Chinese Car Plate Recognition

Question Restatement

In this project, the main task is to implement a neural network that processes car plate images which recognizes each character correctly in the picture. Image process skills are also practical in this task.

Problem Design and Analysis

There are two notebooks, $char\_cnn\_new$ and $car\ plate\ character\ segmentation$. The first one is responsible for the neural network model construction and the second one conduct edge detection, dilation, enhancement, and others preprocessing tasks then perform detection by using saved parameters. In the first notebook, the workflow is:

Load character data from the dataset subfolder

1. Create train data set and test data set
2. Decide some hyper-parameters
3. Data normalization
4. Create, compile, and train a CNN model for character recognition
5. Visualize training results
6. Model Evaluation and save parameters

In step 1, image files are in folders whose names are the class name. Store all the images in a list whose element is a dictionary with key is the image array and the value is the class name. In step 2, train and test set are divided by 80/20 rule. In order to get a better convergence, data normalization is applied to the input array, which squashes numbers to range from 0 to 1.

Here is the model structure:

Layer	**Description**
Input	$20 \times 20 \times 1$ gray image
Convolution $\mathbf{3} \times \mathbf{3}$	$18 \times 18 \times 32$ output
RELU	
Convolution $\mathbf{3} \times \mathbf{3}$	$16 \times 16 \times 64$ output
RELU	
Max pooling $\mathbf{2} \times \mathbf{2}$	$8 \times 8 \times 64$ output
Dropout	0.25 dropout rate
Flatten	4096 output
Fully connected	128 output
RELU	
Dropout	0.2 dropout rate
Fully connected	65 output
Softmax	

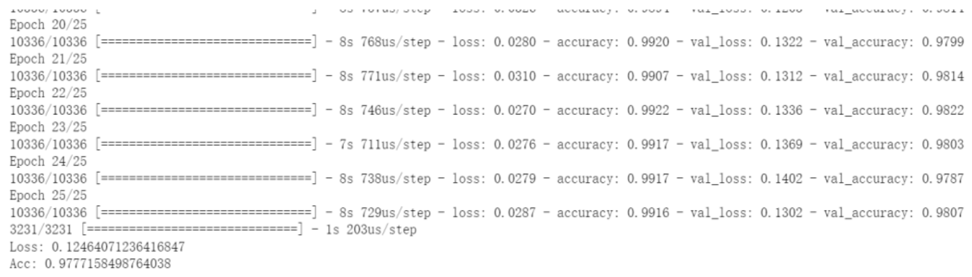
After model training, I plot accuracy and loss on training and validation sets to visualize model performance. Finally, evaluates our model on test set and saves parameters. In the second notebook the workflow is:

1. Load one car plate image
2. Transform to Grey Image
3. Apply canny edge detection and dilation
4. Label and region proposal
5. Retrieve each image character
6. Image enhancement
7. Character recognition

Most of the codes are given in this notebook, however, in order to improve performance, I modify code in part 3 and 6. In edge detection, $feature.canny()$ function is called. However, the sample code only make use of one of parameters $sigma$, standard deviation of the Gaussian filter. In order to get a better edge detection, I added $low\_threshold$ and $high\_threshold$, which are the lower bound and upper bound for hysteresis thresholding respectively. In the image enhancement, I also introduce gaussian blur to the gray scale picture to smooth the edges of each character by calling $cv2.GaussianBlur()$ with a 5×5 kernel.

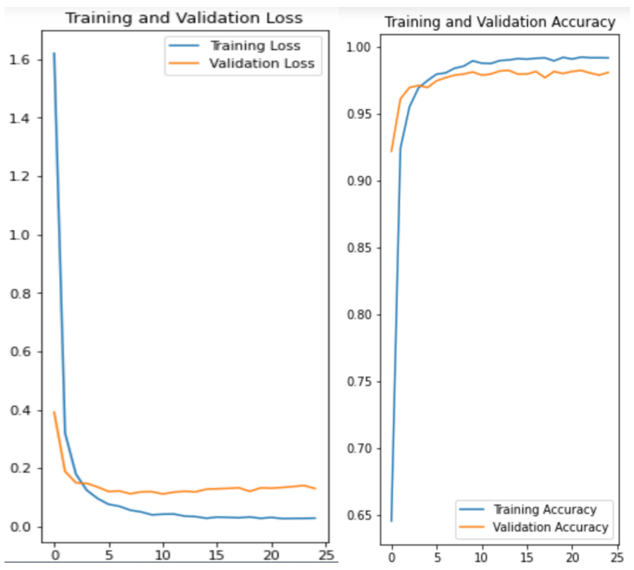
Result Presentation

Model Training:

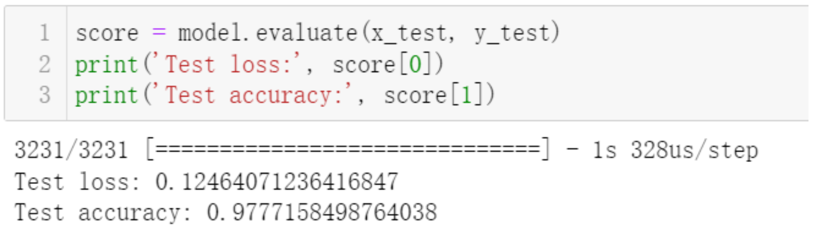


```
Epoch 20/25
10336/10336 [==============================] - 8s 768us/step - loss: 0.0280 - accuracy: 0.9920 - val_loss: 0.1322 - val_accuracy: 0.9799
Epoch 21/25
10336/10336 [==============================] - 8s 771us/step - loss: 0.0310 - accuracy: 0.9907 - val_loss: 0.1312 - val_accuracy: 0.9814
Epoch 22/25
10336/10336 [==============================] - 8s 746us/step - loss: 0.0270 - accuracy: 0.9922 - val_loss: 0.1336 - val_accuracy: 0.9822
Epoch 23/25
10336/10336 [==============================] - 7s 711us/step - loss: 0.0276 - accuracy: 0.9917 - val_loss: 0.1369 - val_accuracy: 0.9803
Epoch 24/25
10336/10336 [==============================] - 8s 738us/step - loss: 0.0279 - accuracy: 0.9917 - val_loss: 0.1402 - val_accuracy: 0.9787
Epoch 25/25
10336/10336 [==============================] - 8s 729us/step - loss: 0.0287 - accuracy: 0.9916 - val_loss: 0.1302 - val_accuracy: 0.9807
3231/3231 [==============================] - 1s 203us/step
Loss: 0.12464071236416847
Acc: 0.9777158498764038
```

Visualize result:



Model Evaluation:

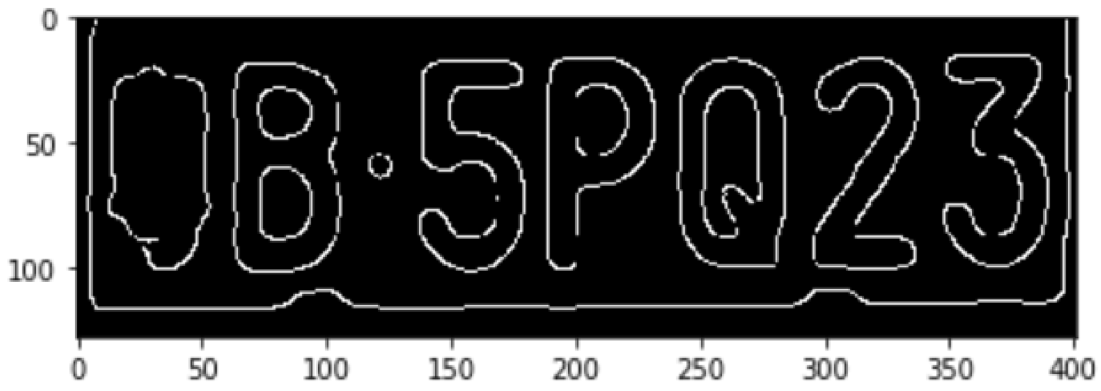


```
score = model.evaluate(x_test, y_test)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
3231/3231 [==============================] - 1s 328us/step
Test loss: 0.12464071236416847
Test accuracy: 0.9777158498764038
```

Edge Detection

Original:



Modified:

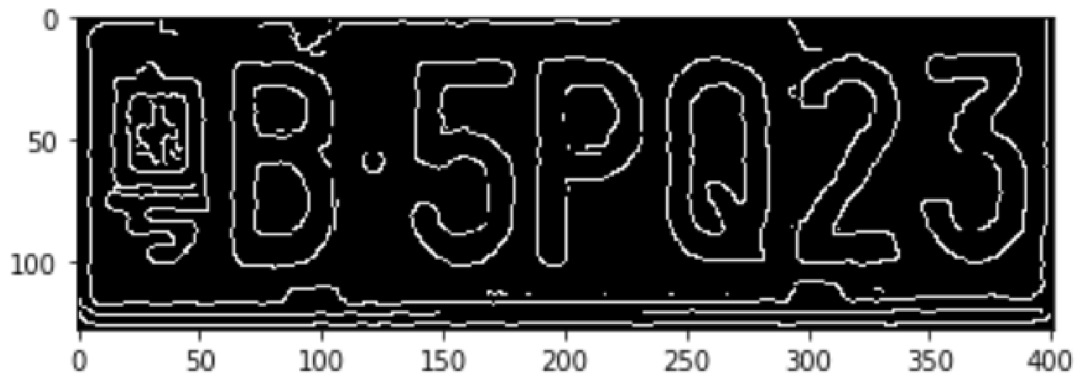
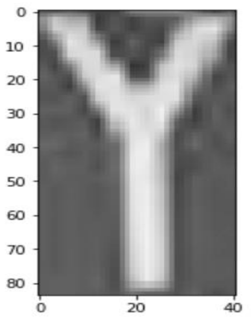
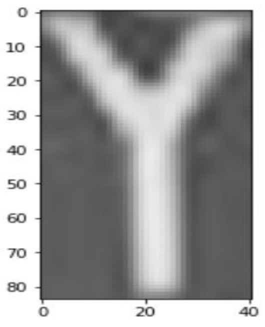


Image Enhancement:

Original:



Modified:



Main problem

In car plate recognition notebook, when I modified code, I encounter two new functions that I never seen or used, $feature.canny()$ and $cv2.GaussianBlur()$. In order to search parameters, I found the official document of skimage and explanation of $GaissianBlur()$ function.
https://scikit-image.org/docs/stable/api/skimage.feature.html#skimage.feature.canny,
https://blog.csdn.net/weixin_44657197/article/details/102679434