dogs-image-classifier-with-vgg16

August 5, 2024

0.1 1. Downloading Data

[1]:
```
!kaggle datasets download -d haroon669/
↪cats-vs-dogs-dataset-10k-cat-10k-dog-images
```

```
Warning: Looks like you're using an outdated API Version, please consider
updating (server 1.6.17 / client 1.6.14)
Dataset URL: https://www.kaggle.com/datasets/haroon669/cats-vs-dogs-
dataset-10k-cat-10k-dog-images
License(s): CC0-1.0
Downloading cats-vs-dogs-dataset-10k-cat-10k-dog-images.zip to /kaggle/working
 98%|                    | 1.04G/1.06G [00:10<00:00, 217MB/s]
100%|                    | 1.06G/1.06G [00:10<00:00, 105MB/s]
```

0.2 2. Unzip Data

[2]:
```
!unzip /kaggle/working/cats-vs-dogs-dataset-10k-cat-10k-dog-images.zip
```

```
Archive:  /kaggle/working/cats-vs-dogs-dataset-10k-cat-10k-dog-images.zip
  inflating: dogs_vs_cats/test/cats/cat.10.jpg
  inflating: dogs_vs_cats/test/cats/cat.10000.jpg
  inflating: dogs_vs_cats/test/cats/cat.10001.jpg
  inflating: dogs_vs_cats/test/cats/cat.10007.jpg
  inflating: dogs_vs_cats/test/cats/cat.10017.jpg
  inflating: dogs_vs_cats/test/cats/cat.10021.jpg
  inflating: dogs_vs_cats/test/cats/cat.10026.jpg
  inflating: dogs_vs_cats/test/cats/cat.10030.jpg
  inflating: dogs_vs_cats/test/cats/cat.10033.jpg
  inflating: dogs_vs_cats/test/cats/cat.10035.jpg
  inflating: dogs_vs_cats/test/cats/cat.10036.jpg
  inflating: dogs_vs_cats/test/cats/cat.10046.jpg
  inflating: dogs_vs_cats/test/cats/cat.10048.jpg
  inflating: dogs_vs_cats/test/cats/cat.10052.jpg
  inflating: dogs_vs_cats/test/cats/cat.10057.jpg
  inflating: dogs_vs_cats/test/cats/cat.10064.jpg
  inflating: dogs_vs_cats/test/cats/cat.10074.jpg
  inflating: dogs_vs_cats/test/cats/cat.10086.jpg
  inflating: dogs_vs_cats/test/cats/cat.10091.jpg
```

```
  inflating: train/dogs/dog.9984.jpg
  inflating: train/dogs/dog.9985.jpg
  inflating: train/dogs/dog.9987.jpg
  inflating: train/dogs/dog.9988.jpg
  inflating: train/dogs/dog.999.jpg
  inflating: train/dogs/dog.9990.jpg
  inflating: train/dogs/dog.9992.jpg
  inflating: train/dogs/dog.9993.jpg
  inflating: train/dogs/dog.9994.jpg
  inflating: train/dogs/dog.9996.jpg
  inflating: train/dogs/dog.9997.jpg
  inflating: train/dogs/dog.9998.jpg
  inflating: train/dogs/dog.9999.jpg
```

0.3 3. Import Necessary Libraries

[3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os
import cv2
from PIL import Image
import random
import warnings
warnings.simplefilter('ignore')
```

[4]:
```
os.listdir("/kaggle/working/dogs_vs_cats")
```

[4]:
```
['train', 'test']
```

[5]:
```
print("Dog Images", len(os.listdir("/kaggle/working/dogs_vs_cats/train/dogs")))
print("Cat Images", len(os.listdir("/kaggle/working/dogs_vs_cats/train/cats")))
```

```
Dog Images 10000
Cat Images 10000
```

0.4 4. Visualizing Data

[6]:
```
dog_directory = "/kaggle/working/dogs_vs_cats/train/dogs/"
dog_name = random.sample(os.listdir(dog_directory),2)
img = os.path.join(dog_directory,dog_name[0])
image = Image.open(img)
plt.imshow(image)
plt.tight_layout()
plt.axis('off')
plt.show()
```

[7]:
```
cat_directory = "/kaggle/working/dogs_vs_cats/train/cats"
cat_image = random.sample(os.listdir(cat_directory),1)
plt.figure(figsize=(5,5))
img_path = os.path.join(cat_directory,cat_image[0])
image = Image.open(img_path)

plt.axis('off')
plt.tight_layout()
plt.imshow(image)
plt.show()
```

[42]:
```
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from keras.applications.vgg16 import VGG16
from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_img,␣
 ↪img_to_array
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
```

0.5 5. Model Saving as conv_base

[9]:
```
conv_base = VGG16(weights = 'imagenet',
include_top = False, # include_top means Dense layers is not included(ann)
input_shape=(150,150,3))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256              0s
0us/step
```

[10]:
```
conv_base.summary()
```

```
Model: "vgg16"
```

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 150, 150, 3)	0
block1_conv1 (Conv2D)	(None, 150, 150, 64)	1,792
block1_conv2 (Conv2D)	(None, 150, 150, 64)	36,928
block1_pool (MaxPooling2D)	(None, 75, 75, 64)	0
block2_conv1 (Conv2D)	(None, 75, 75, 128)	73,856
block2_conv2 (Conv2D)	(None, 75, 75, 128)	147,584
block2_pool (MaxPooling2D)	(None, 37, 37, 128)	0
block3_conv1 (Conv2D)	(None, 37, 37, 256)	295,168
block3_conv2 (Conv2D)	(None, 37, 37, 256)	590,080
block3_conv3 (Conv2D)	(None, 37, 37, 256)	590,080
block3_pool (MaxPooling2D)	(None, 18, 18, 256)	0
block4_conv1 (Conv2D)	(None, 18, 18, 512)	1,180,160
block4_conv2 (Conv2D)	(None, 18, 18, 512)	2,359,808
block4_conv3 (Conv2D)	(None, 18, 18, 512)	2,359,808
block4_pool (MaxPooling2D)	(None, 9, 9, 512)	0
block5_conv1 (Conv2D)	(None, 9, 9, 512)	2,359,808
block5_conv2 (Conv2D)	(None, 9, 9, 512)	2,359,808
block5_conv3 (Conv2D)	(None, 9, 9, 512)	2,359,808
block5_pool (MaxPooling2D)	(None, 4, 4, 512)	0

Total params: 14,714,688 (56.13 MB)

Trainable params: 14,714,688 (56.13 MB)

```
Non-trainable params: 0 (0.00 B)
```

0.6 6. Fine Tuning

[11]:
```
conv_base.trainable = True

set_trainable = False

for layer in conv_base.layers:
  if layer.name == 'block5_conv1':
    set_trainable = True
  if set_trainable:
    layer.trainable = True
  else:
    layer.trainable = False

for layer in conv_base.layers:
  print(layer.name,layer.trainable)
```

```
input_layer False
block1_conv1 False
block1_conv2 False
block1_pool False
block2_conv1 False
block2_conv2 False
block2_pool False
block3_conv1 False
block3_conv2 False
block3_conv3 False
block3_pool False
block4_conv1 False
block4_conv2 False
block4_conv3 False
block4_pool False
block5_conv1 True
block5_conv2 True
block5_conv3 True
block5_pool True
```

[12]:
```
from keras.models import Model
num_classes = 1
x = conv_base.output
model = Sequential()
#model.add(conv_base)
x = Flatten()(x)
x = Dense(512, activation='relu')(x)
```

```
x = Dropout(0.2)(x)
x = Dense(1024, activation='relu')(x)
x = Dropout(0.2)(x)
x = Dense(num_classes, activation='sigmoid')(x)

model = Model(inputs=conv_base.input, outputs=x)
```

[13]:
```
model.summary()
```

Model: "functional"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 150, 150, 3)	0
block1_conv1 (Conv2D)	(None, 150, 150, 64)	1,792
block1_conv2 (Conv2D)	(None, 150, 150, 64)	36,928
block1_pool (MaxPooling2D)	(None, 75, 75, 64)	0
block2_conv1 (Conv2D)	(None, 75, 75, 128)	73,856
block2_conv2 (Conv2D)	(None, 75, 75, 128)	147,584
block2_pool (MaxPooling2D)	(None, 37, 37, 128)	0
block3_conv1 (Conv2D)	(None, 37, 37, 256)	295,168
block3_conv2 (Conv2D)	(None, 37, 37, 256)	590,080
block3_conv3 (Conv2D)	(None, 37, 37, 256)	590,080
block3_pool (MaxPooling2D)	(None, 18, 18, 256)	0
block4_conv1 (Conv2D)	(None, 18, 18, 512)	1,180,160
block4_conv2 (Conv2D)	(None, 18, 18, 512)	2,359,808
block4_conv3 (Conv2D)	(None, 18, 18, 512)	2,359,808
block4_pool (MaxPooling2D)	(None, 9, 9, 512)	0
block5_conv1 (Conv2D)	(None, 9, 9, 512)	2,359,808
block5_conv2 (Conv2D)	(None, 9, 9, 512)	2,359,808

block5_conv3 (Conv2D)	(None, 9, 9, 512)	2,359,808
block5_pool (MaxPooling2D)	(None, 4, 4, 512)	0
flatten (Flatten)	(None, 8192)	0
dense (Dense)	(None, 512)	4,194,816
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 1024)	525,312
dropout_1 (Dropout)	(None, 1024)	0
dense_2 (Dense)	(None, 1)	1,025

Total params: 19,435,841 (74.14 MB)

Trainable params: 11,800,577 (45.02 MB)

Non-trainable params: 7,635,264 (29.13 MB)

0.7 7. Data Augmentation

[14]:
```
base_dir = "/kaggle/working/dogs_vs_cats/train"
train_datagen = ImageDataGenerator(
rescale = 1/255,
validation_split = 0.2,
horizontal_flip = True,
zoom_range = 0.2,
rotation_range = 30)
```

[15]:
```
train_generator = train_datagen.flow_from_directory(
base_dir,
batch_size=32,
target_size = (150,150),
class_mode = 'binary')
```

```
Found 20000 images belonging to 2 classes.
```

[16]:
```
validation_datagen = ImageDataGenerator(
rescale = 1/255)
```

[18]:
```
data_dir = "/kaggle/working/dogs_vs_cats/test"
validation_data = validation_datagen.flow_from_directory(
data_dir,
target_size = (150,150), batch_size=32,
class_mode = 'binary')
```

```
Found 5000 images belonging to 2 classes.
```

[19]:
```
early_stopping = EarlyStopping(monitor='val_loss',patience=5, mode='auto',␣
↪restore_best_weights=True)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', patience = 0.3, factor = 0.1,␣
↪min_lr = 0.00001)
```

0.8 8. Modeling

[20]:
```
model.compile(optimizer=keras.optimizers.SGD(), loss='binary_crossentropy',␣
↪metrics = ['accuracy'])
history = model.fit(train_generator, epochs=50, validation_data =␣
↪validation_data, callbacks = [early_stopping])
```

```
Epoch 1/50
  1/625              3:55:36 23s/step -
accuracy: 0.5625 - loss: 0.6959

WARNING: All log messages before absl::InitializeLog() is called are written to
STDERR
I0000 00:00:1722775702.579163     124 device_compiler.h:186] Compiled cluster
using XLA!  This line is logged at most once for the lifetime of the process.
W0000 00:00:1722775702.603972     124 graph_launch.cc:671] Fallback to op-by-op
mode because memset node breaks graph update

625/625              0s 182ms/step -
accuracy: 0.8009 - loss: 0.4160

W0000 00:00:1722775817.202939     124 graph_launch.cc:671] Fallback to op-by-op
mode because memset node breaks graph update
W0000 00:00:1722775833.058472     124 graph_launch.cc:671] Fallback to op-by-op
mode because memset node breaks graph update

625/625              153s 209ms/step -
accuracy: 0.8010 - loss: 0.4158 - val_accuracy: 0.9172 - val_loss: 0.2023
Epoch 2/50
625/625              126s 199ms/step -
accuracy: 0.9130 - loss: 0.2083 - val_accuracy: 0.9284 - val_loss: 0.1661
Epoch 3/50
625/625              125s 198ms/step -
accuracy: 0.9279 - loss: 0.1706 - val_accuracy: 0.9364 - val_loss: 0.1680
Epoch 4/50
625/625              125s 197ms/step -
```

```
accuracy: 0.9383 - loss: 0.1496 - val_accuracy: 0.9436 - val_loss: 0.1307
Epoch 5/50
625/625             125s 198ms/step -
accuracy: 0.9465 - loss: 0.1351 - val_accuracy: 0.9538 - val_loss: 0.1096
Epoch 6/50
625/625             126s 199ms/step -
accuracy: 0.9542 - loss: 0.1143 - val_accuracy: 0.9416 - val_loss: 0.1397
Epoch 7/50
625/625             127s 201ms/step -
accuracy: 0.9588 - loss: 0.1088 - val_accuracy: 0.9616 - val_loss: 0.0975
Epoch 8/50
625/625             127s 201ms/step -
accuracy: 0.9579 - loss: 0.1031 - val_accuracy: 0.9616 - val_loss: 0.1043
Epoch 9/50
625/625             129s 203ms/step -
accuracy: 0.9655 - loss: 0.0870 - val_accuracy: 0.9578 - val_loss: 0.1072
Epoch 10/50
625/625             127s 201ms/step -
accuracy: 0.9655 - loss: 0.0888 - val_accuracy: 0.9588 - val_loss: 0.1067
Epoch 11/50
625/625             126s 200ms/step -
accuracy: 0.9697 - loss: 0.0806 - val_accuracy: 0.9584 - val_loss: 0.1008
Epoch 12/50
625/625             128s 203ms/step -
accuracy: 0.9709 - loss: 0.0758 - val_accuracy: 0.9630 - val_loss: 0.1085
```

0.9 9. Model Saving

[21]:
```
model.save('model.h5')
```

0.10 10. Prediction

[25]:
```
train_pred = model.predict(train_generator)
train_pred = train_pred>0.5
```

```
625/625             112s 179ms/step
```

0.11 Random Image Prediction

[69]:
```
image_dirs = os.listdir("/kaggle/working/test")
img_dir = random.sample(image_dirs,1)[0]
images = os.listdir("/kaggle/working/test/" + img_dir)
img_name = random.sample(images,1)[0]
image_path = os.path.join("/kaggle/working/test/dogs", img_name)
image = load_img(image_path, target_size=(150, 150))  # VGG16 input size
image_array = img_to_array(image) / 255.0  # Normalize the image
image_array = np.expand_dims(image_array, axis=0)
```

[70]:
```
pred = model.predict(image_array)>0.5
if pred == True:
    print("It's Dog")
    plt.imshow(Image.open(image_path))
    plt.axis('off')
else:
    print("It's Cat")
    plt.imshow(Image.open(image_path))
    plt.axis('off')
```

```
1/1             0s 17ms/step
It's Dog
```

[ ]: