Data Arguments

January 5, 2024

[ ]:
```
pip install --upgrade keras
```

[6]:
```
from tensorflow import keras
```

[ ]:
```
pip show tensorflow
```

[ ]:
```
pip show keras
```

[ ]:
```
pip install tensorflow
```

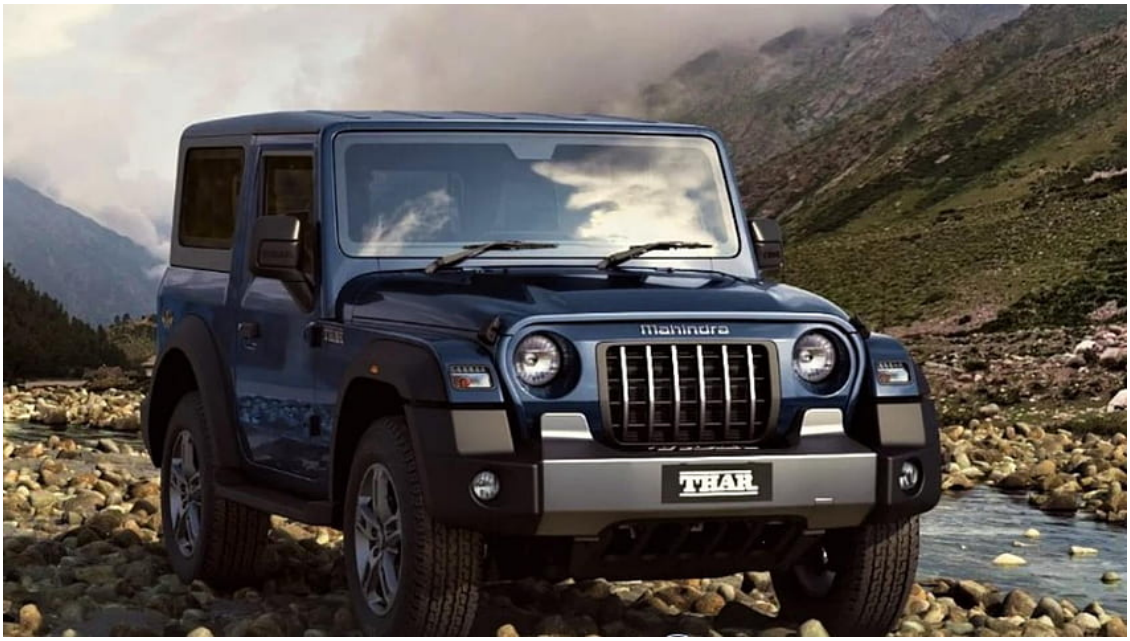
[27]:
```
#from tensorflow.keras.preprocessing.image import ImageDataGenerator,␣
 ↪array_to_img, img_to_array, load_img
#from keras.preprocessing.image import ImageDataGenerator ,  array_to_img ,␣
 ↪img_to_array , load_img

from tensorflow.keras.utils import image_dataset_from_directory
from keras.utils import array_to_img ,img_to_array, load_img

datagen = ImageDataGenerator(
        rotation_range=40,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='nearest')
```

[4]:
```
import tensorflow.keras as keras
```

[28]:
```
img = load_img(r"C:\Users\srava\Downloads\mahindra-thar-2020-black.jpg")
```

[29]:
```
img
```

[29]:

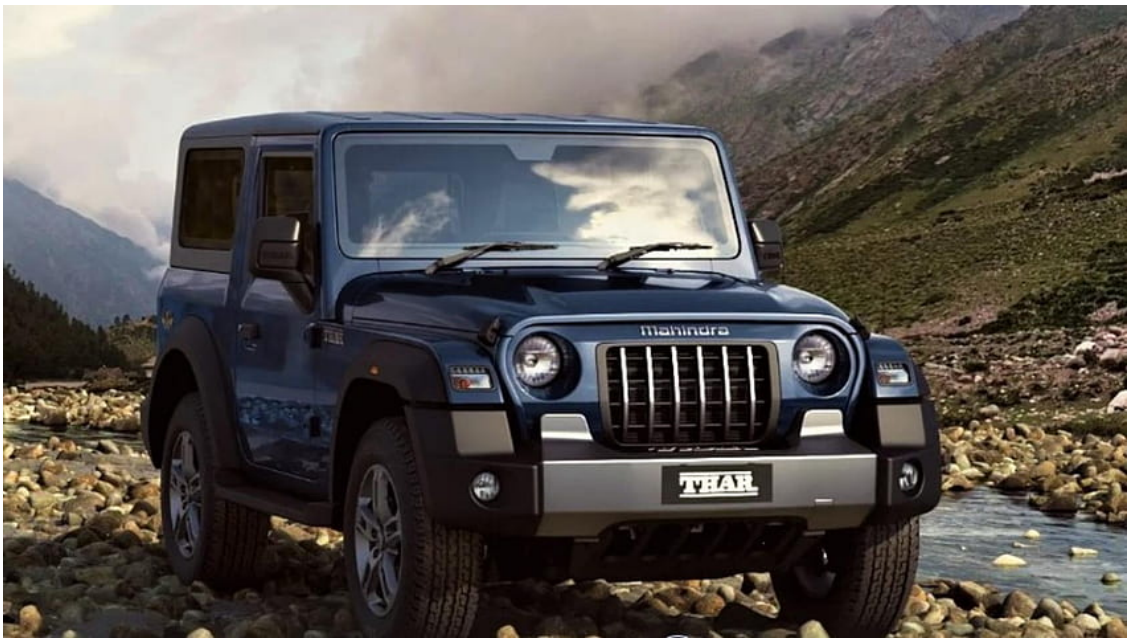


[30]:
```
x = img_to_array(img)  # this is a Numpy array with shape (3, 150, 150)
x = x.reshape((1,) + x.shape)  # this is a Numpy array with shape (1, 3, 150,␣
 ↪150)

# the .flow() command below generates batches of randomly transformed images
# and saves the results to the `preview/` directory
i = 0
for batch in datagen.flow(x, batch_size=1,
                          save_to_dir=r"C:\Users\srava\OneDrive\Desktop\Data␣
 ↪Arguments", save_prefix='cat', save_format='jpeg'):
    i += 1
    if i > 30:
        break  # otherwise the generator would loop indefinitely
```

[24]:
```
from tensorflow.keras.utils import image_dataset_from_directory
from keras.utils import array_to_img ,img_to_array, load_img

datagen = ImageDataGenerator(
        rotation_range=40,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='reflect')
img = load_img(r"C:\Users\srava\Downloads\mahindra-thar-2020-black.jpg")
```

```
img
```

[24]:



[26]:

```
x = img_to_array(img)  # this is a Numpy array with shape (3, 150, 150)
x = x.reshape((1,) + x.shape)  # this is a Numpy array with shape (1, 3, 150,␣
 ↪150)

# the .flow() command below generates batches of randomly transformed images
# and saves the results to the `preview/` directory
i = 0
for batch in datagen.flow(x, batch_size=1,
                          save_to_dir=r"C:\Users\srava\OneDrive\Desktop\data␣
 ↪Arguments Image creation", save_prefix='cat', save_format='jpeg'):
    i += 1
    if i > 30:
        break  # otherwise the generator would loop indefinitely
```

[12]:

```
keras.applications.ResNet50(
    include_top=True,
    weights="imagenet",
    input_tensor=None,
    input_shape=None,
    pooling=None,
    classes=1000,
    classifier_activation="softmax",
)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
```

```
applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels.h5
102967424/102967424 [==============================] - 8s 0us/step
```

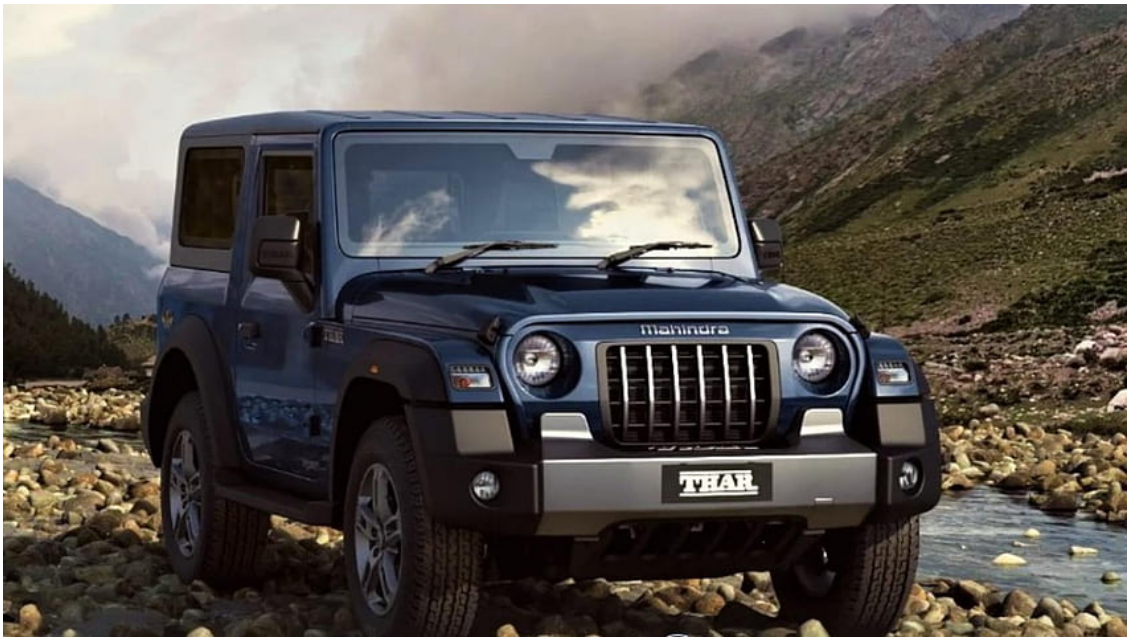
[12]:
```
<keras.engine.functional.Functional at 0x21885b9a8f0>
```

1 Basic Predictions of the Image

2 RESNET50

[7]:
```
from keras.utils import array_to_img ,img_to_array, load_img
img = load_img(r"C:\Users\srava\Downloads\mahindra-thar-2020-black.jpg")
img
```

[7]:



[32]:
```
import numpy as np
from tensorflow.keras.applications.resnet50 import ResNet50, preprocess_input,␣
 ↪decode_predictions
from tensorflow.keras.preprocessing import image

# Load the ResNet-50 model pre-trained on ImageNet data
model = ResNet50(weights='imagenet')

# Load and preprocess the input image
img_path = r"C:\Users\srava\Downloads\mahindra-thar-2020-black.jpg"  # replace␣
 ↪with the path to your image file
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)

# Make predictions
predictions = model.predict(img_array)

# Decode and print the top-3 predicted classes
decoded_predictions = decode_predictions(predictions, top=3)[0]

print("Predictions:")
for i, (imagenet_id, label, score) in enumerate(decoded_predictions):
    print(f"{i + 1}: {label} ({score:.2f})")

# Optionally, you can obtain the class index for the top prediction
top_class_index = np.argmax(predictions[0])
print(f"\nTop Prediction Class Index: {top_class_index}")
```

```
1/1 [==============================] - 2s 2s/step
Downloading data from https://storage.googleapis.com/download.tensorflow.org/dat
a/imagenet_class_index.json
35363/35363 [==============================] - 0s 0us/step
Predictions:
1: jeep (1.00)
2: pickup (0.00)
3: grille (0.00)

Top Prediction Class Index: 609
```

3 ResNet50V2

[36]:
```
import numpy as np
from tensorflow.keras.applications.resnet_v2 import ResNet50V2,␣
 ↪preprocess_input, decode_predictions
from tensorflow.keras.preprocessing import image

# Load the ResNet50V2 model pre-trained on ImageNet data
model = ResNet50V2(weights='imagenet')

# Load and preprocess the input image
img_path = r"C:\Users\srava\Downloads\mahindra-thar-2020-black.jpg"  # replace␣
 ↪with the path to your image file
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)

# Make predictions
```

```
predictions = model.predict(img_array)

# Decode and print the top-3 predicted classes
decoded_predictions = decode_predictions(predictions, top=3)[0]

print("Predictions:")
for i, (imagenet_id, label, score) in enumerate(decoded_predictions):
    print(f"{i + 1}: {label} ({score:.2f})")

# Optionally, you can obtain the class index for the top prediction
top_class_index = np.argmax(predictions[0])
print(f"\nTop Prediction Class Index: {top_class_index}")
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/resnet/resnet50v2_weights_tf_dim_ordering_tf_kernels.h5
102869336/102869336 [==============================] - 8s 0us/step
1/1 [==============================] - 1s 1s/step
Predictions:
1: jeep (1.00)
2: pickup (0.00)
3: grille (0.00)

Top Prediction Class Index: 609
```

4 VGG16

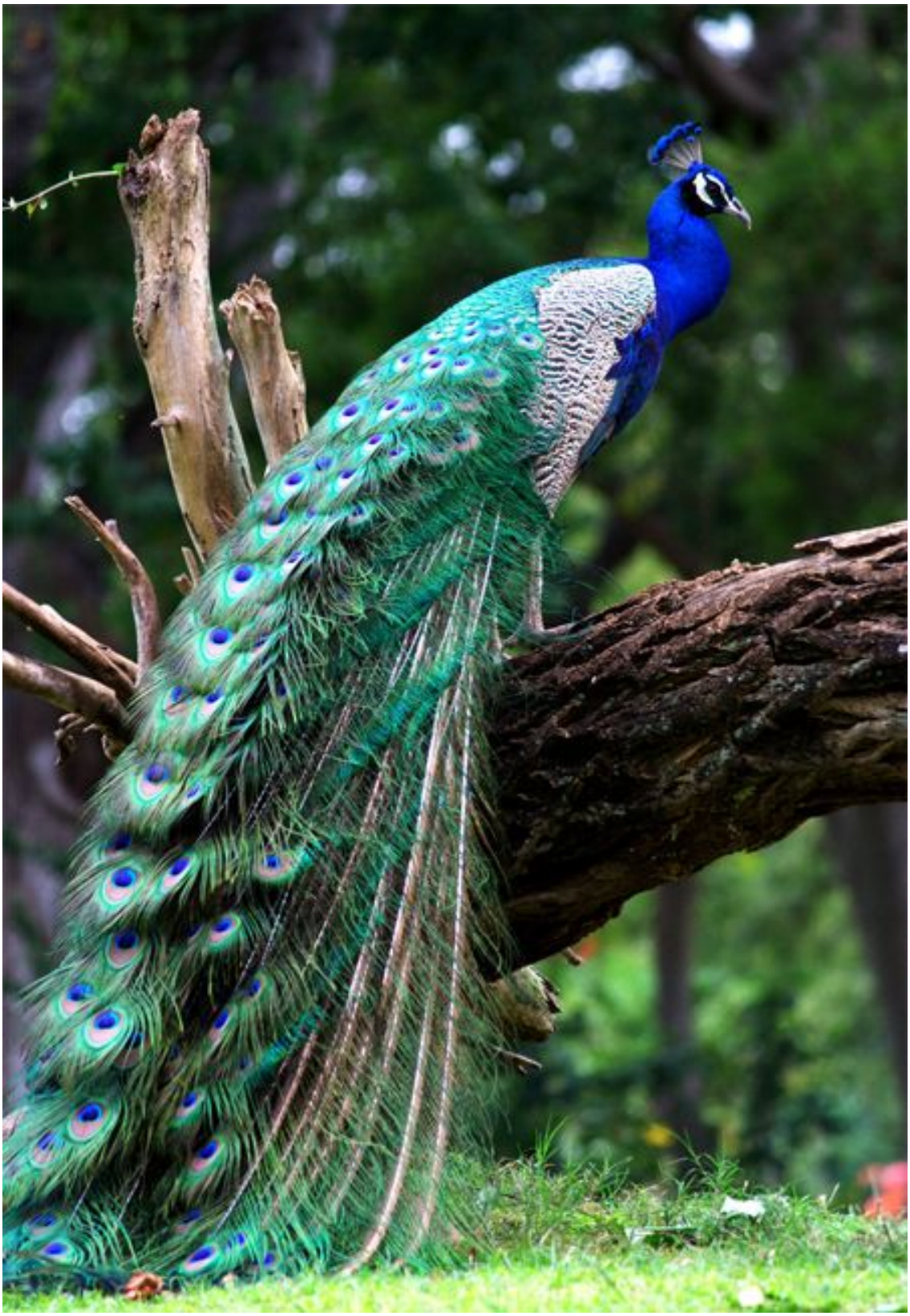
[37]:
```
import numpy as np
from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input,␣
 ↪decode_predictions
from tensorflow.keras.preprocessing import image

# Load the VGG16 model pre-trained on ImageNet data
model = VGG16(weights='imagenet')

# Load and preprocess the input image
img_path = r"C:\Users\srava\Downloads\mahindra-thar-2020-black.jpg"  # replace␣
 ↪with the path to your image file
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)

# Make predictions
predictions = model.predict(img_array)

# Decode and print the top-3 predicted classes
decoded_predictions = decode_predictions(predictions, top=3)[0]
```

```
print("Predictions:")
for i, (imagenet_id, label, score) in enumerate(decoded_predictions):
    print(f"{i + 1}: {label} ({score:.2f})")

# Optionally, you can obtain the class index for the top prediction
top_class_index = np.argmax(predictions[0])
print(f"\nTop Prediction Class Index: {top_class_index}")
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels.h5
553467096/553467096 [==============================] - 42s 0us/step
1/1 [==============================] - 1s 555ms/step
Predictions:
1: jeep (0.99)
2: plow (0.00)
3: tractor (0.00)

Top Prediction Class Index: 609
```

[8]:
```
from keras.utils import array_to_img ,img_to_array, load_img
img = load_img(r"C:\Users\srava\Downloads\peacock.jpg")
img
```

[8]:



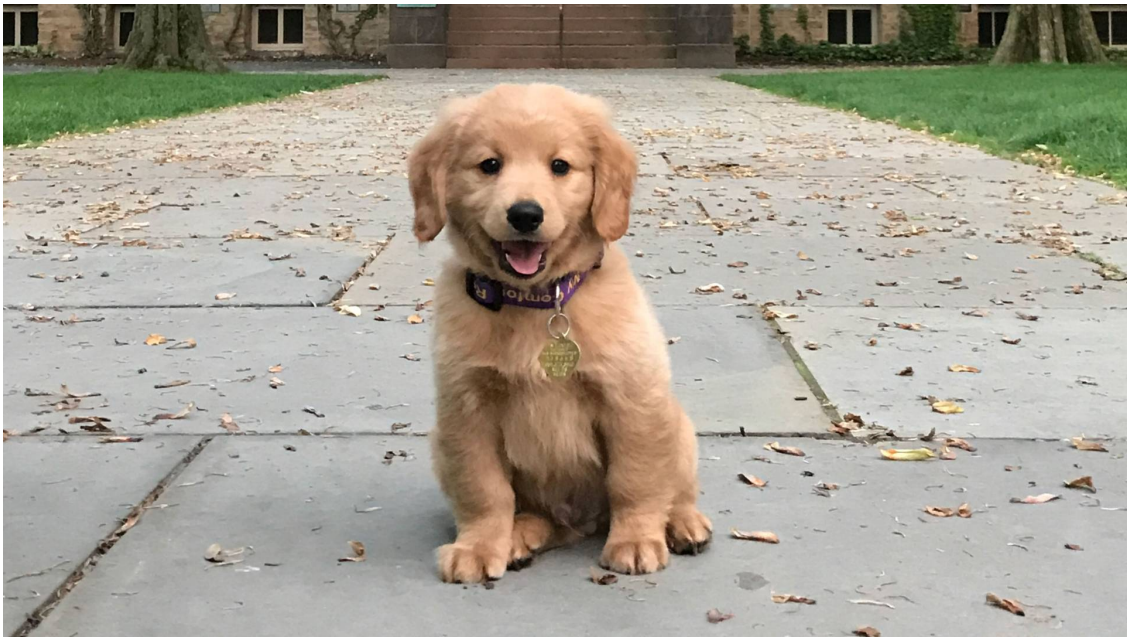
[1]:
```
import numpy as np
from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input,␣
 ↪decode_predictions
from tensorflow.keras.preprocessing import image

# Load the VGG16 model pre-trained on ImageNet data
model = VGG16(weights='imagenet')

# Load and preprocess the input image
img_path = r"C:\Users\srava\Downloads\peacock.jpg" # replace with the path to␣
 ↪your image file
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)

# Make predictions
predictions = model.predict(img_array)

# Decode and print the top-3 predicted classes
decoded_predictions = decode_predictions(predictions, top=3)[0]

print("Predictions:")
for i, (imagenet_id, label, score) in enumerate(decoded_predictions):
    print(f"{i + 1}: {label} ({score:.2f})")

# Optionally, you can obtain the class index for the top prediction
top_class_index = np.argmax(predictions[0])
print(f"\nTop Prediction Class Index: {top_class_index}")
```

```
1/1 [==============================] - 1s 699ms/step
Predictions:
1: peacock (1.00)
2: indigo_bunting (0.00)
3: hummingbird (0.00)

Top Prediction Class Index: 84
```

5 VGG19

[9]:
```
from keras.utils import array_to_img ,img_to_array, load_img
img = load_img(r"C:\Users\srava\Downloads\Nassau_2697x1517.jpg")
img
```

[9]:



[2]:
```
import numpy as np
from tensorflow.keras.applications.vgg19 import VGG19, preprocess_input,␣
 ↪decode_predictions
from tensorflow.keras.preprocessing import image

# Load the VGG19 model pre-trained on ImageNet data
model = VGG19(weights='imagenet')

# Load and preprocess the input image
img_path = r"C:\Users\srava\Downloads\Nassau_2697x1517.jpg"  # replace with the␣
 ↪path to your image file
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)

# Make predictions
predictions = model.predict(img_array)

# Decode and print the top-3 predicted classes
decoded_predictions = decode_predictions(predictions, top=3)[0]

print("Predictions:")
for i, (imagenet_id, label, score) in enumerate(decoded_predictions):
    print(f"{i + 1}: {label} ({score:.2f})")
```

```
# Optionally, you can obtain the class index for the top prediction
top_class_index = np.argmax(predictions[0])
print(f"\nTop Prediction Class Index: {top_class_index}")
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/vgg19/vgg19_weights_tf_dim_ordering_tf_kernels.h5
574710816/574710816 [==============================] - 246s 0us/step
1/1 [==============================] - 1s 583ms/step
Predictions:
1: golden_retriever (0.78)
2: Saluki (0.13)
3: Afghan_hound (0.03)

Top Prediction Class Index: 207
```

[4]:
```
import numpy as np
from tensorflow.keras.applications.vgg19 import VGG19, preprocess_input,␣
 ↪decode_predictions
from tensorflow.keras.preprocessing import image

# Load the VGG19 model pre-trained on ImageNet data
model = VGG19(weights='imagenet')

# Load and preprocess the input image
img_path = r"C:\Users\srava\Downloads\Nassau_2697x1517.jpg"  # replace with the␣
 ↪path to your image file
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)

# Make predictions
predictions = model.predict(img_array)

# Decode and print the top-3 predicted classes
decoded_predictions = decode_predictions(predictions, top=8)[0]

print("Predictions:")
for i, (imagenet_id, label, score) in enumerate(decoded_predictions):
    print(f"{i + 1}: {label} ({score:.2f})")

# Optionally, you can obtain the class index for the top prediction
top_class_index = np.argmax(predictions[0])
print(f"\nTop Prediction Class Index: {top_class_index}")
```

```
1/1 [==============================] - 1s 599ms/step
Predictions:
1: golden_retriever (0.78)
```

```
2: Saluki (0.13)
3: Afghan_hound (0.03)
4: standard_poodle (0.02)
5: Labrador_retriever (0.01)
6: kuvasz (0.01)
7: redbone (0.00)
8: Irish_setter (0.00)

Top Prediction Class Index: 207
```

[5]:
```python
import numpy as np
from tensorflow.keras.applications.resnet_v2 import ResNet50V2,␣
 ↪preprocess_input, decode_predictions
from tensorflow.keras.preprocessing import image

# Load the ResNet50V2 model pre-trained on ImageNet data
model = ResNet50V2(weights='imagenet')

# Load and preprocess the input image
img_path = r"C:\Users\srava\Downloads\Nassau_2697x1517.jpg"  # replace with the␣
 ↪path to your image file
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)

# Make predictions
predictions = model.predict(img_array)

# Decode and print the top-3 predicted classes
decoded_predictions = decode_predictions(predictions, top=10)[0]

print("Predictions:")
for i, (imagenet_id, label, score) in enumerate(decoded_predictions):
    print(f"{i + 1}: {label} ({score:.2f})")

# Optionally, you can obtain the class index for the top prediction
top_class_index = np.argmax(predictions[0])
print(f"\nTop Prediction Class Index: {top_class_index}")
```

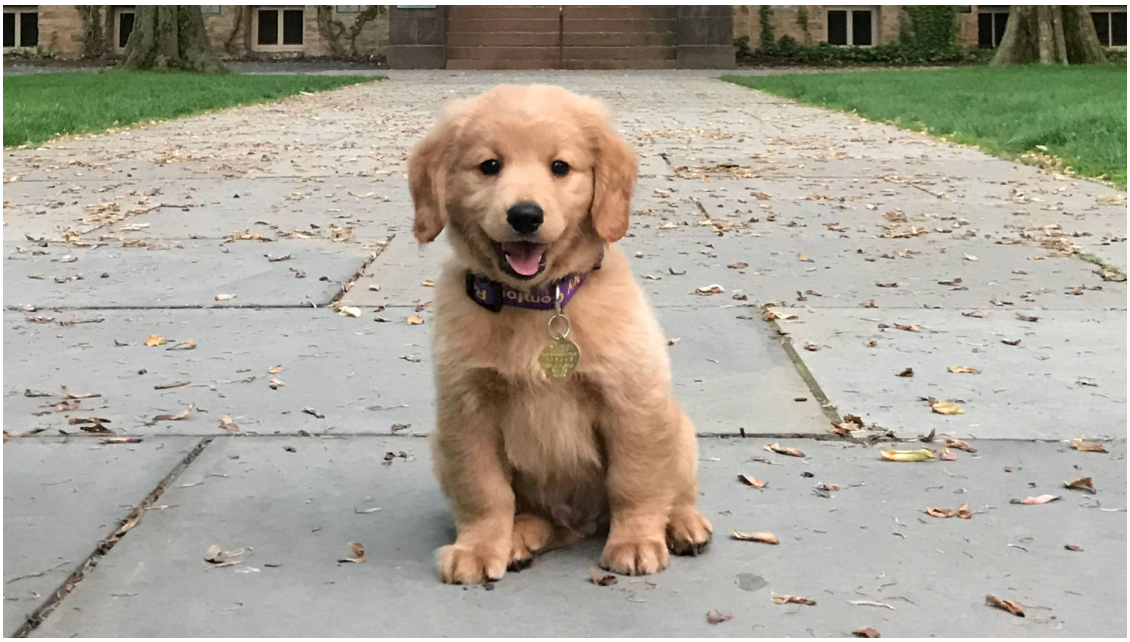
```
1/1 [==============================] - 2s 2s/step
Predictions:
1: golden_retriever (1.00)
2: Labrador_retriever (0.00)
3: kuvasz (0.00)
4: cocker_spaniel (0.00)
5: redbone (0.00)
6: Great_Pyrenees (0.00)
```

```
7: Tibetan_mastiff (0.00)
8: English_setter (0.00)
9: Leonberg (0.00)
10: Newfoundland (0.00)

Top Prediction Class Index: 207
```

6 Predictions for dog image using Keras Api Applications.....

[49]:
```
from keras.utils import array_to_img ,img_to_array, load_img
img = load_img(r"C:\Users\srava\Downloads\Nassau_2697x1517.jpg")
img
```

[49]:



[21]:
```
import numpy as np
from tensorflow.keras.applications.resnet50 import ResNet50, preprocess_input,␣
 ↪decode_predictions
from tensorflow.keras.preprocessing import image

# Load the ResNet-50 model pre-trained on ImageNet data
model = ResNet50(weights='imagenet')


# Load and preprocess the input image
img_path = r"C:\Users\srava\Downloads\Nassau_2697x1517.jpg"  # replace with the␣
 ↪path to your image file
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)

# Measure inference time
start_time = time.time()
predictions = model.predict(img_array)
end_time = time.time()

# Decode and print the top-3 predicted classes
decoded_predictions = decode_predictions(predictions, top=5)[0]

print("Predictions:")
for i, (imagenet_id, label, score) in enumerate(decoded_predictions):
    print(f"{i + 1}: {label} ({score:.2f})")

# Optionally, you can obtain the class index for the top prediction
top_class_index = np.argmax(predictions[0])
print(f"\nTop Prediction Class Index: {top_class_index}")

# Calculate and print the inference time per step
inference_time_ms = (end_time - start_time) * 1000.0
print(f"Inference Time: {inference_time_ms:.2f} ms")

# Model summary provides information about parameters and layers
#model.summary()

# Get the size of the model in megabytes
model_size_MB = model.count_params() * 4 / (1024 ** 2)  # 4 bytes for float32
print(f"Size (MB): {model_size_MB:.2f} MB")


# Get the number of parameters and depth from the model's layers
num_parameters = model.count_params()
model_depth = len(model.layers)

print(f"Parameters: {num_parameters}")
print(f"Depth: {model_depth}")
```

```
1/1 [==============================] - 1s 732ms/step
Predictions:
1: golden_retriever (0.99)
2: Labrador_retriever (0.00)
3: kuvasz (0.00)
4: Afghan_hound (0.00)
5: standard_poodle (0.00)

Top Prediction Class Index: 207
```

```
Inference Time: 765.26 ms
Size (MB): 97.80 MB
Parameters: 25636712
Depth: 177
```

[22]:
```python
import numpy as np
from tensorflow.keras.applications.resnet_v2 import ResNet50V2,␣
 ↪preprocess_input, decode_predictions
from tensorflow.keras.preprocessing import image

# Load the ResNet50V2 model pre-trained on ImageNet data
model = ResNet50V2(weights='imagenet')

# Load and preprocess the input image
img_path = r"C:\Users\srava\Downloads\Nassau_2697x1517.jpg"  # replace with the␣
 ↪path to your image file
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)

# Measure inference time
start_time = time.time()
predictions = model.predict(img_array)
end_time = time.time()

# Decode and print the top-3 predicted classes
decoded_predictions = decode_predictions(predictions, top=5)[0]

print("Predictions:")
for i, (imagenet_id, label, score) in enumerate(decoded_predictions):
    print(f"{i + 1}: {label} ({score:.2f})")

# Optionally, you can obtain the class index for the top prediction
top_class_index = np.argmax(predictions[0])
print(f"\nTop Prediction Class Index: {top_class_index}")

# Calculate and print the inference time per step
inference_time_ms = (end_time - start_time) * 1000.0
print(f"Inference Time: {inference_time_ms:.2f} ms")

# Model summary provides information about parameters and layers
#model.summary()

# Get the size of the model in megabytes
model_size_MB = model.count_params() * 4 / (1024 ** 2)  # 4 bytes for float32
print(f"Size (MB): {model_size_MB:.2f} MB")
```

```
# Get the number of parameters and depth from the model's layers
num_parameters = model.count_params()
model_depth = len(model.layers)

print(f"Parameters: {num_parameters}")
print(f"Depth: {model_depth}")
```

```
1/1 [==============================] - 1s 1s/step
Predictions:
1: golden_retriever (1.00)
2: Labrador_retriever (0.00)
3: kuvasz (0.00)
4: cocker_spaniel (0.00)
5: redbone (0.00)

Top Prediction Class Index: 207
Inference Time: 1275.22 ms
Size (MB): 97.71 MB
Parameters: 25613800
Depth: 192
```

[25]:
```
import numpy as np
from tensorflow.keras.applications.vgg19 import VGG19, preprocess_input,␣
 ↪decode_predictions
from tensorflow.keras.preprocessing import image
import time

# Load the VGG19 model pre-trained on ImageNet data
model = VGG19(weights='imagenet')


# Load and preprocess the input image
img_path = r"C:\Users\srava\Downloads\Nassau_2697x1517.jpg"  # replace with the␣
 ↪path to your image file
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)

# Measure inference time
start_time = time.time()
predictions = model.predict(img_array)
end_time = time.time()

# Decode and print the top-3 predicted classes
```

```
decoded_predictions = decode_predictions(predictions, top=5)[0]

print("Predictions:")
for i, (imagenet_id, label, score) in enumerate(decoded_predictions):
    print(f"{i + 1}: {label} ({score:.2f})")

# Optionally, you can obtain the class index for the top prediction
top_class_index = np.argmax(predictions[0])
print(f"\nTop Prediction Class Index: {top_class_index}")

# Calculate and print the inference time per step
inference_time_ms = (end_time - start_time) * 1000.0
print(f"Inference Time: {inference_time_ms:.2f} ms")

# Model summary provides information about parameters and layers
#model.summary()

# Get the size of the model in megabytes
model_size_MB = model.count_params() * 4 / (1024 ** 2)  # 4 bytes for float32
print(f"Size (MB): {model_size_MB:.2f} MB")


# Get the number of parameters and depth from the model's layers
num_parameters = model.count_params()
model_depth = len(model.layers)

print(f"Parameters: {num_parameters}")
print(f"Depth: {model_depth}")
```

```
1/1 [==============================] - 0s 209ms/step
Predictions:
1: golden_retriever (0.78)
2: Saluki (0.13)
3: Afghan_hound (0.03)
4: standard_poodle (0.02)
5: Labrador_retriever (0.01)

Top Prediction Class Index: 207
Inference Time: 233.69 ms
Size (MB): 548.05 MB
Parameters: 143667240
Depth: 26
```

[26]:
```
import numpy as np
from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input,␣
 ↪decode_predictions
from tensorflow.keras.preprocessing import image
```

```
# Load the VGG16 model pre-trained on ImageNet data
model = VGG16(weights='imagenet')

# Load and preprocess the input image
img_path = r"C:\Users\srava\Downloads\Nassau_2697x1517.jpg"  # replace with the␣
 ↪path to your image file
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)

# Measure inference time
start_time = time.time()
predictions = model.predict(img_array)
end_time = time.time()

# Decode and print the top-3 predicted classes
decoded_predictions = decode_predictions(predictions, top=5)[0]

print("Predictions:")
for i, (imagenet_id, label, score) in enumerate(decoded_predictions):
    print(f"{i + 1}: {label} ({score:.2f})")

# Optionally, you can obtain the class index for the top prediction
top_class_index = np.argmax(predictions[0])
print(f"\nTop Prediction Class Index: {top_class_index}")

# Calculate and print the inference time per step
inference_time_ms = (end_time - start_time) * 1000.0
print(f"Inference Time: {inference_time_ms:.2f} ms")

# Model summary provides information about parameters and layers
#model.summary()

# Get the size of the model in megabytes
model_size_MB = model.count_params() * 4 / (1024 ** 2)  # 4 bytes for float32
print(f"Size (MB): {model_size_MB:.2f} MB")


# Get the number of parameters and depth from the model's layers
num_parameters = model.count_params()
model_depth = len(model.layers)

print(f"Parameters: {num_parameters}")
print(f"Depth: {model_depth}")
```

```
1/1 [==============================] - 0s 181ms/step
```

```
Predictions:
1: golden_retriever (0.89)
2: Afghan_hound (0.05)
3: Saluki (0.02)
4: bloodhound (0.01)
5: Labrador_retriever (0.01)

Top Prediction Class Index: 207
Inference Time: 214.87 ms
Size (MB): 527.79 MB
Parameters: 138357544
Depth: 23
```

[28]:
```python
import numpy as np
from tensorflow.keras.applications import Xception
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.xception import preprocess_input,␣
 ↪decode_predictions

# Load the Xception model pre-trained on ImageNet data
model = Xception(weights='imagenet')

# Load and preprocess the input image
img_path = r"C:\Users\srava\Downloads\Nassau_2697x1517.jpg"  # replace with the␣
 ↪path to your image file
img = image.load_img(img_path, target_size=(299, 299))  # Xception requires␣
 ↪input shape (299, 299)
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
img_array = preprocess_input(img_array)

# Measure inference time
start_time = time.time()
predictions = model.predict(img_array)
end_time = time.time()

# Decode and print the top-3 predicted classes
decoded_predictions = decode_predictions(predictions, top=5)[0]

print("Predictions:")
for i, (imagenet_id, label, score) in enumerate(decoded_predictions):
    print(f"{i + 1}: {label} ({score:.2f})")

# Optionally, you can obtain the class index for the top prediction
top_class_index = np.argmax(predictions[0])
print(f"\nTop Prediction Class Index: {top_class_index}")
```

```
# Calculate and print the inference time per step
inference_time_ms = (end_time - start_time) * 1000.0
print(f"Inference Time: {inference_time_ms:.2f} ms")

# Model summary provides information about parameters and layers
#model.summary()

# Get the size of the model in megabytes
model_size_MB = model.count_params() * 4 / (1024 ** 2)  # 4 bytes for float32
print(f"Size (MB): {model_size_MB:.2f} MB")


# Get the number of parameters and depth from the model's layers
num_parameters = model.count_params()
model_depth = len(model.layers)

print(f"Parameters: {num_parameters}")
print(f"Depth: {model_depth}")
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/xception/xception_weights_tf_dim_ordering_tf_kernels.h5
91884032/91884032 [==============================] - 26s 0us/step
1/1 [==============================] - 2s 2s/step
Predictions:
1: golden_retriever (0.97)
2: Labrador_retriever (0.01)
3: cocker_spaniel (0.00)
4: tennis_ball (0.00)
5: kuvasz (0.00)

Top Prediction Class Index: 207
Inference Time: 1623.19 ms
Size (MB): 87.40 MB
Parameters: 22910480
Depth: 134
```

[30]:
```
import numpy as np
from tensorflow.keras.applications import InceptionV3
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.inception_v3 import preprocess_input,␣
 ↪decode_predictions

# Load the InceptionV3 model pre-trained on ImageNet data
model = InceptionV3(weights='imagenet')

# Load and preprocess the input image
```

```
img_path = r"C:\Users\srava\Downloads\Nassau_2697x1517.jpg"  # replace with the␣
 ↪path to your image file
img = image.load_img(img_path, target_size=(299, 299))  # Xception requires␣
 ↪input shape (299, 299)
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
img_array = preprocess_input(img_array)

# Measure inference time
start_time = time.time()
predictions = model.predict(img_array)
end_time = time.time()

# Decode and print the top-3 predicted classes
decoded_predictions = decode_predictions(predictions, top=5)[0]

print("Predictions:")
for i, (imagenet_id, label, score) in enumerate(decoded_predictions):
    print(f"{i + 1}: {label} ({score:.2f})")

# Optionally, you can obtain the class index for the top prediction
top_class_index = np.argmax(predictions[0])
print(f"\nTop Prediction Class Index: {top_class_index}")

# Calculate and print the inference time per step
inference_time_ms = (end_time - start_time) * 1000.0
print(f"Inference Time: {inference_time_ms:.2f} ms")

# Model summary provides information about parameters and layers
#model.summary()

# Get the size of the model in megabytes
model_size_MB = model.count_params() * 4 / (1024 ** 2)  # 4 bytes for float32
print(f"Size (MB): {model_size_MB:.2f} MB")


# Get the number of parameters and depth from the model's layers
num_parameters = model.count_params()
model_depth = len(model.layers)

print(f"Parameters: {num_parameters}")
print(f"Depth: {model_depth}")
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/inception_v3/inception_v3_weights_tf_dim_ordering_tf_kernels.h5
96112376/96112376 [==============================] - 20s 0us/step
1/1 [==============================] - 2s 2s/step
```

```
Predictions:
1: golden_retriever (0.95)
2: Labrador_retriever (0.01)
3: tennis_ball (0.00)
4: doormat (0.00)
5: cocker_spaniel (0.00)

Top Prediction Class Index: 207
Inference Time: 2438.82 ms
Size (MB): 90.99 MB
Parameters: 23851784
Depth: 313
```

[33]:
```python
import numpy as np
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input,␣
 ↪decode_predictions

# Load the MobileNetV2 model pre-trained on ImageNet data
model = MobileNetV2(weights='imagenet')

# Load and preprocess the input image
img_path = r"C:\Users\srava\Downloads\Nassau_2697x1517.jpg"  # replace with the␣
 ↪path to your image file
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)

# Measure inference time
start_time = time.time()
predictions = model.predict(img_array)
end_time = time.time()

# Decode and print the top-3 predicted classes
decoded_predictions = decode_predictions(predictions, top=5)[0]

print("Predictions:")
for i, (imagenet_id, label, score) in enumerate(decoded_predictions):
    print(f"{i + 1}: {label} ({score:.2f})")

# Optionally, you can obtain the class index for the top prediction
top_class_index = np.argmax(predictions[0])
print(f"\nTop Prediction Class Index: {top_class_index}")

# Calculate and print the inference time per step
```

```
inference_time_ms = (end_time - start_time) * 1000.0
print(f"Inference Time: {inference_time_ms:.2f} ms")

# Model summary provides information about parameters and layers
#model.summary()

# Get the size of the model in megabytes
model_size_MB = model.count_params() * 4 / (1024 ** 2)  # 4 bytes for float32
print(f"Size (MB): {model_size_MB:.2f} MB")


# Get the number of parameters and depth from the model's layers
num_parameters = model.count_params()
model_depth = len(model.layers)

print(f"Parameters: {num_parameters}")
print(f"Depth: {model_depth}")
```

```
1/1 [==============================] - 1s 1s/step
Predictions:
1: golden_retriever (0.97)
2: Labrador_retriever (0.01)
3: Chesapeake_Bay_retriever (0.00)
4: cocker_spaniel (0.00)
5: flat-coated_retriever (0.00)

Top Prediction Class Index: 207
Inference Time: 1306.48 ms
Size (MB): 13.50 MB
Parameters: 3538984
Depth: 156
```

[38]:
```
import numpy as np
from tensorflow.keras.applications import DenseNet121
from tensorflow.keras.applications.densenet import preprocess_input,␣
 ↪decode_predictions
from tensorflow.keras.preprocessing import image

# Load the DenseNet121 model pre-trained on ImageNet data
model = DenseNet121(weights='imagenet')

# Load and preprocess the input image
img_path = r"C:\Users\srava\Downloads\Nassau_2697x1517.jpg"  # replace with the␣
 ↪path to your image file
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)

# Measure inference time
start_time = time.time()
predictions = model.predict(img_array)
end_time = time.time()

# Decode and print the top-3 predicted classes
decoded_predictions = decode_predictions(predictions, top=5)[0]

print("Predictions:")
for i, (imagenet_id, label, score) in enumerate(decoded_predictions):
    print(f"{i + 1}: {label} ({score:.2f})")

# Optionally, you can obtain the class index for the top prediction
top_class_index = np.argmax(predictions[0])
print(f"\nTop Prediction Class Index: {top_class_index}")

# Calculate and print the inference time per step
inference_time_ms = (end_time - start_time) * 1000.0
print(f"Inference Time: {inference_time_ms:.2f} ms")

# Model summary provides information about parameters and layers
#model.summary()

# Get the size of the model in megabytes
model_size_MB = model.count_params() * 4 / (1024 ** 2)  # 4 bytes for float32
print(f"Size (MB): {model_size_MB:.2f} MB")


# Get the number of parameters and depth from the model's layers
num_parameters = model.count_params()
model_depth = len(model.layers)

print(f"Parameters: {num_parameters}")
print(f"Depth: {model_depth}")
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/densenet/densenet121_weights_tf_dim_ordering_tf_kernels.h5
33188688/33188688 [==============================] - 11s 0us/step
1/1 [==============================] - 3s 3s/step
Predictions:
1: golden_retriever (0.87)
2: Labrador_retriever (0.11)
3: cocker_spaniel (0.00)
4: kuvasz (0.00)
5: Great_Pyrenees (0.00)
```

```
Top Prediction Class Index: 207
Inference Time: 3114.43 ms
Size (MB): 30.76 MB
Parameters: 8062504
Depth: 429
```

[42]:
```python
import numpy as np
from tensorflow.keras.applications import NASNetMobile
from tensorflow.keras.applications.nasnet import preprocess_input,␣
 ↪decode_predictions
from tensorflow.keras.preprocessing import image

# Load the NASNetMobile model pre-trained on ImageNet data
model = NASNetMobile(weights='imagenet')

# Load and preprocess the input image
img_path = r"C:\Users\srava\Downloads\Nassau_2697x1517.jpg"  # replace with the␣
 ↪path to your image file
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)

# Measure inference time
start_time = time.time()
predictions = model.predict(img_array)
end_time = time.time()

# Decode and print the top-3 predicted classes
decoded_predictions = decode_predictions(predictions, top=5)[0]

print("Predictions:")
for i, (imagenet_id, label, score) in enumerate(decoded_predictions):
    print(f"{i + 1}: {label} ({score:.2f})")

# Optionally, you can obtain the class index for the top prediction
top_class_index = np.argmax(predictions[0])
print(f"\nTop Prediction Class Index: {top_class_index}")

# Calculate and print the inference time per step
inference_time_ms = (end_time - start_time) * 1000.0
print(f"Inference Time: {inference_time_ms:.2f} ms")

# Model summary provides information about parameters and layers
#model.summary()

# Get the size of the model in megabytes
```

```
model_size_MB = model.count_params() * 4 / (1024 ** 2)  # 4 bytes for float32
print(f"Size (MB): {model_size_MB:.2f} MB")


# Get the number of parameters and depth from the model's layers
num_parameters = model.count_params()
model_depth = len(model.layers)

print(f"Parameters: {num_parameters}")
print(f"Depth: {model_depth}")
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/nasnet/NASNet-mobile.h5
24227760/24227760 [==============================] - 9s 0us/step
1/1 [==============================] - 6s 6s/step
Predictions:
1: golden_retriever (0.91)
2: Labrador_retriever (0.02)
3: redbone (0.01)
4: tennis_ball (0.00)
5: Irish_setter (0.00)

Top Prediction Class Index: 207
Inference Time: 6383.15 ms
Size (MB): 20.32 MB
Parameters: 5326716
Depth: 771
```

[45]:
```
import numpy as np
from tensorflow.keras.applications import NASNetLarge
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.nasnet import preprocess_input,␣
 ↪decode_predictions

# Load the NASNetLarge model pre-trained on ImageNet data
model = NASNetLarge(weights='imagenet')

# Load and preprocess the input image
img_path = r"C:\Users\srava\Downloads\Nassau_2697x1517.jpg"  # replace with the␣
 ↪path to your image file
img = image.load_img(img_path, target_size=(331, 331))  # NASNetLarge requires␣
 ↪input shape (331, 331)
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
img_array = preprocess_input(img_array)

# Measure inference time
```

```
start_time = time.time()
predictions = model.predict(img_array)
end_time = time.time()

# Decode and print the top-3 predicted classes
decoded_predictions = decode_predictions(predictions, top=5)[0]

print("Predictions:")
for i, (imagenet_id, label, score) in enumerate(decoded_predictions):
    print(f"{i + 1}: {label} ({score:.2f})")

# Optionally, you can obtain the class index for the top prediction
top_class_index = np.argmax(predictions[0])
print(f"\nTop Prediction Class Index: {top_class_index}")

# Calculate and print the inference time per step
inference_time_ms = (end_time - start_time) * 1000.0
print(f"Inference Time: {inference_time_ms:.2f} ms")

# Model summary provides information about parameters and layers
#model.summary()

# Get the size of the model in megabytes
model_size_MB = model.count_params() * 4 / (1024 ** 2)  # 4 bytes for float32
print(f"Size (MB): {model_size_MB:.2f} MB")


# Get the number of parameters and depth from the model's layers
num_parameters = model.count_params()
model_depth = len(model.layers)

print(f"Parameters: {num_parameters}")
print(f"Depth: {model_depth}")
```

```
1/1 [==============================] - 9s 9s/step
Predictions:
1: golden_retriever (0.91)
2: Labrador_retriever (0.01)
3: tennis_ball (0.00)
4: Airedale (0.00)
5: Irish_setter (0.00)

Top Prediction Class Index: 207
Inference Time: 9299.89 ms
Size (MB): 339.32 MB
Parameters: 88949818
Depth: 1041
```

[47]:
```
import numpy as np
from tensorflow.keras.applications import EfficientNetV2B0
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.efficientnet_v2 import preprocess_input,␣
 ↪decode_predictions

# Load the EfficientNetV2B0 model pre-trained on ImageNet data
model = EfficientNetV2B0(weights='imagenet')

# Load and preprocess the input image
img_path = r"C:\Users\srava\Downloads\Nassau_2697x1517.jpg"  # replace with the␣
 ↪path to your image file
img = image.load_img(img_path, target_size=(224, 224))  # NASNetLarge requires␣
 ↪input shape (331, 331)
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
img_array = preprocess_input(img_array)

# Measure inference time
start_time = time.time()
predictions = model.predict(img_array)
end_time = time.time()

# Decode and print the top-3 predicted classes
decoded_predictions = decode_predictions(predictions, top=5)[0]

print("Predictions:")
for i, (imagenet_id, label, score) in enumerate(decoded_predictions):
    print(f"{i + 1}: {label} ({score:.2f})")

# Optionally, you can obtain the class index for the top prediction
top_class_index = np.argmax(predictions[0])
print(f"\nTop Prediction Class Index: {top_class_index}")

# Calculate and print the inference time per step
inference_time_ms = (end_time - start_time) * 1000.0
print(f"Inference Time: {inference_time_ms:.2f} ms")

# Model summary provides information about parameters and layers
#model.summary()

# Get the size of the model in megabytes
model_size_MB = model.count_params() * 4 / (1024 ** 2)  # 4 bytes for float32
print(f"Size (MB): {model_size_MB:.2f} MB")


# Get the number of parameters and depth from the model's layers
```

```
num_parameters = model.count_params()
model_depth = len(model.layers)

print(f"Parameters: {num_parameters}")
print(f"Depth: {model_depth}")
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/efficientnet_v2/efficientnetv2-b0.h5
29403144/29403144 [==============================] - 11s 0us/step
1/1 [==============================] - 3s 3s/step
Predictions:
1: golden_retriever (0.92)
2: Labrador_retriever (0.01)
3: Tibetan_mastiff (0.00)
4: cocker_spaniel (0.00)
5: kuvasz (0.00)

Top Prediction Class Index: 207
Inference Time: 2635.39 ms
Size (MB): 27.47 MB
Parameters: 7200312
Depth: 273
```

[ ]: