5/8/2023

# CS-351L-Artificial Intelligence Lab

2020244 - Mohsin Zia

INSTRUCTOR: PROFESSOR MUNEEB BAIG

# *Cats And Dogs Classification using CNN*

- First, I've imported all the required libraries needed.

```
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from sklearn.model_selection import train_test_split
import cv2
import glob
import random
import tensorflow as tf
import tensorflow_hub as hub
import shutil
```

- Extracting the zip file which contains the dataset of cats and dogs.

```
from zipfile import ZipFile

dataset = './archive.zip'

with ZipFile(dataset, 'r') as zip:
    zip.extractall()
    print('Complete dataset is extracted')
```

- Saving images from the train folder.

```
import os
# counting the number of files in train folder
path, dirs, files = next(os.walk('./train'))
file_count = len(files)
print('Number of images: ', file_count)
```

➢ Counting the images of Cats and Dogs.

```
file_names = os.listdir('./train/')

dog_count = 0
cat_count = 0

for img_file in file_names:

    name = img_file[0:3]

    if name == 'dog':
        dog_count += 1

    else:
        cat_count += 1

print('Number of dog images =', dog_count)
print('Number of cat images =', cat_count)
```

➢ Converting the images to the size that our CNN architecture supports, and then saving them to a new folder.

```
original_folder = './train/'
resized_folder = './resizedImages/'

for i in range(697):

    filename = os.listdir(original_folder)[i]
    img_path = original_folder+filename

    img = Image.open(img_path)
    img = img.resize((224, 224))
    img = img.convert('RGB')

    newImgPath = resized_folder+filename
    img.save(newImgPath)
```

➢ Appending labels for each image to a list. 1 if image contains dog else 0.

```
filenames = os.listdir('./resizedImages/')


labels = []

for i in range(697):

    file_name = filenames[i]
    label = file_name[0:3]

    if label == 'dog':
        labels.append(1)

    else:
        labels.append(0)
```

➢ Counting the length of each label from the list.

```
values, counts = np.unique(labels, return_counts=True)
print(values)
print(counts)
```

➢ Type casting the images to 2D Matrix, single pixel values.

```
image_directory = './resizedImages/'
image_extension = ['png', 'jpg']

files = []

[files.extend(glob.glob(image_directory + '*.' + e)) for e in image_extension]
dog_cat_images = np.asarray([cv2.imread(file) for file in files])
```

➢ Splitting the data set into train and test set. Test set is 20% of the training data.

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_s
tate
                                                    =2)
```

➢ Normalizing the data, scaling between 0 and 1.

```
X_train_scaled = X_train/255

X_test_scaled = X_test/255
```

➢ Importing the CNN architecture of MobileNetV2 from tensorflow. MobileNet V2 is a family of neural network architectures for efficient on-device image classification and related tasks.

```
mobilenet_model = 'https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4'

pretrained_model = hub.KerasLayer(mobilenet_model, input_shape=(224,224,3), trainable=False)
```

➢ Assigning the number of classes and creating the CNN model using pretrained model.

```
num_of_classes = 2

model = tf.keras.Sequential([

    pretrained_model,
    tf.keras.layers.Dense(num_of_classes)

])

model.summary()
```

| Layer (type) | Output Shape | Param # |
|---|---|---|
| keras_layer (KerasLayer) | (None, 1280) | 2257984 |
| dense (Dense) | (None, 2) | 2562 |

```
================================================================================
Total params: 2,260,546
Trainable params: 2,562
Non-trainable params: 2,257,984
================================================================================
```

- Compiling the model.

```
model.compile(
    optimizer = 'adam',
    loss = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
    metrics = ['acc']
)
```

- Training the data on 5 iterations.

```
model.fit(X_train_scaled, Y_train, epochs=5)
```

- Evaluating the model on test set and got **95.7%** of accuracy.

```
score, acc = model.evaluate(X_test_scaled, Y_test)
print('Test Accuracy =', acc*100)
# Test Accuracy = 95.71428298950195
```

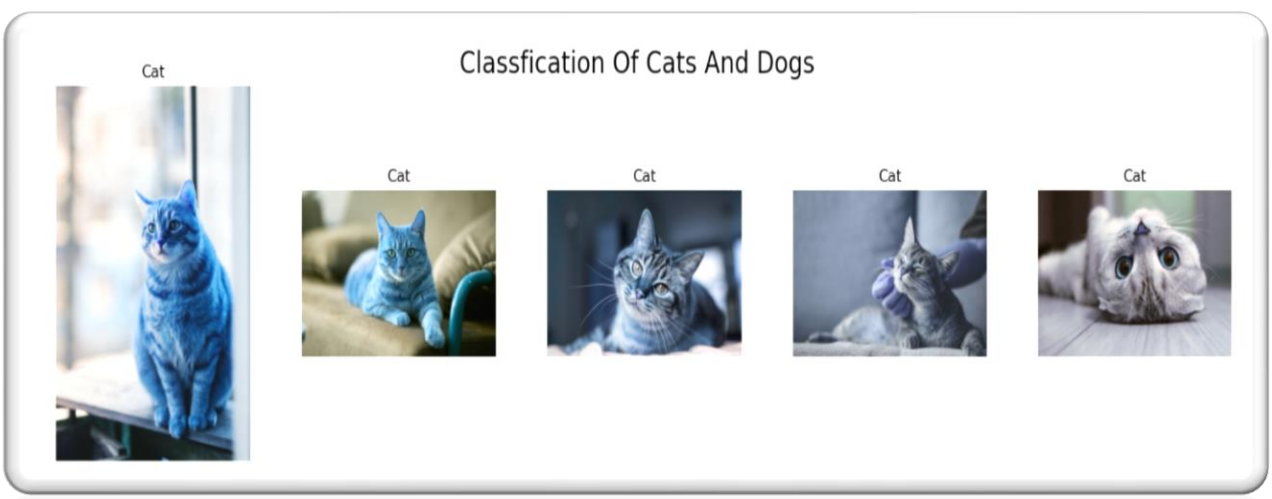
- Function that classifies the image as a cat or a dog. It reads the image from the given path and then displays it. Then the image is reshaped to (1x224x224x3) and then passed to the model for prediction. Then the index with max probability is printed along with it's class.

```python
def classifyImage(path):

    input_image = cv2.imread(path)
    plt.imshow(input_image)
    plt.show()

    input_image_resize = cv2.resize(input_image, (224,224))

    input_image_scaled = input_image_resize/255

    image_reshaped = np.reshape(input_image_scaled, [1,224,224,3])

    input_prediction = model.predict(image_reshaped)

    print(input_prediction)

    input_pred_label = np.argmax(input_prediction)

    print(f"Label Predicted: {input_pred_label}")

    if input_pred_label == 0:
        print('The image represents a Cat')

    else:
        print('The image represents a Dog')

```

➢ Function that does same as the above function. It just plots the data on subplots.

```python
def plotData(imagesPath: list):

    classes = ['Cat', 'Dog']
    LabelsList = np.empty(shape=(len(imagesPath)))

    for i in range(len(imagesPath)):
        input_image = cv2.imread(imagesPath[i])
        input_image_resize = cv2.resize(input_image, (224,224))
        input_image_scaled = input_image_resize/255
        image_reshaped = np.reshape(input_image_scaled, [1,224,224,3])
        input_prediction = model.predict(image_reshaped)
        input_pred_label = np.argmax(input_prediction)
        LabelsList[i] = input_pred_label

    fig, axes = plt.subplots(nrows= 1, ncols=len(imagesPath), figsize=(15, 10))

    fig.suptitle('Classfication Of Cats And Dogs', fontsize = 20)
    fig.tight_layout()
    fig.subplots_adjust(top=1.5)

    for i in range(len(imagesPath)):
        input_image = cv2.imread(imagesPath[i])
        axes[i].imshow(input_image)
        axes[i].axis('off')
        axes[i].title.set_text(classes[int(LabelsList[i])])
```

➢ Output for cat's classification on unseen data for the model.



➢ Output for dog's classification on unseen data for the model.


➢ **CONCLUSION:**

During the completion of this assignment, I have gained profound insights into the remarkable capabilities offered by the CNN architecture known as "MobilenetV2," which is made available through the tensorflow hub. Developed by Google, MobileNet-v2 stands as a 53-layer deep convolutional neural network primarily tailored for training classifiers in the domain of computer vision. The network's utilization of depthwise convolutions facilitates a notable reduction in parameter count when compared to alternative models, consequently culminating in a lightweight deep neural network. By effectively accentuating salient image features, this approach renders classification tasks more accessible for the employed CNN model.

---

*THE END.*

---