Untitled27

December 20, 2023

1 Cat and Dog Image Classifier

Introduction:

In this project, the aim to develop an image classification model using data science techniques in Python to distinguish between images of cats and dogs. Image classification is a fundamental task in computer vision, and it has numerous applications, including content moderation, surveillance, and medical image analysis.

Objective:

The primary objective of this project is to create a robust and accurate image classifier that can correctly identify whether an input image contains a cat or a dog. To achieve this, we will leverage machine learning techniques, specifically deep learning, which has proven to be highly effective in image classification tasks.

1.1 Step 1 : Importing Libraries

[1]:
```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing import image
import os
import numpy as np
import matplotlib.pyplot as plt
```

```
WARNING:tensorflow:From C:\Users\indhu\downloads\anaconda3\Lib\site-
packages\keras\src\losses.py:2976: The name
tf.losses.sparse_softmax_cross_entropy is deprecated. Please use
tf.compat.v1.losses.sparse_softmax_cross_entropy instead.
```

1.2 Step 2 : Load pre-trained VGG16 model

[2]:
```
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224,␣
↪224, 3))
```

```
WARNING:tensorflow:From C:\Users\indhu\downloads\anaconda3\Lib\site-
packages\keras\src\backend.py:1398: The name
tf.executing_eagerly_outside_functions is deprecated. Please use
tf.compat.v1.executing_eagerly_outside_functions instead.

WARNING:tensorflow:From C:\Users\indhu\downloads\anaconda3\Lib\site-
packages\keras\src\layers\pooling\max_pooling2d.py:161: The name tf.nn.max_pool
is deprecated. Please use tf.nn.max_pool2d instead.
```

1.3 Step 3 : Freeze the pre-trained layers

[3]:
```
for layer in base_model.layers:
    layer.trainable = False
```

1.4 Step 4 : Build Binary Classification Model

[4]:
```
model = Sequential()
model.add(base_model)
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

1.5 Step 5 : Compile Model

[5]:
```
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),␣
↪loss='binary_crossentropy', metrics=['accuracy'])
```

1.6 Step 6 : Data Augmentation and Preprocessing

[6]:
```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2
)

test_datagen = ImageDataGenerator(rescale=1./255)
```

1.7 Step 7 : Load and Preprocess Data

[7]:
```
train_generator = train_datagen.flow_from_directory(r'C:
 ↪\Users\indhu\Downloads\train', target_size=(224, 224), batch_size=32,␣
 ↪class_mode='binary')
test_generator = test_datagen.flow_from_directory(r'C:
 ↪\Users\indhu\Downloads\test', target_size=(224, 224), batch_size=32,␣
 ↪class_mode='binary')
```

```
Found 557 images belonging to 2 classes.
Found 140 images belonging to 2 classes.
```

1.8 Step 8 : Train and Evaluate Model

[8]:
```
history = model.fit(train_generator, epochs=10, validation_data=test_generator)
test_loss, test_acc = model.evaluate(test_generator)
print(f"Test Accuracy: {test_acc}")
```

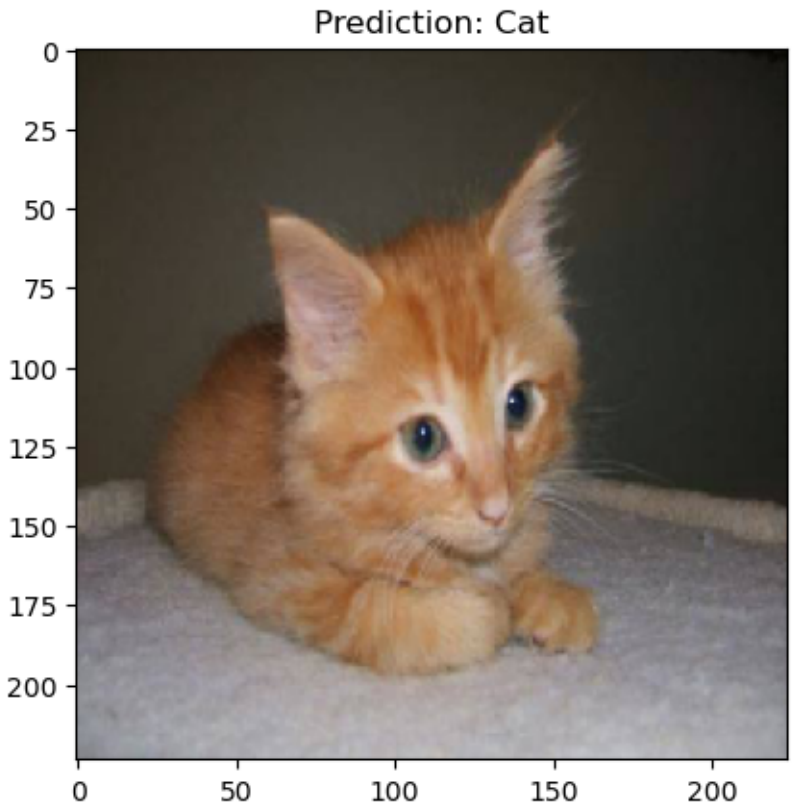
```
Epoch 1/10
WARNING:tensorflow:From C:\Users\indhu\downloads\anaconda3\Lib\site-
packages\keras\src\utils\tf_utils.py:492: The name tf.ragged.RaggedTensorValue
is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.

WARNING:tensorflow:From C:\Users\indhu\downloads\anaconda3\Lib\site-
packages\keras\src\engine\base_layer_utils.py:384: The name
tf.executing_eagerly_outside_functions is deprecated. Please use
tf.compat.v1.executing_eagerly_outside_functions instead.

18/18 [==============================] - 122s 7s/step - loss: 0.8396 - accuracy:
0.5566 - val_loss: 0.5069 - val_accuracy: 0.7857
Epoch 2/10
18/18 [==============================] - 119s 7s/step - loss: 0.4728 - accuracy:
0.7666 - val_loss: 0.4465 - val_accuracy: 0.7857
Epoch 3/10
18/18 [==============================] - 119s 7s/step - loss: 0.4090 - accuracy:
0.8187 - val_loss: 0.5628 - val_accuracy: 0.7143
Epoch 4/10
18/18 [==============================] - 119s 7s/step - loss: 0.3542 - accuracy:
0.8528 - val_loss: 0.4792 - val_accuracy: 0.7500
Epoch 5/10
18/18 [==============================] - 119s 7s/step - loss: 0.3098 - accuracy:
0.8689 - val_loss: 0.4690 - val_accuracy: 0.7786
Epoch 6/10
18/18 [==============================] - 119s 7s/step - loss: 0.2796 - accuracy:
0.8905 - val_loss: 0.4650 - val_accuracy: 0.7786
Epoch 7/10
18/18 [==============================] - 119s 7s/step - loss: 0.2577 - accuracy:
0.8815 - val_loss: 0.4320 - val_accuracy: 0.7857
```

```
Epoch 8/10
18/18 [==============================] - 119s 7s/step - loss: 0.2876 - accuracy:
0.8654 - val_loss: 0.4208 - val_accuracy: 0.8071
Epoch 9/10
18/18 [==============================] - 119s 7s/step - loss: 0.2238 - accuracy:
0.9174 - val_loss: 0.4440 - val_accuracy: 0.8071
Epoch 10/10
18/18 [==============================] - 119s 7s/step - loss: 0.2431 - accuracy:
0.8941 - val_loss: 0.4326 - val_accuracy: 0.8071
5/5 [==============================] - 24s 4s/step - loss: 0.4326 - accuracy:
0.8071
Test Accuracy: 0.8071428537368774
```

1.9 Step 9 : Prediction

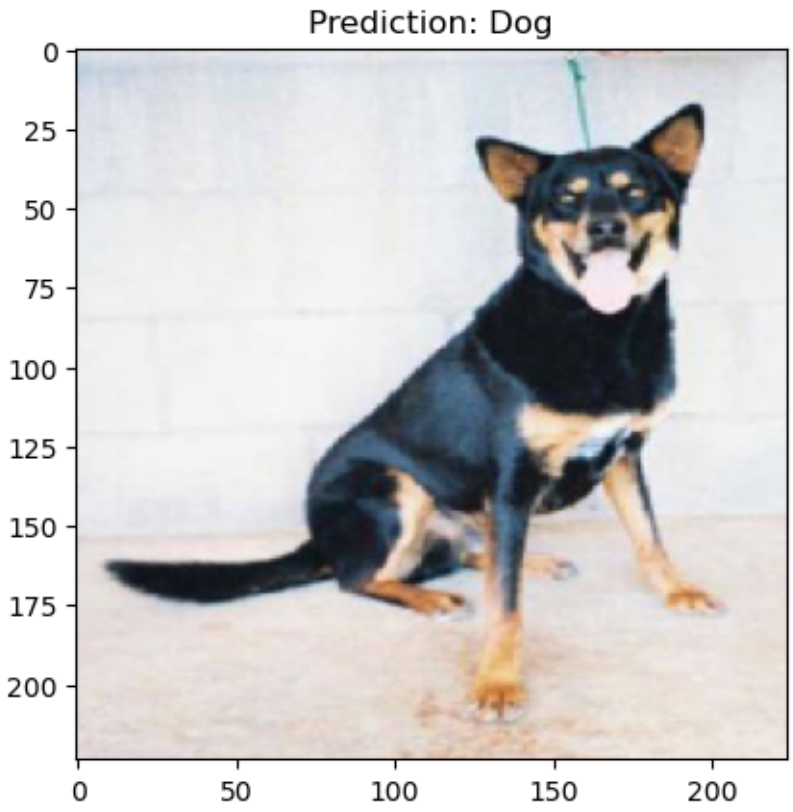
[9]:
```
img_path = r'C:\Users\indhu\Desktop\classification\test\cats\cat.325.jpg'
img = image.load_img(img_path, target_size=(224, 224))
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
img_array = preprocess_input(img_array)
predictions = model.predict(img_array)
if predictions[0] > 0.5:
    result = "Dog"
else:
    result = "Cat"
plt.imshow(img)
plt.title(f"Prediction: {result}")
plt.show()
```

```
1/1 [==============================] - 1s 644ms/step
```



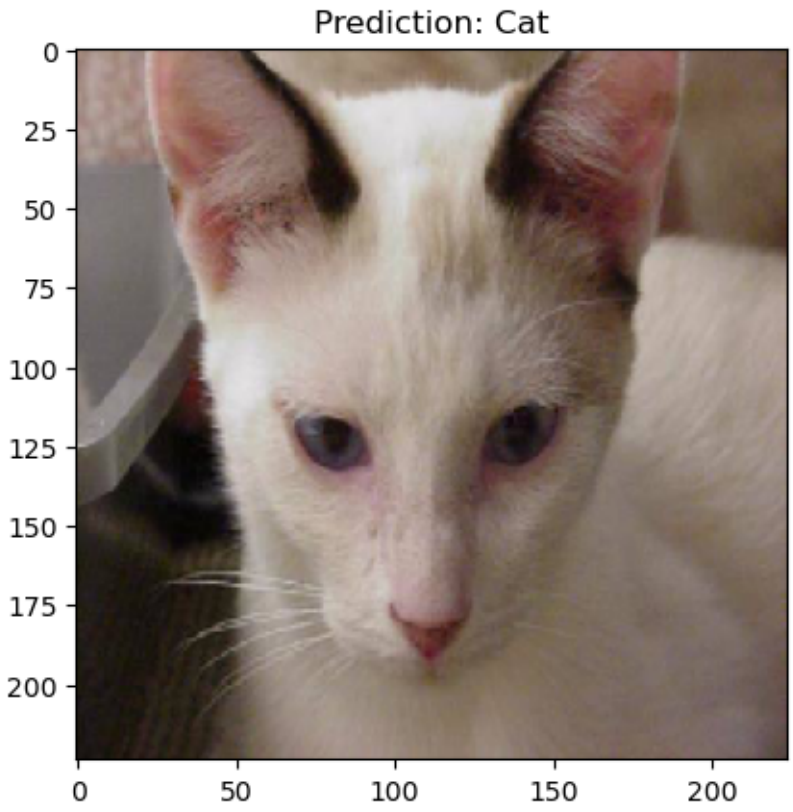
[10]:
```
img_path = r'C:\Users\indhu\Desktop\classification\test\dogs\dog.368.jpg'
img = image.load_img(img_path, target_size=(224, 224))
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
img_array = preprocess_input(img_array)
predictions = model.predict(img_array)
if predictions[0] > 0.5:
    result = "Dog"
else:
    result = "Cat"
plt.imshow(img)
plt.title(f"Prediction: {result}")
plt.show()
```

```
1/1 [==============================] - 0s 257ms/step
```



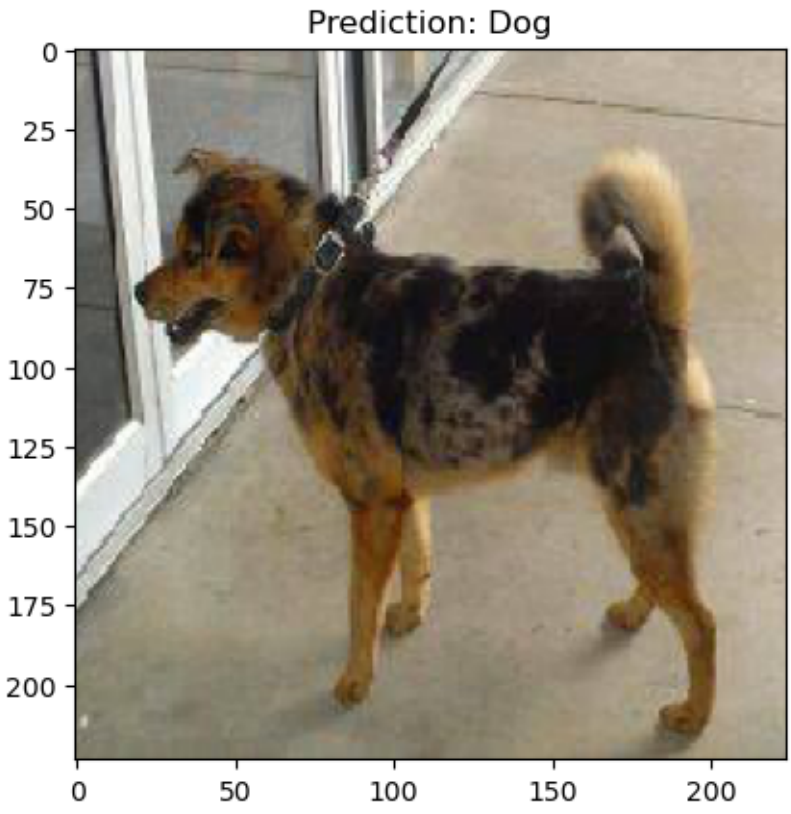
[12]:
```
img_path = r'C:\Users\indhu\Desktop\classification\train\cats\cat.64.jpg'
img = image.load_img(img_path, target_size=(224, 224))
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
img_array = preprocess_input(img_array)
predictions = model.predict(img_array)
if predictions[0] > 0.5:
    result = "Dog"
else:
    result = "Cat"
plt.imshow(img)
plt.title(f"Prediction: {result}")
plt.show()
```

```
1/1 [==============================] - 0s 290ms/step
```



[13]:
```
img_path = r'C:\Users\indhu\Desktop\classification\train\dogs\dog.142.jpg'
img = image.load_img(img_path, target_size=(224, 224))
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
img_array = preprocess_input(img_array)
predictions = model.predict(img_array)
if predictions[0] > 0.5:
    result = "Dog"
else:
    result = "Cat"
plt.imshow(img)
plt.title(f"Prediction: {result}")
plt.show()
```

```
1/1 [==============================] - 0s 270ms/step
```


Prediction: Dog
0
25
50
75
100
125
150
175
200
0
50
100
150
200