Deep Learning Laboratory Experience with VGG16

A Reflective Journal

Author Note

This reflective journal was prepared by Varit Kobutra as a Lab 02 assignment submission for ITAI 2376 - Deep Learning in Artificial Intelligence (CRN: 19519) at Houston Community College. The author is currently enrolled in the Associates in Artificial Intelligence Program (Student ID: W216632608). This work was submitted to Professor Patricia McManus on January 20, 2025.

Table of Contents

Reflective Journal: Deep Learning Laboratory Experience with VGG16 .... *3*

Initial Setup and Environment *3*

Technical Learning Experience *3*

Model Architecture 3

Data Preprocessing 4

Testing and Results 5

Professional Development Implications 6

References *7*

Reflective Journal: Deep Learning Laboratory Experience with VGG16

This reflective journal documents my experience with the deep learning laboratory exercise, focusing on implementing and testing the *VGG16* model for image classification using *Google Colab*. Through this hands-on experience, I gained valuable insights into both the technical aspects of deep learning and its practical applications in computer vision.

Initial Setup and Environment

The laboratory began with setting up the Google Colab environment, which proved to be an excellent choice due to its pre-installed deep learning packages and GPU support. The process taught me the importance of proper environment configuration, particularly when working with resource-intensive deep learning models. I learned that even before working with AI models, careful consideration must be given to package dependencies and version compatibility.

Technical Learning Experience

Model Architecture

One of the most significant learning outcomes was understanding the VGG16 model's architecture. The model summary revealed its complexity with multiple convolutional layers, pooling layers, and dense layers, demonstrating the sophisticated nature of deep learning models

```
[3] # Load the VGG16 model
    model = VGG16(weights='imagenet')

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

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim
553467096/553467096 ━━━━━━━━━━━━━━━━━━━━ 14s 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)
```

Figure 1: Model architecture as displayed by invoking model.summary()

Data Preprocessing

The practical implementation of image preprocessing was particularly enlightening. I observed how raw images need to be:

- Resized to 224x224 pixels

- Converted to arrays
- Normalized using `preprocess_input`
- Expanded in dimensions to match the model's input requirements

```
[5] # 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

    # Load and preprocess a sample image
    sample_image, processed_image = load_and_preprocess_image('dog.jpg')

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

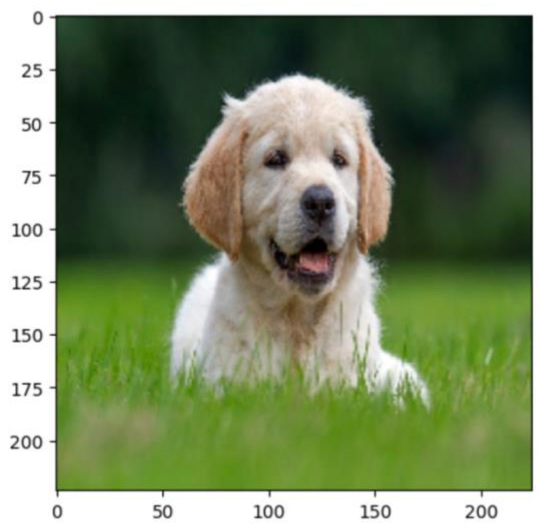

Figure 2: Data preprocessing, using sample image of a dog from Wikipedia

Testing and Results

The model demonstrated impressive accuracy in classifying a dog image as a Golden Retriever with high confidence (approximately 85% probability).

This experience helped me understand the real-world applications of pre-trained models and transfer learning in computer vision tasks.

```
[7] # Upload button to load images
    upload = widgets.FileUpload()
    display(upload)

    # Button to make predictions
    predict_button = widgets.Button(description="Make Prediction")
    display(predict_button)

    # Function to handle button click
    def on_click(change):
        img_data = list(upload.value.values())[0]['content']
        img = Image.open(io.BytesIO(img_data))
        img = img.resize((224, 224))

        # Preprocess and predict
        img_array = img_to_array(img)
        img_array = np.expand_dims(img_array, axis=0)
        img_array = preprocess_input(img_array)
        predictions = model.predict(img_array)
        decoded_predictions = decode_predictions(predictions, top=3)[0]

        # Display predictions
        print(decoded_predictions)

    predict_button.on_click(on_click)
```

pretty good prediction

```
Upload (1)
Make Prediction
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 617ms/step
[('n02099601', 'golden_retriever', 0.84284985), ('n02111500', 'Great_Pyrenees', 0.038634963), ('n02108551', '
```

Figure 3: Prediction using interactive prediction function with accuracy of almost 85%

Professional Development Implications

This laboratory experience has significant implications for my understanding of deep learning applications. It has demonstrated how pre-trained models can be effectively utilized for practical tasks, providing me with hands-on experience that aligns with industry practices.

References

Matio, H. (2020, October 16). *Dog Breeds* [Photograph]. Wikimedia Commons. Retrieved January 20, 2025, from Wikimedia Commons.