Candace Edwards

ICS 635: Homework 5

Part 1: Notebook Link

Part 2: Q4: Notebook Link

```
! pip install tensorflow keras
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-packages (2.12.0)
Requirement already satisfied: keras in /usr/local/lib/python3.10/dist-packages (2.12.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=2.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (23.3.3)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.4.0)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.54.0)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.8.0)
Requirement already satisfied: jax>=0.3.15 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.4.8)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (16.0.0)
Requirement already satisfied: numpy<1.24,>=1.22 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.22.4)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow) (23.1)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in /usr/local/lib/pythor
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow) (67.7.2)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: tensorboard<2.13,>=2.12 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.12.2)
Requirement already satisfied: tensorflow-estimator<2.13,>=2.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.12.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.3.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (4.5.0)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.14.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.32.0
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorflow) (0.40.
Requirement already satisfied: ml-dtypes>=0.0.3 in /usr/local/lib/python3.10/dist-packages (from jax>=0.3.15->tensorflow) (0.1.0)
Requirement already satisfied: scipy>=1.7 in /usr/local/lib/python3.10/dist-packages (from jax>=0.3.15->tensorflow) (1.10.1)
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.13,>=2.12->tensorflo
Requirement already satisfied: google-auth-oauthlib<1.1,>=0.5 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.13,>=2.12->
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.13,>=2.12->tensorflow) (3.
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.13,>=2.12->tensorflow)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.13,>
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.13,>=2.12->t
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.13,>=2.12->tensorflow) (2.
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboar
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.13,>=
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from google-auth-oauthlib<1.1,>=0.5-
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.13
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboa
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.13,>=2.1
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1->tensorboard<2.13,>=2.
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1->google-auth<
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from requests-oauthlib>=0.7.0->google-auth-oa
```

```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score

import cv2

faces = datasets.fetch_olivetti_faces()

import tensorflow as tf
from tensorflow.keras import datasets, layers, models, utils

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd


###
### Load the dataset.
###
```

```
X_grey = faces['images']
y = faces['target']

X = []
for a in X_grey:
  image_rgb = cv2.cvtColor(a, cv2.COLOR_GRAY2RGB)
  X.append(image_rgb)
X = np.array(X)

y = utils.to_categorical(
    y, num_classes=40, dtype='float32'
)


###
### Split data into train and test sets. Do not change. For reference only.
###

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=42
)


###
### Sourced from class notes
###
def plot_acc(history, ax = None, xlabel = 'Epoch #'):
    if hasattr(history, 'history_'):
      history = history.history_
    else:
      history = history.history
    history.update({'epoch':list(range(len(history['val_accuracy'])))})
    history = pd.DataFrame.from_dict(history)

    best_epoch = history.sort_values(by = 'val_accuracy', ascending = False).iloc[0]['epoch']

    if not ax:
      f, ax = plt.subplots(1,1)
    sns.lineplot(x = 'epoch', y = 'val_accuracy', data = history, label = 'Validation', ax = ax)
    sns.lineplot(x = 'epoch', y = 'accuracy', data = history, label = 'Training', ax = ax)
    ax.axhline(0.5, linestyle = '--',color='red', label = 'Chance')
    ax.axvline(x = best_epoch, linestyle = '--', color = 'green', label = 'Best Epoch')
    ax.legend(loc = 7)
    ax.set_ylim([0.4, 1])

    ax.set_xlabel(xlabel)
    ax.set_ylabel('Accuracy (Fraction)')

    plt.show()


###
### CNN Implementation
###

def get_neural_network(X_train, y_train):
  """
  Define, train, and return the neural network.

  You may replace this code with any TensorFlow model that you wish.
  """

  model = models.Sequential()
  model.add(layers.Conv2D(65, (3, 3), activation='relu', input_shape=(64, 64, 3)))
  ##model.add(layers.Conv2D(128, (3, 3), activation='relu', input_shape=(64, 64, 3)))

  model.add(layers.MaxPooling2D((2, 2)))
  model.add(layers.Dropout(0.05)) #added dropout layer [0.05,0.15,0.15,0.05,0.07]
  '''
  model.add(layers.Conv2D(64, (3, 3), activation='relu'))
  '''
  #changed to paddin =same
  model.add(layers.Conv2D(64,(3,3), activation ='relu', padding="same"))
  model.add(layers.MaxPooling2D((2, 2)))
  model.add(layers.Dropout(0.005)) #[0.005, 0.15,0.20,0.005,0.05]

  #model.add(layers.Conv2D(64, (3, 3), activation='relu'))
```

```
    model.add(layers.Conv2D(32, (3, 3), activation='relu'))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Dropout(0.005)) #[0.0005, 0.15,0.30]
    ##
    model.add(layers.Flatten())
    model.add(layers.Dense(64, activation='relu'))
    model.add(layers.Dense(40, activation='softmax'))

    model.compile(optimizer='adam',
                loss=tf.keras.losses.CategoricalCrossentropy(from_logits=False),
                metrics=['accuracy'])

    model.fit(X_train, y_train,
            epochs=34)




    # #validation data: resplit traininging data
    # X_train, X_val,y_train,y_val = train_test_split(X_train,y_train,test_size= 0.25,random_state=10)

    # history = model.fit(X_train, y_train, \
    #                   validation_data=(X_val, y_val), \
    #                   epochs=40)
    # plot_acc(history)
    # ##

    return model


###
### Train the neural network.
###

model = get_neural_network(X_train, y_train)
```

```
Epoch 1/34
10/10 [==============================] - 5s 379ms/step - loss: 3.7020 - accuracy: 0.0367
Epoch 2/34
10/10 [==============================] - 5s 538ms/step - loss: 3.6887 - accuracy: 0.0300
Epoch 3/34
10/10 [==============================] - 4s 376ms/step - loss: 3.6872 - accuracy: 0.0233
Epoch 4/34
10/10 [==============================] - 4s 366ms/step - loss: 3.6812 - accuracy: 0.0467
Epoch 5/34
10/10 [==============================] - 5s 530ms/step - loss: 3.6758 - accuracy: 0.0267
Epoch 6/34
10/10 [==============================] - 4s 363ms/step - loss: 3.6707 - accuracy: 0.0367
Epoch 7/34
10/10 [==============================] - 4s 371ms/step - loss: 3.6608 - accuracy: 0.0733
Epoch 8/34
10/10 [==============================] - 5s 533ms/step - loss: 3.6502 - accuracy: 0.0333
Epoch 9/34
10/10 [==============================] - 4s 363ms/step - loss: 3.6167 - accuracy: 0.0567
Epoch 10/34
10/10 [==============================] - 4s 369ms/step - loss: 3.5504 - accuracy: 0.0533
Epoch 11/34
10/10 [==============================] - 6s 636ms/step - loss: 3.3509 - accuracy: 0.1233
Epoch 12/34
10/10 [==============================] - 4s 395ms/step - loss: 3.1081 - accuracy: 0.1600
Epoch 13/34
10/10 [==============================] - 4s 370ms/step - loss: 2.7392 - accuracy: 0.2733
Epoch 14/34
10/10 [==============================] - 4s 405ms/step - loss: 2.1422 - accuracy: 0.4400
Epoch 15/34
10/10 [==============================] - 5s 454ms/step - loss: 1.7623 - accuracy: 0.5367
Epoch 16/34
10/10 [==============================] - 4s 369ms/step - loss: 1.3453 - accuracy: 0.6267
Epoch 17/34
10/10 [==============================] - 4s 374ms/step - loss: 1.3684 - accuracy: 0.6200
Epoch 18/34
10/10 [==============================] - 5s 511ms/step - loss: 0.9997 - accuracy: 0.7467
Epoch 19/34
10/10 [==============================] - 4s 367ms/step - loss: 0.8564 - accuracy: 0.7367
Epoch 20/34
10/10 [==============================] - 4s 365ms/step - loss: 0.6803 - accuracy: 0.8100
Epoch 21/34
10/10 [==============================] - 5s 525ms/step - loss: 0.4695 - accuracy: 0.8767
Epoch 22/34
10/10 [==============================] - 4s 359ms/step - loss: 0.4808 - accuracy: 0.8567
Epoch 23/34
```

```
10/10 [==============================] - 4s 368ms/step - loss: 0.4862 - accuracy: 0.8567
Epoch 24/34
10/10 [==============================] - 6s 583ms/step - loss: 0.3250 - accuracy: 0.9100
Epoch 25/34
10/10 [==============================] - 4s 369ms/step - loss: 0.2805 - accuracy: 0.9100
Epoch 26/34
10/10 [==============================] - 4s 371ms/step - loss: 0.2819 - accuracy: 0.9067
Epoch 27/34
10/10 [==============================] - 5s 535ms/step - loss: 0.3014 - accuracy: 0.9133
Epoch 28/34
10/10 [==============================] - 4s 384ms/step - loss: 0.1883 - accuracy: 0.9500
Epoch 29/34
10/10 [==============================] - 4s 381ms/step - loss: 0.1988 - accuracy: 0.9433
```

```
###
### Get the result on the test set.
###

y_pred = model.predict(X_test).argmax(1)
y_true = y_test.argmax(1)

print('Predicted values: ', y_pred)
print('Actual values: ', y_true)

print('F1 score: ', f1_score(y_true, y_pred, average='weighted'))
```

```
4/4 [==============================] - 1s 123ms/step
Predicted values:  [20 28  3 21  9  8 32  9 26 12  0 36  5  7 13  4 27 37 23 38  7  1 39 25
  0 20 11 22 26 14 39  3 26  5 23 11  8 34 15 14  9  5  7 36  8 38 14 18
  2 17  4 32 33  7 37  3 22 22  3 15 12 29 36 20 10  3 35 26 39  7 32 14
  0  4 38 24 22 36 17 28 12  1 20 36 27  6 24 30 10  9 23 33 11 22 18 31
 37 38 23 15]
Actual values:  [20 28  3 21  9  8 32  9 26 12  0 36  5  7 13  4 27 37 23 38  7  1 39 27
  0 39 11 22 26 10 39 19 26  5 23 11 11 34 15 14 38  5  7  2  8 38 14 18
  2 17  4 32 33  7 37  3 22 17  3 15 12 29 25  7 10  3 35 26 39  7 32 14
  0  4 38 24 22 36 17 28  0  1 20 25 27  6 24 30 10  9 23 33 11 22 18 31
 37 38 23  7]
F1 score:  0.8662698412698413
```

```
###
### Save model and weights. These will be used for grading.
### The weights will be downloaded to your computer. Use Chrome.
###

# TODO: CHANGE TO YOUR NAME
student_name = 'candace_edwards'

model.save(student_name + '_model.h5')

from google.colab import files
files.download(student_name + '_model.h5')
```