深度視覺 Final Project 結果分析

M113040064 李冠宏

Data preparation

這邊首先會先掛載 google drive，並且將需要的資料解壓縮。

Training Section

Import packages

這邊首先引入一些必要的套件

```
import pandas as pd
import numpy as np
import cv2
import matplotlib.pyplot as plt
import torch
import torchvision
import torchvision.transforms as transforms
import csv
import os
import random
from torch import nn
from torch.utils.data import Dataset, DataLoader
from tqdm import tqdm
from PIL import Image
```

Augmented functions

這邊定義了做隨機資料增量的函式，其中包括了深度改變、邊緣小範圍裁切、長寬比改變、雜訊、平面旋轉和深度傾斜

```
# 隨機做資料增量之 function
def random_transform(image):

  #整體深度改變
  picked = random.choice([True, False])
  if(picked):
```

```
    rand = random.randint(-300, 300)
    to_tensor = transforms.ToTensor()
    image = to_tensor(image)
    image += rand
    to_PIL = transforms.ToPILImage()
    image = to_PIL(image)

  #邊緣小範圍裁切
  picked = random.choice([True, False])
  if(picked):
    random_number = random.randint(285, 295)
    transform = transforms.CenterCrop((random_number,
random_number))
    image = transform(image)

  #長寬比小範圍改變
  # picked = random.choice([True, False])
  # if(picked):
  #   random_number = random.randint(285, 295)
  #   transform = transforms.RandomResizedCrop((random_number,
random_number))
  #   image = transform(image)

    # 定義目標寬度和高度
    target_width = random.randint(290, 310)
    target_height = random.randint(290, 310)
    # 計算當前圖像的寬度和高度
    current_width, current_height = image.size
    # 計算調整後的寬度和高度
    if current_width / current_height > target_width /
target_height:
        # 圖像的寬度比目標寬度大，根據目標高度計算新的寬度
        new_width = int(target_height * (current_width /
current_height))
        new_height = target_height
    else:
        # 圖像的寬度比目標寬度小，根據目標寬度計算新的高度
        new_width = target_width
```

```
        new_height = int(target_width * (current_height / 
current_width))
    # 調整圖像大小以適應新的寬度和高度
    resized_image = image.resize((new_width, new_height))
    # 計算裁剪區域的邊界
    left = (new_width - target_width) // 2
    top = (new_height - target_height) // 2
    right = left + target_width
    bottom = top + target_height
    # 裁剪圖像
    image = resized_image.crop((left, top, right, bottom))

  #雜訊(高斯雜訊或少量鹽噪點)
  picked = random.choice([True, False])
  if(picked):
    transform = transforms.GaussianBlur(7,2)
    image = transform(image)


  #平面旋轉(約 15°內)
  picked = random.choice([True, False])
  if(picked):
    transform = transforms.RandomRotation(degrees=(-15, 15))
    image = transform(image)

  #深度傾斜
  picked = random.choice([True, False])
  if(picked):
    transform = transforms.RandomAffine(degrees=0, shear=(-20, 
20))
    image = transform(image)

  return image
```

而以下的函式將圖片的尺寸都變成 300*300

```
def resize(image):
  transform = transforms.Compose([transforms.Resize((300,
300))])
  resized_image = transform(image)

  return resized_image
```

Test the augmented image

這邊會測試剛剛的增量函式

```
# 測試資料增量之圖片
image = Image.open("/content/Mydata/images/1V727/0017.png")
image = random_transform(image)
image
```

結果:

CSV file with Anchor, Positive and Negative

這邊會將原本的圖片做資料增量，以產生 positive，並挑選一張和 anchor 不一樣的圖片當作是 negative，最後將這些檔案的路徑都寫進 CSV 檔案。

首先將標題寫進 CSV 檔案中

```
# 將標題列寫入 CSV 檔中
with open("/content/Mydata/train.csv", 'w', newline="") as
csvfile:
    writer = csv.writer(csvfile)
    writer.writerows([["Anchor", "Positive", "Negative"]])

with open("/content/Mydata/valid.csv", 'w', newline="") as
csvfile:
    writer = csv.writer(csvfile)
    writer.writerows([["Anchor", "Positive", "Negative"]])
```

這邊定義了一些變數和計算走訪資料夾的總數量

```
# variables
count = 0
total = 0
limit = float('inf')
pre = ""
valid_data = False

# 計算總資料數量
DATA_DIR = '/content/Mydata/images/'
for root, dirs, files in os.walk(DATA_DIR):
  for file in files:
    total += 1
```

以下是做資料增量產生 positive，挑選 negative 和將檔名寫進 CSV 檔案

```
# 做資料增量 + 將檔名寫入 CSV 檔
for root, dirs, files in os.walk(DATA_DIR):
  for file in files:
    file_path = os.path.join(root, file) #檔案路徑
    current_dir = file_path[:-8] #現在處於的資料夾 ->
/content/Mydata/images/1UR77/
```

```python
    current_file_name = file_path[-8:-4] #現在的檔案名稱 -> 0013
    count += 1 #總檔案數量

    #產生資料增量之圖片(positive)
    image = Image.open(file_path)
    augmented_image = random_transform(image)
    positive_path = current_dir + current_file_name + "_pos.png"
    augmented_image.save(positive_path)


    #挑選 negative
    file_list = os.listdir(current_dir)
    if len(file_list) <= 1:
      filtered_files = file_list
    else:
      # alternative: file[-8:] != file_path[-8:] and file[-7:] !=
"pos.png"
      # file[-8:] != file_path[-8:] and file[-12:] !=
f"{file_path[-8:-4]}_pos.png"
      filtered_files = [file for file in file_list if file[-8:] !=
file_path[-8:] and file[-7:] != "pos.png"]
    if filtered_files:
      for i in range(len(filtered_files)):
        filtered_files[i] = current_dir[23:] + filtered_files[i]


    #判斷要寫入 train data 還是 valid data
    if count > 50000 and pre[23:29] != file_path[23:29]:
      valid_data = True


    #將檔案寫入 csv 檔中
    for i in range(len(filtered_files)):
      if valid_data:
        with open("/content/Mydata/valid.csv", 'a', newline="")
as csvfile:
          writer = csv.writer(csvfile)
          writer.writerows([[file_path[23:], positive_path[23:],
filtered_files[i]]])
```

```
      else:
        with open("/content/Mydata/train.csv", 'a', newline="")
as csvfile:
          writer = csv.writer(csvfile)
          writer.writerows([[file_path[23:], positive_path[23:],
filtered_files[i]]])

    pre = file_path

    #tdqm
    print('\r' + 'Augmenting images progressing :[%s%s]%.2f%%;' %
(
    '█' * int(count*20/total), ' ' * (20-int(count*20/total)),
    float(count/total*100)), end='')

  if count > limit:
    break
```

執行結果:

```
Augmenting images progressing :[████████████████████]100.00%;
```

之後印出 CSV 檔案裡面的內容，確定有寫進去

```
#從 CSV 檔案載入 train data
train_df = pd.read_csv(TRAIN_CSV_FILE)
train_df = train_df.sample(n=10000, replace=True, random_state=1)
train_df.head()
```

	Anchor	Positive	Negative
128037	2GW05S/0019.png	2GW05S/0019_pos.png	2GW05S/0001.png
491755	3MS77S/0011.png	3MS77S/0011_pos.png	3MS77S/0001.png
491263	2GS81/0007.png	2GS81/0007_pos.png	2GS81/0014.png
836489	2H736/0003.png	2H736/0003_pos.png	2H736/0050.png
73349	1V975/0007.png	1V975/0007_pos.png	1V975/0002.png

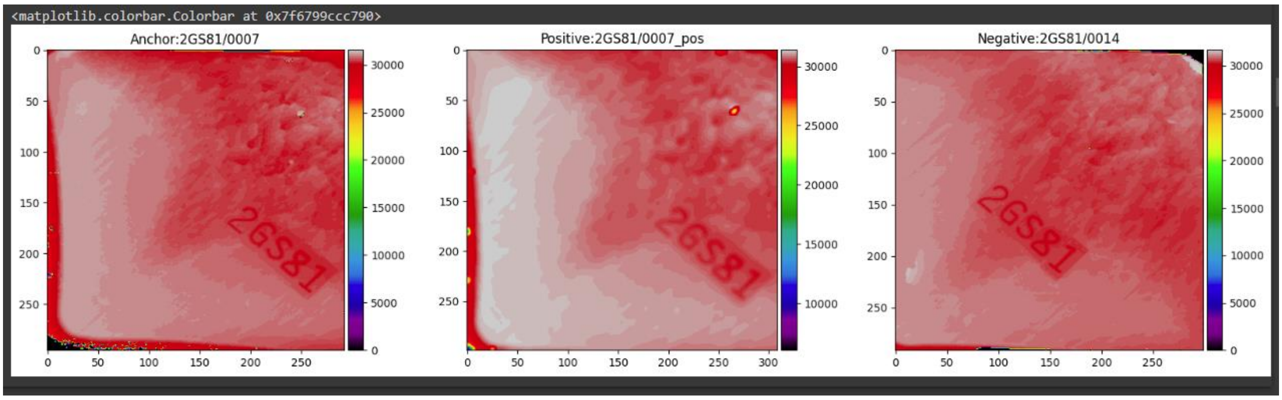
```
#從 CSV 檔案載入 valid data
valid_df = pd.read_csv(VALID_CSV_FILE)
```

```
valid_df = valid_df.sample(n=3000, replace=True, random_state=1)
valid_df.head()
```

	Anchor	Positive	Negative
128037	2H020/0035.png	2H020/0035_pos.png	2H020/0015.png
491755	1X572/0048.png	1X572/0048_pos.png	1X572/0018.png
470924	3MR92/0053.png	3MR92/0053_pos.png	3MR92/0037.png
791624	2H784/0057.png	2H784/0057_pos.png	2H784/0037.png
491263	1X572/0017.png	1X572/0017_pos.png	1X572/0003.png

Visualization of Anchor, Positive and Negative

這部分會視覺化原本的圖片，做過資料增量之 positive 和 negative



Dataset and Dataloader

這邊將資料載入 dataset，之後再透過 dataloader 做批次處理

Model

定義了 model

(ResNet50)

```
#定義model class
class Model(nn.Module):
  def __init__(self):
    super(Model, self).__init__()
    self.prelayer = nn.Conv2d(1, 3, kernel_size=1, bias=False)
```

```
    self.net = torchvision.models.resnet50()
  def forward(self, images):
    images = self.net(self.prelayer(images))
    return images
```

(ResNet + Attention)

```
class AttentionModule(nn.Module):
    def __init__(self, in_channels):
        super(AttentionModule, self).__init__()
        self.query_conv = nn.Conv2d(in_channels, in_channels // 8, kernel_size=1)
        self.key_conv = nn.Conv2d(in_channels, in_channels // 8, kernel_size=1)
        self.value_conv = nn.Conv2d(in_channels, in_channels, kernel_size=1)
        self.gamma = nn.Parameter(torch.zeros(1))

    def forward(self, x):
        # Reshape input tensor
        batch_size, channels, height, width = x.size()
        query = self.query_conv(x).view(batch_size, -1, height * width).permute(0, 2, 1)
        key = self.key_conv(x).view(batch_size, -1, height * width)
        value = self.value_conv(x).view(batch_size, -1, height * width)

        # Calculate attention map
        attention_map = F.softmax(torch.bmm(query, key), dim=2)

        # Apply attention map to value
        attention_out = torch.bmm(value, attention_map.permute(0, 2, 1))
        attention_out = attention_out.view(batch_size, channels, height, width)

        # Apply scaling factor and add to original input
        out = self.gamma * attention_out + x
```

```python
        return out


class ResNetWithAttention(nn.Module):
    def __init__(self, num_classes=1000):
        super(ResNetWithAttention, self).__init__()
        self.prelayer = nn.Conv2d(1, 3, kernel_size=1, bias=False)
        self.resnet = models.resnet50()
        self.attention = AttentionModule(2048)
        self.fc = nn.Linear(2048, num_classes)

    def forward(self, x):
        x = self.prelayer(x)
        x = self.resnet.conv1(x)
        x = self.resnet.bn1(x)
        x = self.resnet.relu(x)
        x = self.resnet.maxpool(x)

        x = self.resnet.layer1(x)
        x = self.resnet.layer2(x)
        x = self.resnet.layer3(x)
        x = self.resnet.layer4(x)

        x = self.attention(x)

        x = self.resnet.avgpool(x)
        x = torch.flatten(x, 1)
        x = self.fc(x)

        return x
```

由於用 ResNet50 的訓練結果(大約 76%準確率)比 ResNet + Attention(大約 68%準確率)好，因此最後採用 ResNet50 來當作訓練模型

Train and Evaluation Function

這邊定義了訓練的方式

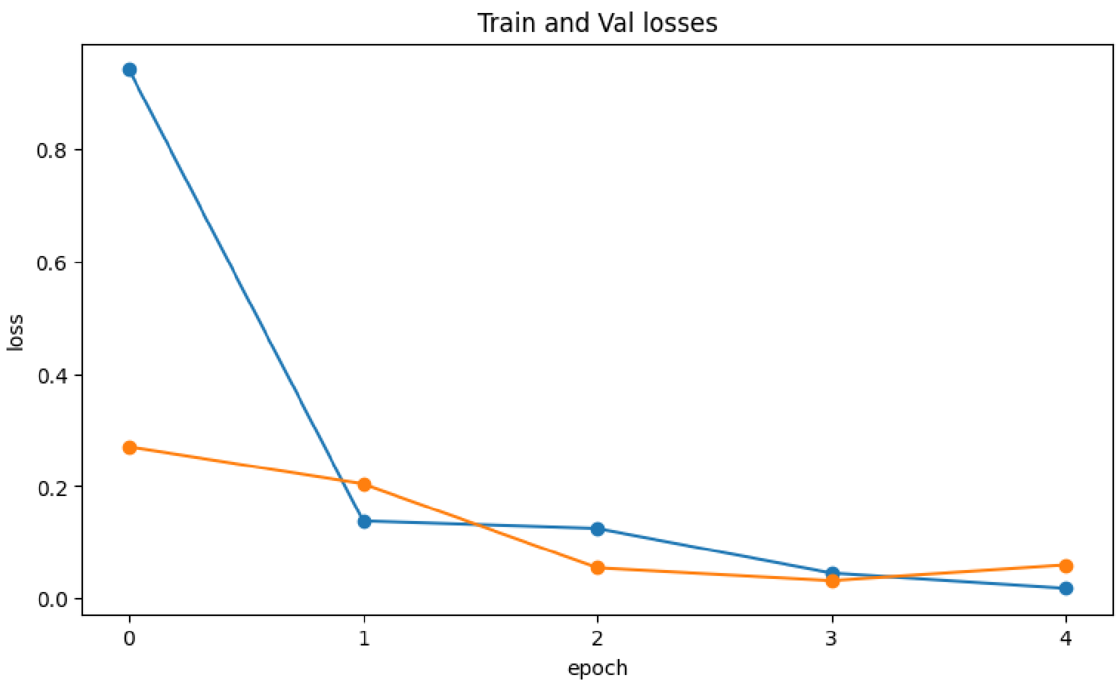
Training

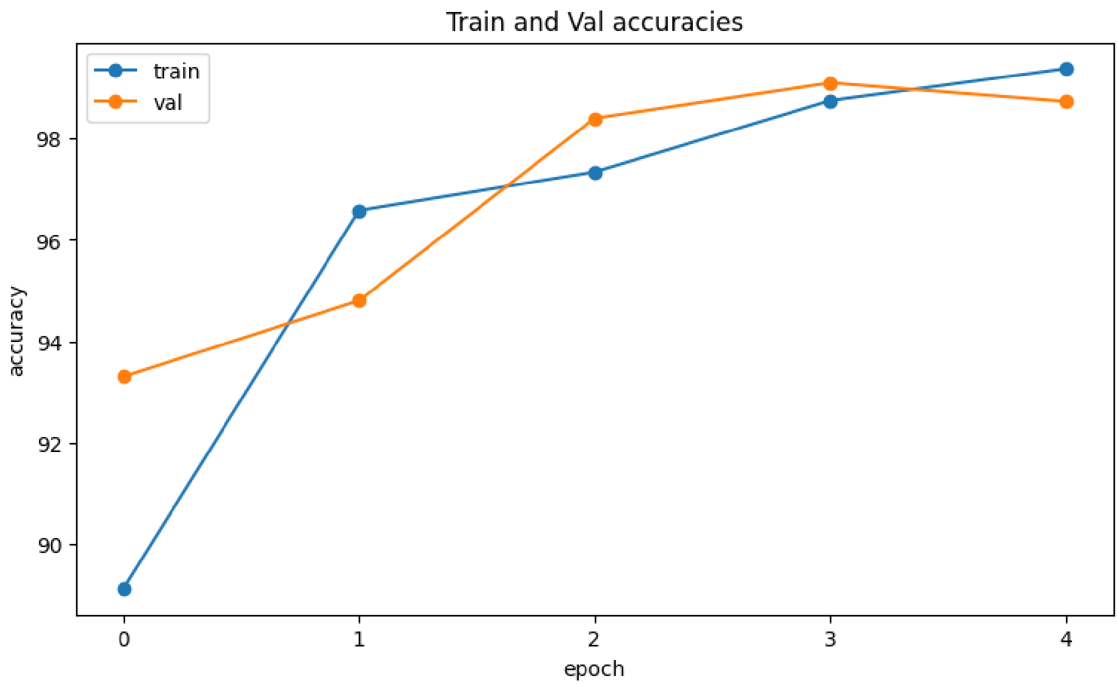
進行訓練，以下為訓練過程

```
EPOCH : 1
100%|██████████| 625/625 [08:37<00:00,  1.21it/s]
100%|██████████| 188/188 [01:17<00:00,  2.44it/s]
[WEIGHTS SAVED]
train_loss : 0.9414058455228805 valid_loss : 0.2702511450711717
train_acc : 89.11% valid_acc : 93.30000000000001%

EPOCH : 2
100%|██████████| 625/625 [08:27<00:00,  1.23it/s]
100%|██████████| 188/188 [01:16<00:00,  2.44it/s]
[WEIGHTS SAVED]
train_loss : 0.13825388845205308 valid_loss : 0.20422357079037962
train_acc : 96.58% valid_acc : 94.8%

EPOCH : 3
100%|██████████| 625/625 [08:24<00:00,  1.24it/s]
100%|██████████| 188/188 [01:16<00:00,  2.44it/s]
[WEIGHTS SAVED]
train_loss : 0.12479071687459946 valid_loss : 0.05497013535746868
train_acc : 97.34% valid_acc : 98.4%

EPOCH : 4
100%|██████████| 625/625 [08:20<00:00,  1.25it/s]
100%|██████████| 188/188 [01:16<00:00,  2.44it/s]
[WEIGHTS SAVED]
train_loss : 0.045715463435649875 valid_loss : 0.03226048071333702
train_acc : 98.75% valid_acc : 99.1%

EPOCH : 5
100%|██████████| 625/625 [08:17<00:00,  1.26it/s]
100%|██████████| 188/188 [01:17<00:00,  2.44it/s]train_loss : 0.01874799406528473 valid_loss : 0.059963318578740384
train_acc : 99.38% valid_acc : 98.73333%
```

Training Learning Curve

以下為訓練過程中 training data 和 valid data 的 loss 及 accuracy 變化





Test Section

最後是測試的部分，一樣也是先引入套件，定義好 model，最後進行測試

```
def dist(a_emb, b_emb):
  dist = a_emb - b_emb
  dist = np.dot(dist, dist.T)
  dist[dist < 0] = 0
  dist = np.sqrt(dist)[0][0]
  return dist
```

```
def test(model, folder, candidates):
  query_img = torch.from_numpy(cv2.imread(DATA_DIR + folder +
'query.png', -1).astype(np.int32)).unsqueeze(2).permute(2, 0, 1)
/ 65535.0
  query_img = query_img.unsqueeze(0).to(DEVICE)
  query_emb = model(query_img)
  dists = {}
  for cand in candidates:
    cand_img = torch.from_numpy(cv2.imread(DATA_DIR + folder +
cand, -1).astype(np.int32)).unsqueeze(2).permute(2, 0, 1) /
65535.0
    cand_img = cand_img.unsqueeze(0).to(DEVICE)
    cand_emb = model(cand_img)
    dists[cand] = dist(query_emb.cpu().detach().numpy(),
cand_emb.cpu().detach().numpy())
  return len(candidates)-1, np.count_nonzero(list(dists.values())
> dists['pos.png'])
```

```
testcases = os.listdir(DATA_DIR)
total_eval = 0
total_corr = 0

for testcase in testcases:
  testcase += '/'
  files = os.listdir(DATA_DIR + testcase)
  files.remove('query.png')
  eval, corr = test(model, testcase, files)
  total_eval += eval
```

```
  total_corr += corr
tz = timezone(timedelta(hours=+8))
print(f'Current time: {datetime.now(tz)}')
print(f'Number of test cases: {len(testcases)}')
print(f'Test accuracy: {round(total_corr/total_eval, 7)*100}%')
```

測試結果:

```
Current time: 2023-05-23 21:05:06.494204+08:00
Number of test cases: 15
Test accuracy: 76.63043%
```