Image Classification Using VGG - Cat or Dog Kaggle Competition

Import libraries

```
1 import pandas as pd
2 import IPython.display as display
3 import tensorflow as tf
4 from tensorflow.keras import layers
5 import numpy as np
6 import os,random
7 import matplotlib.pyplot as plt
8 print(tf.__version__)
```

```
2.4.0
```

Prepare Data

The dataset has been split into two parts - training set which contains 25,000 images of dags and cats. In this project, the model is trained on the training set and tested on test set which includes 12,500 images. Since uploading the dataset to google colab is time-consuming. Yura Istomin gives me an idea of uploading the dataset to github and cloning the repository, which works perfectly to use the dataset in Google colab.

Clonning

```
1 ! git clone https://github.com/patrick013/Image-Classification-CNN-and-VGG.git
```

```
Cloning into 'Image-Classification-CNN-and-VGG'...
remote: Enumerating objects: 96, done.
remote: Counting objects: 100% (96/96), done.
remote: Compressing objects: 100% (88/88), done.
remote: Total 37510 (delta 43), reused 36 (delta 7), pack-reused 37414
Receiving objects: 100% (37510/37510), 812.51 MiB | 45.53 MiB/s, done.
Resolving deltas: 100% (48/48), done.
Checking out files: 100% (50016/50016), done.
```

Labelling of data

```
IMAGE_HEIGHT=128
IMAGE_WIDTH=128
BATCH_SIZE=64
def get_pathframe(path):
  '''
  Get all the images paths and its corresponding labels
  Store them in pandas dataframe
  '''
  filenames = os.listdir(path)
  categories = []
  paths=[]
  for filename in filenames:
    paths.append(path+filename)
    category = filename.split('.')[0]
    if category == 'dog':
      categories.append(1)
    else:
      categories.append(0)

  df= pd.DataFrame({
      'filename': filenames,
      'category': categories,
      'paths':paths
  })
  return df
```

Dataframe

```
df=get_pathframe("Image-Classification-CNN-and-VGG/dataset/dataset/")
df.tail(5)
```

	filename	category	paths
24995	cat.8309.jpg	0	Image-Classification-CNN-and-VGG/dataset/datas...
24996	cat.4669.jpg	0	Image-Classification-CNN-and-VGG/dataset/datas...
24997	cat.7917.jpg	0	Image-Classification-CNN-and-VGG/dataset/datas...
24998	cat.5124.jpg	0	Image-Classification-CNN-and-VGG/dataset/datas...
24999	cat.10920.jpg	0	Image-Classification-CNN-and-VGG/dataset/datas...

Images loading

```
def load_and_preprocess_image(path):
  '''
  Load each image and resize it to desired shape
  '''
  image = tf.io.read_file(path)
  image = tf.image.decode_jpeg(image, channels=3)
  image = tf.image.resize(image, [IMAGE_WIDTH, IMAGE_HEIGHT])
  image /= 255.0  # normalize to [0,1] range
  return image

def convert_to_tensor(df):
  '''
  Convert each data and labels to tensor
  '''
  path_ds = tf.data.Dataset.from_tensor_slices(df['paths'])
  image_ds = path_ds.map(load_and_preprocess_image)
  # onehot_label=tf.one_hot(tf.cast(df['category'], tf.int64),2) if using softmax
  onehot_label=tf.cast(df['category'], tf.int64)
  label_ds = tf.data.Dataset.from_tensor_slices(onehot_label)

  return image_ds,label_ds
```

Train and test split

```
X,Y=convert_to_tensor(df)
print("Shape of X in data:", X)
print("Shape of Y in data:", Y)
```

```
Shape of X in data: <DatasetV1Adapter shapes: (128, 128, 3), types: tf.float32>
Shape of Y in data: <DatasetV1Adapter shapes: (), types: tf.int64>
```

```
dataset=tf.data.Dataset.zip((X,Y)).shuffle(buffer_size=2000)
dataset_train=dataset.take(22500)
dataset_test=dataset.skip(22500)

dataset_train=dataset_train.batch(BATCH_SIZE, drop_remainder=True)
dataset_test=dataset_test.batch(BATCH_SIZE, drop_remainder=True)
dataset_train
```

```
<BatchDataset shapes: ((64, 128, 128, 3), (64,)), types: (tf.float32, tf.int64)>
```

Check images

```
def plotimages(imagesls):
  fig, axes = plt.subplots(1, 5, figsize=(20,20))
  axes = axes.flatten()
  for image,ax in zip(imagesls, axes):
    ax.imshow(image)
    ax.axis('off')

imagesls=[]
for n, image in enumerate(X.take(5)):
  imagesls.append(image)

plotimages(imagesls)
```

    

Task

1. Build your CNN architecture
2. Train CNN model
3. Test Model