Assignment – AOI_image_ recognition

1. 概述

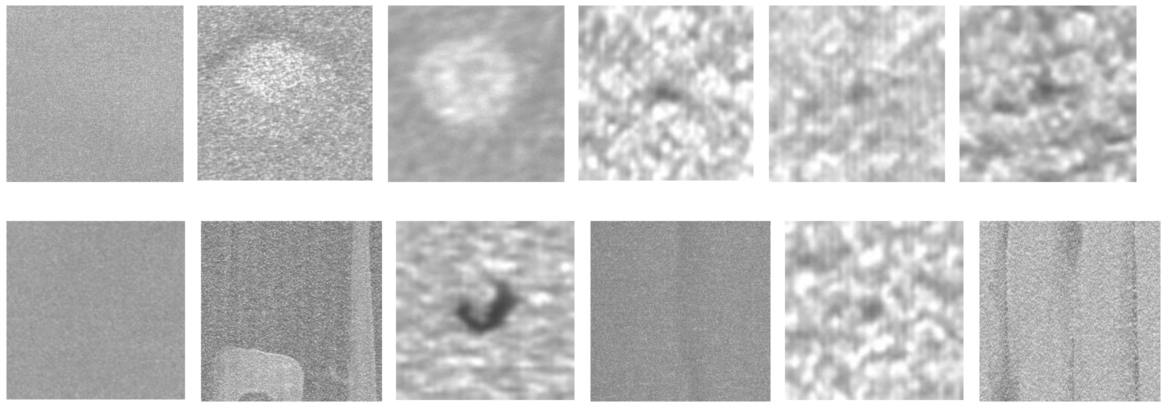
a. 本題目藉由 AOI 影像訓練深度學習模型辨識產品表面瑕疵，使用框架為 Pytorch。實作結果顯示，預訓練 LeNet5 模型的測試準確已達到 0.7775585。

b. 硬體環境:
- Google colab GPU
- Google drive

c. 影像資料
- 訓練資料： 2,528 張(隨機抽取 20%作為驗證資料)
- 測試資料：10,142 張
- 影像類別：6 個類別(正常類別 + 5 種瑕疵類別)
- 影像尺寸：512x512

2. AOI 程式碼說明

在訓練方面，本次影像資料是由工研院電光所在 Aidea(人工智慧共創平台)釋出作為開放性議題，提供參賽者建立瑕疵辨識模型。圖面樣態如下所示：



訓練及測試的平台為 google 的 colab 上做訓練及測試。相關程式碼說明如下：

a. 載入 google drive。

```
# mount google’s drive
from google.colab import drive
drive.mount('/content/drive')

# 切換路徑至資料夾。
%cd /content/drive/MyDrive/work/NCKU/10902/dl/HW05/dl05

# 讀取目前路徑
!pwd
```

b. 執行訓練主程式

```
import os
import argparse
import logging
import time
import pickle
import time
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import scipy as sp
import torch
import torch.nn as nn
import torch.optim as optim
import torch.nn.functional as F
import torchvision.datasets as datasets
import torchvision.transforms as transforms
from torchvision import models
from load_data import CreateList, CustomDataset
from models import LeNet5, VGG
from utils import updateBN, savemodel, Log

trial_info = 'lenet_init'
log = Log(trial_info)

first = 1e-4
last  = 1e-4
#%% All parameters setting
para = {
    # Dataset
    'dataset': 'aoi',
    'batch_size': 48,
    'split': 0.8,  # ratio of training data
    # Model
    'resume': '',  # a path of trained model
    'pruned': '',  # a path of pruned model
    'pretrain': False,
    'cfg': [],  # None or a list of integers and 'M'
    # Training
    'cuda': True, # True
    'workers': 0,
    'epochs': 100,
    'checkpoint_freq': 5,
    'early_stop': False,
    # Hyperparameters
    'lr': 1e-2,
    'decay': 1e-5,
    'channel_sparsity': True,  #Ture whether adding L1-norm of BN gamma factor
    'sparsity_rate': 0,
    'patience': 8,
    # Trial id
    'trial': trial_info}

log.log('Parameters Setting:\n{}'.format(para).replace(', ', ',\n '))

#%% Prepare data pipeline
#dir_img_train = 'C:/Dataset/AOI/train_images/'
dir_img_train = './aoi/train_images/'
#path_label_train = 'C:/Dataset/AOI/train.csv'
path_label_train = './aoi/train.csv'

# Split image list and label list into train and valid.
train_list = CreateList(dir_img_train, path_label_train, shuffle=True)
train_valid_split = round(train_list.length * para['split'])
train_img = train_list.img[:train_valid_split]
train_label = train_list.label[:train_valid_split]
valid_img = train_list.img[train_valid_split:]
valid_label = train_list.label[train_valid_split:]

# Image preprocessing
transform = {
    'train': transforms.Compose([
        transforms.RandomHorizontalFlip(),
        transforms.RandomRotation(15),
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Normalize((0.5,), (0.5,))
    ]),
    'valid': transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Normalize((0.5,), (0.5,))
    ])
}

log.log('Data Preprocessing:\n{}'.format(transform))

# Create DataLoader
train_dataset = CustomDataset(train_img,
                                train_label,
                                transform['train'])
valid_dataset = CustomDataset(valid_img,
                                valid_label,
                                transform['valid'])

train_loader = torch.utils.data.DataLoader(dataset=train_dataset,
                                        batch_size=para['batch_size'],
                                        shuffle=False,
                                        num_workers=para['workers'],
                                        pin_memory=True)
valid_loader = torch.utils.data.DataLoader(dataset=valid_dataset,
                                        batch_size=para['batch_size'],
                                        shuffle=False,
                                        num_workers=para['workers'],
                                        pin_memory=True)

#%% Build a model
#net = VGG(dataset=para['dataset'], pretrained=para['pretrain'])
net = LeNet5('aoi')

# Send model into gpu memory
if para['cuda']:
    net.cuda()

log.log('Model Structure:\n{}'.format(net))

#%% Create loss function, optimzier and training scheduler
criterion = nn.CrossEntropyLoss()

optimizer = optim.SGD(net.parameters(),
                    lr=para['lr'],
                    weight_decay=para['decay'],
                    momentum=0.9,
                    nesterov=True)

scheduler = torch.optim.lr_scheduler.ReduceLROnPlateau(optimizer, mode='max',
                                                       factor=0.1,
patience=para['patience'], verbose=True, threshold=1e-4, min_lr=1e-6)

log.log('Optimizer:\n{}'.format(optimizer))

#%% Train the Model
start_epoch = 0
best_prec1 = 0
if __name__ == '__main__':
    start_training = time.time()
    log.log('Start training model...')

    for epoch in range(start_epoch, start_epoch + para['epochs']):
        # loss list
        list_loss_train = []
        list_loss_valid = []
        # training
        train_correct = 0
        train_total = 0
        net.train()  # activate autograd
        for i, (images, label) in enumerate(train_loader):
            if para['cuda']:
                images, label = images.cuda(), label.cuda()

            optimizer.zero_grad()  # clear buffer
            out = net(images)
            train_loss = criterion(out, label)
            train_loss.backward()
            # subgradient decent
            if para['channel_sparsity']:
                updateBN(net, para['sparsity_rate'], False, first, last)
            optimizer.step()  # update weights

            _, pred = torch.max(out.data, 1)  # max() return maximum and its index in each
row
            train_total += float(label.size(0))
            train_correct += float((pred == label).sum())

        # validation
        valid_correct = 0
        valid_total = 0
        net.eval()
        with torch.no_grad():
            for images, label in valid_loader:
                if para['cuda']:
                    images, label = images.cuda(), label.cuda()

                out = net(images)  # forward
                valid_loss = criterion(out, label)
                _, pred = torch.max(out.data, 1)  # max() return maximum and its index in
each row
                valid_total += float(label.size(0))
                valid_correct += float((pred == label).sum())

        # metrics
        train_acc = 100*train_correct / train_total
        valid_acc = 100*valid_correct / valid_total
        is_best = valid_acc > best_prec1
        best_prec1 = max(valid_acc, best_prec1)
        list_loss_train.append(train_loss)
        list_loss_valid.append(valid_loss)

        scheduler.step(valid_acc)
        # save model
        state = {
            'epoch': epoch,  # last epoch
            'state_dict': net.state_dict(),
            'best_prec1': best_prec1,
            'optimizer': optimizer.state_dict(),
            'scheduler': scheduler.state_dict()
            }
        state.update(para)
        suffix = para['trial']
        # save pruned structure
        if para['pruned']:
            state['cfg'] = pruned_pkl['cfg']
            suffix += '_' + args.pruned.split('_')[-1][:-4]

        save = savemodel(state, is_best, para['checkpoint_freq'], suffix, False)
        if save:
            log.log(save)

        # print result
        if (epoch+1) % 1 == 0:

            log.log('Epoch:{}/{}\nAccuracy(Train/Valid):{:.02f}/{:.02f}%
Loss(Train/Valid):{:.3f}/{:.3f}'.format(
                epoch, start_epoch + para['epochs']-1, train_acc, valid_acc, train_loss,
valid_loss))

        # early stopping
        if para['early_stop'] and valid_acc > 99.5:
            log.log('Early stop beacause valid accuracy > 99.5.')
            break

    end_training = time.time()
    #log.log('Time:', round((end_training - start_training)/60, 2), 'mins')
    log.log('Time:{}  mins'.format(round((end_training - start_training)/60, 2)))
```

c. 執行測試主程式

```python
import math, time
from PIL import Image
import pandas as pd
import matplotlib.pyplot as plt
from tqdm import tqdm
import torch
import torch.nn as nn
import torchvision
from torchvision import datasets, transforms
from load_data import CreateList, CustomDataset
from models import VGG, LeNet5

#%% Paths
#dir_img_test = 'C:/Dataset/AOI/test_images/'
dir_img_test = './aoi/test_images/'
#path_label_test = 'C:/Dataset/AOI/test.csv'
path_label_test = './aoi/test.csv'
#path_model = './model/bestmodel0721_vgg_pre_bn01.pkl'
path_model = './model/bestmodel0531_lenet_init.pkl'
save_submit = './submit/{}_submit.csv'.format(path_model.split('/')[-1].replace('.pkl', ''))

#%% Parameters
cuda = True
workers = 2
batch_size = 128
#%% Load the Model
#net = VGG('aoi', True)
net = LeNet5('aoi')
save = torch.load(path_model)
save['best_prec1']
net.load_state_dict(save['state_dict'])
net.eval()

# Send model into gpu memory
if cuda:
    net.cuda()
#%% Prepare the data
test_list = CreateList(dir_img_test, path_label_test, shuffle=False, train=False)

transform = {
    'test': transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Normalize((0.5,), (0.5,))
    ])
}

fake_list = [i for i in range(len(test_list.img))]

test_dataset = CustomDataset(test_list.img,
                             label_list=fake_list,
                             transform=transform['test'])

test_loader = torch.utils.data.DataLoader(dataset=test_dataset,
                                          batch_size=batch_size,
                                          shuffle=False,
                                          num_workers=workers,
                                          pin_memory=True)

#%% Predict test images
# Collect prediction values
test_predict = []
net.eval()
with torch.no_grad():
    for images, _ in tqdm(test_loader):
        images = images.cuda()

        out = net(images)  # forward
        _, pred = torch.max(out.data, 1)
        test_predict += pred.cpu().numpy().tolist()
# Check number of class of predicitons
len(set(test_predict))
# Check whether the number of predictions match test images
len(test_predict) == len(test_list.filename)

#%% Create submit data
df_submit = pd.DataFrame({'ID': test_list.filename,
                            'Label': test_predict})

df_submit.to_csv(save_submit,
                header=True,
                sep=',',
                encoding='utf-8',
                index=False)
```

3. 訓練及測試結果

a. 訓練

```
05-31 00:08 Start training model...
05-31 00:23 Epoch:0/99
Accuracy(Train/Valid):35.46/45.85% Loss(Train/Valid):1.408/1.566
05-31 00:23 Epoch:1/99
Accuracy(Train/Valid):31.75/22.73% Loss(Train/Valid):1.618/1.662
05-31 00:23 Epoch:2/99
Accuracy(Train/Valid):29.13/26.48% Loss(Train/Valid):1.918/1.762
05-31 00:24 Epoch:3/99
Accuracy(Train/Valid):38.72/46.44% Loss(Train/Valid):1.312/1.216

                              … …

05-31 00:56 Epoch:93/99
Accuracy(Train/Valid):96.88/94.47% Loss(Train/Valid):0.014/0.334
05-31 00:56 Model saved.
05-31 00:56 Epoch:94/99
Accuracy(Train/Valid):95.99/94.47% Loss(Train/Valid):0.017/0.335
05-31 00:57 Epoch:95/99
Accuracy(Train/Valid):96.34/94.47% Loss(Train/Valid):0.007/0.335
05-31 00:57 Epoch:96/99
Accuracy(Train/Valid):95.90/94.47% Loss(Train/Valid):0.007/0.335
05-31 00:58 Epoch:97/99
Accuracy(Train/Valid):96.24/94.47% Loss(Train/Valid):0.010/0.335
05-31 00:58 Epoch:98/99
Accuracy(Train/Valid):96.39/94.47% Loss(Train/Valid):0.007/0.335
05-31 00:58 Model saved.
05-31 00:58 Epoch:99/99
Accuracy(Train/Valid):96.14/94.47% Loss(Train/Valid):0.014/0.335
```

b. <u>測試</u>

ID	Label
test_00000.png	1
… …	
test_10105.png	0
test_10106.png	5
test_10107.png	5
test_10108.png	0
test_10109.png	1
test_10110.png	0
test_10111.png	3
test_10112.png	0
test_10113.png	0
test_10114.png	1
test_10115.png	3
test_10116.png	3
test_10117.png	0
test_10118.png	5
test_10119.png	5
test_10120.png	0
test_10121.png	1
test_10122.png	0
test_10123.png	5
test_10124.png	3
test_10125.png	0
test_10126.png	5
test_10127.png	4
test_10128.png	0
test_10129.png	0
test_10130.png	5
test_10131.png	0
test_10132.png	1
test_10133.png	1
test_10134.png	0
test_10135.png	1
test_10136.png	3
test_10137.png	3
test_10138.png	1
test_10139.png	1
test_10140.png	4
test_10141.png	1

4. 結論

整體來說測試的結果不錯，詳如 bestmodel0531_lenet_init_submit.csv 檔案。