

```

1 !pip install tensorflow
2 !pip install matplotlib
3 !pip install ipywidgets

```

```

Requirement already satisfied: tensorflow in /usr/local/lib/python3.12/dist-packages (2.19.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (25.2.10)
Requirement already satisfied: gast!=0.5.0,!0.5.1,!0.5.2,>=0.2.1 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (3.4.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.12/dist-packages (from tensorflow) (25.0)
Requirement already satisfied: protobuf!=4.21.0,!4.21.1,!4.21.2,!4.21.3,!4.21.4,!4.21.5,<6.0.0dev,>=3.20.3 in /usr/local/lib
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (2.32.4)
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from tensorflow) (75.2.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (1.17.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (3.1.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (4.15.0)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (1.17.3)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (1.75.0)
Requirement already satisfied: tensorboard~2.19.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (2.19.0)
Requirement already satisfied: keras>=3.5.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (3.10.0)
Requirement already satisfied: numpy<2.2.0,>=1.26.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (2.0.2)
Requirement already satisfied: h5py>=3.11.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (3.14.0)
Requirement already satisfied: ml-dtypes<1.0.0,>=0.5.1 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (0.5.3)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.12/dist-packages (from astunparse>=1.6.0->tensorflow)
Requirement already satisfied: rich in /usr/local/lib/python3.12/dist-packages (from keras>=3.5.0->tensorflow) (13.9.4)
Requirement already satisfied: namex in /usr/local/lib/python3.12/dist-packages (from keras>=3.5.0->tensorflow) (0.1.0)
Requirement already satisfied: optree in /usr/local/lib/python3.12/dist-packages (from keras>=3.5.0->tensorflow) (0.17.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.21.0->ten
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.21.0->tensorflow) (3.
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.21.0->tensoflo
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.21.0->tensoflo
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.12/dist-packages (from tensorboard~2.19.0->tensorflow)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.12/dist-packages (from tensorboard
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from tensorboard~2.19.0->tensorflow)
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.12/dist-packages (from werkzeug>=1.0.1->tensorboard~2
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.12/dist-packages (from rich->keras>=3.5.0->tensof
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.12/dist-packages (from rich->keras>=3.5.0->tenso
Requirement already satisfied: mdurl~0.1 in /usr/local/lib/python3.12/dist-packages (from markdown-it-py>=2.2.0->rich->keras>=3.
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (3.10.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (4.60.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.4.9)
Requirement already satisfied: numpy>=1.23 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (2.0.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (25.0)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (3.2.4)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.7->matplotlib) (1.17.
Requirement already satisfied: ipywidgets in /usr/local/lib/python3.12/dist-packages (7.7.1)
Requirement already satisfied: ipykernel>=4.5.1 in /usr/local/lib/python3.12/dist-packages (from ipywidgets) (6.17.1)
Requirement already satisfied: ipython-genutils~0.2.0 in /usr/local/lib/python3.12/dist-packages (from ipywidgets) (0.2.0)
Requirement already satisfied: traitlets>=4.3.1 in /usr/local/lib/python3.12/dist-packages (from ipywidgets) (5.7.1)
Requirement already satisfied: widgetsnbextension~3.6.0 in /usr/local/lib/python3.12/dist-packages (from ipywidgets) (3.6.10)
Requirement already satisfied: ipython>=4.0.0 in /usr/local/lib/python3.12/dist-packages (from ipywidgets) (7.34.0)
Requirement already satisfied: jupyterlab-widgets>=1.0.0 in /usr/local/lib/python3.12/dist-packages (from ipywidgets) (3.0.15)
Requirement already satisfied: debugpy>=1.0 in /usr/local/lib/python3.12/dist-packages (from ipykernel>=4.5.1->ipywidgets) (1.8.1
Requirement already satisfied: jupyter-client>=6.1.12 in /usr/local/lib/python3.12/dist-packages (from ipykernel>=4.5.1->ipywidg
Requirement already satisfied: matplotlib-inline>=0.1 in /usr/local/lib/python3.12/dist-packages (from ipykernel>=4.5.1->ipywidg

```

```

1 # Import necessary libraries
2 from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input, decode_predictions
3 from tensorflow.keras.preprocessing.image import load_img, img_to_array
4 from IPython.display import display
5 import numpy as np
6 import matplotlib.pyplot as plt
7 from PIL import Image, ImageFilter
8 import io
9 import ipywidgets as widgets
10

```

✧ Introduction

This lab is designed to introduce you to the basics of deep learning by interacting with a pre-built model. You'll understand the workflow of a deep learning project, including data preprocessing, model architecture, and making predictions. The goal is to familiarize yourself with the basics of deep learning without writing any code.

```
1 # Load the VGG16 model
2 model = VGG16(weights='imagenet')
3
4 # Display the model architecture
5 model.summary()
6
```

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels.h5
553467096/553467096 ————— 2s 0us/step

Model: "vgg16"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1,792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36,928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73,856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147,584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295,168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590,080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590,080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1,180,160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	(None, 4096)	102,764,544
fc2 (Dense)	(None, 4096)	16,781,312
predictions (Dense)	(None, 1000)	4,097,000

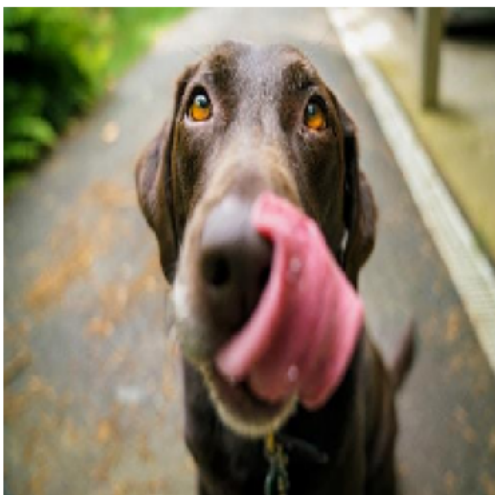
```
1 import requests
2 from PIL import Image
3 import io
4
5 # ☒ Use a direct JPEG image link
6 url = "https://images.unsplash.com/photo-1518717758536-85ae29035b6d?w=500" # Dog photo from Unsplash
7 response = requests.get(url, stream=True)
8 img = Image.open(io.BytesIO(response.content)).convert("RGB") # ensure RGB
9 img.save("sample.jpg") # Save locally
10
11 print("Image downloaded and saved as sample.jpg")
12
```

Image downloaded and saved as sample.jpg

```

1 sample_image, processed_image = load_and_preprocess_image("sample.jpg")
2
3 # Display
4 plt.imshow(sample_image)
5 plt.axis("off")
6 plt.show()
7

```



```

1 # Make predictions
2 predictions = model.predict(processed_image)
3
4 # Decode and print the predictions
5 decoded_predictions = decode_predictions(predictions, top=3)[0]
6 print(decoded_predictions)
7

```

1/1 ————— 1s 904ms/step

Downloading data from https://storage.googleapis.com/download.tensorflow.org/data/imagenet_class_index.json

35363/35363 ————— 0s 0us/step

[('n02089078', 'black-and-tan-coonhound', np.float32(0.58478373)), ('n02107142', 'Doberman', np.float32(0.1641687)), ('n02092339',

```

1 # Upload button to load images
2 upload = widgets.FileUpload()
3 display(upload)
4
5 # Button to make predictions
6 predict_button = widgets.Button(description="Make Prediction")
7 display(predict_button)
8
9 # Function to handle button click
10 def on_click(change):
11     # Check if a file has been uploaded
12     if upload.value:
13         img_data = list(upload.value.values())[0]['content']
14         img = Image.open(io.BytesIO(img_data))
15         img = img.resize((224, 224))
16
17         # Preprocess and predict
18         img_array = img_to_array(img)
19         img_array = np.expand_dims(img_array, axis=0)
20         img_array = preprocess_input(img_array)
21         predictions = model.predict(img_array)
22         decoded_predictions = decode_predictions(predictions, top=3)[0]
23
24         # Display predictions
25         print(decoded_predictions)
26     else:
27         print("Please upload an image first.")
28
29
30 predict_button.on_click(on_click)

```

Upload (5)

Make Prediction

```

1/1 _____ 1s 584ms/step
[('n02123159', 'tiger_cat', np.float32(0.7304771)), ('n02123045', 'tabby', np.float32(0.18864565)), ('n02124075', 'Egyptian_cat', n
1/1 _____ 1s 602ms/step
[('n02085936', 'Maltese_dog', np.float32(0.17133066)), ('n02328150', 'Angora', np.float32(0.1558474)), ('n02326432', 'hare', np.flo
1/1 _____ 1s 605ms/step
[('n02124075', 'Egyptian_cat', np.float32(0.4743577)), ('n02123159', 'tiger_cat', np.float32(0.38248402)), ('n02123045', 'tabby', n
1/1 _____ 1s 755ms/step
[('n02085936', 'Maltese_dog', np.float32(0.9040568)), ('n02113624', 'toy_poodle', np.float32(0.043650363)), ('n02098413', 'Lhasa',
1/1 _____ 1s 996ms/step
[('n02092339', 'Weimaraner', np.float32(0.96446556)), ('n02091032', 'Italian_greyhound', np.float32(0.012432499)), ('n02109047', 'G

```

✓ Conclusion and Discussion

Reflect on the lab activities. Discuss how the pre-trained model was able to make predictions, the role of data preprocessing, and the impact of input modifications on the model's predictions.

In this lab, we used a pre-trained VGG16 model to make predictions on our own images without writing training code. Because VGG16 was trained on the large ImageNet dataset, it already “knows” many visual patterns (edges, textures, shapes). When we upload a new picture, the model passes it through many convolution layers, builds a high-level feature representation, and then uses a final classifier to output class probabilities. We saw this as top-k predictions with confidence scores.

Data preprocessing was essential. The model expects inputs that match its training setup: RGB format, 224×224 size, and ImageNet normalization via `preprocess_input`. When we center-cropped and then resized, predictions were usually more stable because we avoided stretching the image. Adding the batch dimension (`np.expand_dims`) and using the same normalization made sure the distribution of our inputs was consistent with what VGG16 saw during training. In short, good preprocessing reduced errors and improved confidence.

We also explored how input modifications change predictions. Small rotations or crops sometimes shifted the top label or lowered confidence, showing that VGG16 is not perfectly rotation-invariant and is sensitive to what part of the object is visible. Noise and blur removed fine details the model relies on, which often reduced confidence or caused confusion between similar classes (e.g., cat breeds or dog breeds). Changes in brightness/contrast could also hurt results if they pushed the image away from the training distribution. These behaviors are common for pre-trained models that were not fine-tuned with heavy data augmentation for robustness.

Overall, the lab demonstrated three key ideas: (1) pre-trained models can deliver useful predictions out-of-the-box, (2) correct preprocessing is critical, and (3) even strong models are sensitive to input quality and viewpoint—an important lesson when preparing data for real-world deployment.

1 Start coding or [generate](#) with AI.