Problem Statement

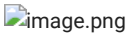
In this problem, you'll train a CNN model to classify whether images contain either a dog or a cat. This is easy for humans, dogs, and cats. Your computer will find it a bit more difficult.
The Dogs vs. Cats dataset is a standard computer vision dataset that involves classifying photos as either containing a dog or cat.

In this notebook, you will discover how to develop a convolutional neural network to classify photos of dogs and cats.

- How to load and prepare photos of dogs and cats for modeling.
- How to develop a convolutional neural network for photo classification from scratch and improve model performance.
- How to develop a model for photo classification using transfer learning.

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```




```
  File "<ipython-input-98-c8e4c43cd605>", line 1
    show(/content/drive/MyDrive/cat.8.jpg)
         ^
SyntaxError: invalid syntax
```

SEARCH STACK OVERFLOW

Importing Libraries

```
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow import keras
import os

import matplotlib.pyplot as plt
import matplotlib.image as img
import seaborn as sns

'''setting seed'''
seed = 0
np.random.seed(seed)
tf.random.set_seed(3)
```

Load and Extract data

```
import zipfile

zip_files = ['test1', 'train']

for zip_file in zip_files:
    with zipfile.ZipFile("/content/drive/MyDrive/dogs-vs-cats.zip".format(zip_file),"r") as z:
        z.extractall(".")
        print("{} unzipped".format(zip_file))
```

```
 test1 unzipped
 train unzipped
```

```
import csv


with open('/content/drive/MyDrive/dogs-vs-cats.zip', 'r') as f:
    reader = csv.reader(f)
```

```
TRAIN_DIR_PATH = '/content/drive/MyDrive/train'
file_names = os.listdir(TRAIN_DIR_PATH )
print('There are {} number of images in directory.'.format(len(file_names)))
```

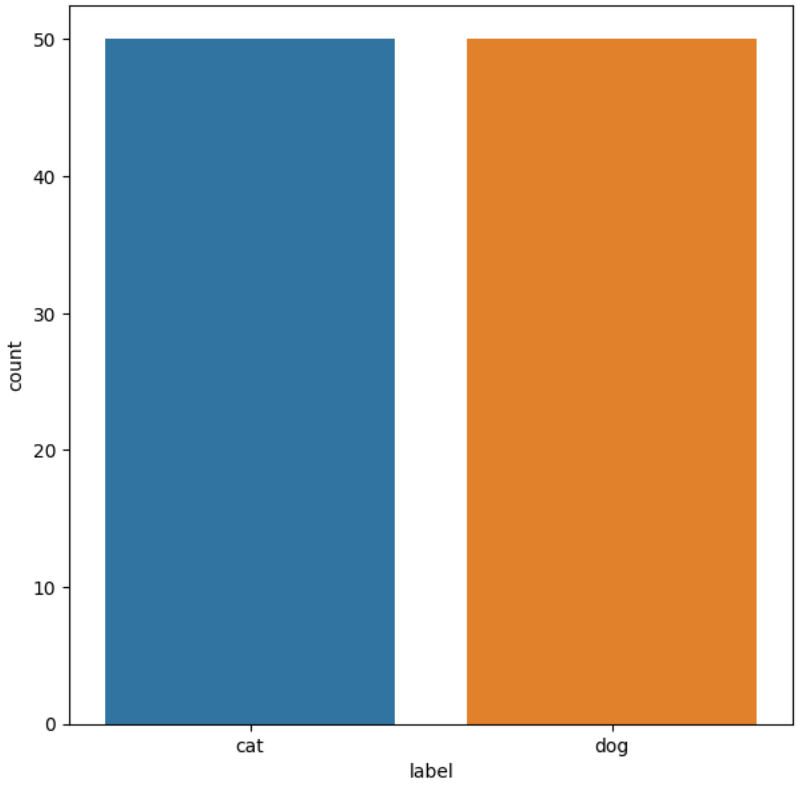
```
 There are 100 number of images in directory.
```

```
def to_dataframe(file_names):
    files, labels = list(), list()
    for file in file_names:
        files.append(file)
        labels.append(file[:3])
    df = pd.DataFrame({'filename':files, 'label':labels})
    return df

df = to_dataframe(file_names)
```

Analyze data

```
print('data set label distribution:\n',df['label'].value_counts())

plt.figure(figsize=(7,7))
sns.countplot(x = df['label'])
plt.show()
```

```
 data set label distribution:
  cat    50
 dog    50
 Name: label, dtype: int64
```




```
cat = [file for file in file_names if file[:3]=='cat']
dog = [file for file in file_names if file[:3]=='dog']
```

```
plt.figure(figsize=(5,3))
for i, c in enumerate(np.random.randint(0,len(cat),31), start=1):
    im = img.imread('/content/drive/MyDrive/train/cat.46.jpg'+cat[c])
    plt.subplot(2,5,i)
    plt.imshow(im)

for i, c in enumerate(np.random.randint(0,len(dog),5), start=6):
    im = img.imread('/content/drive/MyDrive/train/dog.10006.jpg'+dog[c])
    plt.subplot(2,5,i)
    plt.imshow(im)
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-82-b6355bfdedbc> in <cell line: 2>()
      1 plt.figure(figsize=(5,3))
      2 for i, c in enumerate(np.random.randint(0,len(cat),31), start=1):
----> 3     im = img.imread('/content/drive/MyDrive/train/cat.46.jpg'+cat[c])
      4     plt.subplot(2,5,i)
      5     plt.imshow(im)

                                 1 frames
/usr/local/lib/python3.10/dist-packages/PIL/Image.py in open(fp, mode, formats)
   3225
   3226     if filename:
-> 3227         fp = builtins.open(filename, "rb")
   3228         exclusive_fp = True
   3229

FileNotFoundError: [Errno 2] No such file or directory:
'/content/drive/MyDrive/train/cat.46.jpgcat.36.jpg'
```

Looking at a few random photos in the directory, you can see that the photos are color and have different shapes and sizes.

Split data

```
from sklearn.model_selection import train_test_split
train_set, valid_set = train_test_split(df, test_size=0.2,random_state=seed)
```

```
train_set.shape, valid_set.shape
```

```
((80, 2), (20, 2))
```

```
print('train set distribution:\n',train_set['label'].value_counts())
print('\nvalid set distribution:\n',valid_set['label'].value_counts())
print()

plt.figure(figsize=(10,5))
plt.subplot(1,3,1)
sns.countplot(x=train_set['label'], order=['dog','cat'])
plt.title('Train set label distribution')

plt.subplot(1,3,3)
sns.countplot(x=valid_set['label'], order=['dog','cat'])
plt.title('valid set label distribution')
plt.show()
```

Image data Generator and Data Augmentation

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

WIDTH, HEIGHT = 150, 150
batch_size = 32
```

Train set label distribution Valid set label distribution

```
train_datagenerator = ImageDataGenerator(rotation_range=15,
                                  rescale=1./255,
                                  shear_range=0.1,
                                  zoom_range=0.2,
                                  horizontal_flip=True,
                                  width_shift_range=0.1,
                                  height_shift_range=0.1)

training_data = train_datagenerator.flow_from_dataframe(dataframe=train_set,
                                                        directory='./train',
                                                       x_col='filename',
                                                       y_col='label',
                                                       target_size=(WIDTH, HEIGHT),
                                                       class_mode='categorical',
                                                       batch_size=batch_size)
```

```
Found 0 validated image filenames belonging to 0 classes.
/usr/local/lib/python3.10/dist-packages/keras/src/preprocessing/image.py:1137: UserWarning: Found 80 invalid image filename(s) in x_
  warnings.warn(
```

```
valid_datagenerator = ImageDataGenerator(rescale=1./255)
validation_data = valid_datagenerator.flow_from_dataframe(dataframe=valid_set,
                                                         directory='./train',
                                                         x_col='filename',
                                                         y_col='label',
                                                         target_size=(WIDTH,HEIGHT),
                                                         class_mode='categorical',
                                                         batch_size=batch_size)
```

```
Found 0 validated image filenames belonging to 0 classes.
/usr/local/lib/python3.10/dist-packages/keras/src/preprocessing/image.py:1137: UserWarning: Found 20 invalid image filename(s) in x_
  warnings.warn(
```

Making CNN Model

```
from tensorflow.keras.layers import *
from functools import partial

DefaultConv2D = partial(keras.layers.Conv2D, kernel_size=3, activation='relu', padding="SAME")

model = keras.models.Sequential()

model.add(DefaultConv2D(filters=32, kernel_size=5, input_shape=(WIDTH, HEIGHT, 3)))
model.add(MaxPooling2D(pool_size=2))

model.add(DefaultConv2D(filters=64))
model.add(DefaultConv2D(filters=64))
model.add(MaxPooling2D(pool_size=2))

model.add(DefaultConv2D(filters=128))
model.add(DefaultConv2D(filters=128))
model.add(MaxPooling2D(pool_size=2))

model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(32, activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(2, activation='softmax'))


model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_5 (Conv2D)           (None, 150, 150, 32)      2432

 max_pooling2d_3 (MaxPoolin  (None, 75, 75, 32)        0
 g2D)

 conv2d_6 (Conv2D)           (None, 75, 75, 64)        18496

 conv2d_7 (Conv2D)           (None, 75, 75, 64)        36928

 max_pooling2d_4 (MaxPoolin  (None, 37, 37, 64)        0
 g2D)

 conv2d_8 (Conv2D)           (None, 37, 37, 128)       73856

 conv2d_9 (Conv2D)           (None, 37, 37, 128)       147584

 max_pooling2d_5 (MaxPoolin  (None, 18, 18, 128)       0
 g2D)

 flatten_1 (Flatten)         (None, 41472)             0

 dense_3 (Dense)             (None, 64)                2654272

 dropout_2 (Dropout)         (None, 64)                0

 dense_4 (Dense)             (None, 32)                2080

 dropout_3 (Dropout)         (None, 32)                0

 dense_5 (Dense)             (None, 2)                 66

=================================================================
Total params: 2935714 (11.20 MB)
Trainable params: 2935714 (11.20 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Callbacks

```
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau

earlystop_cb = EarlyStopping(patience=10, restore_best_weights=True)
reduce_lr_cb = ReduceLROnPlateau(factor=0.5, patience=5, monitor='val_loss', min_lr=0.00001)
checkpoint_cb = ModelCheckpoint('model.h5', save_best_only=True)

callbacks = [earlystop_cb, reduce_lr_cb, checkpoint_cb]
```

Train the Model

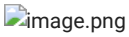
```
opt = keras.optimizers.SGD(learning_rate=0.01, momentum=0.9)
model.compile(loss="categorical_crossentropy", optimizer=opt, metrics=['accuracy'])


history = model.fit(training_data,
                    epochs=5,
                    validation_data=validation_data,
                    validation_steps=valid_set.shape[1]//batch_size,
                    steps_per_epoch=train_set.shape[1]//batch_size,
                    callbacks=callbacks)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-96-516d3eb87198> in <cell line: 1>()
----> 1 history = model.fit(training_data,
      2                     epochs=5,
```

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(acc))

plt.figure(figsize=(15,5))
plt.subplot(1,2,1)
plt.plot(epochs, acc, 'bo', label='Training accuracy')
plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.legend()


plt.subplot(1,2,2)
plt.plot(epochs, loss, 'go', label='Training Loss')
plt.plot(epochs, val_loss, 'g', label='Validation Loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-94-d19084472359> in <cell line: 1>()
----> 1 acc = history.history['accuracy']
      2 val_acc = history.history['val_accuracy']
      3 loss = history.history['loss']
      4 val_loss = history.history['val_loss']
      5

NameError: name 'history' is not defined
```

SEARCH STACK OVERFLOW



Evaluate Model

```
model1 = keras.models.load_model('model.h5')


test_loss, test_acc = model1.evaluate(validation_data, steps=len(validation_data), verbose=1)
print('Loss: %.3f' % (test_loss))
print('Accuracy: %.3f' % (test_acc * 100.0))
```

The VGG Architecture works well on the data

So the next we'll going to train VGG16 architecture (transfer learning)

Transfer Learning

```
base_model = keras.applications.vgg16.VGG16(weights="imagenet", include_top=False, input_shape=(WIDTH,HEIGHT,3))
base_model.trainable = False ## Not trainable weights


base_model.summary()


from tensorflow.keras.layers import *

model = keras.models.Sequential()
model.add(base_model)
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(2, activation='softmax'))
```

```
opt = keras.optimizers.SGD(learning_rate=0.01, momentum=0.9)
model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])


history = model.fit(training_data,
            epochs=4,
            validation_data=validation_data,
            validation_steps=valid_set.shape[0]//batch_size,
            steps_per_epoch=train_set.shape[0]//batch_size,
            )


model.layers[0].trainable=True


model.summary()


checkpoint_cb = keras.callbacks.ModelCheckpoint('model1.h5',save_best_only=True)
earlystop_cb = keras.callbacks.EarlyStopping(patience=5, restore_best_weights=True)

opt = keras.optimizers.SGD(learning_rate=0.001, momentum=0.9)
model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])

history = model.fit(training_data,
            epochs=20,
            validation_data=validation_data,
            validation_steps=valid_set.shape[0]//batch_size,
            steps_per_epoch=train_set.shape[0]//batch_size,
            callbacks=[checkpoint_cb, earlystop_cb])


acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(acc))

plt.figure(figsize=(15,5))
plt.subplot(1,2,1)
plt.plot(epochs, acc, 'bo', label='Training accuracy')
plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.legend()


plt.subplot(1,2,2)
plt.plot(epochs, loss, 'go', label='Training Loss')
plt.plot(epochs, val_loss, 'g', label='Validation Loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()


model2 = keras.models.load_model('model1.h5')


test_loss, test_acc = model2.evaluate(validation_data, steps=len(validation_data), verbose=1)
print('Loss: %.3f' % (test_loss))
print('Accuracy: %.3f' % (test_acc * 100.0))
```

Nothing better than this....

Make Predictions

```
sampleSubmission = pd.read_csv('../input/dogs-vs-cats/sampleSubmission.csv')
test_df = sampleSubmission.copy()
test_df['id'] = test_df['id'].apply(lambda x : str(x)+'.jpg')


test_gen = ImageDataGenerator(rescale=1./255)
test_generator = test_gen.flow_from_dataframe(test_df,
                                              "./test1/",
                                              x_col='id',
                                              y_col=None,
                                              class_mode=None,
                                              target_size=(WIDTH,HEIGHT),
```

```
    batch_size=batch_size,
    shuffle=False)
```

```
predict = model1.predict(test_generator, steps=np.ceil(test_df.shape[0]/batch_size))
```

```
sampleSubmission['label'] = np.argmax(predict, axis=-1)
sampleSubmission.to_csv('submission.csv', index=False)
```

```
test_df['label'] = sampleSubmission['label'].replace({ 1: 'dog', 0: 'cat' })
```

```
sample_test = test_df.values[np.random.randint(0, len(test_df), 16)]
sample_test
plt.figure(figsize=(15, 17))
for index, row in enumerate(sample_test):
    filename = row[0]
    category = row[1]
    image = img.imread("./test1/"+filename)
    plt.subplot(4, 4, index+1)
    plt.imshow(image)
    plt.xlabel('Predicted: ' + "{}".format(category) )
plt.tight_layout()
plt.show()
```

Hurrayyy

ALL are correctlty predicted

Upvote If you like it !