image-classification-resnet

March 11, 2024

0.0.1 Import Libraries

[1]:
```
import numpy as np
import pandas as pd
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Models
import tensorflow as tf
from tensorflow.keras.layers import Input
from tensorflow.keras.layers import Flatten, Dense, Dropout, Conv2D,␣
↪MaxPooling2D

import os

folder_path = '../input/animal-image-datasetdog-cat-and-panda/animals/animals/'
for dirname, _, filenames in os.walk(folder_path):
    print(dirname)
```

```
../input/animal-image-datasetdog-cat-and-panda/animals/animals/
../input/animal-image-datasetdog-cat-and-panda/animals/animals/dogs
../input/animal-image-datasetdog-cat-and-panda/animals/animals/panda
../input/animal-image-datasetdog-cat-and-panda/animals/animals/cats
```

[2]:
```
print('Total cats images:', len(os.listdir(folder_path + 'cats')))
print('Total dogs images:', len(os.listdir(folder_path + 'dogs')))
print('Total pandas images:', len(os.listdir(folder_path + 'panda')))
```

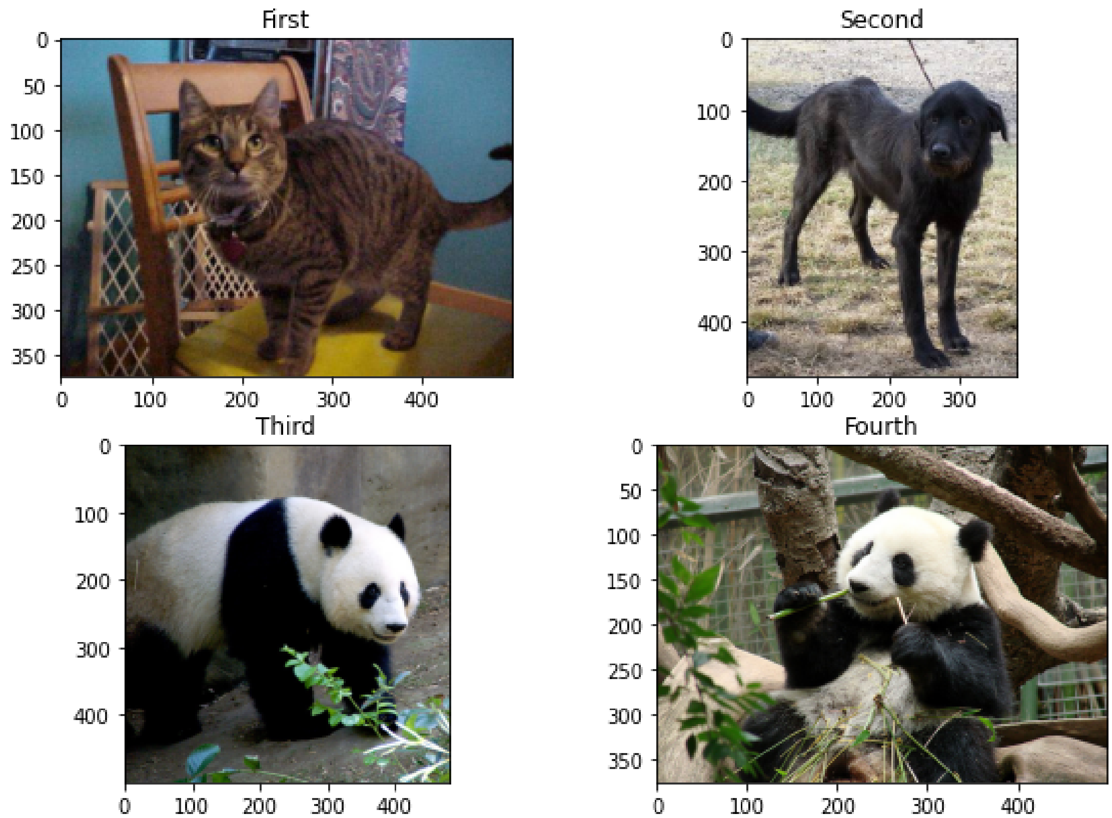
```
Total cats images: 1000
Total dogs images: 1000
Total pandas images: 1000
```

[3]:
```
from keras.preprocessing import image
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
%matplotlib inline
```

0.0.2 Show some images

[4]:
```python
fig = plt.figure(figsize=(10, 7))

# setting values to rows and column variables
rows = 2
columns = 2

# reading images
Image1 = image.load_img(folder_path + 'cats/cats_00001.jpg')
Image2 = image.load_img(folder_path + 'dogs/dogs_00001.jpg')
Image3 = image.load_img(folder_path + 'panda/panda_00001.jpg')
Image4 = image.load_img(folder_path + 'panda/panda_00003.jpg')

fig.add_subplot(rows, columns, 1)
plt.imshow(Image1)
plt.title("First")

fig.add_subplot(rows, columns, 2)
plt.imshow(Image2)
plt.title("Second")

fig.add_subplot(rows, columns, 3)
plt.imshow(Image3)
plt.title("Third")

fig.add_subplot(rows, columns, 4)
plt.imshow(Image4)
plt.title("Fourth")
```

[4]:
```
Text(0.5, 1.0, 'Fourth')
```



Split data : 80% train, 20% validation

[5]:
```
train_datagen = ImageDataGenerator(
                fill_mode = 'nearest',
                validation_split=0.2
)
```

[6]:
```
train_generator=train_datagen.flow_from_directory(
    folder_path,
    target_size=(108,108),
    color_mode='rgb',
    class_mode='categorical',
    subset='training',
)
validation_generator=train_datagen.flow_from_directory(
    folder_path,
    target_size=(108,108),
    color_mode='rgb',
    class_mode='categorical',
    subset='validation',
)
```

```
Found 2400 images belonging to 3 classes.
Found 600 images belonging to 3 classes.
```

[7]:

```
from keras.applications import ResNet50
model = tf.keras.models.Sequential([
    ResNet50(input_shape=(108,108,3), include_top=False),
])
for layer in model.layers:
  layer.trainable = False

model.add(Conv2D(64, (3,3), activation='relu'))
model.add(MaxPooling2D(2,2))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(3, activation='softmax'))

model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94773248/94765736 [==============================] - 1s 0us/step
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50 (Functional)        (None, 4, 4, 2048)        23587712
_________________________________________________________________
conv2d (Conv2D)              (None, 2, 2, 64)          1179712
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 1, 1, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 64)                0
_________________________________________________________________
dense (Dense)                (None, 64)                4160
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 195
=================================================================
Total params: 24,771,779
Trainable params: 1,184,067
Non-trainable params: 23,587,712
_________________________________________________________________
```

[8]:

```
model.compile(optimizer='Adam',
              loss='categorical_crossentropy',
```

```python
                metrics=['accuracy'])
```

[9]:
```python
history = model.fit(train_generator,
                    validation_data=validation_generator,
                    epochs=25,
                    verbose=1,
                    )
```

```
Epoch 1/25
75/75 [==============================] - 38s 392ms/step - loss: 1.8237 -
accuracy: 0.6816 - val_loss: 0.4622 - val_accuracy: 0.8500
Epoch 2/25
75/75 [==============================] - 13s 174ms/step - loss: 0.3802 -
accuracy: 0.8593 - val_loss: 0.2975 - val_accuracy: 0.8967
Epoch 3/25
75/75 [==============================] - 13s 169ms/step - loss: 0.3030 -
accuracy: 0.9205 - val_loss: 0.3336 - val_accuracy: 0.8867
Epoch 4/25
75/75 [==============================] - 13s 171ms/step - loss: 0.1920 -
accuracy: 0.9296 - val_loss: 0.2697 - val_accuracy: 0.9167
Epoch 5/25
75/75 [==============================] - 13s 167ms/step - loss: 0.1613 -
accuracy: 0.9421 - val_loss: 0.2644 - val_accuracy: 0.9233
Epoch 6/25
75/75 [==============================] - 12s 166ms/step - loss: 0.1257 -
accuracy: 0.9500 - val_loss: 0.3849 - val_accuracy: 0.9017
Epoch 7/25
75/75 [==============================] - 13s 173ms/step - loss: 0.0635 -
accuracy: 0.9773 - val_loss: 0.3215 - val_accuracy: 0.9133
Epoch 8/25
75/75 [==============================] - 13s 170ms/step - loss: 0.0423 -
accuracy: 0.9842 - val_loss: 0.4613 - val_accuracy: 0.9017
Epoch 9/25
75/75 [==============================] - 13s 172ms/step - loss: 0.0488 -
accuracy: 0.9852 - val_loss: 0.3908 - val_accuracy: 0.9183
Epoch 10/25
75/75 [==============================] - 13s 167ms/step - loss: 0.0388 -
accuracy: 0.9869 - val_loss: 0.4905 - val_accuracy: 0.9083
Epoch 11/25
75/75 [==============================] - 13s 168ms/step - loss: 0.0534 -
accuracy: 0.9818 - val_loss: 0.5380 - val_accuracy: 0.9067
Epoch 12/25
75/75 [==============================] - 13s 170ms/step - loss: 0.0248 -
accuracy: 0.9899 - val_loss: 0.5889 - val_accuracy: 0.9100
Epoch 13/25
75/75 [==============================] - 13s 167ms/step - loss: 0.0330 -
accuracy: 0.9934 - val_loss: 0.3782 - val_accuracy: 0.8967
```

```
Epoch 14/25
75/75 [==============================] - 13s 169ms/step - loss: 0.0355 -
accuracy: 0.9888 - val_loss: 0.4316 - val_accuracy: 0.9133
Epoch 15/25
75/75 [==============================] - 13s 167ms/step - loss: 0.0310 -
accuracy: 0.9924 - val_loss: 0.4650 - val_accuracy: 0.9117
Epoch 16/25
75/75 [==============================] - 13s 168ms/step - loss: 0.0161 -
accuracy: 0.9926 - val_loss: 0.5947 - val_accuracy: 0.9083
Epoch 17/25
75/75 [==============================] - 13s 169ms/step - loss: 0.0364 -
accuracy: 0.9876 - val_loss: 0.4491 - val_accuracy: 0.9067
Epoch 18/25
75/75 [==============================] - 13s 168ms/step - loss: 0.0170 -
accuracy: 0.9968 - val_loss: 0.5354 - val_accuracy: 0.9033
Epoch 19/25
75/75 [==============================] - 13s 177ms/step - loss: 0.0394 -
accuracy: 0.9852 - val_loss: 0.5163 - val_accuracy: 0.9000
Epoch 20/25
75/75 [==============================] - 13s 167ms/step - loss: 0.0267 -
accuracy: 0.9914 - val_loss: 0.4460 - val_accuracy: 0.9200
Epoch 21/25
75/75 [==============================] - 13s 177ms/step - loss: 0.0200 -
accuracy: 0.9951 - val_loss: 0.5121 - val_accuracy: 0.9167
Epoch 22/25
75/75 [==============================] - 13s 169ms/step - loss: 0.0309 -
accuracy: 0.9924 - val_loss: 0.7349 - val_accuracy: 0.9100
Epoch 23/25
75/75 [==============================] - 13s 169ms/step - loss: 0.0140 -
accuracy: 0.9965 - val_loss: 0.6946 - val_accuracy: 0.9033
Epoch 24/25
75/75 [==============================] - 13s 169ms/step - loss: 0.0085 -
accuracy: 0.9973 - val_loss: 0.4800 - val_accuracy: 0.9100
Epoch 25/25
75/75 [==============================] - 13s 167ms/step - loss: 0.0505 -
accuracy: 0.9874 - val_loss: 0.6157 - val_accuracy: 0.9100
```

[13]:

```python
# Plot Model accuracy and loss
plt.figure(figsize=(12, 6))

# Plot Model accuracy
plt.subplot(1, 2, 1)  # 1 row, 2 columns, subplot 1
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
```

```
plt.legend(['Train', 'Validation'], loc='lower right')

# Plot Model Loss
plt.subplot(1, 2, 2)  # 1 row, 2 columns, subplot 2
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper right')

plt.tight_layout()
plt.show()
```