NAME : AAKASH MOHAN

ID : 1001656408

**SUBJECT CODE** : EE5353

**SUBJECT** : Neural Networks and Deep Learning

**PROGRAM ASSIGNMENT 9:**

**Transfer Learning using Google Colab**

import tensorflow as tf import random as rn import os,cv2

import numpy as np

os.environ['PYTHONHASHSEED'] = '0'

# Setting the seed for numpy-generated random numbers np.random.seed(37)

# Setting the seed for python random numbers rn.seed(1254)

# Setting the graph-level random seed. tf.set\_random\_seed(89)

from keras import backend as K

session\_conf = tf.ConfigProto(intra\_op\_parallelism\_threads=1,inter\_op\_parallelism\_threads=

#Force Tensorflow to use a single thread

sess = tf.Session(graph=tf.get\_default\_graph(), config=session\_conf) K.set\_session(sess)

import glob

from sklearn.utils import shuffle

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

from keras.models import Sequential from keras import models

from keras import layers from keras import optimizers

from keras.applications import VGG16

from tensorflow.keras.preprocessing.image import ImageDataGenerator from keras.layers import Dense, Conv2D, Flatten, MaxPooling2D, Dropout from sklearn.utils import shuffle

from PIL import Image

from google.colab.patches import cv2\_imshow import matplotlib.pyplot as plt

import glob

def gen\_image(img): plt.imshow(img) return plt

#from sklearn.cross\_validation import train\_test\_split

from google.colab import drive drive.mount('/content/drive')

PATH = os.getcwd()

from google.colab import drive drive.mount('/content/drive')

train\_dir = '/content/drive/My Drive/Colab Notebooks/cats\_dogs\_horse\_humans/train' validation\_dir = '/content/drive/My Drive/Colab Notebooks/cats\_dogs\_horse\_humans/validatio

#create model

model = Sequential() #add model layers

model.add(Conv2D(128, kernel\_size=3,strides=1, activation='relu',input\_shape=(256,256,3))) # 64 are the number of filters, kernel size is the size of the filters example 3\*3

model.add(Conv2D(128, kernel\_size=3,strides=1, activation='relu')) model.add(MaxPooling2D(pool\_size=(3,3)))

model.add(Conv2D(256, kernel\_size=3,strides=1, activation='relu')) model.add(Conv2D(256, kernel\_size=3,strides=1, activation='relu')) model.add(Flatten())

model.add(Dense(64, activation='relu')) model.add(Dropout(0.5)) model.add(Dense(64, activation='relu')) model.add(Dropout(0.5)) model.add(Dense(4, activation='softmax'))

image\_size=256

train\_datagen = ImageDataGenerator( rescale=1./255, rotation\_range=20, width\_shift\_range=0.2, height\_shift\_range=0.2, horizontal\_flip=True, fill\_mode='nearest')

validation\_datagen = ImageDataGenerator(rescale=1./255)

# Change the batchsize according to your system RAM train\_batchsize = 50

val\_batchsize = 10

train\_generator = train\_datagen.flow\_from\_directory( train\_dir,

target\_size=(image\_size, image\_size), batch\_size=train\_batchsize, class\_mode='categorical')

validation\_generator = validation\_datagen.flow\_from\_directory( validation\_dir,

target\_size=(image\_size, image\_size), batch\_size=val\_batchsize, class\_mode='categorical', shuffle=False)

# Compile the model model.compile(loss='categorical\_crossentropy',

optimizer=optimizers.RMSprop(lr=1e-4), metrics=['acc'])

# Train the model

history = model.fit\_generator( train\_generator,

steps\_per\_epoch=train\_generator.samples/train\_generator.batch\_size , epochs=5,

validation\_data=validation\_generator, validation\_steps=validation\_generator.samples/validation\_generator.batch\_size, verbose=1)

'''#TESTING

test\_dir='/content/drive/My Drive/images' test\_datagen = ImageDataGenerator(rescale=1./255)

test\_generator = test\_datagen.flow\_from\_directory( test\_dir, target\_size=(256, 256),batch\_ test\_generator=test\_generator.reshape(10,256,256,3)

ytested = model.predict\_classes(test\_generator) labels=['human','horse','dogs','cats']

for i in range(len(ytested)):

#print("The Predicted Testing image is =%s verify below" ,labels[ytested[i]]) print(np.argmax(ytested[i]))

gen\_image(test\_generator[i]).show() '''

#TESTING

data\_path = '/content/drive/My Drive/Colab Notebooks/cats\_dogs\_horse\_humans/test/all\_data' img\_data\_list=[]

img\_list = glob.glob(data\_path+'/\*.jpg') for img in img\_list:

input\_img=cv2.imread(img,1 ) input\_img\_resize=cv2.resize(input\_img,(256,256)) img\_data\_list.append(input\_img\_resize)

img\_data = np.array(img\_data\_list) img\_data = img\_data.astype('float32') x\_test = shuffle(img\_data, random\_state=2) x\_test=x\_test/255

#Nv\_test=x\_test.shape[0]

x\_test = x\_test.reshape(40,256,256,3) label=['humans','horse','dogs','cats'] ytested = model.predict\_classes(x\_test) for i in range(40):

print("The Predicted Testing image is =%s verify below" %label[ytested[i]]) gen\_image(x\_test[i]).show() # printing image vs the predicted image below





he default version of TensorFlow in Colab will soon switch to TensorFlow 2.x.

T

We recommend you [upgrade](https://www.tensorflow.org/guide/migrate) now or ensure your notebook will continue to use TensorFlow 1.x via the %ten

Using TensorFlow backend.

Go to this URL in a browser: [https://accounts.google.com/o/oauth2/auth?client\_id=94731](https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_type=code&scope=email%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.readonly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly)

Enter your authorization code:

··········

Mounted at /content/drive

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mo WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflo

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflo WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflo WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflo

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflo Instructions for updating:

Please use `rate` instead of `keep\_prob`. Rate should be set to `rate = 1 - keep\_prob` Found 608 images belonging to 4 classes.

Found 160 images belonging to 4 classes.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optimizers.py:793

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflo

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow\_core/python/ Instructions for updating:

Use tf.where in 2.0, which has the same broadcast rule as np.where WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflo

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflo Epoch 1/5

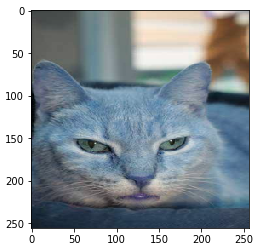
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflo

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflo WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflo WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflo

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 13/12 | [================================] | - | 221s 17s/step | - loss: 1.7662 | - acc: 0.2818 |
| Epoch | 2/5 |  |  |  |  |
| 13/12 | [================================] | - | 7s 546ms/step | - loss: 1.3769 | - acc: 0.2716 |
| Epoch | 3/5 |  |  |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 13/12 | [================================] | - 13s | 1s/step | - loss: | 1.3905 | - acc: | 0.2853 | - |
| Epoch | 4/5 |  |  |  |  |  |  |  |
| 13/12 | [================================] | - 13s | 1s/step | - loss: | 1.3920 | - acc: | 0.2856 | - |
| Epoch | 5/5 |  |  |  |  |  |  |  |
| 13/12 | [================================] | - 13s | 1s/step | - loss: | 1.3838 | - acc: | 0.2796 | - |

The Predicted Testing image is =humans verify below



The Predicted Testing image is =humans verify below

The Predicted Testing image is =humans verify below

The Predicted Testing image is =horse verify below

The Predicted Testing image is =humans verify below



The Predicted Testing image is =humans verify below

The Predicted Testing image is =cats verify below

The Predicted Testing image is =humans verify below

The Predicted Testing image is =humans verify below



The Predicted Testing image is =humans verify below

The Predicted Testing image is =humans verify below

The Predicted Testing image is =humans verify below

The Predicted Testing image is =humans verify below



The Predicted Testing image is =cats verify below

The Predicted Testing image is =humans verify below

The Predicted Testing image is =cats verify below

The Predicted Testing image is =humans verify below



The Predicted Testing image is =humans verify below

The Predicted Testing image is =humans verify below

The Predicted Testing image is =humans verify below

he Predicted Testing image is =humans verify below



The Predicted Testing image is =cats verify below

The Predicted Testing image is =cats verify below

The Predicted Testing image is =humans verify below



The Predicted Testing image is =humans verify below

The Predicted Testing image is =humans verify below

The Predicted Testing image is =humans verify below

The Predicted Testing image is =cats verify below



The Predicted Testing image is =humans verify below

The Predicted Testing image is =humans verify below

The Predicted Testing image is =humans verify below

The Predicted Testing image is =humans verify below



The Predicted Testing image is =humans verify below

The Predicted Testing image is =humans verify below

The Predicted Testing image is =cats verify below

The Predicted Testing image is =humans verify below



The Predicted Testing image is =humans verify below

The Predicted Testing image is =humans verify below

The Predicted Testing image is =humans verify below

The Predicted Testing image is =humans verify below



from keras import models from keras import layers from keras import optimizers

from keras.applications import VGG16

from tensorflow.keras.preprocessing.image import ImageDataGenerator from keras.layers import Dense, Conv2D, Flatten, MaxPooling2D, Dropout from sklearn.utils import shuffle

from PIL import Image

from google.colab.patches import cv2\_imshow import matplotlib.pyplot as plt

import glob import numpy as np import os,cv2

#Load the VGG model

def gen\_image(img): plt.imshow(img) return plt

from google.colab import drive drive.mount('/content/drive')

train\_dir = '/content/drive/My Drive/Colab Notebooks/cats\_dogs\_horse\_humans/train' validation\_dir = '/content/drive/My Drive/Colab Notebooks/cats\_dogs\_horse\_humans/validatio

img\_width, img\_height = 256, 256 image\_size = img\_height

vgg\_conv = VGG16(weights='imagenet', include\_top=False, input\_shape=(img\_width, img\_height

# Freeze the layers except the last 4 layers for layer in vgg\_conv.layers[:-2]:

layer.trainable = False

# Check the trainable status of the individual layers for layer in vgg\_conv.layers:

print(layer, layer.trainable)

# Create the model

model = models.Sequential()

# Add the vgg convolutional base model model.add(vgg\_conv)

# Add new layers model.add(Flatten())

model.add(Dense(64, activation='relu')) model.add(Dropout(0.5)) model.add(Dense(64, activation='relu')) model.add(Dropout(0.5)) model.add(Dense(4, activation='softmax'))

# Show a summary of the model. Check the number of trainable parameters model.summary()

train\_datagen = ImageDataGenerator( rescale=1./255, rotation\_range=20, width\_shift\_range=0.2, height\_shift\_range=0.2, horizontal\_flip=True, fill\_mode='nearest')

validation\_datagen = ImageDataGenerator(rescale=1./255)

# Change the batchsize according to your system RAM train\_batchsize = 50

val\_batchsize = 10

train\_generator = train\_datagen.flow\_from\_directory( train\_dir,

target\_size=(image\_size, image\_size), batch\_size=train\_batchsize, class\_mode='categorical')

validation\_generator = validation\_datagen.flow\_from\_directory( validation\_dir,

target\_size=(image\_size, image\_size), batch\_size=val\_batchsize, class\_mode='categorical', shuffle=False)

# Compile the model model.compile(loss='categorical\_crossentropy',

optimizer=optimizers.RMSprop(lr=1e-4), metrics=['acc'])

# Train the model

history = model.fit\_generator( train\_generator,

steps\_per\_epoch=train\_generator.samples/train\_generator.batch\_size , epochs=5,

validation\_data=validation\_generator, validation\_steps=validation\_generator.samples/validation\_generator.batch\_size, verbose=1)

acc = history.history['acc']

val\_acc = history.history['val\_acc'] loss = history.history['loss'] val\_loss = history.history['val\_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, 'b', label='Training acc')

plt.plot(epochs, val\_acc, 'r', label='Validation acc') plt.title('Training and validation accuracy') plt.legend()

plt.figure()

plt.plot(epochs, loss, 'b', label='Training loss') plt.plot(epochs, val\_loss, 'r', label='Validation loss') plt.title('Training and validation loss')

plt.legend() plt.show()

###########################################333

# insert the testing code #TESTING

data\_path = '/content/drive/My Drive/Colab Notebooks/cats\_dogs\_horse\_humans/test/all\_data' img\_data\_list=[]

img\_list = glob.glob(data\_path+'/\*.jpg') for img in img\_list:

input\_img=cv2.imread(img,1 ) input\_img\_resize=cv2.resize(input\_img,(256,256)) img\_data\_list.append(input\_img\_resize)

img\_data = np.array(img\_data\_list) img\_data = img\_data.astype('float32') x\_test = shuffle(img\_data, random\_state=2) x\_test=x\_test/255

#Nv\_test=x\_test.shape[0]

x\_test = x\_test.reshape(40,256,256,3) label=['humans','horse','dogs','cats'] ytested = model.predict\_classes(x\_test) for i in range(40):

print("The Predicted Testing image is =%s verify below" %label[ytested[i]]) gen\_image(x\_test[i]).show() # printing image vs the predicted image below



Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m

<keras.engine.input\_layer.InputLayer object at 0x7ff970bb54e0> False

<keras.layers.convolutional.Conv2D object at 0x7ff970bb54a8> False

<keras.layers.convolutional.Conv2D object at 0x7ff970bb5780> False

<keras.layers.pooling.MaxPooling2D object at 0x7ff970bbc5f8> False

<keras.layers.convolutional.Conv2D object at 0x7ff970bafd68> False

<keras.layers.convolutional.Conv2D object at 0x7ff970b40e10> False

<keras.layers.pooling.MaxPooling2D object at 0x7ff970b44cf8> False

<keras.layers.convolutional.Conv2D object at 0x7ff970b4fbe0> False

<keras.layers.convolutional.Conv2D object at 0x7ff970b54438> False

<keras.layers.convolutional.Conv2D object at 0x7ff970b5b198> False

<keras.layers.pooling.MaxPooling2D object at 0x7ff970b60c50> False

<keras.layers.convolutional.Conv2D object at 0x7ff970b6db00> False

<keras.layers.convolutional.Conv2D object at 0x7ff970b75358> False

<keras.layers.convolutional.Conv2D object at 0x7ff970b7b278> False

<keras.layers.pooling.MaxPooling2D object at 0x7ff970b7ab70> False

<keras.layers.convolutional.Conv2D object at 0x7ff970b0ea20> False

<keras.layers.convolutional.Conv2D object at 0x7ff970b172b0> False

<keras.layers.convolutional.Conv2D object at 0x7ff970b1e5c0> True

<keras.layers.pooling.MaxPooling2D object at 0x7ff970b22a20> True Model: "sequential\_3"

Layer (type) Output Shape Param #

=================================================================

|  |  |  |  |
| --- | --- | --- | --- |
| vgg16 (Model) | (None, | 8, 8, 512) | 14714688 |
| flatten\_3 (Flatten) | (None, | 32768) | 0 |
| dense\_7 (Dense) | (None, | 64) | 2097216 |
| dropout\_5 (Dropout) | (None, | 64) | 0 |
| dense\_8 (Dense) | (None, | 64) | 4160 |
| dropout\_6 (Dropout) | (None, | 64) | 0 |
| dense\_9 (Dense) | (None, | 4) | 260 |

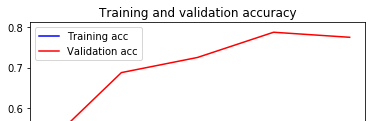
=================================================================

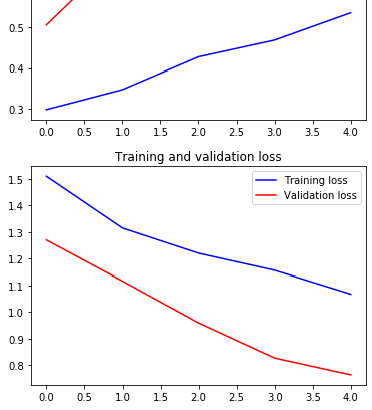
Total params: 16,816,324

Trainable params: 4,461,444

Non-trainable params: 12,354,880

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Found Found | 608 images belonging to 4 classes.  160 images belonging to 4 classes. |  | | | | | |
| Epoch | 1/5 |
| 13/12 | [================================] | - 12s | 923ms/step | - loss: | 1.5013 | - acc: | 0.287 |
| Epoch | 2/5 |  |  |  |  |  |  |
| 13/12 | [================================] | - 10s | 794ms/step | - loss: | 1.3094 | - acc: | 0.356 |
| Epoch | 3/5 |  |  |  |  |  |  |
| 13/12 | [================================] | - 11s | 815ms/step | - loss: | 1.2177 | - acc: | 0.432 |
| Epoch | 4/5 |  |  |  |  |  |  |
| 13/12 | [================================] | - 10s | 804ms/step | - loss: | 1.1500 | - acc: | 0.470 |
| Epoch | 5/5 |  |  |  |  |  |  |
| 13/12 | [================================] | - 11s | 811ms/step | - loss: | 1.0769 | - acc: | 0.534 |





The Predicted Testing image is =cats verify below



The Predicted Testing image is =horse verify below

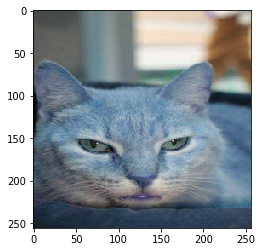


The Predicted Testing image is =dogs verify below





The Predicted Testing image is =horse verify below



The Predicted Testing image is =horse verify below



The Predicted Testing image is =cats verify below



The Predicted Testing image is =dogs verify below





The Predicted Testing image is =humans verify below



The Predicted Testing image is =horse verify below

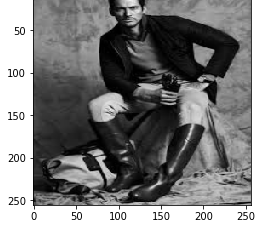


The Predicted Testing image is =humans verify below



The Predicted Testing image is =humans verify below





The Predicted Testing image is =dogs verify below



The Predicted Testing image is =humans verify below



The Predicted Testing image is =horse verify below



The Predicted Testing image is =dogs verify below



The Predicted Testing image is =horse verify below



The Predicted Testing image is =humans verify below



The Predicted Testing image is =humans verify below





The Predicted Testing image is =cats verify below



The Predicted Testing image is =horse verify below



The Predicted Testing image is =humans verify below



The Predicted Testing image is =horse verify below





The Predicted Testing image is =horse verify below



The Predicted Testing image is =horse verify below



The Predicted Testing image is =horse verify below



The Predicted Testing image is =humans verify below





The Predicted Testing image is =horse verify below



The Predicted Testing image is =cats verify below



The Predicted Testing image is =cats verify below



The Predicted Testing image is =cats verify below





The Predicted Testing image is =humans verify below



The Predicted Testing image is =horse verify below



The Predicted Testing image is =humans verify below



The Predicted Testing image is =horse verify below





The Predicted Testing image is =horse verify below



The Predicted Testing image is =horse verify below



The Predicted Testing image is =horse verify below



The Predicted Testing image is =horse verify below





The Predicted Testing image is =horse verify below



The Predicted Testing image is =horse verify below



**EXPLANATION:**

**TASK 1:**

* At the beginning we are including few libraries and lines of code to generate reproduceable result, (i,e) to get same accuracy no matter how many times we run the code.
* The environ[‘PYTHONHASHSEED’] is used to maintain the precision which helps us to get same results.
* This seed is set by means of numpy a library which generates random numbers at python level.
* We use the tensor flow for all the functions in this program.
* We import all the files necessary for the code to run.
* We give separate data paths for training, testing and validation.
* Now add the model layers to create the CNN, this time we are using 4 convolution layers, 1 pooling layer, 1 flatten, 2 Dropout and 3 dense layers.
* With train and validation batch size we generate the train and validation data.
* In previous assignments we split the training data into training and validation, but in this assignment, we use separate data.
* Then we compile and train the model with the necessary arguments.
* Then we test the data by creating a data list, reshaping it, normalizing it and predicting the results with the image.

**TASK 2:**

* For task 2 we include all the libraries and the tensor flow functions along with some files which is imported necessary for the code.
* In this task for transfer learning we use VGG 16 model layer, so we need to include all the libraries for the VGG 16 model.
* We then add the model layers and we also include some code for freezing the layer as long it will be false once it’s true the flow of code will be normal.
* Then we add the code to analyses the data and represent it in the form of graph.
* We give separate data paths for training, testing and validation.
* Again, with train and validation batch size we generate the train and validation data.
* Then we test the data by creating a data list, reshaping it, normalizing it and predicting the results with the image.

**CONCLUSION:**

In the first task we are training and validating the data by generating the weights and trying to predict the data. In previous assignments we are training and validating by splitting the data, but in this assignment, we are giving separate directory for these data. When we are testing and printing the result for task 1 we notice that the accuracy is very poor. This is because we are generating the weights. To overcome this in task 2 we are using VGG 16 model layer, which has in built layers that can generate pre trained weights. In task 2 when we test the data, we get a better accuracy than task 1 but not very good accuracy. In this assignment the images are not sectioned into different folders, they are in a single folder. There is a possibility for mismatch. So, we use VGG 16 for better performance.