NYCU Pattern Recognition, Final Project

311553010, 陳姿羽

1. Environment details

a、 numpy: Version: 1.21.6
Read .csv file.

b、 OpenCV: Version: 4.7.0
Read image data, Resize the data.

c、 Pytorch and Cuda: Version: 1.13.1 + cu117
Loader data to model, Create CNN model, Set the loss function and optimizer.

d、 Matplotlib: Version: 3.5.3
Draw line graph for Training loss, Validation loss, Training accuracy and Validation accuracy.

e、 Pre-trained weight:
File name: model_Task2.pt, model_Task3.pt.

f、 Path setting:
As shown in the diagram below, the paths for the input data, output file, and pre-trained weights are set in the second cell.

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
import os
os.chdir('/content/drive/MyDrive')
os.listdir()

!pip install opencv-python
!pip install matplotlib
```

Out[1]:
```
"\nfrom google.colab import drive\ndrive.mount('/content/drive')\nimport
\n!pip install opencv-python\n!pip install matplotlib\n"
```

In [ ]
```
TRAIN_PATH = "kaggle/input/captcha-hacker-2023-spring/dataset/train"
TEST_PATH = "kaggle/input/captcha-hacker-2023-spring/dataset/test"
OUTPUT_PATH = "kaggle/output"
PRETRAINED_WEIGHT_PATH = "pre-trained"


device = "cuda"
Enable_Line_Graph = True
```

2. Implementation details

a、 Data pre-processing

I think that the color of the image is not very relevant to the text of captcha, so during data pre-processing, I converted the images to one channel. This method avoids the model from relying on color as a feature, and also simplifies the input data, reduces the complexity of the model's learning process.

First, I use csv reader function to read the contents of the “annotations.csv” file, split the data into 80% for training data and 20% for validation data. I create training data loaders and validation data loaders using DataLoader function from Pytorch. Next, I use OpenCV to load image data and convert the image from three channels to one. In Task 1, I randomly rotated the images around their center point by -30 to 30 degrees, to make the data more complex.

Finally, the image data is converted to a float tensor, the label data is converted to an integer tensor, and a tuple is formed with the corresponding label. After processing a task, print the information of the first image (image data, size, label), indicating that the data is loaded successfully.

b、 CNN model (Convolutional Neural Network)

I created a model using 9 convolutional layers, 5 max-pooling layers, and 2 fully connected layers for Task 1. Each convolutional layer has a kernel size of 3 and uses the ReLU activation function. The max-pooling layers have a kernel size of 2 and a stride of 2. The model follows a pattern of two convolutional layers followed by one pooling layer, with the last layer consisting of one convolutional layer followed by one pooling layer.

After each pooling layer, a dropout of 0.3 is applied to prevent overfitting of the model. Batch normalization is performed after every two convolutional layers to address the issues of vanishing or exploding gradients. The two fully connected layers are used to transform the output of the convolutional layers into the final predictions.

c、 Train CNN model

I use Adam algorithm to adjust the learning rate and update the parameters, with an initial learning rate set to 1e-3, and use Cross Entropy as the loss function. The batch size is set to 100. During the validation process, if the

input data is less than 100, use the torch.cat function to create a tensor array filled with 1s to pad the data up to 100. Then discard redundant results. The number of epochs is set to 50. After each epoch, the training accuracy and validation accuracy are recorded.

After the training is finished, I plot the training loss, validation loss, training accuracy, and validation accuracy as line graphs. In this way, it is more intuitive to evaluate whether the model is converging, whether there are problems such as overfitting or underfitting. And we can easily make necessary adjustments and improvements.

g、 CRNN model (Convolutional Recurrent Neural Network)

For Task 2 and Task 3, the order of captcha is important, so I use CRNN to implement. CRNN combines the characteristics of CNN and RNN. Firstly, CNN is used to extract features from the input images, and then RNN is employed for inference and sequence modeling.

I used a CNN layer composed of two convolutional layers and one max-pooling layer. The kernel size of the first convolutional layer is 9, and the second convolutional layer is (4, 3). Both convolutional layers uses the ReLU activation function. The max-pooling layer has a kernel size of 3 and a stride of 3. For the RNN layer, I used a single bidirectional GRU layer, hidden size is 1024. The CNN and RNN layers are connected using a linear layer, and the output is generated through another linear layer after RNN.

h、 Train CRNN model

Since the training speed slows down after surpassing 50 epochs, I divided the training process into two parts. In the first part of training, the initial learning rate was set to 0.001 and the number of epochs was set to 50. In the second part of training, I adjusted the learning rate to 0.0001 and reduced the number of epochs to 20 for Task 2, 50 for Task 3. After completing the first part of training, I saved the weights to "model_Task2.pt" and "model_Task3.pt". For the second part of training, I swapped the weights, using "model_Task3.pt" for Task 2 and "model_Task2.pt" for Task 3.

Like Task 1, I use the Adam algorithm to adjust the learning rate and update the parameters, and a batch size of 100. Additionally, I use CTCloss function as the loss function. During the calculation of the loss, I move the model's

output and labels to the CPU, then calculate their lengths, and finally use CTCloss to compute the loss. After each epoch, I record the training accuracy and validation accuracy. To prevent overfitting, I implemented an early stopping method. If the validation loss is consistently higher than the minimum validation loss for 8 epochs, the training process is stopped.

For calculating accuracy, I apply the softmax function to convert the output into the index of the character. Then I compare adjacent indices to check if they are the same character. After converting the output to a string, I compare it with the label to calculate the accuracy.

i、 Prediction

I read the test data from "sample_submission.csv" and "test" to make predictions. Similar to the validation process, I filled the input data which batch size less than 100 up to 100, and discarded the excess results after prediction. Finally, I converted the predicted results into strings and stored the image names and results in "submission.csv".