深度學習視覺辨識 - 第G組報告

口罩辨識

- 結合科技與時下疫情 -

成員名單: 吳浩瑋 | 葉駿成 | 梁芷芸

動機

109年12月1日起，政府公告出入八大類場所強制戴口罩。在公共場合未配戴口罩，除了引起周圍人的反感之外，更會被處以罰鍰。

選擇「人臉口罩辨識」的主題，**第一、**因應目前疫情所需，必須準確分辨哪些人是有戴口罩，哪些是沒戴口罩的；當然，是在不使用肉眼判斷的前提下。**第二、**即使在疫情過後，許多人會持續配戴口罩。許多科技產品有提供人臉解鎖的功能(Apple FaceID)，如何透過少量特徵判斷身份成了重要課題。

本報告將著重於對第一點的探討，第二點將建立在第一點的基礎上，日後自行延伸發揮應用。

資料來源

kaggle

Search

資料集名稱

Create

Home

Competitions

Datasets

Code

Data Tasks Code (95) Discussion (1) Activity Metadata

Download (345 MB)

New Noteb

Data Explorer
344.9 MB

- Face Mask Dataset
  - Test
    - WithMask
    - WithoutMask
  - Train
    - WithMask
    - WithoutMask
  - Validation
    - WithMask
    - WithoutMask

Summary
- 11.8k files

包含檔案

Usability 7.5

License CC0: Public Domain

檔案大小

可用性

Description

Context

This dataset is used for Face Mask Detection Classification with images. The dataset consists of almost 12K images which are almost 328.92MB

Acknowledgments

All the images with the face mask (~6K) are scrapped from google search and all the images without the face mask are preprocessed from the CelebFace dataset created by Jessica Li (https://www.kaggle.com/jessicali9530). Thank you so much Jessica for providing a wonderful dataset community.

View Active Events

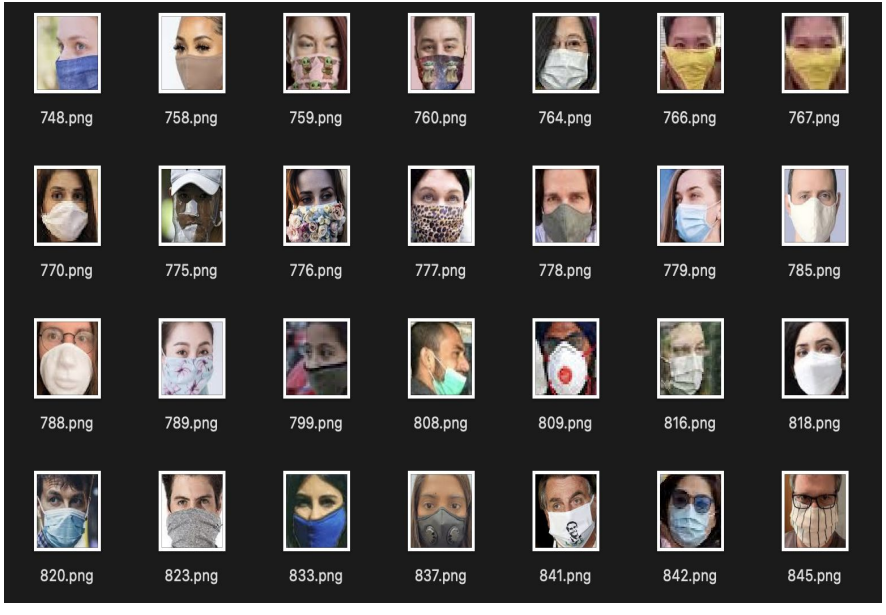
資料簡介

1.三個資料夾 train 10000張 test 995張 validation 803張

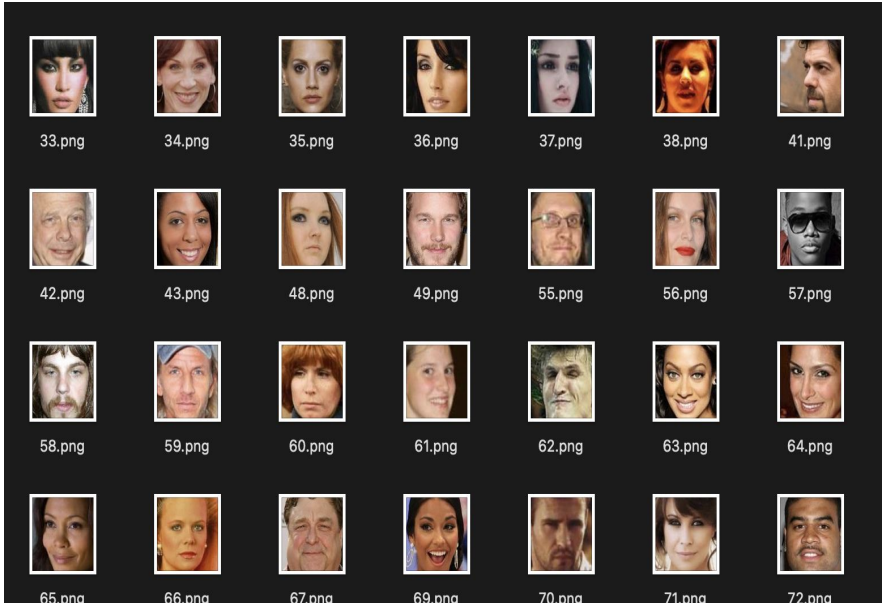
2.每個資料夾中各有 WithMask, WithoutMask

3. train 314.4MB test 30.4MB validation 24.7MB

有戴口罩



沒戴口罩



訓練流程

一、尋找合適資料

二、HOG特徵抓取

三、建立模型

(CNN,ResNET)

四、測試模型 valid

(CNN,ResNET)

五、實際運用

資料前處理

程式碼

```python
from skimage import exposure
from skimage import feature
import cv2
import sys
from pathlib import Path
 (variable) Folder_Dir: Path
Folder_Dir = Path(sys.argv[1])
pics = [x for x in Folder_Dir.iterdir()]
# print(pics)
i=0

# import torch
for pic in pics:
    img = cv2.imread(str(pic))
    img = cv2.resize(img,(128*2, 64*2))
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    (out, hog_image) = feature.hog(gray,orientations=9,pixels_per_cell=(8, 8),
                                   cells_per_block=(2, 2), visualize=True,
                                   transform_sqrt=True)
    hog_image = exposure.rescale_intensity(hog_image,out_range=(0,255))
    hog_img = hog_image.astype("uint8")
    # cv2.imshow(str(pic),hog_img)
    # img = torch.tensor(img)
    # print(img.size())
    # cv2.waitKey(0)

    dir = f"dataHog/valid/withMaskHog/{i}.jpg"
    cv2.imwrite(dir,hog_img)
    i += 1

# cv2.destroyAllWindows()
```

使用 HOG 抓取特徵

1. 使用 pathlib 模組讀入資料夾檔案
   將資料夾中的檔案路徑存成串列，後面用 for 一個個讀入。

2. Opencv 處理
   - ❏ resize（128*2, 64*2）
   - ❏ 影像灰階化
   - ❏ HOG
   - ❏ 輸出

3.
   1. 指定輸出路徑
   2. 一次跑一個資料夾
   3. 一共必須跑六次

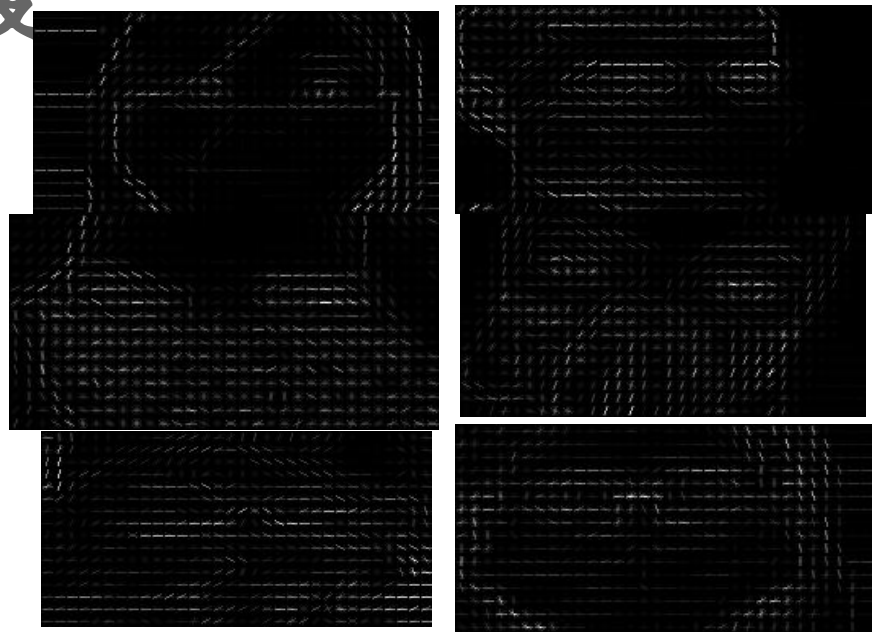
HOG處理後

Before [train 314.4MB test 30.4MB validation 24.7MB]

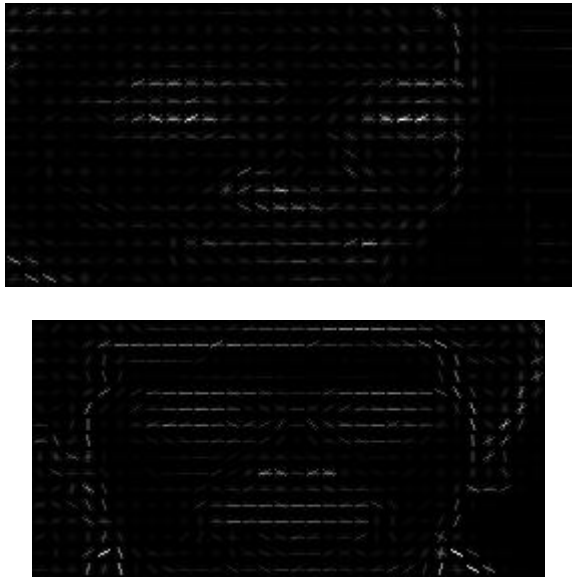
減少一半

After [train 149.9 MB test 14.8MB validation 12.1MB]

有戴口罩



沒戴口罩



建立模型

CNN

```python
import torch
import torch.nn as nn
train_on_gpu = torch.cuda.is_available()
if not train_on_gpu:
    print("GPU 不支援 CUDA。使用 CPU ..")
else:
    print('GPU 支援 CUDA。使用 GPU ..')

TRAIN_DIR = "dataHog/train"
TEST_DIR = "dataHog/test"
VAL_DIR = "dataHog/valid"

from pathlib import Path
TRAIN = Path(TRAIN_DIR)
TEST = Path(TEST_DIR)
VALID = Path(VAL_DIR)
```

1.資料夾路徑

```python
def forward(self,x):
    out = self.cnn1(x)
    out = self.relu1(out)
    out = self.maxpool1(out)
    out = self.cnn2(out)
    out = self.relu2(out)
    out = self.maxpool2(out)
    out = self.cnn3(out)
    out = self.relu3(out)
    out = self.maxpool3(out)
    out = self.cnn4(out)
    out = self.relu4(out)
    out = self.maxpool4(out)
    out = out.view(out.size(0), -1)
    out = self.fc1(out)
    out = self.drop1(out)
    out = self.relu5(out)
    out = self.fc2(out)
    out = self.drop2(out)
    out = self.output(out)

    return out

model = CNN()
```

3.建模型

```python
class CNN(nn.Module):
    def __init__(self):
        super(CNN,self).__init__()
        self.cnn1 = nn.Conv2d(in_channels=3, out_channels=16, kernel_size=5,stride=1,padding=0)
        self.relu1=nn.ReLU()
        self.maxpool1 = nn.MaxPool2d(kernel_size=2)

        self. (variable) relu2: ReLU nels=16,out_channels=32,kernel_size=5,stride=1,padding=0)
        self.relu2 =nn.ReLU()
        self.maxpool2 = nn.MaxPool2d(kernel_size=2)

        self.cnn3 = nn.Conv2d(in_channels=32,out_channels=16,kernel_size=3,stride=1,padding=0)
        self.relu3 = nn.ReLU()
        self.maxpool3 = nn.MaxPool2d(kernel_size=2)

        self.cnn4 = nn.Conv2d(in_channels=16,out_channels=8,kernel_size=3,stride=1,padding=0)
        self.relu4 = nn.ReLU()
        self.maxpool4 = nn.MaxPool2d(kernel_size=2)

        self.fc1 = nn.Linear(8*11*11,512)
        self.drop1 = nn.Dropout(0.5)
        self.relu5 = nn.ReLU()
        self.fc2 = nn.Linear(512,2)
        self.drop2 = nn.Dropout(0.5)
        self.output = nn.Softmax(dim=1)
```

2.
Resize
ToTensor
正規化

```python
from torchvision import datasets, transforms
MEAN = [0.485, 0.456, 0.406]
STD = [0.229, 0.224, 0.225]
train_transforms = transforms.Compose([transforms.Resize((224,224)),
                                       transforms.ToTensor(),
                                       transforms.Normalize(MEAN,STD)])

valid_transforms = transforms.Compose([transforms.Resize((224,224)),
                                       transforms.ToTensor(),
                                       transforms.Normalize(MEAN,STD)])

 transforms = transforms.Compose([transforms.Resize((224,224)),
                                  transforms.ToTensor(),
                                  transforms.Normalize(MEAN,STD)])

train_data = datasets.ImageFolder(TRAIN,transform=train_transforms)
test_data = datasets.ImageFolder(TEST,transform=test_transforms)
valid_data = datasets.ImageFolder(VALID,transform=valid_transforms)

print(train_data.class_to_idx)

NUM_WORKERS = 0
BATCH_SIZE = 50
LR = 0.01
```

一萬張每個
epoch跑200次
共 3 個 epoch

4.訓練模型
model.train( )
model.eval( )

```python
    valid_loss=0.0
    print('running epoch: {}'.format(epoch))
    model.train()
    for data, target in train_loader:

        if train_on_gpu:
            data, target = data.cuda(), target.cuda()

        optimizer.zero_grad()
        output=model(data)
        loss = criterion(output,target)

        loss.backward()
        optimizer.step()
        print(f"bach {bach_i}:{loss.item()*data.size(0)}")
        bach_i += 1
        train_loss += loss.item()*data.size(0)

    print(f"epoch {epoch}:training_loss {train_loss}")
    torch.save(model.state_dict(),"model_CNN.pth")
```

建立模型

ResNet

```python
import os
os.environ["KMP_DUPLICATE_LIB_OK"]="TRUE"
import torch
import torch.nn as nn
from torchvision import datasets ,transforms
from pathlib import Path
from matplotlib import pyplot as plt
import numpy as np
import torch.optim as optim
import torch.nn.functional as F

train_on_gpu = torch.cuda.is_available()
if not train_on_gpu:
    print('CUDA is not available.  Training on CPU ...')
else:
    print('CUDA is available!  Training on GPU ...')

PATH_train=r"C:\Users\user\Desktop\dataHog\train"
PATH_val=r"C:\Users\user\Desktop\dataHog\valid"
PATH_test=r"C:\Users\user\Desktop\dataHog\test"

TRAIN = Path(PATH_train)
VALID = Path(PATH_val)
TEST = Path(PATH_test)
# print(TRAIN)
# print(VALID)
# print(TEST)

# number of subprocesses to use for data loading
num_workers = 0
# how many samples per batch to load
batch_size = 32
# learning rate
LR = 0.01

# convert data to a normalized torch.FloatTensor
train_transforms = transforms.Compose([transforms.Resize((32,32)),
                                       transforms.ToTensor(),
                                       transforms.Normalize([0.485, 0.456, 0.406],
                                                            [0.229, 0.224, 0.225])])

valid_transforms = transforms.Compose([transforms.Resize((32,32)),
                                       transforms.ToTensor(),
                                       transforms.Normalize([0.485, 0.456, 0.406],
                                                            [0.229, 0.224, 0.225])])

test_transforms = transforms.Compose([transforms.Resize((32,32)),
                                      transforms.ToTensor(),
                                      transforms.Normalize([0.485, 0.456, 0.406],
                                                           [0.229, 0.224, 0.225])])

# choose the training and test datasets
train_data = datasets.ImageFolder(TRAIN, transform=train_transforms)
valid_data = datasets.ImageFolder(VALID,transform=valid_transforms)
```

1.資料夾路徑

2.正規化

3.建立模型

```python
    # save model if validation loss has decreased
    if valid_loss <= valid_loss_min:
        print('Validation loss decreased ({:.6f} --> {:.6f}).  Saving model ...'.format(valid_loss_min, valid_loss))

        torch.save(model.state_dict(), 'model_ResNet.pth')
        valid_loss_min = valid_loss

# model.load_state_dict(torch.load('model_CNN.pth'))

def test(loaders, model, criterion, use_cuda):

    # monitor test loss and accuracy
    test_loss = 0.
    correct = 0.
    total = 0.

    model.eval()
    for batch_idx, (data, target) in enumerate(loaders):
        # move to GPU
        if use_cuda:
            data, target = data.cuda(), target.cuda()
        # forward pass: compute predicted outputs by passing inputs to the model
        output = model(data)
        # calculate the loss
        loss = criterion(output, target)
        # update average test loss
        test_loss = test_loss + ((1 / (batch_idx + 1)) * (loss.data - test_loss))
        # convert output probabilities to predicted class
        pred = output.data.max(1, keepdim=True)[1]
        # compare predictions to true label
        correct += np.sum(np.squeeze(pred.eq(target.data.view_as(pred))).cpu().numpy())
        total += data.size(0)

    print('Test Loss: {:.6f}'.format(test_loss))
    print('Test Accuracy: %2d%% (%2d/%2d)' % (100. * correct / total, correct, total))

use_cuda = torch.cuda.is_available()
test(test_loader, model, criterion, use_cuda)
```

4.訓練模型

```python
    for epoch in range(1, n_epochs+1):
        # keep track of training and validation loss
        train_loss = 0.0
        valid_loss = 0.0
        print('running epoch: {}'.format(epoch))

        ##################
        # train the model #
        ##################
        model.train()
        for data, target in train_loader:
            # move tensors to GPU if CUDA is available
            if train_on_gpu:
                data, target = data.cuda(), target.cuda()
            # clear the gradients of all optimized variables
            optimizer.zero_grad()
            # forward pass: compute predicted outputs by passing inputs to the model
            output = model(data)
            # calculate the batch loss
            loss = criterion(output, target)
            # backward pass: compute gradient of the loss with respect to model parameters
            loss.backward()
            # perform a single optimization step (parameter update)
            optimizer.step()
            # update training loss
            train_loss += loss.item()*data.size(0)

        ######################
        # validate the model #
        ######################
        model.eval()
        for data, target in valid_loader:
            # move tensors to GPU if CUDA is available
            if train_on_gpu:
                data, target = data.cuda(), target.cuda()
            # forward pass: compute predicted outputs by passing inputs to the model
            output = model(data)
            # calculate the batch loss
            loss = criterion(output, target)
            # update average validation loss
            valid_loss += loss.item()*data.size(0)

        # calculate average losses
        # train_losses.append(train_loss/len(train_loader.dataset))
        # valid_losses.append(valid_loss.item()/len(valid_loader.dataset))
        train_loss = train_loss/len(train_loader.dataset)
        valid_loss = valid_loss/len(valid_loader.dataset)

        # print training/validation statistics
        print('\tTraining Loss: {:.6f} \tValidation Loss: {:.6f}'.format(train_loss, valid_loss))

        # save model if validation loss has decreased
        if valid_loss <= valid_loss_min:
            print('Validation loss decreased ({:.6f} --> {:.6f}).  Saving model ...'.format(valid_loss_min,

            torch.save(model.state_dict(), 'model_ResNet.pth')
            valid_loss_min = valid_loss
```

準確率

Console 1/A

```
          Conv2d-47          [-1, 512, 4, 4]       2,359,296
     BatchNorm2d-48          [-1, 512, 4, 4]           1,024
          Conv2d-49          [-1, 512, 4, 4]         131,072
     BatchNorm2d-50          [-1, 512, 4, 4]           1,024
   ResidualBlock-51          [-1, 512, 4, 4]               0
          Conv2d-52          [-1, 512, 4, 4]       2,359,296
     BatchNorm2d-53          [-1, 512, 4, 4]           1,024
            ReLU-54          [-1, 512, 4, 4]               0
          Conv2d-55          [-1, 512, 4, 4]       2,359,296
     BatchNorm2d-56          [-1, 512, 4, 4]           1,024
   ResidualBlock-57          [-1, 512, 4, 4]               0
          Linear-58                [-1, 10]           5,130
================================================================
Total params: 11,173,962
Trainable params: 11,173,962
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB):
Params size (MB): 42.63
Estimated Total Size (MB): 56.26
----------------------------------------------------------------
running epoch: 1
    Training Loss: 0.150205     Validation Loss: 0.373844
Validation loss decreased (inf --> 0.373844).  Saving model ...
running epoch: 2
    Training Loss: 0.079118     Validation Loss: 0.160660
Validation loss decreased (0.373844 --> 0.160660).  Saving model ...
running epoch: 3
    Training Loss: 0.055860     Validation Loss: 0.048278
Validation loss decreased (0.160660 --> 0.048278).  Saving model ...
Test Loss: 0.047180
Test Accuracy: 98% (977/992)
```

ResNet

CNN

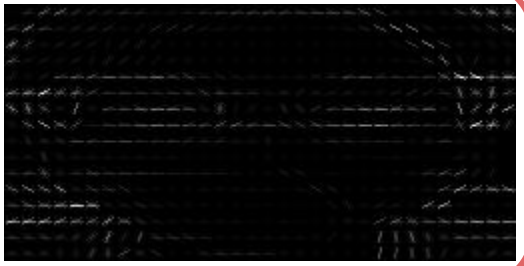
```
GPU 不支援 CUDA。使用 CPU ..
{'withMask': 0, 'withoutMask': 1}
Test Loss: 0.34734
Test Accuracy: 96.63157894736842% (918/950)
```

實際運用

芷芸

浩瑋

駿成



有帶口罩

結果

```
In [5]: runfile('C:/Users/user/Desktop/dl.py', wdir='C:/Users/user/Desktop')
CUDA is not available.  Training on CPU ...
tensor([[ 6.6120,  1.8115, -4.0079, -4.5733, -3.8779, -4.3241, -3.8255, -4.0073,
         -4.0071, -4.5442],
        [ 8.1687,  0.8754, -4.5933, -5.3064, -4.4859, -5.0285, -4.4789, -4.6471,
         -4.5811, -5.1449],
        [ 8.0008,  0.9860, -4.5183, -5.2492, -4.4939, -5.0219, -4.4781, -4.5340,
         -4.6009, -5.1075],
        [ 7.7910,  1.1844, -4.4194, -5.1083, -4.4617, -4.8796, -4.4425, -4.4515,
         -4.5560, -5.0590],
        [ 3.8084,  4.0615, -3.9256, -4.6271, -3.8862, -4.5078, -3.7178, -3.9828,
         -4.1977, -4.6652],
        [ 4.6805,  2.6850, -3.3703, -3.9107, -3.2707, -3.7895, -3.1448, -3.3743,
         -3.3782, -3.8552]], grad_fn=<AddmmBackward>)
Test Loss: 1.553135
Test Accuracy: 66% ( 4/ 6)
```

沒帶口罩

芷芸

浩瑋

駿成