Report of Intro-to-Machine-Learning Homework5

109550027 紀竺均

a. Environment details
   I. Python version: 3.7.10
   II. Torch version: 1.11.0
   III. Numpy verseion: 1.21.6
   IV. %pip install captcha (version: 0.4)

b. Implementation details
   I. Data Preprocessing
   Label: One-hot encodes the string label into an numpy array with length = 36 * len(label).
   Image:
   1. Use cv2.cvtColor to turn it to a 1 channel gray scale image.
   2. Use ImageCaptcha in captcha.image to generate more training data. (5000 images for task2 and 20000 images for task3)

```
3. def gen_train_data(width, height, n_len, task, num_data=5000):
4.     n_class = len(NUM_ALPHA)
5.     for i in range(num_data):
6.         generator = ImageCaptcha(width=width, height=height)
7.         random_str = ''.join([random.choice(NUM_ALPHA) for j in
   range(n_len)])
8.         img = generator.generate_image(random_str)
9.
10.        with open(f'./{TRAIN_PATH}/annotations.csv', 'a',
   newline='') as csvfile:
11.            csv_writer = csv.writer(csvfile)
12.            csv_writer.writerow([f"task{task}/moretrain{i}.png",
   random_str])
13.        img.save(f"./{TRAIN_PATH}/task{task}/moretrain{i}.png")
14.
15.gen_train_data(72, 72, 2, 2)
16.gen_train_data(96, 72, 4, 3, num_data=20000)
```

II. Model architecture

```
self.conv1 = nn.Sequential(
            nn.Conv2d(1, 8, kernel_size=3),
            nn.BatchNorm2d(8),
            nn.Conv2d(8, 16, kernel_size=3),
            nn.AvgPool2d(2),
            nn.BatchNorm2d(16),
            nn.ReLU()
        )
        self.conv2 = nn.Sequential(
            nn.Conv2d(16, 128, kernel_size=5),
            nn.BatchNorm2d(128),
            nn.Conv2d(128, 128, kernel_size=3),
            nn.AvgPool2d(2),
            nn.BatchNorm2d(128),
            nn.ReLU()
        )
        self.conv3 = nn.Sequential(
            nn.Conv2d(128, 256, kernel_size=3),
            nn.BatchNorm2d(256),
            nn.Conv2d(256, 256, kernel_size=5),
            nn.MaxPool2d(2),
            nn.BatchNorm2d(256),
            nn.ReLU()
        )
        self.fc1 = nn.Linear(TEMP_OUT, 500)
        self.drop = nn.Dropout(0.2)
        # linear layer (100 -> 10)
        self.fc2 = nn.Linear(500, self.OUTPUT_LEN)
batch, height, width = x.shape
x = x.view(batch, 1, height, width)

# sequance of convolutional layers with relu activation
x = self.conv1(x)
x = self.conv2(x)
x = self.conv3(x)
x = self.drop(x)
# flatten the image input
```

```
x = x.view(-1, self.TEMP_OUT)
# 1st hidden layer with relu activation
x = F.relu(self.fc1(x))
# output-layer
x = self.fc2(x)
```

I mainly use 6 convolution layers to build my model.
For task1 and task2, with the input of 1 channel * 72 * 72 image, we will get a self.TEMP_OUT with size 4096, then we flatten it and use two linear function to get the output size of 1*36 (task1) and 1*72 (task2).
For task3, the input size is 72 * 96 , TEMP_OUT is 7168, and the output size is 1*144.
Optimizer and loss function:

```
optimizer = torch.optim.Adam(model.parameters(), lr=5e-4)
loss_fn = nn.CrossEntropyLoss()
```

III. Hyper-parameters

```
BATCH = 50
learning rate = 5e-4
epochs = 30, 100, 120
```

IV. Used deep learning framework

1. Convolution 2D layer * 6
2. Batch Normalization 2D layer * 6
3. Average pooling * 2 and Max pooling layer * 1
4. ReLU function
5. Dropout function
6. Linear layer * 2

c. Result

model link:
https://drive.google.com/drive/folders/1uSqNpoPYHxSTqEZA7FzLDl9aguZkMvGc?usp=sharing
(Note: I name task3's model "task4.pt" cause I need to make use of the length of the captcha.)
Public test case result: