DSP Assignment 3

ECE16U017

Making a Automatic Speech Recognition convolutional neural network to recognise digits

Done on PYTHON

(Note - the multiline comments give function description of the below function called and single line comments give the line description)

Data Preparation -

Code:

```
import librosa
import os
import numpy as np
import matplotlib.pyplot as plt
from librosa.display import specshow
import cv2.cv2 as cv2

def printpic(path):
    for i in os.listdir(path):
        p = os.path.join(path,i)
        for j in os.listdir(p):
            pa = os.path.join(p,j)
            wav,sr = librosa.load(pa)
            mfc = librosa.feature.melspectrogram(wav,sr)
            specshow(mfc)
            plt.axis('off')
            plt.savefig((j.replace(".wav","")+".png"),bbox_inches='tight'
                        , transparent=True, pad_inches=0.0)

def createnpy(path):
    data = []
    label = []
    fg = 0
    for i in os.listdir(path):
        p = os.path.join(path,i)
```

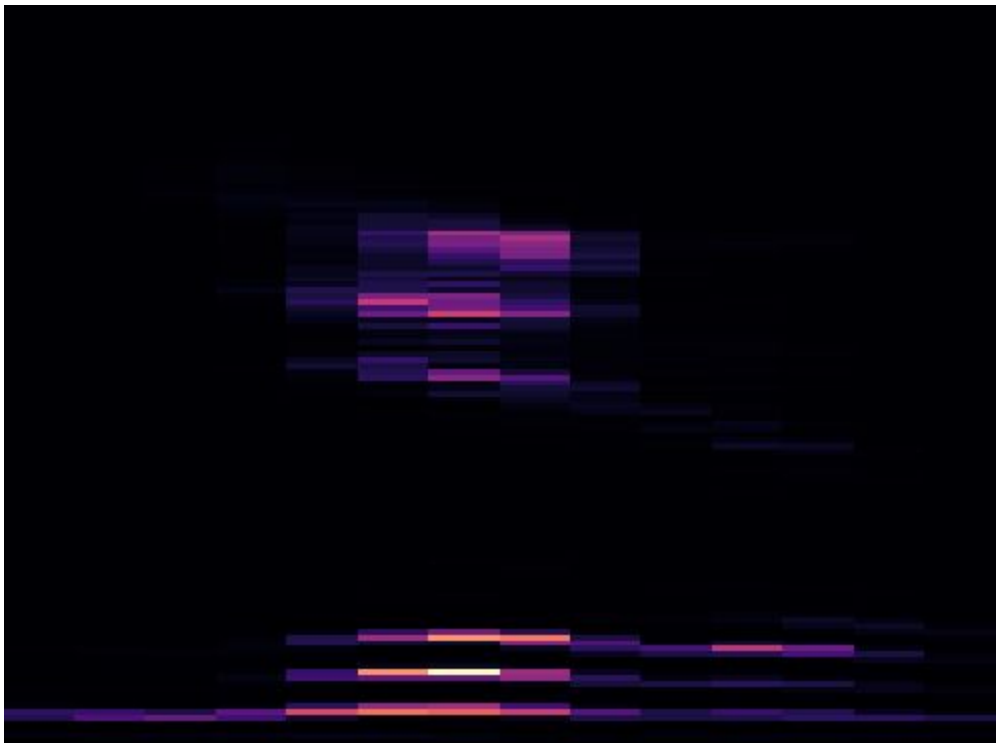
```
        for j in os.listdir(p):
            pa = os.path.join(p,j)
            img = cv2.imread(pa,cv2.IMREAD_COLOR)
            # img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
            im = cv2.resize(img,(100,100))
            pic = np.asarray(im)
            data.append(pic)
            label.append(fg)
        fg += 1
    dat = np.asarray(data)
    lab = np.asarray(label)
    return dat,lab

if __name__ == "__main__":
    """
    First we read all wav files
    we then plot their respective mel spectographs using librosa
    The spectrographs images are then stored in png format without axes
    """

    printpic("C:\\Users\\rhin1\\Downloads\\DSP\\wavs")

    """
    We have the mel spectrogram images, we load the images using opencv
    we are reading the images in BGR channel
    all images are resized to a standard shape (100,100)
    we create a npy of resized images (save file of numpy array)
    for both the data and the corresponding label
    the npy files help importing all the image data easily and efficiently
    """
    d,l = createnpy("C:\\Users\\rhin1\\Downloads\\DSP\\melpic")
    print(d.shape,l.shape) #get the shape of image data array and label
array
    np.save('d.npy',d) # create npy file of data
    np.save('l.npy',l) #create npy file of label
```

Example picture of mel spectrogram:



The shape of the data and label npy:
Data - 2000 img of 100x100 size with 3 channels BGR
Label - 2000

```
PS C:\Users\rhin1\Downloads\DSP> python .\datapicpre.py
(2000, 100, 100, 3) (2000,)
```

Neural Network Model -

Code:

```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3' # turning of logging WARNINGS and
INFO

import tensorflow as tf
import numpy as np
from sklearn.model_selection import train_test_split
```

```
if __name__ == "__main__":
    data = np.load('d.npy') # loading the data images
    label = np.load('l.npy') # loading corresponding labels
    traind,testd,trainl,testl = train_test_split(data,label,test_size =
0.1,random_state=42)

    """
    the above code helps split the data and labels into test and train
data
    and shuffle the contents in the respective numpy arrays
    """

    print(traind.shape,trainl.shape,testd.shape,testl.shape) # get the
shape of all four arrays after split

    traind = traind/255.0 # normalizing the BGR channel values
    testd = testd/255.0 # normalizing the BGR channel values

    # the neural network is defined
    model = tf.keras.models.Sequential([
        tf.keras.layers.Conv2D(64 ,(5,5),activation = 'relu', input_shape
= (100,100,3)),
        tf.keras.layers.MaxPooling2D(2,2),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(32, activation="relu"),
        tf.keras.layers.Dropout(0.2),
        tf.keras.layers.Dense(10, activation="softmax")])

    # all the layers defined are complied with the optimizer and loss
functions
    model.compile(optimizer =
tf.keras.optimizers.Adam(learning_rate=0.001),
                  loss = 'sparse_categorical_crossentropy',
                  metrics = ['accuracy'])

    #training the model with train data and corresponding labels
    model.fit(traind, trainl, epochs = 6, validation_split = 0.1)
```

```
#testing our model with test data
test_loss, test_acc = model.evaluate(testd, testl)

#printing model accuracy percentage
print(test_acc*100)

#saving the model as a h5 file for further use
model.save("adr_model.h5")
```

Cmd Output:

Train data - 1620 img of 100x100 shape with BGR channel

Validation data - 180 img of 100x100 shape with BGR channel

Test data - 200 img of 100x100 shape with BGR channel

Model accuracy - 99.5%

```
PS C:\Users\rhin1\Downloads\DSP> python .\adr_model.py
(1800, 100, 100, 3) (1800,) (200, 100, 100, 3) (200,)
Train on 1620 samples, validate on 180 samples
Epoch 1/6
1620/1620 [==============================] - 17s 11ms/sample - loss: 0.6486 - accuracy: 0.7846 - val_loss: 0.1614 - val_accuracy: 0.9556
Epoch 2/6
1620/1620 [==============================] - 17s 10ms/sample - loss: 0.1013 - accuracy: 0.9679 - val_loss: 0.1029 - val_accuracy: 0.9667
Epoch 3/6
1620/1620 [==============================] - 17s 11ms/sample - loss: 0.1043 - accuracy: 0.9673 - val_loss: 0.0824 - val_accuracy: 0.9833
Epoch 4/6
1620/1620 [==============================] - 21s 13ms/sample - loss: 0.0657 - accuracy: 0.9778 - val_loss: 0.0484 - val_accuracy: 0.9833
Epoch 5/6
1620/1620 [==============================] - 19s 12ms/sample - loss: 0.0424 - accuracy: 0.9864 - val_loss: 0.0668 - val_accuracy: 0.9889
Epoch 6/6
1620/1620 [==============================] - 18s 11ms/sample - loss: 0.0537 - accuracy: 0.9846 - val_loss: 0.1053 - val_accuracy: 0.9889
200/200 [==============================] - 1s 3ms/sample - loss: 0.0129 - accuracy: 0.9950
99.50000047683716
```

Prediction using model -

Code:

```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3' # turning of logging WARNINGS and
INFO


import tensorflow as tf
import numpy as np
import librosa
from librosa.display import specshow
import matplotlib.pyplot as plt
import cv2.cv2 as cv2
```

```
def plots(path):
    for i in os.listdir(path):
        wav,sr = librosa.load(os.path.join(path,i))
        mfc = librosa.feature.melspectrogram(wav,sr)
        specshow(mfc)
        plt.axis('off')
        plt.savefig((i.replace(".wav","")+".png"),bbox_inches='tight',
transparent=True, pad_inches=0.0)

def arr(path):
    pic = []
    for i in os.listdir(path):
        img = cv2.imread(os.path.join(path,i),cv2.IMREAD_COLOR)
        j = cv2.resize(img,(100,100))
        j = np.asarray(j)
        pic.append(j)
    pic = np.asarray(pic)
    return pic

if __name__ == "__main__":
    """
    we have 10 audio samples each for a digit
    we will test our model using these test audio
    first we'll make mel spectrogram plots of 10 audio samples
    then we will save the spectrograph as images
    """
    plots("C:\\Users\\rhin1\\Downloads\\DSP\\dig1")
    """
    we are loading the 10 image files saved
    we are resizing the images to 100x100
    we are making a numpy array with BGR channel values
    """
    pred = arr("C:\\Users\\rhin1\\Downloads\\DSP\\meltest1")

    print(pred.shape) # print numpy array shape
    model = tf.keras.models.load_model("adr_model.h5") #loading saved
model
```

```
    predic = model.predict(pred) # making a prediction on test array
    print(predic) # printing all model probabilities
    val = [np.argmax(i) for i in predic] # segregating index of max
probability
    print(val) # printing the index value indirectly the digit recognised
```

Cmd Output:

Test arr shape - 10 img of 100x100 shape in BGR channel

Last is the Prediction array

```
PS C:\Users\rhin1\Downloads\DSP> python .\adr_pred.py
(10, 100, 100, 3)
[[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```