Group Members: Morathi Mnkandla, Ali Shan Ahad

Class: Deep Learning ITAI 2376
October 3 2025

L02 Exploring Deep Learning Tools: A No-Code Introduction to TensorFlow and Keras

1. Introduction

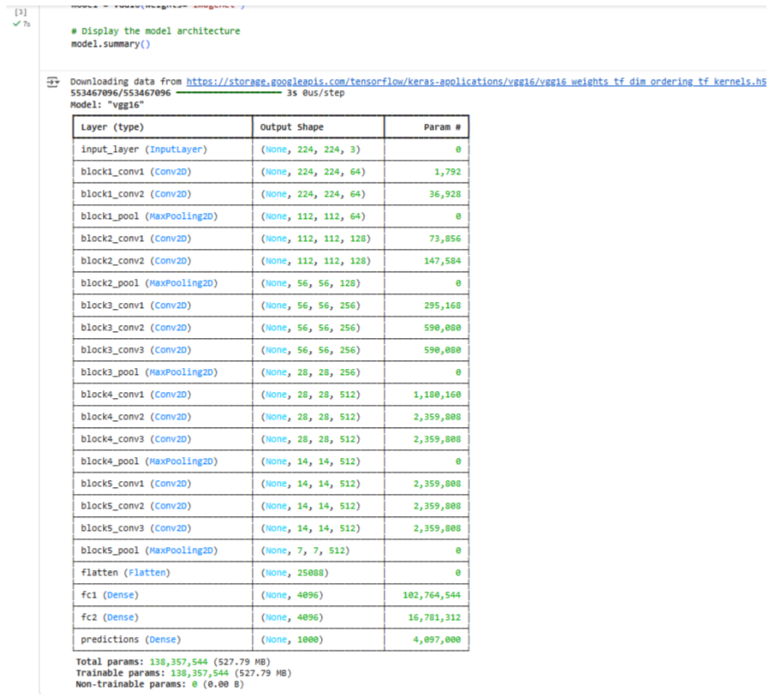
Task: Read the introduction provided in the markdown cell.
Objective: Understand the goals of the lab and the basics of deep learning.

The lab aimed to introduce the workflow of using a pre-trained deep learning model, specifically VGG16, to make predictions on images. The focus was on understanding how models process data, make predictions, and react to variations in input.

2. Overview of the Pre-built Model

Task: Interact with the pre-loaded code cell to load the VGG16 model and view its architecture.
Objective: Familiarize yourself with the components of a deep learning model.

The VGG16 model was loaded with pretrained weights from ImageNet. Its architecture includes multiple convolutional layers, pooling layers, and fully connected layers, designed for image classification tasks. Observing the model structure helped understand how deep neural networks extract features and make predictions.



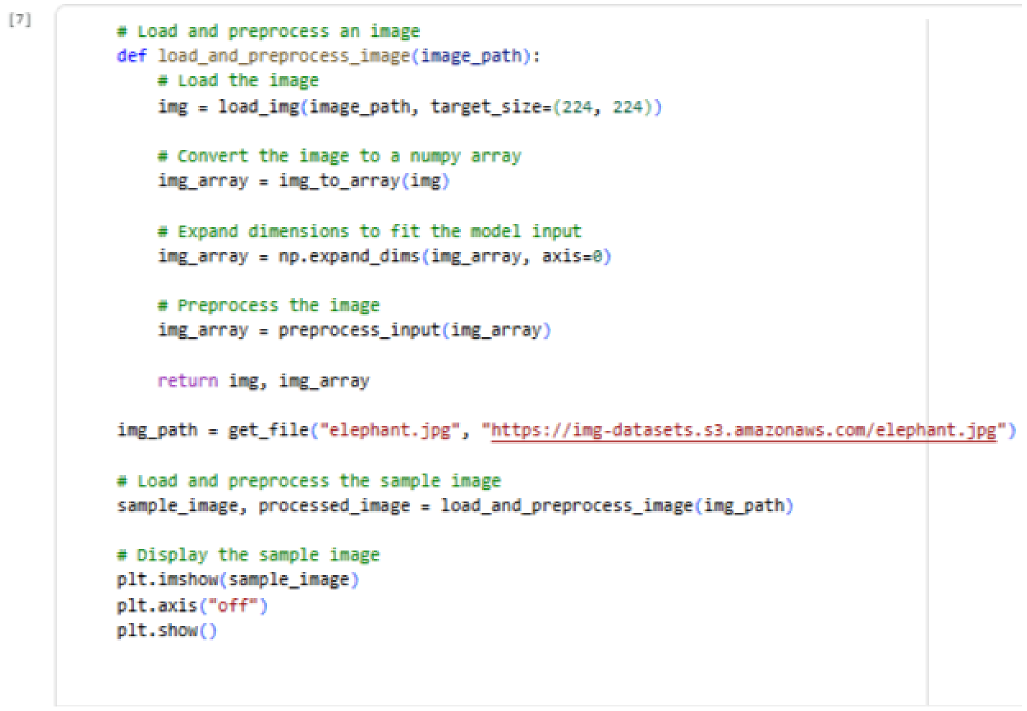
```
# Display the model architecture
model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels.h5
553467096/553467096 ━━━━━━━━━━━━━━━━━━━━ 3s 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 │
└─────────────────────────────┴───────────────────────┴─────────────┘
 Total params: 138,357,544 (527.79 MB)
 Trainable params: 138,357,544 (527.79 MB)
 Non-trainable params: 0 (0.00 B)
```

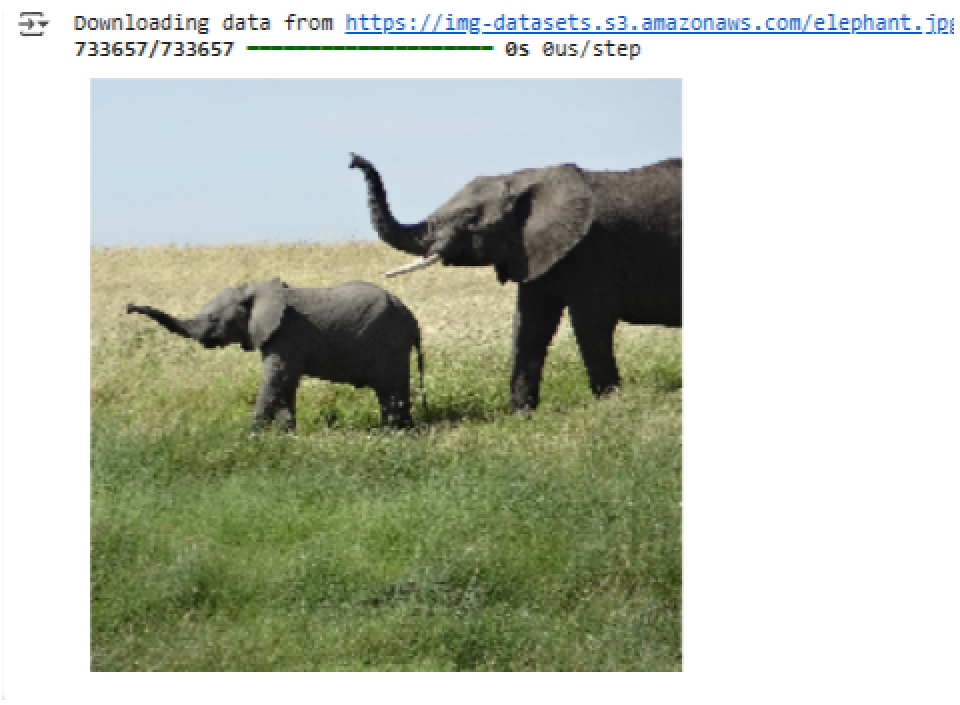
3. Data Loading and Preprocessing

Task: Observe the code and outputs in the read-only code cell that demonstrates data loading and preprocessing.
Objective: Understand the importance of and methods for preparing data for a deep learning model.

Initially, attempting to load `sample.jpg` caused a `FileNotFoundError`. This highlighted the necessity of correct input data. To resolve this, a test image (elephant) was used. Preprocessing steps included resizing the image to 224×224 pixels, converting it to a numerical array, expanding dimensions to simulate a batch, and normalizing pixel values.



```
[7]
# Load and preprocess an image
def load_and_preprocess_image(image_path):
    # Load the image
    img = load_img(image_path, target_size=(224, 224))

    # Convert the image to a numpy array
    img_array = img_to_array(img)

    # Expand dimensions to fit the model input
    img_array = np.expand_dims(img_array, axis=0)

    # Preprocess the image
    img_array = preprocess_input(img_array)

    return img, img_array

img_path = get_file("elephant.jpg", "https://img-datasets.s3.amazonaws.com/elephant.jpg")

# Load and preprocess the sample image
sample_image, processed_image = load_and_preprocess_image(img_path)

# Display the sample image
plt.imshow(sample_image)
plt.axis("off")
plt.show()
```

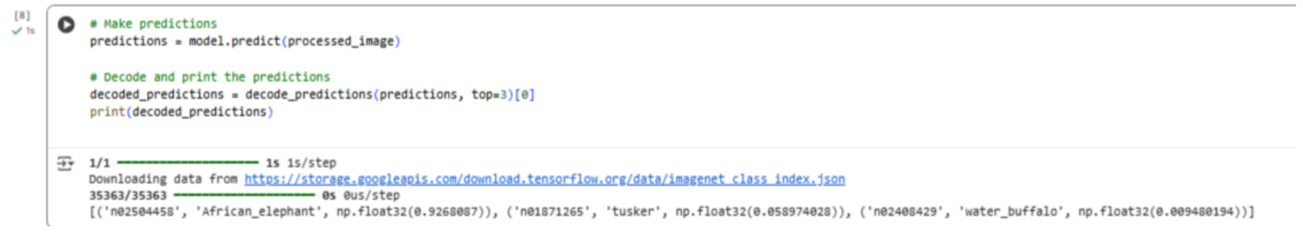


```
Downloading data from https://img-datasets.s3.amazonaws.com/elephant.jp
733657/733657 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

4. Making Predictions

Task: Upload an image using the interactive widget and observe the model making predictions on the uploaded image.
Objective: Learn how a pre-trained model can be used to make predictions on new data.

After preprocessing, the image was passed into the model. VGG16 returned probabilities for multiple object categories. For the elephant image, the top prediction was “African elephant” with high confidence, followed by “Tusker” and “Water buffalo” with lower probabilities.



```
[8]
# Make predictions
predictions = model.predict(processed_image)

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

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step
Downloading data from https://storage.googleapis.com/download.tensorflow.org/data/imagenet_class_index.json
35363/35363 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
[('n02504458', 'African_elephant', np.float32(0.9268087)), ('n01871265', 'tusker', np.float32(0.058974028)), ('n02408429', 'water_buffalo', np.float32(0.009480194))]
```

5. Understanding Predictions

Task: Use interactive widgets to see how slight modifications to the input image (like rotation or adding noise) affect the model's predictions.
Objective: Gain insights into the model's behavior and its sensitivity to input data changes.

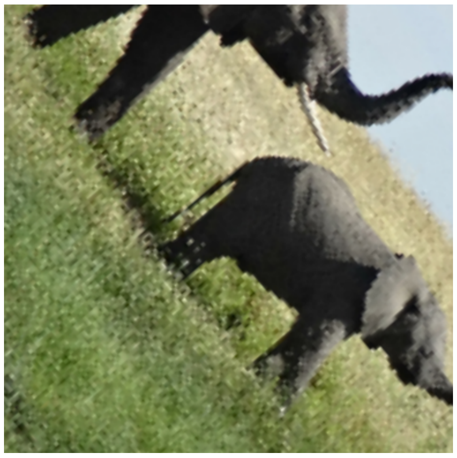
Observation: After **rotating and enlarging** the original image, the model made the following predictions:

```
[('n02437616', 'llama', 0.091),

 ('n02113799', 'standard_poodle', 0.072),

 ('n02488702', 'colobus', 0.060)]
```

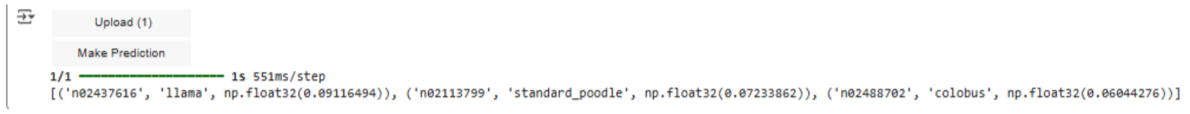
Interpretation:

- The model's **top prediction** changed from the original image (African elephant) to **llama**, but with a **low confidence of ~9%**.
- The probabilities for the other classes were similarly low, indicating **uncertainty**.
- This shows that modifying the input image (rotation and enlargement) can significantly impact the model's prediction and reduce confidence, even causing it to misclassify the object.

Observation: Pre-trained models are **sensitive to large changes** in input, and transformations like rotation or scaling can confuse the model if the altered image deviates too much from the training data distribution.

: 

Modifying the image changed the model’s confidence levels or altered the top predicted class. This demonstrated the model’s reliance on clear, correctly formatted input and highlighted the sensitivity of deep learning models to variations in data.



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 551ms/step
[('n02437616', 'llama', np.float32(0.09116494)), ('n02113799', 'standard_poodle', np.float32(0.07233862)), ('n02488702', 'colobus', np.float32(0.06044276))]
```

6. Conclusion and Discussion

In this lab, we explored how a pre-trained **VGG16 model** predicts objects in images. Initially, using a **sample image from Keras**, the model correctly identified an **African elephant** with high confidence (~93%), demonstrating its ability to leverage learned features from ImageNet. When we **modified the input image** by rotating and enlarging it, the model’s confidence dropped, and the top prediction shifted to **llama** with only ~9% confidence.

This demonstrates that:

- **Pre-trained models** perform well on standard, unaltered images.
- **Data preprocessing** is crucial for consistent predictions.
- The model is **sensitive to input changes**, such as rotation or scaling, which can significantly affect accuracy.

Overall, the lab helped us understand how deep learning models interpret images, the importance of preprocessing, and the impact of input modifications on predictions.

7. Feedback and Assessment

Feedback:

- The lab was effective in showing how **pre-trained models from Keras** can be used without training from scratch.
- Tasks like uploading images and observing predictions made the lab **interactive and insightful**.
- A minor issue was the initial **sample image not being found**, but using the Keras image resolved this smoothly.

Assessment of Understanding:

- We now understand how **VGG16 extracts features** to make predictions.
- We can explain the role of **preprocessing** and why image modifications affect results.
- We learned to **interpret prediction outputs**, including class labels and probabilities, and relate them to model confidence.