1072 Deep Learning – Homework 2

Due: May 24, 2019, 11:55pm

1. (50%) Please use logistic regression and neural net to classify two datasets, “Iris” and “Ionosphere.’ You can check UCI repository to see more details regarding these datasets.

Iris - <http://archive.ics.uci.edu/ml/datasets/Iris> (15%)

Ionosphere - <https://archive.ics.uci.edu/ml/datasets/ionosphere>

To save you some time, you could use “fisheriris.mat” and “ionosphere.mat” released to you.

The “Iris” dataset contains 150 feature vectors in 4 dimensions, where the label contains three different values, {setosa, versicolor, virginica}, each of which refers to a type of iris plant.

The “Ionosphere” dataset contains radar data collected by a system in Goose Bay, Labrador, which consists of a phased array of 16 high-frequency antennas with a total transmitted power on the order of 6.4 kilowatts. It has 34 real-valued features and categorical response: "b" for bad radar returns and "g" for good radar returns.

* 1. First randomly select 80% of the dataset as your training set and the rest 20% as your testing set. (5% for each)

import scipy.io

from sklearn.model\_selection import train\_test\_split

fisheriris = scipy.io.loadmat('fisheriris.mat')

ionosphere = scipy.io.loadmat('ionosphere.mat')

# iris

iris\_train\_feature, iris\_test\_feature, iris\_train\_label, iris\_test\_label = \

train\_test\_split(fisheriris['meas'], fisheriris['species'], test\_size=0.2, shuffle=True, random\_state=113)

for i in range(len(iris\_train\_label)):

iris\_train\_label[i] = iris\_train\_label[i][0]

for i in range(len(iris\_test\_label)):

iris\_test\_label[i] = iris\_test\_label[i][0]

iris\_train\_label = iris\_train\_label.ravel()

iris\_test\_label = iris\_test\_label.ravel()

# ionosphere

iono\_train\_feature, iono\_test\_feature, iono\_train\_label, iono\_test\_label = \

train\_test\_split(ionosphere['X'], ionosphere['Y'], test\_size=0.2, shuffle=True, random\_state=113)

for i in range(len(iono\_train\_label)):

iono\_train\_label[i] = iono\_train\_label[i][0]

for i in range(len(iono\_test\_label)):

iono\_test\_label[i] = iono\_test\_label[i][0]

iono\_train\_label = iono\_train\_label.ravel()

iono\_test\_label = iono\_test\_label.ravel()

* 1. Train logistic regression and neural net classifiers for these two datasets and report accuracy for training and testing datasets as (20% for each)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Classifier | Iris | | Ionosphere | |
| Training | Testing | Training | Testing |
| Logistic Regression | 0.96 | 0.96 | 0.91 | 0.87 |
| Neural Net | 0.97 | 1.0 | 1.0 | 0.98 |

Please detail your models in the report. For neural net, your model must at least have three hidden layers, meaning including the input and output layers, in total, it has at least five layers. You are encouraged to try to use more layers or neurons in your model to see if they help increase the prediction accuracy.

(程式碼在另外檔案附上)

logistic regression:

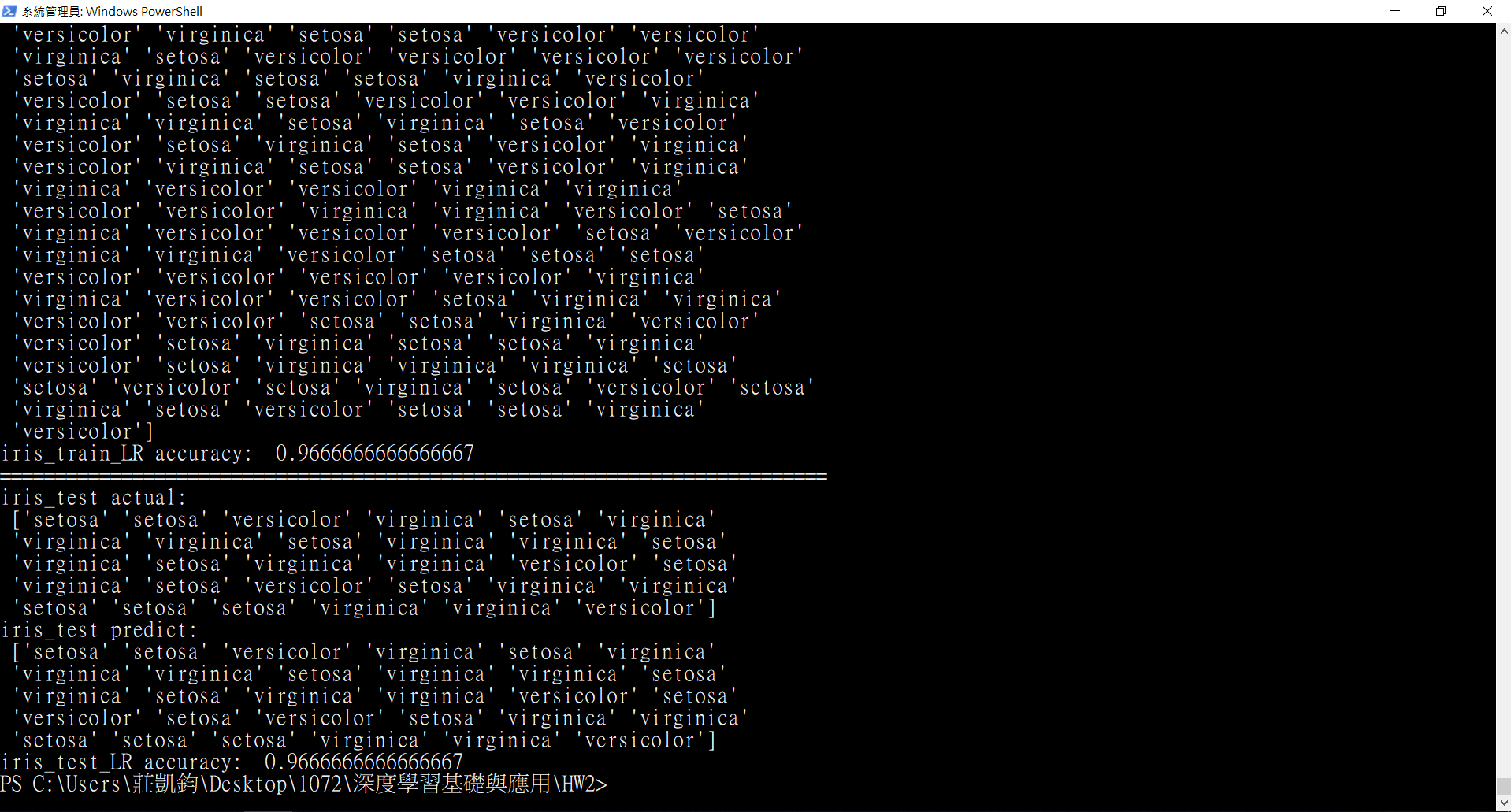
將label處理成一維純字串格式後，傳入sklearn.linear\_model的LogisticRegression

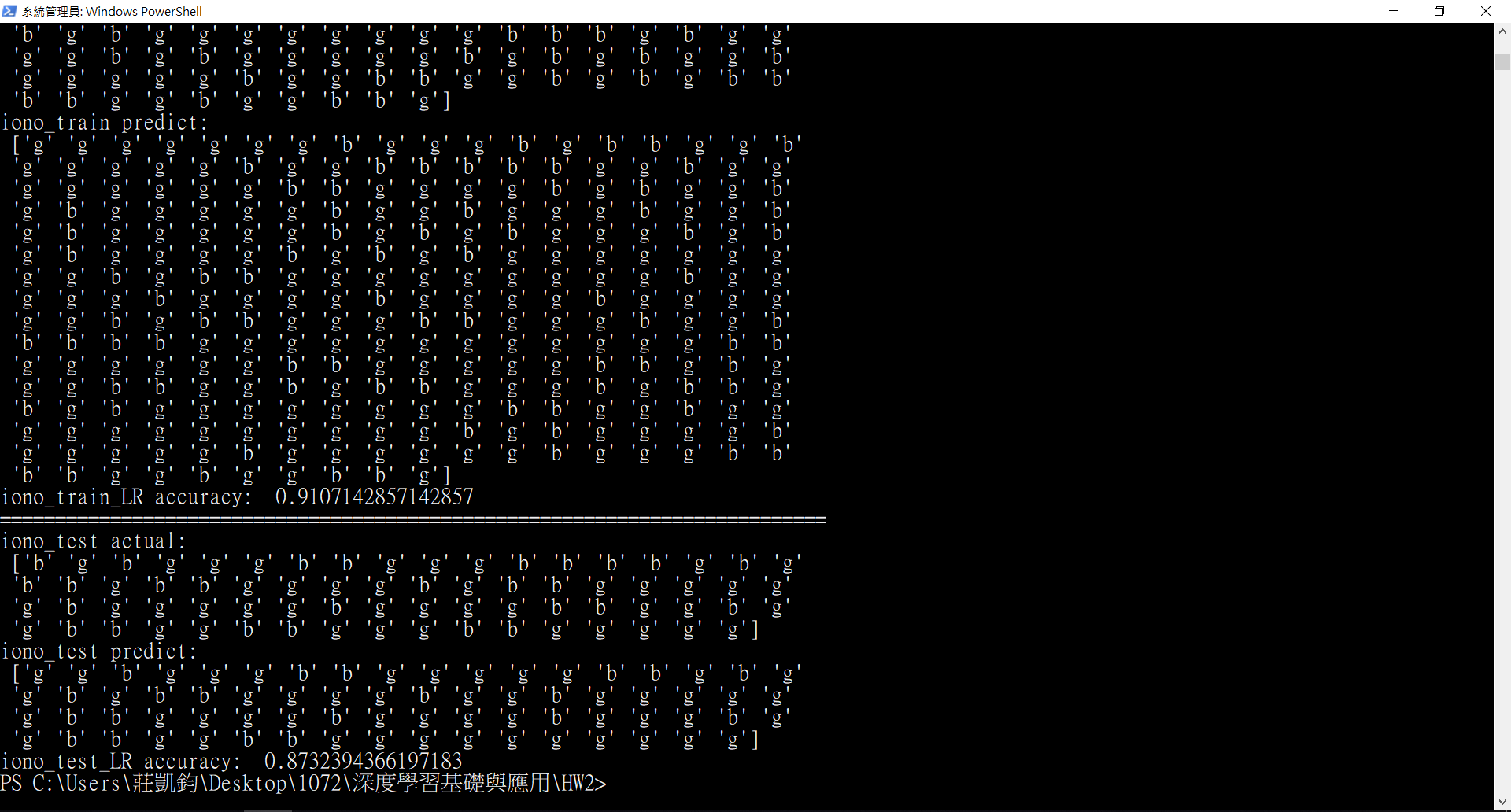
Neural Net:

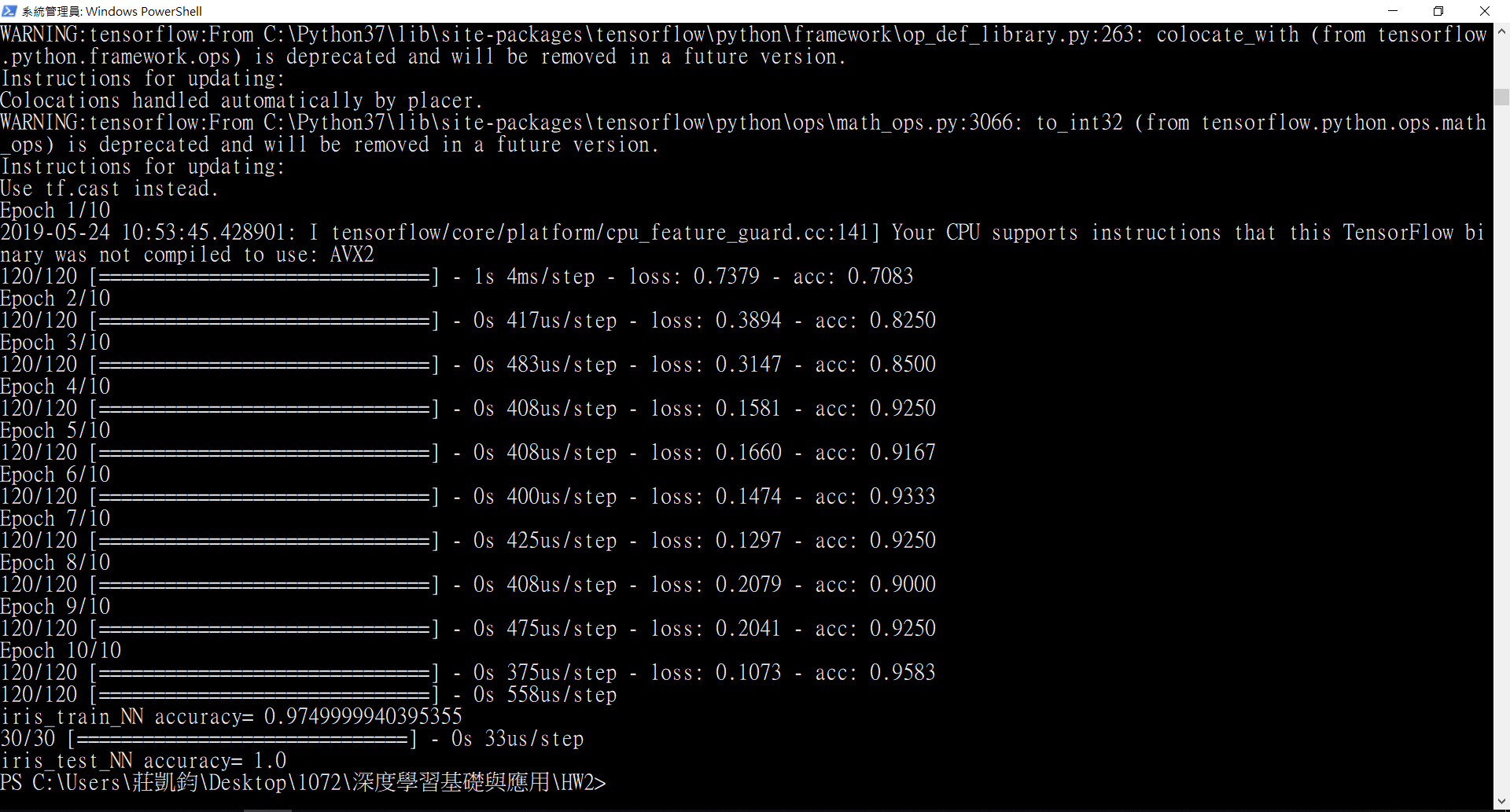
Step1:將label處理成one-hot encoding

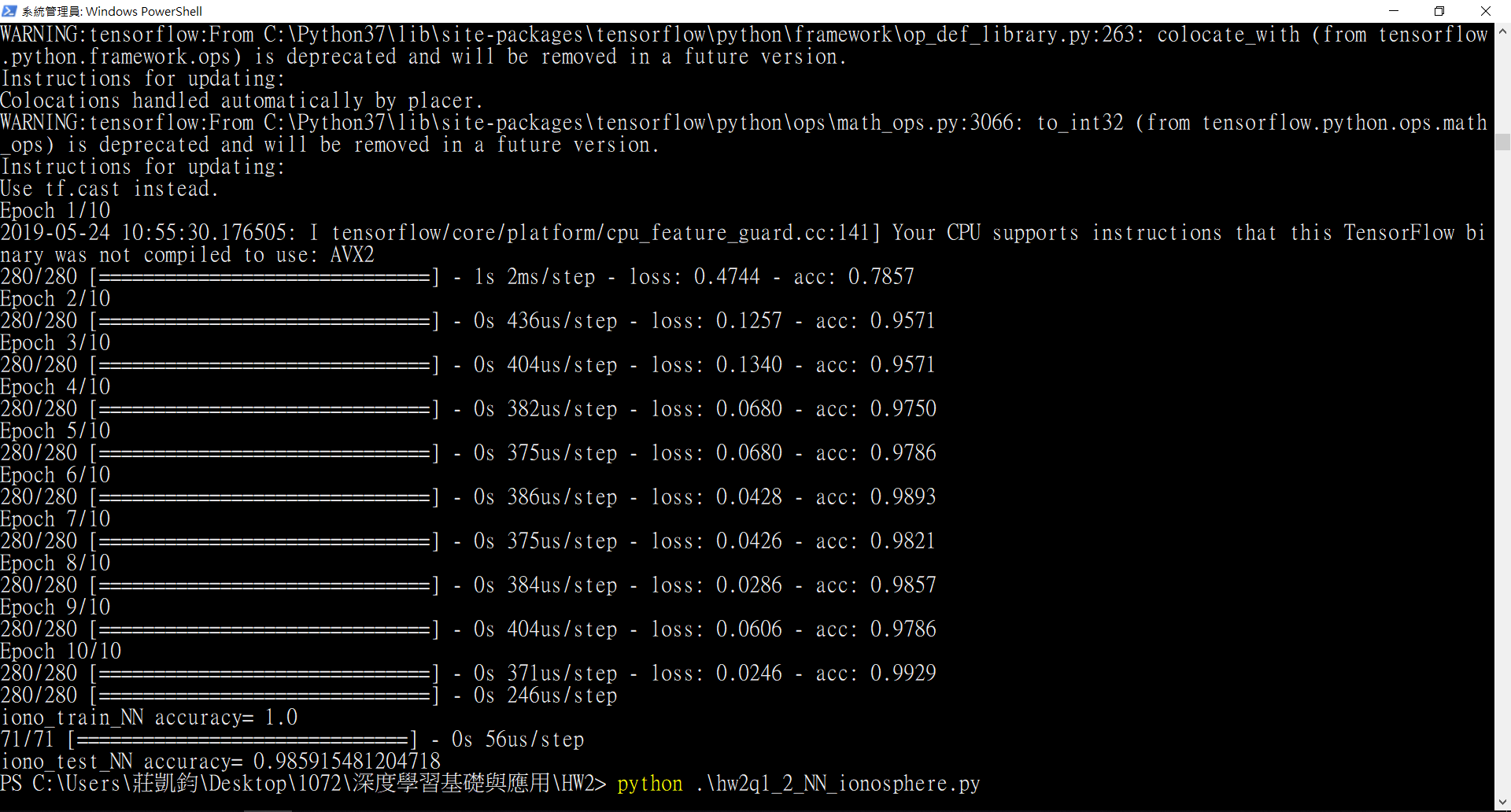
Step2: input layer傳入iris:4維feature， ionosphere:34維feature，neural net共有4層hidden layer，output layer回傳iris:3個class， ionosphere:2個class，每層neuron數皆設128，activation function為relu，output layer則為softmax

Step3: loss採用cross entropy，optimizer採用adam，metrics採用accuracy









1. (30%) In the mnist example where a classification model based on convolutional neural networks is provided, please add 10%, 20%, and 40% of noise to all the images and report the prediction accuracy for the training and testing datasets.

To add noise to the images, you could use the example code below:

import random

import numpy as np

noise\_lv = 0.1

img\_size = 28\*28

for i in range(len(X\_train)):

ran\_seq = random.sample([n for n in range(img\_size)], np.int(img\_size\*noise\_lv))

x = X\_train[i].reshape(-1, img\_size)

x[0, ran\_seq]=255

Step1: 將feature都除以255，label皆one-hot encoding成10類

Step2: 以salt\_and\_pepper法(某pixel全白或全黑)添加noise(10%,20%,40%)

Step3: 根據channels\_first或channels\_last將data都加上channel axis，設channel為1，表grayscale

Step4: 建立cnn model，其中每個filter為3\*3，第一次conv2D有32張filter，activation為relu，接著maxpooling2D，pool\_size取2\*2，第二次conv2D有50張filter，接著maxpooling2D，pool\_size取2\*2，最後flatten成一維，再丟進fully connected feedforward network

model = Sequential()

model.add(Conv2D(filters=32, kernel\_size=(3, 3), activation='relu', input\_shape=input\_shape))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Conv2D(filters=50, kernel\_size=(3, 3), activation='relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Flatten())

model.add(Dense(units=128, activation='relu'))

model.add(Dense(units=num\_classes, activation='softmax'))

Step5: apply cnn model

model.compile(loss=keras.losses.categorical\_crossentropy, optimizer=keras.optimizers.adadelta(), metrics=['accuracy'])

model.fit(x\_train, y\_train, batch\_size=batch\_size, epochs=epochs)

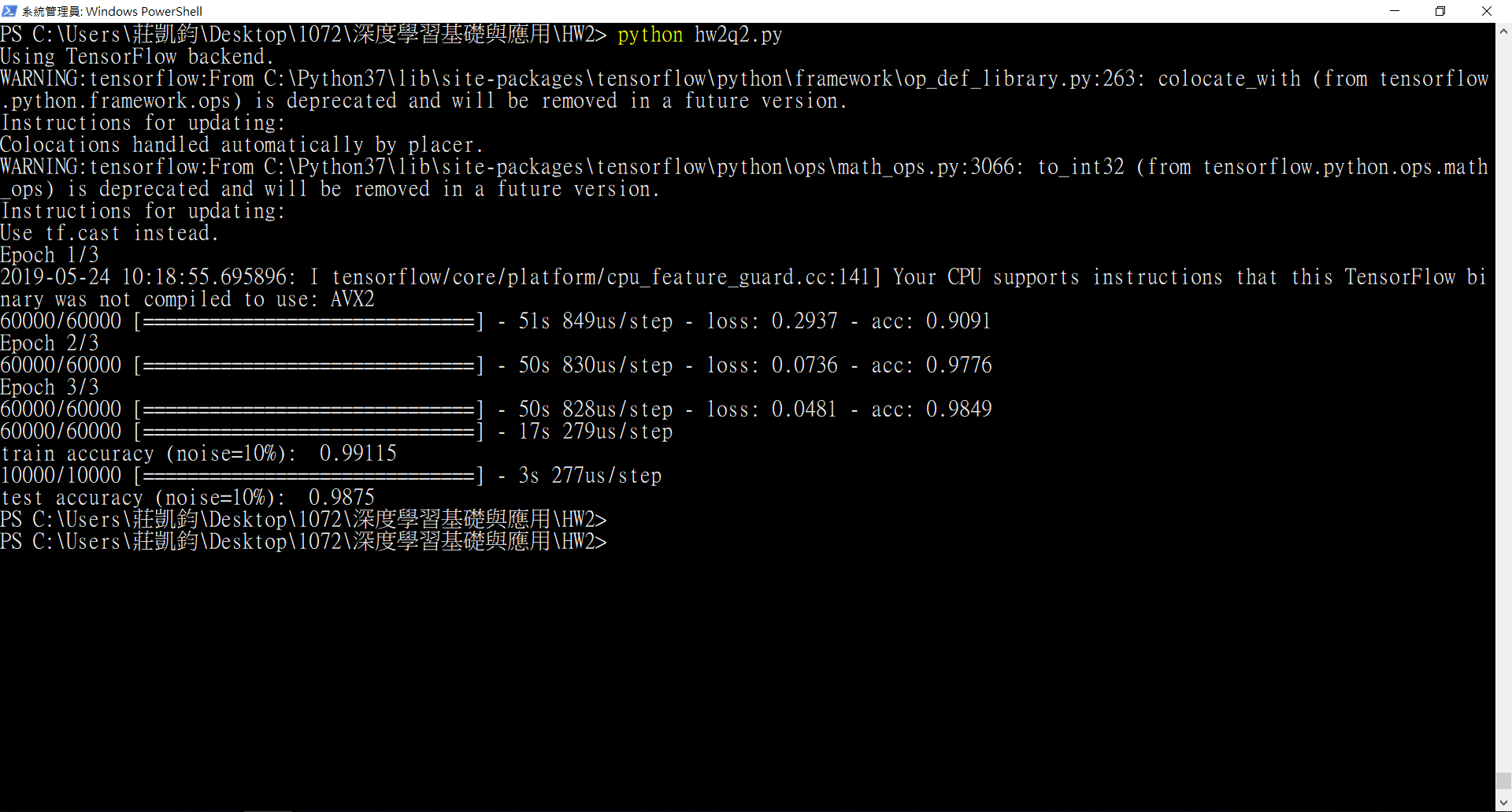
result = model.evaluate(x\_train, y\_train, verbose=1)

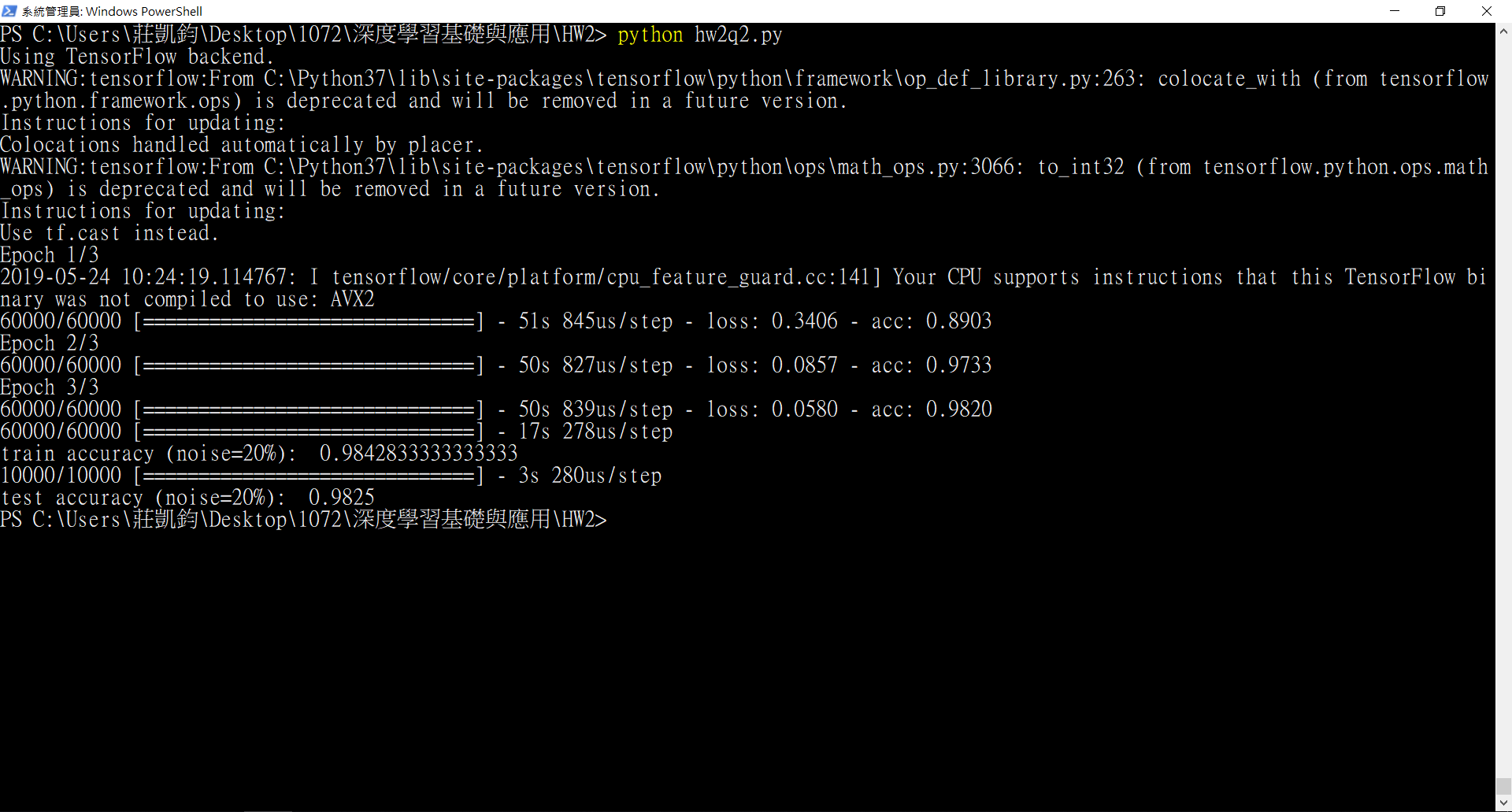
print("train accuracy (noise=10%): ", result[1])

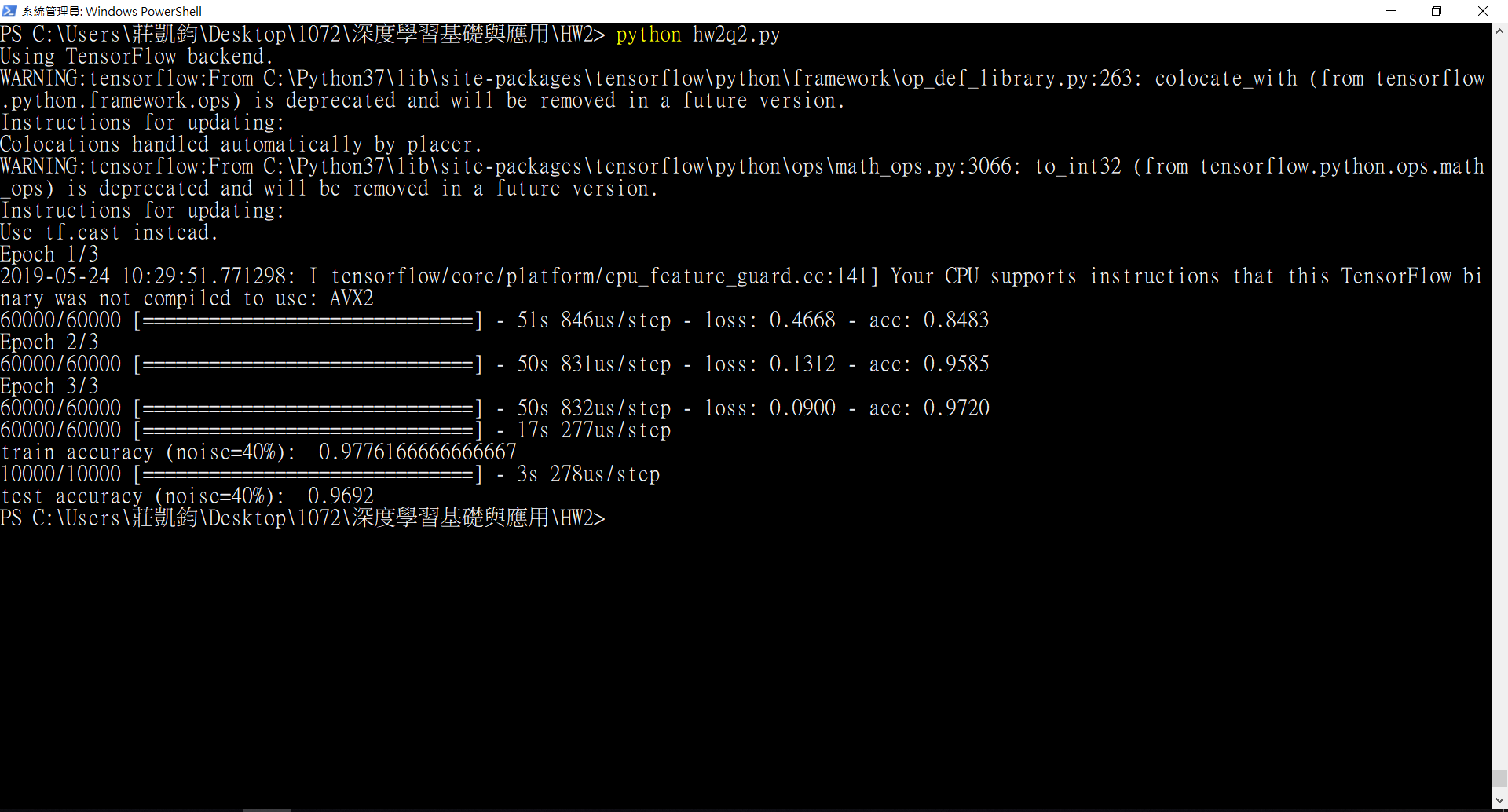
result = model.evaluate(x\_test, y\_test, verbose=1)

print("test accuracy (noise=10%): ", result[1])

|  |  |  |
| --- | --- | --- |
| noise\dataset | train | test |
| 10% | 0.9911 | 0.9875 |
| 20% | 0.9842 | 0.9825 |
| 40% | 0.9776 | 0.9692 |



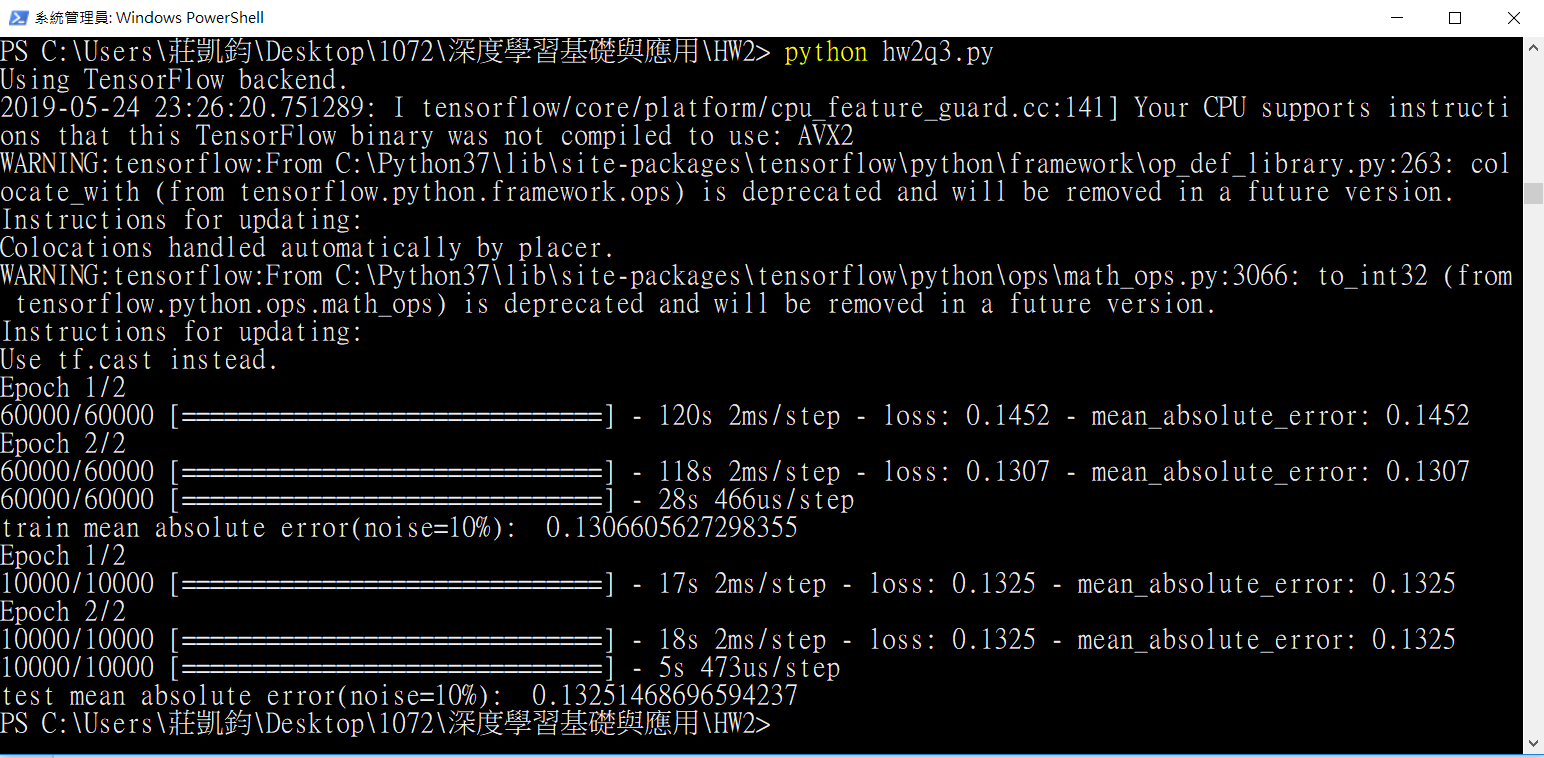




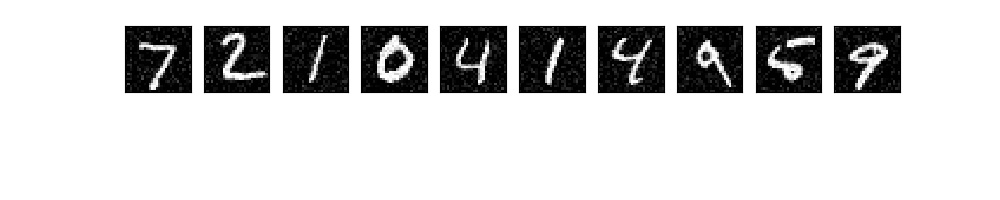
1. (20%) Following the previous question, please design a noise removal model based on convolutional neural networks for the images with added noise. Report the mean absolute error for the training and testing datasets.

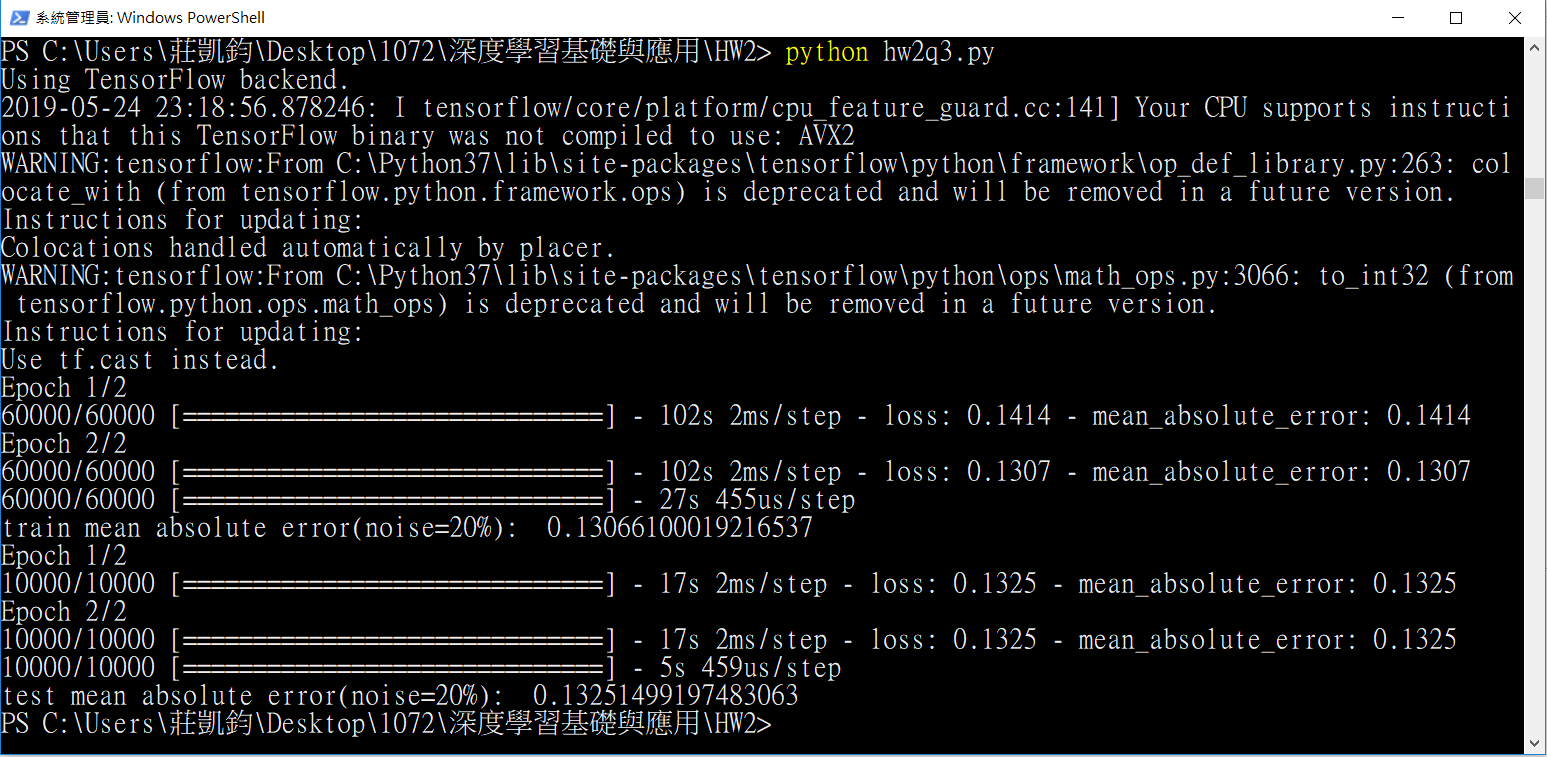
採用autoencode，首先encoding做了各3次Conv2D以及MaxPooling2D，接著做decoding，各做了3次Conv2D以及UpSampling，最後compile使用adam做為optimizer，loss及metrics採用mean absolute error

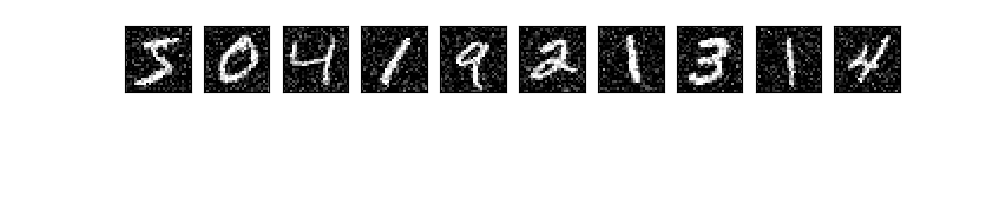
|  |  |  |
| --- | --- | --- |
| mean absolute error | train | test |
| 10% | 0.1306 | 0.1325 |
| 20% | 0.1306 | 0.1325 |
| 40% | 0.1306 | 0.1325 |



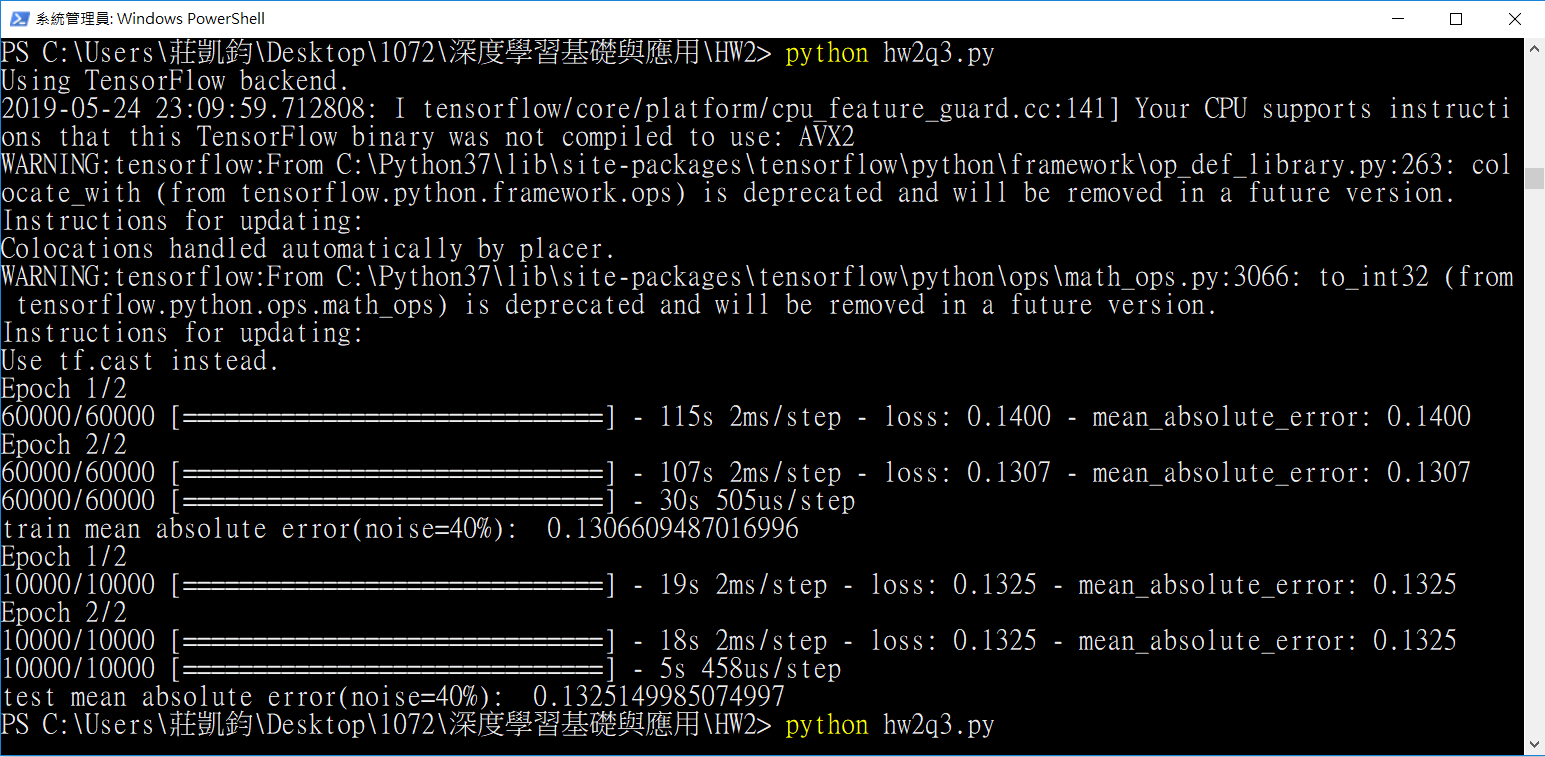
















1. (10% Bonus) Please use the noise removal model to pre-process the images before training the classification model. Would this help improve the prediction accuracy for the images with 10%, 20%, and 40% of noise?

是的，能夠小幅improve accuracy，因為經過auto encoder處理後能夠移除部分noise，找到使loss最小的weight，使得與原圖的相似性最高