Data Arguments

```
pip install --upgrade keras
```

```
Requirement already satisfied: keras in c:\users\admin\appdata\local\
programs\python\python310\lib\site-packages (3.10.0)
Requirement already satisfied: absl-py in c:\users\admin\appdata\
local\programs\python\python310\lib\site-packages (from keras) (2.2.2)
Requirement already satisfied: numpy in c:\users\admin\appdata\local\
programs\python\python310\lib\site-packages (from keras) (1.26.4)
Requirement already satisfied: rich in c:\users\admin\appdata\local\
programs\python\python310\lib\site-packages (from keras) (14.0.0)
Requirement already satisfied: namex in c:\users\admin\appdata\local\
programs\python\python310\lib\site-packages (from keras) (0.1.0)
Requirement already satisfied: h5py in c:\users\admin\appdata\local\
programs\python\python310\lib\site-packages (from keras) (3.14.0)
Requirement already satisfied: optree in c:\users\admin\appdata\local\
programs\python\python310\lib\site-packages (from keras) (0.16.0)
Requirement already satisfied: ml-dtypes in c:\users\admin\appdata\
local\programs\python\python310\lib\site-packages (from keras) (0.5.1)
Requirement already satisfied: packaging in c:\users\admin\appdata\
local\programs\python\python310\lib\site-packages (from keras) (24.2)
Requirement already satisfied: typing-extensions>=4.6.0 in c:\users\
admin\appdata\local\programs\python\python310\lib\site-packages (from
optree->keras) (4.14.0)
Requirement already satisfied: markdown-it-py>=2.2.0 in c:\users\
admin\appdata\local\programs\python\python310\lib\site-packages (from
rich->keras) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in c:\users\
admin\appdata\local\programs\python\python310\lib\site-packages (from
rich->keras) (2.19.1)
Requirement already satisfied: mdurl~=0.1 in c:\users\admin\appdata\
local\programs\python\python310\lib\site-packages (from markdown-it-
py>=2.2.0->rich->keras) (0.1.2)
Note: you may need to restart the kernel to use updated packages.
```

```python
from tensorflow import keras
```

```
pip show tensorflow
```

```
Name: tensorflow
Version: 2.19.0
Summary: TensorFlow is an open source machine learning framework for
everyone.
Home-page: https://www.tensorflow.org/
Author: Google Inc.
Author-email: packages@tensorflow.org
License: Apache 2.0
```

```
Location: c:\users\admin\appdata\local\programs\python\python310\lib\
site-packages
Requires: absl-py, astunparse, flatbuffers, gast, google-pasta,
grpcio, h5py, keras, libclang, ml-dtypes, numpy, opt-einsum,
packaging, protobuf, requests, setuptools, six, tensorboard,
tensorflow-io-gcs-filesystem, termcolor, typing-extensions, wrapt
Required-by: tf_keras
Note: you may need to restart the kernel to use updated packages.
```

```
pip show keras
```

```
Name: keras
Version: 3.10.0
Summary: Multi-backend Keras
Home-page:
Author:
Author-email: Keras team <keras-users@googlegroups.com>
License: Apache License 2.0
Location: c:\users\admin\appdata\local\programs\python\python310\lib\
site-packages
Requires: absl-py, h5py, ml-dtypes, namex, numpy, optree, packaging,
rich
Required-by: tensorflow
Note: you may need to restart the kernel to use updated packages.
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator,
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')
```

```
import tensorflow.keras as keras
```

```
img = load_img(r"D:\FSDS\Tensorflow_Keras\butterfly.jpg")
img
```

```
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"D:\FSDS\Tensorflow_Keras\data
arguments",save_prefix='butterfly',save_format='jpeg'):
    i += 1
    if i > 20:
        break # otherwise the generator would loop indefinitely

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"D:\FSDS\Tensorflow_Keras\New-Mahindra-Thar-Front-grey.jpg")

img
```

```
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"D:\FSDS\Tensorflow_Keras\data arguments",save_prefix='butterfly',save_format='jpeg'):
    i += 1
    if i > 20:
        break # otherwise the generator would loop indefinitely
```

Resnet 50

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

```
<Functional name=resnet50, built=True>
```

1.Basic Predictions of the Image

2. Resnet50 model

```
from keras.utils import array_to_img, img_to_array, load_img
img = load_img(r"D:\FSDS\Tensorflow_Keras\butterfly.jpg")
img
```

```
import numpy as np
from tensorflow.keras.applications.resnet50 import ResNet50,
preprocess_input, decode_predictions
from tensorflow.keras.preprocessing import image

# Load the ResNet50 model and make predictions on an image
model = ResNet50(weights='imagenet')
```

```
# Load and preprocess the input image
img_path = r"D:\FSDS\Tensorflow_Keras\butterfly.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)

# Make predictions
predictions = model.predict(img_array)

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

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

 # optionally, you can also print the top class index for the
predictions
top_class_index = np.argmax(predictions[0])
print(f"\nTop class index: {top_class_index}")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
Predicted:
1: monarch (0.99)
2: admiral (0.00)
3: sulphur_butterfly (0.00)

Top class index: 323
```

ResNet50V2 Model

```
import numpy as np
from tensorflow.keras.applications.resnet_v2 import ResNet50V2,
preprocess_input, decode_predictions
from tensorflow.keras.preprocessing import image

# Load the ResNet50 model and make predictions on an image
model = ResNet50V2(weights='imagenet')

# Load and preprocess the input image
img_path = r"D:\FSDS\Tensorflow_Keras\husky.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)

# Make predictions
predictions = model.predict(img_array)
```

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

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

 # optionally, you can also print the top class index for the
predictions
top_class_index = np.argmax(predictions[0])
print(f"\nTop class index: {top_class_index}")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 3s 3s/step
Predicted:
1: Eskimo_dog (0.60)
2: Siberian_husky (0.40)
3: malamute (0.00)

Top class index: 248
```

VGG16 Model

```
import numpy as np
from tensorflow.keras.applications.vgg16 import VGG16,
preprocess_input, decode_predictions
from tensorflow.keras.preprocessing import image

# Load the ResNet50 model and make predictions on an image
model = VGG16(weights='imagenet')

# Load and preprocess the input image
img_path = r"D:\FSDS\Tensorflow_Keras\2560px-A-Cat.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)

# Make predictions
predictions = model.predict(img_array)

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

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

 # optionally, you can also print the top class index for the
predictions
```

```
top_class_index = np.argmax(predictions[0])
print(f"\nTop class index: {top_class_index}")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 655ms/step
Predicted:
1: tiger_cat (0.49)
2: tabby (0.25)
3: Egyptian_cat (0.18)

Top class index: 282
```

VGG19 Model

```
from keras.utils import array_to_img, img_to_array, load_img
img= load_img(r"D:\FSDS\Tensorflow_Keras\dog.jpg")
img
```

```
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"D:\FSDS\Tensorflow_Keras\dog.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)

# 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
Predictions:
1: golden_retriever (0.52)
2: Great_Pyrenees (0.22)
3: kuvasz (0.15)

Top Prediction Class Index: 207

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"D:\FSDS\Tensorflow_Keras\h.jpg" # replace with thepath 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 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
Predictions:
1: golden_retriever (0.91)
2: Labrador_retriever (0.02)
3: kuvasz (0.01)
4: redbone (0.01)
5: tennis_ball (0.01)
6: flat-coated_retriever (0.00)
7: curly-coated_retriever (0.00)
8: Chesapeake_Bay_retriever (0.00)

Top Prediction Class Index: 207
```

```
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"D:\FSDS\Tensorflow_Keras\h.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 ━━━━━━━━━━━━━━━━━━━━ 4s 4s/step
Predictions:
```

```
1: golden_retriever (1.00)
2: Labrador_retriever (0.00)
3: kuvasz (0.00)
4: Tibetan_mastiff (0.00)
5: Great_Pyrenees (0.00)
6: Brittany_spaniel (0.00)
7: flat-coated_retriever (0.00)
8: Chesapeake_Bay_retriever (0.00)
9: tennis_ball (0.00)
10: Leonberg (0.00)

Top Prediction Class Index: 207
```

predictions using predictions using keras Api Applications

```
from keras.utils import array_to_img ,img_to_array, load_img
img = load_img("D:\FSDS\Tensorflow_Keras\h.jpg" )
img
```

```
import time
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"D:\FSDS\Tensorflow_Keras\h.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 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
Predictions:
1: golden_retriever (0.98)
2: Labrador_retriever (0.01)
3: tennis_ball (0.00)
4: Brittany_spaniel (0.00)
5: soccer_ball (0.00)

Top Prediction Class Index: 207
Inference Time: 2455.22 ms
Size (MB): 97.80 MB
Parameters: 25636712
Depth: 177
```

```python
import time
import numpy as np
from tensorflow.keras.applications.resnet_v2 import ResNet50V2,
preprocess_input, decode_predictions
from tensorflow.keras.preprocessing import image

 # Load the ResNet-50 model pre-trained on ImageNet data
model = ResNet50V2(weights='imagenet')
 # Load and preprocess the input image
img_path = r"D:\FSDS\Tensorflow_Keras\h.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 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
Predictions:
1: golden_retriever (1.00)
2: Labrador_retriever (0.00)
3: kuvasz (0.00)
4: Tibetan_mastiff (0.00)
5: Great_Pyrenees (0.00)

Top Prediction Class Index: 207
Inference Time: 2169.36 ms
Size (MB): 97.71 MB
Parameters: 25613800
Depth: 192
```

```
import time
import numpy as np
from tensorflow.keras.applications.vgg19 import VGG19,
preprocess_input, decode_predictions
from tensorflow.keras.preprocessing import image

 # Load the ResNet-50 model pre-trained on ImageNet data
model = VGG19(weights='imagenet')

 # Load and preprocess the input image
img_path = r"D:\FSDS\Tensorflow_Keras\h.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 740ms/step
Predictions:
1: golden_retriever (0.91)
2: Labrador_retriever (0.02)
3: kuvasz (0.01)
4: redbone (0.01)
5: tennis_ball (0.01)

Top Prediction Class Index: 207
Inference Time: 829.94 ms
Size (MB): 548.05 MB
Parameters: 143667240
Depth: 26

import time
import numpy as np
from tensorflow.keras.applications.vgg16 import VGG16,
preprocess_input, decode_predictions
from tensorflow.keras.preprocessing import image

 # Load the ResNet-50 model pre-trained on ImageNet data
model = VGG16(weights='imagenet')
```

```python
 # Load and preprocess the input image
img_path = r"D:\FSDS\Tensorflow_Keras\h.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 629ms/step
Predictions:
1: golden_retriever (0.96)
2: Labrador_retriever (0.01)
3: tennis_ball (0.01)
4: kuvasz (0.00)
5: flat-coated_retriever (0.00)

Top Prediction Class Index: 207
```

```
Inference Time: 844.49 ms
Size (MB): 527.79 MB
Parameters: 138357544
Depth: 23
```

```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"D:\FSDS\Tensorflow_Keras\h.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
```

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

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

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
Predictions:
1: golden_retriever (0.99)
2: tennis_ball (0.00)
3: Brittany_spaniel (0.00)
4: Labrador_retriever (0.00)
5: Irish_setter (0.00)

Top Prediction Class Index: 207
Inference Time: 1714.25 ms
Size (MB): 87.40 MB
Parameters: 22910480
Depth: 134
```

```python
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"D:\FSDS\Tensorflow_Keras\h.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}")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 3s 3s/step
Predictions:
1: golden_retriever (0.99)
2: tennis_ball (0.00)
3: flat-coated_retriever (0.00)
4: kuvasz (0.00)
5: Irish_setter (0.00)

Top Prediction Class Index: 207
Inference Time: 2769.18 ms
Size (MB): 90.99 MB
Parameters: 23851784
Depth: 313
```

```
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"D:\FSDS\Tensorflow_Keras\h.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)

 # 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 (0.00)
3: tennis_ball (0.00)
4: Tibetan_mastiff (0.00)
5: Chesapeake_Bay_retriever (0.00)

Top Prediction Class Index: 207
Inference Time: 1335.85 ms
Size (MB): 13.50 MB
Parameters: 3538984
Depth: 156
```

```
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"D:\FSDS\Tensorflow_Keras\h.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 ━━━━━━━━━━━━━━━━━━━━ 9s 0us/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 6s 6s/step
Predictions:
1: golden_retriever (0.99)
2: Labrador_retriever (0.00)
3: Brittany_spaniel (0.00)
4: kuvasz (0.00)
5: Irish_setter (0.00)

Top Prediction Class Index: 207
Inference Time: 6191.55 ms
Size (MB): 30.76 MB
Parameters: 8062504
Depth: 429

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"D:\FSDS\Tensorflow_Keras\h.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 ━━━━━━━━━━━━━━━━━━━━ 7s 0us/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 6s 6s/step
Predictions:
1: golden_retriever (0.93)
2: tennis_ball (0.01)
3: Irish_setter (0.01)
4: Labrador_retriever (0.00)
5: flat-coated_retriever (0.00)

Top Prediction Class Index: 207
Inference Time: 6439.96 ms
Size (MB): 20.32 MB
Parameters: 5326716
Depth: 771

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"D:\FSDS\Tensorflow_Keras\h.jpg" # replace with the path
to your image file

img = image.load_img(img_path, target_size=(331, 331)) # NASNetLarge
requiresinput 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/nasnet/NASNet-large.h5
359748576/359748576 ━━━━━━━━━━━━━━━━━━━━ 95s 0us/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 9s 9s/step
Predictions:
1: golden_retriever (0.89)
2: tennis_ball (0.01)
3: Labrador_retriever (0.00)
4: Irish_setter (0.00)
5: Pembroke (0.00)

Top Prediction Class Index: 207
```

```
Inference Time: 8789.68 ms
Size (MB): 339.32 MB
Parameters: 88949818
Depth: 1041
```

```python
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"D:\FSDS\Tensorflow_Keras\h.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 ━━━━━━━━━━━━━━━━━━━━ 9s 0us/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
Predictions:
1: golden_retriever (0.95)
2: Labrador_retriever (0.00)
3: tennis_ball (0.00)
4: Brittany_spaniel (0.00)
5: Tibetan_mastiff (0.00)

Top Prediction Class Index: 207
Inference Time: 2236.97 ms
Size (MB): 27.47 MB
Parameters: 7200312
Depth: 273
```