HW5

Coding (100%):

Environment details:

使用 Kaggle 進行 train 與 inference ，因此環境設置與 Kaggle 一樣

python version: 3.7.12

accelerator: GPU P100

Python機器學習庫: PyTorch

```
import csv
import numpy as np
import random
import os

from PIL import Image

import torch as t
import torch.nn as nn
import torch.nn.functional as F
from torchvision import transforms as  T
from torch.utils.data import Dataset, DataLoader
```

Implementation details:

generate dataset

生成 training data set:

亂數生成由 1-9 、 a-z 組成的 code ，再使用以下套件生成圖片 總共生成近 1GB 的資料，再 Load 到 Kaggle 上

```
from captcha.image import ImageCaptcha
```

程式如附件：109550206_HW5_generate.ipynb

train

讀入 train data 時，先確 label 長度是否符合 task (因生成問題，存在部分問題資料)

Model 使用如下的 ResNet 架構

Model code:

```
class Residual(nn.Module):
    def __init__(self, inChannel, outChannel, stride=1):
        super(Residual, self).__init__()
        self.conv2_1 = nn.Conv2d(inChannel, outChannel, kernel_size=3, stride=stride, padding=1,
bias=False)
        self.bn2_1 = nn.BatchNorm2d(outChannel, track_running_stats=True)
        self.relu = nn.ReLU(inplace=True)
        self.conv2_2 = nn.Conv2d(outChannel, outChannel, kernel_size=3, stride=1, padding=1,
bias=False)
        self.bn2_2 = nn.BatchNorm2d(outChannel, track_running_stats=True)
        self.shortcut = nn.Sequential()
        if stride != 1 or inChannel != outChannel:
```

```
            self.shortcut = nn.Sequential(
                nn.Conv2d(inChannel, outChannel, kernel_size=1, stride=stride, bias=False),
                nn.BatchNorm2d(outChannel, track_running_stats=True)
            )

    def forward(self, x):
        out = self.conv2_1(x)
        out = self.bn2_1(out)
        out = self.relu(out)
        out = self.conv2_2(out)
        out = self.bn2_2(out)
        out += self.shortcut(x)
        out = self.relu(out)
        return out
```

```
class Model(nn.Module):
    def __init__(self, Residual, task='task1'):
        super(Model, self).__init__()
        self.task = task
        self.inChannel = 36
        self.conv1 = nn.Sequential(
            nn.Conv2d(3, 36, kernel_size=3, stride=1, padding=1, bias=False),
            nn.BatchNorm2d(36, track_running_stats=True),
            nn.ReLU()
        )
        self.layer1 = self.make_layer(Residual, outChannel=36, num=2, stride=1)
        self.layer2 = self.make_layer(Residual, outChannel=72, num=2, stride=2)
        self.layer3 = self.make_layer(Residual, outChannel=144, num=2, stride=2)
        self.layer4 = self.make_layer(Residual, outChannel=288, num=2, stride=2)
        self.fc1 = nn.Linear(288, numType(task))
        self.fc2 = nn.Linear(288, numType(task))
        self.fc3 = nn.Linear(288, numType(task))
        self.fc4 = nn.Linear(288, numType(task))

    def make_layer(self, Residual, outChannel, num, stride):
        layers = list()
        strides = [stride] + [1] * (num - 1)
        for stride in strides:
            layers.append(Residual(self.inChannel, outChannel, stride))
            self.inChannel = outChannel
        return nn.Sequential(*layers)

    def forward(self, x):       #print("IN")#print(x.shape)
        x = self.conv1(x)       #print("conv1")#print(x.shape)
        x = self.layer1(x)      #print("layer1")#print(x.shape)
        x = self.layer2(x)      #print("layer2")#print(x.shape)
        x = self.layer3(x)      #print("layer3")#print(x.shape)
        x = self.layer4(x)      #print("layer4")#print(x.shape)
        x = F.avg_pool2d(x, 9)  #print("avg_pool2d")#print(x.shape)
        x = x.view(-1, 288)
        y = list()
        y.append(self.fc1(x))
        if self.task != 'task1':
            y.append(self.fc2(x))
            if self.task != 'task2':
                y.append(self.fc3(x))
                y.append(self.fc4(x))
        return y
```

Model 架構:

```
Model(
  (conv1): Sequential(
    (0): Conv2d(3, 36, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (1): BatchNorm2d(36, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
  )
  (layer1): Sequential(
    (0): Residual(
      (conv2_1): Conv2d(36, 36, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2_1): BatchNorm2d(36, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2_2): Conv2d(36, 36, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2_2): BatchNorm2d(36, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (shortcut): Sequential()
    )
    (1): Residual(
      (conv2_1): Conv2d(36, 36, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2_1): BatchNorm2d(36, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2_2): Conv2d(36, 36, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2_2): BatchNorm2d(36, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (shortcut): Sequential()
    )
  )
  (layer2): Sequential(
    (0): Residual(
      (conv2_1): Conv2d(36, 72, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
      (bn2_1): BatchNorm2d(72, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2_2): Conv2d(72, 72, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2_2): BatchNorm2d(72, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (shortcut): Sequential(
        (0): Conv2d(36, 72, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(72, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (1): Residual(
      (conv2_1): Conv2d(72, 72, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2_1): BatchNorm2d(72, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2_2): Conv2d(72, 72, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2_2): BatchNorm2d(72, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (shortcut): Sequential()
    )
  )
  (layer3): Sequential(
    (0): Residual(
      (conv2_1): Conv2d(72, 144, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
      (bn2_1): BatchNorm2d(144, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2_2): Conv2d(144, 144, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2_2): BatchNorm2d(144, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (shortcut): Sequential(
        (0): Conv2d(72, 144, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(144, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (1): Residual(
      (conv2_1): Conv2d(144, 144, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2_1): BatchNorm2d(144, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2_2): Conv2d(144, 144, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
```

```
      (bn2_2): BatchNorm2d(144, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (shortcut): Sequential()
    )
  )
  (layer4): Sequential(
    (0): Residual(
      (conv2_1): Conv2d(144, 288, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
      (bn2_1): BatchNorm2d(288, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2_2): Conv2d(288, 288, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2_2): BatchNorm2d(288, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (shortcut): Sequential(
        (0): Conv2d(144, 288, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(288, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (1): Residual(
      (conv2_1): Conv2d(288, 288, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2_1): BatchNorm2d(288, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2_2): Conv2d(288, 288, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2_2): BatchNorm2d(288, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (shortcut): Sequential()
    )
  )
  (fc1): Linear(in_features=288, out_features=36, bias=True)
  (fc2): Linear(in_features=288, out_features=36, bias=True)
  (fc3): Linear(in_features=288, out_features=36, bias=True)
  (fc4): Linear(in_features=288, out_features=36, bias=True)
)
```

Hyperparameters:

```
#task1
learningRate = 1e-3
epochs = 100
batchSize = 128
```

```
#task2
learningRate = 1e-3
epochs = 300
batchSize = 256
```

```
#task3
learningRate = 1e-3
epochs = 300
batchSize = 256
```

save model as task[1-3]_model.pth

inference

Load test data from captcha-hacker.

Load model task[1-3]_model.pth.

Get all files which needed to predict from sample_submission.csv.

Predict and write into submission.csv.

Results

Submission and Description	Public Score	Select
submission.csv Complete · 1h ago	**0.996**	☐

14	**109550206**		0.99600	8	42m