CS420: Assignment 1

Hartono Tjakrawinata @ Jonathan Chow, SID: 01339300, hartonot.2019

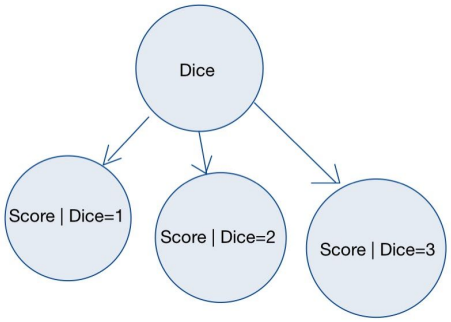
Question 1: Bayesian Networks

1. True
    a. Blocked Trail (C > E > G, Cascade)
    b. Blocked Trail (C > A < D, V Structure)
2. False
    a. Blocked Trail (F > D > A, Cascade)
    b. Active Trail (C > E > G < F, V Structure)
3. False
    a. Active Trail (E < C > A < D, V Structure)
    b. Blocked Trail (E > G < F, V Structure)
4. True
    a. Blocked Trail (C > A < D, V Structure)
    b. Blocked Trail (E > G < F, V Structure)
5. True
    a. Blocked Trail (A > D > F, Cascade)
    b. Blocked Trail (A < C > E, Common Cause)

Hartono Tjakrawinata @ Jonathan Chow, SID: 01339300, hartonot.2019

Question 2: Bayesian Networks

(i) Draw the Bayes net corresponding to this setup



Variable Name	Domain	Interpretation
Dice	{1,2,3}	Set of dice that could be picked up
Score | Dice=1	{1,2,3,4}	Set of Scores, given dice 1 was picked
Score | Dice=2	{1,2,3,4}	Set of Scores, given dice 2 was picked
Score | Dice=3	{1,2,3,4}	Set of Scores, given dice 3 was picked

(ii) Conditional Probabilities Tables for Random Variables

P(D=1)	P(D=2)	P(D=3)
0.3333	0.3333	0.3333

Table 1: Conditional Probabilities for Dice

P(S=1|D=1)	P(S=2|D=1)	P(S=3|D=1)	P(S=4|D=1)
0.25	0.25	0.25	0.25

Table 2: Conditional Probabilities for Score given Dice=1

P(S=1|D=2)	P(S=2|D=2)	P(S=3|D=2)	P(S=4|D=2)
0.5	0.1666	0.1666	0.1666

Table 3: Conditional Probabilities for Score given Dice=2

P(S=1|D=3)	P(S=2|D=3)	P(S=3|D=3)	P(S=4|D=3)
0.1666	0.5	0.1666	0.1666

Table 4: Conditional Probabilities for Score given Dice=3

(iii) Assume that the observed outcome was X1=1, X2=3, X3=2. Calculate which dice (D1 or D2 or D3) was most likely to have been picked from the box.

$$P(Dice \mid X1, X2, X3) = \frac{P(Dice) \times P(X1, X2, X3 \mid Dice)}{P(X1, X2, X3)}$$

$$P(Dice) = \frac{1}{3}$$

$$P(X1 = 1, X2 = 3, X3 = 2 \mid Dice = 1) = \frac{1}{4}^{3} = \frac{1}{64}$$

$$P(X1 = 1, X2 = 3, X3 = 2 \mid Dice = 2) = \frac{1}{6}^{2} \times \frac{1}{2} = \frac{1}{72}$$

$$P(X1 = 1, X2 = 3, X3 = 2 \mid Dice = 3) = \frac{1}{6}^{2} \times \frac{1}{2} = \frac{1}{72}$$

$$P(X1 = 1, X2 = 3, X3 = 2) = \frac{1}{64} \times \frac{1}{3} + \frac{1}{72} \times \frac{1}{3} + \frac{1}{72} \times \frac{1}{3} = \frac{25}{1728}$$

$$P(Dice = 1 \mid X1 = 1, X2 = 3, X3 = 2) = \frac{\frac{1}{3} \times \frac{1}{64}}{\frac{25}{1728}} = \frac{9}{25} = 0.36$$

$$P(Dice = 2 \mid X1 = 1, X2 = 3, X3 = 2) = \frac{\frac{1}{3} \times \frac{1}{72}}{\frac{25}{1728}} = \frac{8}{25} = 0.32$$

$$P(Dice = 3 \mid X1 = 1, X2 = 3, X3 = 2) = \frac{\frac{1}{3} \times \frac{1}{72}}{\frac{25}{1728}} = \frac{8}{25} = 0.32$$

Since, $P(Dice = 1 \mid X1 = 1, X2 = 3, X3 = 2) = 0.36$ has the greatest probability. Therefore, **Dice 1** would most likely have been picked from the box.

CS420: Assignment 1
Hartono Tjakrawinata @ Jonathan Chow, SID: 01339300, hartonot.2019

Question 3: Bayesian Networks

Print out your code showing the implementation in pgmpy and attach as part of your solution pdf. [2.5 points]

```
!pip install pgmpy
# Import relevant packages
from pgmpy.models import BayesianModel
from pgmpy.factors.discrete import TabularCPD
import sys
```

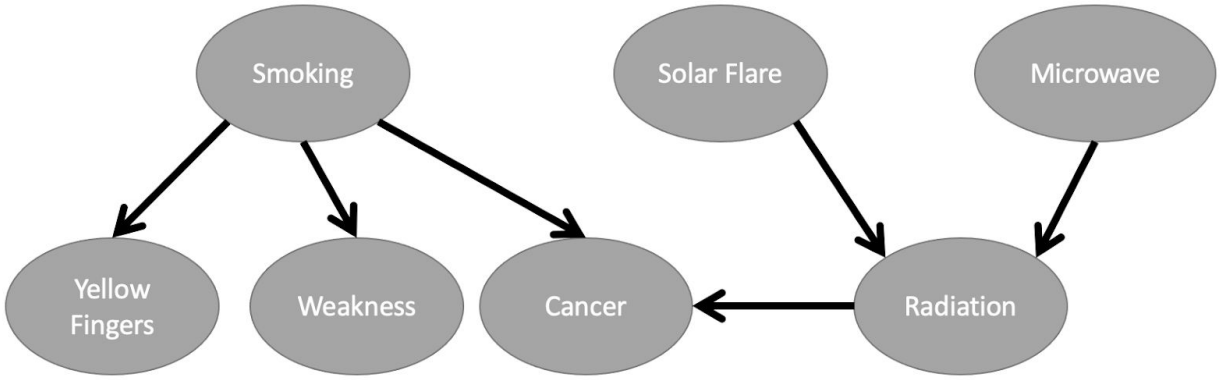
```
# We first create a model which containts edges of the graph
model = BayesianModel([('Smoking', 'Yellow'), ('Smoking', 'Weakness'), ('Smoking', 'Cancer'), ('Flare', 'Radiation'),
                       ('Microwave', 'Radiation'), ('Radiation', 'Cancer')])

# Prior probabilities
cpd_S = TabularCPD(variable='Smoking', variable_card=2, values=[[0.85], [0.15]])
cpd_F = TabularCPD(variable='Flare', variable_card=2, values=[[0.99], [0.01]])
cpd_M = TabularCPD(variable='Microwave', variable_card=2, values=[[0.05], [0.95]])

# Conditional probabilities
cpd_Y = TabularCPD(variable='Yellow', variable_card=2, values = [[0.89, 0.2], [0.11, 0.8]],
                                    evidence = ['Smoking'],
                                    evidence_card=[2])
cpd_W = TabularCPD(variable='Weakness', variable_card=2, values = [[0.8, 0.1], [0.2, 0.9]],
                                    evidence = ['Smoking'],
                                    evidence_card=[2])
cpd_R = TabularCPD(variable='Radiation',variable_card=2, values = [
                                    [0.9, 0.8, 0.8, 0.1], [0.1, 0.2, 0.2, 0.9]],
                                    evidence = ['Flare', 'Microwave'],
                                    evidence_card=[2, 2])
cpd_C = TabularCPD(variable='Cancer',variable_card=2, values = [
                                    [0.9, 0.4, 0.7, 0.1], [0.1, 0.6, 0.3, 0.9]],
                                    evidence = ['Smoking', 'Radiation'],
                                    evidence_card=[2, 2])
model.add_cpds(cpd_S, cpd_F, cpd_M, cpd_Y, cpd_W, cpd_R, cpd_C)
print(model.check_model())
print(model.get_cpds('Radiation'))
print(model.get_cpds('Cancer'))
```

```
############### Inference ##################

from pgmpy.inference import VariableElimination
infer = VariableElimination(model)
print('\n')

# P(C | W=1)
phi_query = infer.query(['Cancer'], evidence={'Weakness':1}, joint = False)
factor = phi_query['Cancer']
print('P(C | W=1): Probability of cancer given weakness = 1')
print(factor)

# P(S | C=1)
phi_query = infer.query(['Smoking'], evidence={'Cancer':1}, joint = False)
factor = phi_query['Smoking']
print('P(S | C=1): Probability of smoking given cancer = 1')
print(factor)

# P(C | M=0)
phi_query = infer.query(['Cancer'], evidence={'Microwave':0}, joint = False)
factor = phi_query['Cancer']
print('P(C | M=0): Probability of cancer given microwave = 0')
print(factor)
```

Hartono Tjakrawinata @ Jonathan Chow, SID: 01339300, hartonot.2019

---

(1) Draw the Bayesian network clearly showing the nodes and arrows showing relationship among all the variables (you can use drawing tools in PowerPoint or Keynote)



(2) What is the probability of cancer given weakness = 1?

P(Cancer=1 | Weakness=1) = 0.2983

(3) What is the probability of smoking given cancer=1?

P(Smoking=1 | Cancer=1) = 0.2700

(4) Are Smoking and Radiation independent given cancer? Justify your answer.

Not Independent: By V Structure, Observing cancer couples Smoking and Radiation.

(5) What is the probability of cancer if you never use a microwave?

P(Cancer=1 | Microwave=0) = 0.1820

Question 4: ANN

(a) Download the fashion_mnist dataset (you can use tf.keras.datasets as covered in class). [1 point]

```
%tensorflow_version 1.x
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.layers import Flatten, Dense, Dropout, BatchNormalization
from keras.datasets import fashion_mnist
```

```
[30] class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat','Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
     (train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
     print ('Training data size:', train_images.shape, 'Test data size', test_images.shape)

     Training data size: (60000, 28, 28) Test data size (10000, 28, 28)
```

(b) Create an ANN with 1 input layer, at least two hidden layer with at least 10 nodes per layer. [3 points]

(c) Create 1 output layer. What is the size of output layer? What should be the activation function for output layer? [2 points]

Size of output is 10, 1 node for each “fashion_mnist” category. The activation function should be softmax as the 10 outputs will then be values between 0 and 1 that sums to 1, which allows us to treat said values as probabilities.

(d) Compile and train the neural network with the appropriate loss function (what should be the loss function type?) [2 points]

Loss function used is “sparse_categorical_crossentropy” for the categorical classification used, where each item only has 1 category.

```
model = keras.Sequential([
    Flatten(input_shape=(28, 28)),
    BatchNormalization(),
    Dense(512, activation='relu'),
    Dropout(0.3),
    BatchNormalization(),
    Dense(256, activation='relu'),
    Dropout(0.3),
    BatchNormalization(),
    Dense(64, activation='relu'),
    Dropout(0.3),
    BatchNormalization(),
    Dense(24, activation='relu'),
    Dropout(0.3),
    BatchNormalization(),
    Dense(10, activation='softmax')
])

model.compile(loss="sparse_categorical_crossentropy",
              optimizer="adam",
              metrics = ['accuracy'])

history = model.fit(train_images, train_labels, batch_size=256, epochs=30)
```

(e) What is the final accuracy on the testing data? [2 points]

Test Data Accuracy = 89.49%

```
Epoch 30/30
60000/60000 [==============================] - 7s 121us/sample - loss: 0.2295 - acc: 0.9226
```

```
test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)
print('\nTest accuracy:', test_acc)
```

```
10000/10000 - 1s - loss: 0.3192 - acc: 0.8949

Test accuracy: 0.8949
```

Hartono Tjakrawinata @ Jonathan Chow, SID: 01339300, hartonot.2019

Question 5: CNN

```
[1] %tensorflow_version 1.x
    import tensorflow as tf
    from keras.datasets import cifar10
    import cv2
    import numpy as np
    import matplotlib.pyplot as plt
```

```
TensorFlow 1.x selected.
Using TensorFlow backend.
```

```
[2] class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer','dog', 'frog', 'horse', 'ship', 'truck']
    (train_images, train_labels), (test_images, test_labels) = cifar10.load_data()
    print ('Training data size:', train_images.shape, 'Test data size', test_images.shape)
```

```
Training data size: (50000, 32, 32, 3) Test data size (10000, 32, 32, 3)
```

(a) Download the pre-trained MobileNetV2 network from Keras Applications (https://keras.io/applications/) [1 point]

(b) Remove the final output layer of the downloaded network (to do this, please check the flag “include_top” in Keras) [1 point]

```
[3] # RESIZE
    minSize = 96
    def resize(images):
      imageCount = len(images)
      resized = np.zeros((imageCount, minSize, minSize, 3), dtype=np.float32)
      for i in range(0, imageCount):
        resized[i] = cv2.resize(images[i], (minSize, minSize), interpolation = cv2.INTER_AREA)
      return resized

    train_images = resize(train_images)
    test_images = resize(test_images)
    print(test_images.shape, train_images.shape)

    # PREPROCESSING (between -1 and 1)
    preprocess_input = tf.keras.applications.mobilenet_v2.preprocess_input
    train_images = preprocess_input(train_images)
    test_images = preprocess_input(test_images)

    # DOWNLOAD MODEL
    base_model = tf.keras.applications.MobileNetV2(input_shape=(minSize,minSize,3),
                                                   include_top=False,
                                                   weights='imagenet')
```

```
(10000, 96, 96, 3) (50000, 96, 96, 3)
```

(c) Extend the MobileNetV2 model [2 points]

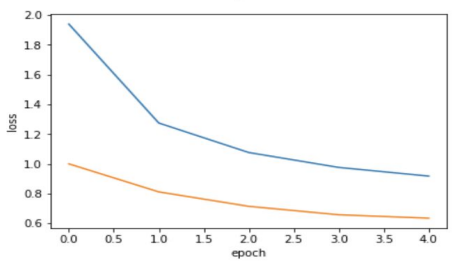
(d) Add the appropriate loss function, compile and train the modified network. [2 points]

CS420: Assignment 1
Hartono Tjakrawinata @ Jonathan Chow, SID: 01339300, hartonot.2019

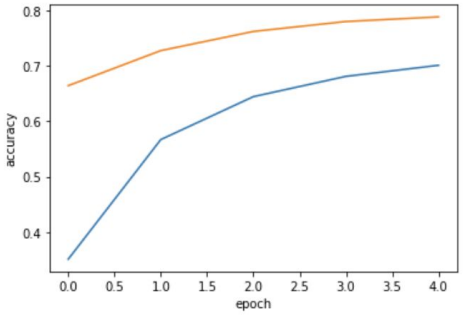
Training 1 - learning rate = 0.0001

```
[4]  base_model.trainable=False

     inputs = tf.keras.Input(shape=(minSize, minSize, 3))
     x = base_model(inputs, training=False)
     x = tf.keras.layers.MaxPooling2D(pool_size=(3, 3), strides=None, padding="valid")(x)
     x = tf.keras.layers.Flatten()(x)

     x = tf.keras.layers.Dense(1024, activation='relu')(x)
     x = tf.keras.layers.Dropout(0.4)(x)
     x = tf.keras.layers.BatchNormalization()(x)

     x = tf.keras.layers.Dense(512, activation='relu')(x)
     x = tf.keras.layers.Dropout(0.4)(x)
     x = tf.keras.layers.BatchNormalization()(x)

     x = tf.keras.layers.Dense(128, activation='relu')(x)
     x = tf.keras.layers.Dropout(0.4)(x)
     x = tf.keras.layers.BatchNormalization()(x)

     x = tf.keras.layers.Dense(64, activation='relu')(x)
     x = tf.keras.layers.Dropout(0.4)(x)
     x = tf.keras.layers.BatchNormalization()(x)

     outputs = tf.keras.layers.Dense(10, activation='softmax')(x)
     model = tf.keras.Model(inputs, outputs)

     base_learning_rate=0.0001
     batch_size=32
     epochs=5

     model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=base_learning_rate),
                   loss='sparse_categorical_crossentropy',
                   metrics=['accuracy'])
     model.summary()
```

```
[5]  history = model.fit(train_images, train_labels, batch_size=batch_size, epochs=epochs, validation_data=(test_images, test_labels))

     Train on 50000 samples, validate on 10000 samples
     Epoch 1/5
     50000/50000 [==============================] - 25s 499us/sample - loss: 1.9387 - acc: 0.3509 - val_loss: 0.9998 - val_acc: 0.6646
     Epoch 2/5
     50000/50000 [==============================] - 22s 441us/sample - loss: 1.2741 - acc: 0.5672 - val_loss: 0.8117 - val_acc: 0.7280
     Epoch 3/5
     50000/50000 [==============================] - 22s 439us/sample - loss: 1.0760 - acc: 0.6447 - val_loss: 0.7146 - val_acc: 0.7626
     Epoch 4/5
     50000/50000 [==============================] - 22s 442us/sample - loss: 0.9759 - acc: 0.6814 - val_loss: 0.6579 - val_acc: 0.7806
     Epoch 5/5
     50000/50000 [==============================] - 22s 442us/sample - loss: 0.9177 - acc: 0.7014 - val_loss: 0.6350 - val_acc: 0.7889
```




```
10000/10000 - 3s - loss: 0.6350 - acc: 0.7889

Test accuracy: 0.7889
```

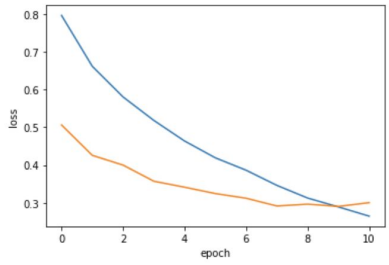


Training 2 (fine tuning) - learning rate = 0.00001

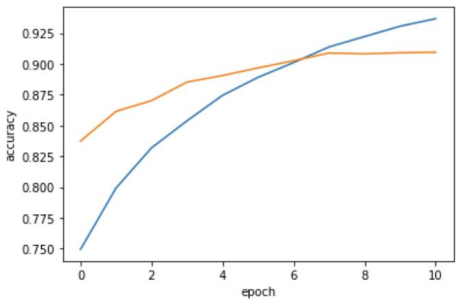
```
[7]  base_model.trainable = True
     print("Number of layers in the base model: ", len(base_model.layers))
     fine_tune_at = 80

     for layer in base_model.layers[:fine_tune_at]:
       layer.trainable =  False

     model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=base_learning_rate/10),
                   loss='sparse_categorical_crossentropy',
                   metrics=['accuracy'])
     model.summary()
```

```
fine_tune_epochs = 10
total_epochs =  epochs + fine_tune_epochs

history_fine = model.fit(train_images, train_labels,
                         epochs=total_epochs,
                         initial_epoch=history.epoch[-1],
                         validation_data=(test_images, test_labels))
```

```
Train on 50000 samples, validate on 10000 samples
Epoch 5/15
50000/50000 [==============================] - 36s 722us/sample - loss: 0.7959 - acc: 0.7494 - val_loss: 0.5062 - val_acc: 0.8374
Epoch 6/15
50000/50000 [==============================] - 34s 681us/sample - loss: 0.6612 - acc: 0.7992 - val_loss: 0.4258 - val_acc: 0.8615
Epoch 7/15
50000/50000 [==============================] - 34s 681us/sample - loss: 0.5802 - acc: 0.8319 - val_loss: 0.4000 - val_acc: 0.8702
Epoch 8/15
50000/50000 [==============================] - 34s 679us/sample - loss: 0.5182 - acc: 0.8539 - val_loss: 0.3573 - val_acc: 0.8853
Epoch 9/15
50000/50000 [==============================] - 34s 683us/sample - loss: 0.4637 - acc: 0.8745 - val_loss: 0.3415 - val_acc: 0.8906
Epoch 10/15
50000/50000 [==============================] - 34s 684us/sample - loss: 0.4192 - acc: 0.8891 - val_loss: 0.3247 - val_acc: 0.8968
Epoch 11/15
50000/50000 [==============================] - 34s 683us/sample - loss: 0.3864 - acc: 0.9011 - val_loss: 0.3124 - val_acc: 0.9026
Epoch 12/15
50000/50000 [==============================] - 34s 688us/sample - loss: 0.3464 - acc: 0.9138 - val_loss: 0.2919 - val_acc: 0.9089
Epoch 13/15
50000/50000 [==============================] - 34s 677us/sample - loss: 0.3129 - acc: 0.9223 - val_loss: 0.2968 - val_acc: 0.9082
Epoch 14/15
50000/50000 [==============================] - 34s 689us/sample - loss: 0.2894 - acc: 0.9307 - val_loss: 0.2908 - val_acc: 0.9091
Epoch 15/15
50000/50000 [==============================] - 34s 685us/sample - loss: 0.2650 - acc: 0.9368 - val_loss: 0.3007 - val_acc: 0.9095
```




```
10000/10000 - 3s - loss: 0.3007 - acc: 0.9095

Test accuracy: 0.9095
```



CS420: Assignment 1
Hartono Tjakrawinata @ Jonathan Chow, SID: 01339300, hartonot.2019

---

(1) Write how you extended the MobileNetV2 model (how many layers you added, what type of layers, how many nodes per layer, their activation function etc). [2 points]

I added 16 layers in total. Many of them being Dropout and Normalisation layers due to overfitting issues I faced early on.

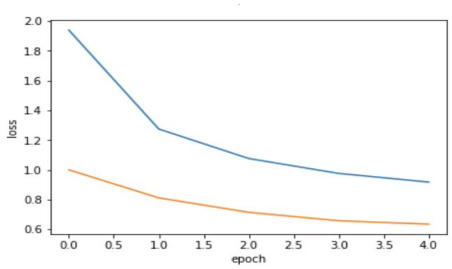
Layer Type	No. of layers	No. of Nodes	Activations
Input	1	NA	NA
Output (Dense)	1	10	softmax
Hidden (Dense)	4	1024, 512, 128, 64	relu
MaxPooling2D	1	NA	NA
Flatten	1	NA	NA
Dropout(0.4)	4	NA	NA
BatchNormalisation	4	NA	NA

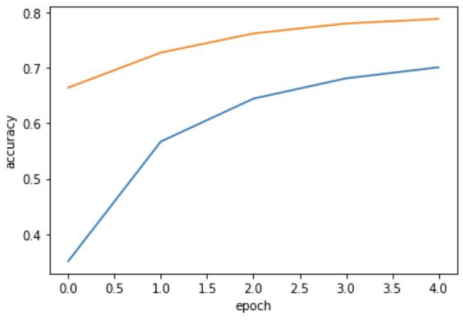
(2) Plot the loss function value with respect to the epoch number on the training data. How did you decide when to terminate training? [1 point]

I terminated the training, after the loss graph for the test data started to plateau.

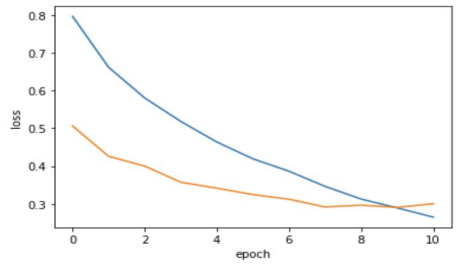
In Training 1, it plateaued after 5 epochs, where the training was terminated in favour of a lower training rate where it was terminated at 10 epochs, despite plateauing at 7 epochs.

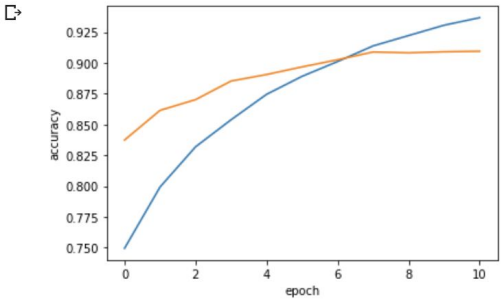
Graph 1 & 2: Training 1 - learning rate = 0.0001





Graph 3 & 4: Training 2 - learning rate = 0.00001 (fine tuning)





(3) Show the accuracy of the trained classifier over the entire testing dataset. [1 point]

Test Accuracy=0.9095