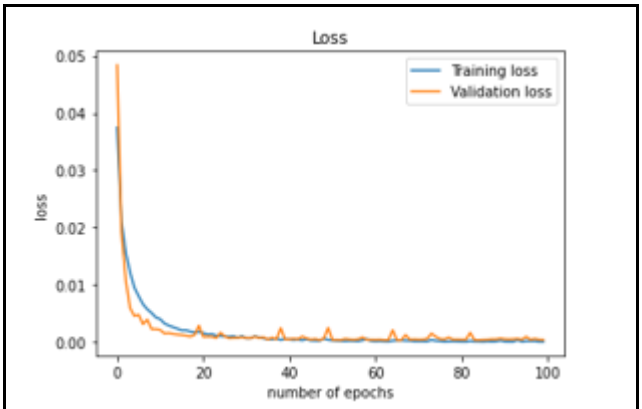

Figure 13: Loss from improved model during training with a dataset of 2000 images

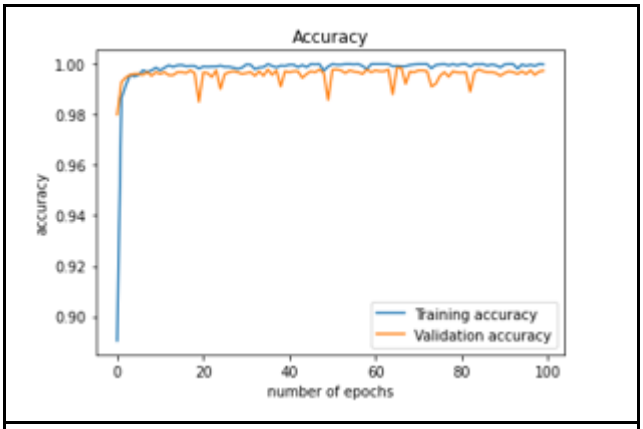

Figure 14: Accuracy of improved model on the training and validation sets using a dataset of 2000 images

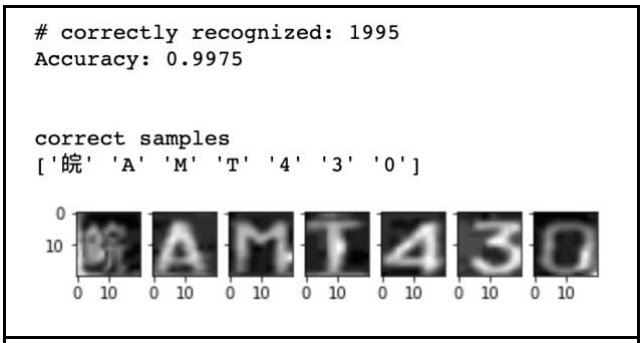

Figure 15: Output of the improved approach using a dataset of 2000 images

In an attempt to further improve performance, the model was trained on 60,000 images picked randomly. Validation accuracy during training actually decreased somewhat as shown in figure 17. However when the model was evaluated on the test set, 59,965 out of 60,000 license plates were correctly recognized for an accuracy of 99.94% shown in figure 18.

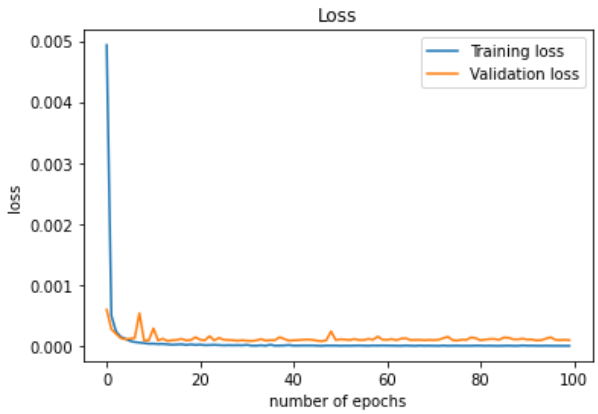

Figure 16: Loss from improved model during training with a dataset of 60000 images

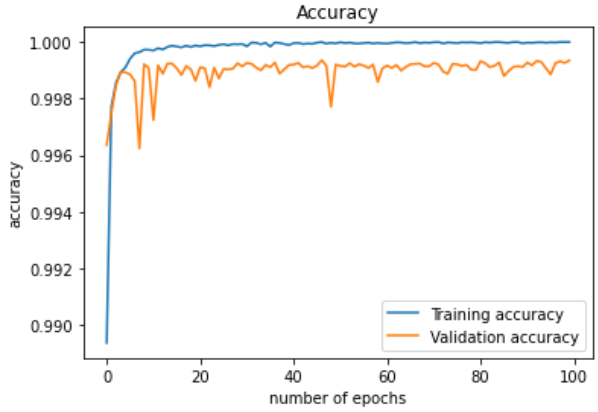

Figure 17: Accuracy of improved model on the training and validation sets using a dataset of 60000 images

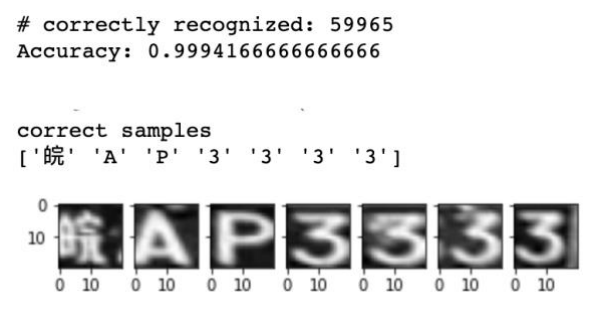

Figure 18: Output of the improved approach using a dataset of 60000 images

V. Future Work

This work could be improved in many ways. The model could be trained on augmented data or images taken during adverse weather conditions to improve its ability to detect blurry or obscured images. Also the dataset could be improved by adding more images of license plates from certain chinese provinces. The data set seems to be dominated by a few provinces with some being very rare.

The biggest form of improvement that would be made would be to get the neural network to detect the vertices of the license plate and find individual characters without preprocessing the images.

VI. Group Member Roles

The preprocessing code was written by Kenneth Collins. This includes downloading the dataset, creating labels, transforming the images, and one hot encoding. Mahima Sunny split the dataset, constructed the convolutional neural network, trained the model, and evaluated the performance. We collaborated with each other throughout and we both worked on the reports and presentation.

V. References

Xu, Z., Yang, W., Meng, A., Lu, N., Huang, H., Ying, C., & Huang, L. (2018). Towards end-to-end license plate detection and recognition. *Computer Vision Foundation.*

https://openaccess.thecvf.com/content_ECCV_2018/papers/Zhenbo_Xu_Towards_End-to-End_License_ECCV_2018_paper.pdf

(2021). *CCPD (Chinese city parking dataset, ECCV).* Github. https://github.com/detectRecog/CCPD

(2021). *Facts + statistics: auto theft.* Insurance information institute. https://www.iii.org/fact-statistic/facts-statistics-auto-theft