**Practical Assignment**

**Objective: - Image Classification with Stanford Dogs Dataset**

The Stanford Dogs dataset contains images of 120 breeds of dogs from around the world. This dataset has been built using images and annotation from ImageNet for the task of fine-grained image categorization. Contents of this dataset:

* **Number of categories: 120**
* **Number of images: 20,580**
* **Annotations: Class labels, Bounding boxes**

**Dataset Link: -**

Images Link :-<http://vision.stanford.edu/aditya86/ImageNetDogs/images.tar>

Annotations Link :-http://vision.stanford.edu/aditya86/ImageNetDogs/annotation.tar

**Task: -** Create a Web Application using Flask. Use the end user should be able to upload an image and get results with the prediction score.

**Deployment: -** Any Free Platform(Try to look out for free options.)

**Assignment Submission: -** Only submit the hosted app link.

#Import data from drive

from google.colab import drive

import scipy.io

import numpy as np

import pandas as pd

import os

import seaborn as sns

from pandas import DataFrame, Series

import matplotlib.pyplot as plt

import imageio

drive.mount('/content/drive/')

path= '/content/drive/My Drive/CMT307 Applied Machine Learning/Coursework2/lists/file\_list.mat'

path1= '/content/drive/My Drive/CMT307 Applied Machine Learning/Coursework2/lists/file\_list.csv'

path\_train= '/content/drive/My Drive/CMT307 Applied Machine Learning/Coursework2/lists/train\_list.mat'

path\_test= '/content/drive/My Drive/CMT307 Applied Machine Learning/Coursework2/lists/test\_list.mat'

# Create dataframe for full list, the given test and the given training set

#Full

data = scipy.io.loadmat(path)

df = pd.DataFrame()

df1 = pd.DataFrame()

df2 = pd.DataFrame()

list1 = []

for i in data:

if '\_\_' not in i and 'readme' not in i:

data\_array = data[i]

if df.empty:

df = pd.DataFrame(data\_array)

elif df1.empty:

df1 = pd.DataFrame(data\_array)

else:

df2 = pd.DataFrame(data\_array)

frames = [df, df2]

full\_list = pd.concat(frames, axis=1)

#train\_list = df

full\_list.columns = ['File', 'Index']

full\_list['File'] = full\_list['File'].astype(str)

full\_list['Breed'] = full\_list['File'].str.split('-').str[1]

full\_list['Breed'] = full\_list['Breed'].str.split('/').str[0]

full\_list['File'] = full\_list['File'].str.split('[').str[1]

full\_list['File'] = full\_list['File'].str.split(']').str[0]

full\_list['File'] = full\_list['File'].str.strip("'")

# Training

data = scipy.io.loadmat(path\_train)

df = pd.DataFrame()

df1 = pd.DataFrame()

df2 = pd.DataFrame()

list1 = []

for i in data:

if '\_\_' not in i and 'readme' not in i:

data\_array = data[i]

if df.empty:

df = pd.DataFrame(data\_array)

list1.append(str(data\_array).split("-"))

elif df1.empty:

df1 = pd.DataFrame(data\_array)

else:

df2 = pd.DataFrame(data\_array)

frames = [df, df2]

train\_list = pd.concat(frames, axis=1)

#train\_list = df

train\_list.columns = ['File', 'Index']

train\_list['File'] = train\_list['File'].astype(str)

train\_list['Breed'] = train\_list['File'].str.split('-').str[1]

train\_list['Breed'] = train\_list['Breed'].str.split('/').str[0]

train\_list['File'] = train\_list['File'].str.split('[').str[1]

train\_list['File'] = train\_list['File'].str.split(']').str[0]

train\_list['File'] = train\_list['File'].str.strip("'")

# Test

data = scipy.io.loadmat(path\_test)

df = pd.DataFrame()

df1 = pd.DataFrame()

df2 = pd.DataFrame()

for i in data:

if '\_\_' not in i and 'readme' not in i:

data\_array = data[i]

if df.empty:

df = pd.DataFrame(data\_array)

elif df1.empty:

df1 = pd.DataFrame(data\_array)

else:

df2 = pd.DataFrame(data\_array)

frames = [df, df2]

test\_list = pd.concat(frames, axis=1)

test\_list.columns = ['File', 'Index']

test\_list['File'] = test\_list['File'].astype(str)

test\_list['Breed'] = test\_list['File'].str.split('-').str[1]

test\_list['Breed'] = test\_list['Breed'].str.split('/').str[0]

test\_list['File'] = test\_list['File'].str.split('[').str[1]

test\_list['File'] = test\_list['File'].str.split(']').str[0]

test\_list['File'] = test\_list['File'].str.strip("'")

print('Total')

print('There are ' + str(len(full\_list.groupby('Breed'))) + ' breeds')

print('There are ' + str(sum(full\_list['Breed'].value\_counts())) + ' Images')

print('Train')

print('There are ' + str(len(train\_list.groupby('Breed'))) + ' breeds')

print('There are ' + str(sum(train\_list['Breed'].value\_counts())) + ' Images')

print('Test')

print('There are ' + str(len(test\_list.groupby('Breed'))) + ' breeds')

print('There are ' + str(sum(test\_list['Breed'].value\_counts())) + ' Images')

# Create data frame of the number of images per breed

path = '/content/drive/My Drive/CMT307 Applied Machine Learning/Coursework2/Images/'

mn = 20

d = pd.DataFrame()

folders = ([name for name in os.listdir(path)

if os.path.isdir(os.path.join(path, name)) and name.startswith("")]) # get all directories

for folder in folders:

contents = os.listdir(os.path.join(path,folder)) # get list of contents

folder1 =folder.split("-",1)[1]

if len(contents) > mn: # if greater than the limit, print folder and number of contents

temp = pd.DataFrame({'Dog': [folder1], 'Count': [len(contents)], })

d = pd.concat([d, temp])

print(folder1,len(contents))

## Plot a barplot for the dog breeds

fig, ax = plt.subplots()

fig.set\_size\_inches(15, 9)

sns.set\_style("whitegrid")

d1 = d.sort\_values('Count',ascending=False)

ax = sns.barplot(x = d1.Dog, y = d1.Count, data = d1)

ax.set\_xticklabels(ax.get\_xticklabels(), rotation = 90, fontsize = 8)

ax.set(xlabel='Dog Breed', ylabel='Count')

ax.set\_title('Distribution of Dog breeds')

plt.show()

#See breeds with the most and least amount of pictures

d1.head(5)

d1.tail(5)

# Descriptive analysis of the number of breeds per image

d1.describe()

## Add the average/max/min for image sizes and how this will affect us

path = '/content/drive/My Drive/CMT307 Applied Machine Learning/Coursework2/Images/'

mn = 20

ImageDetails = pd.DataFrame()

folders = ([name for name in os.listdir(path)

if os.path.isdir(os.path.join(path, name)) and name.startswith("")]) # get all directories

for folder in folders:

contents = os.listdir(os.path.join(path,folder)) # get list of contents

folder1 =folder.split("-",1)[1]

if len(contents) > mn: # if greater than the limit, print folder and number of contents

#temp = pd.DataFrame({'Dog': [folder1], 'Count': [len(contents)], })

for image in contents:

pic = imageio.imread(path + folder+ "/" + image)

temp = pd.DataFrame({ 'ID': str(image), 'Breed':folder1 ,'Width': pic.shape[1], 'Height': pic.shape[0],}, index=[0])

ImageDetails = pd.concat([ImageDetails, temp])

#print("Folder:" + folder1 + " Complete. ")

#Summary of image height and width for all of the photos

print('Summary of image height and width for all of the photos')

ImageDetails.describe()

# See top 10 by height

ImageDetailsHeight = ImageDetails.sort\_values('Height',ascending=False)

ImageDetailsHeight.head(10)

# See top 10 by width

ImageDetailsWidth = ImageDetails.sort\_values('Width',ascending=False)

ImageDetailsWidth.head(10)

# See distribution of height and width

box\_plot\_data=[ImageDetails['Height'], ImageDetails['Width']]

box=plt.boxplot(box\_plot\_data,vert=0,patch\_artist=True,labels=['Height','Width'],

)

colors = ['cyan', 'lightblue']

for patch, color in zip(box['boxes'], colors):

patch.set\_facecolor(color)

plt.show()