S. B. JAIN INSTITUTE OF TECHNOLOGY, MANAGEMENT & RESEARCH, NAGPUR.

Practical No. 10

Aim: Apply the knowledge to study and implement VGG16 in deep learning.

Name of Student	: Shrutika Pradeep Bagdi
Roll No	**:** CS22130
Semester/Year	**:** VIIth Sem / IVth Year
Academic Session	**:** 2025-2026 [ODD]
Date of Performance	: ______________________________
Date of Submission	**:** ______________________________

AIM: Apply the knowledge to study and implement VGG16 in deep learning.

OBJECTIVE/EXPECTED LEARNING OUTCOME:

The objectives and expected learning outcome of this practical are:

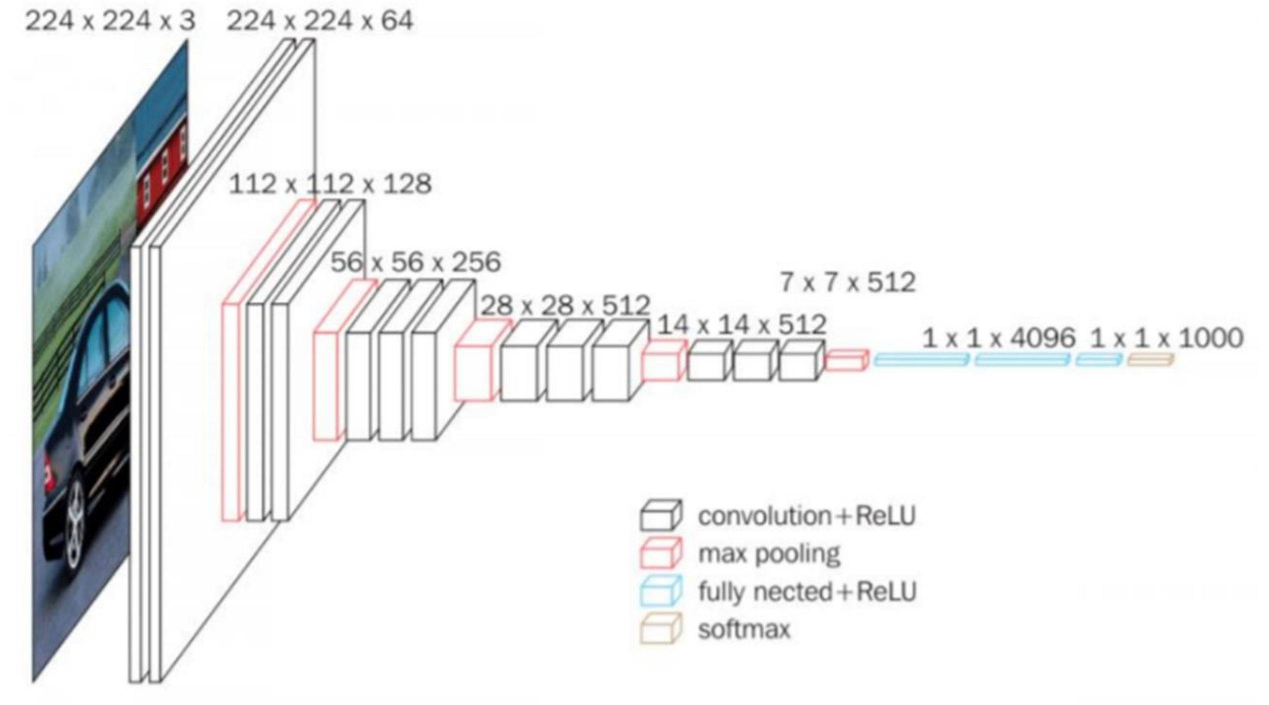
- To understand the architecture and working principles of VGGNet/VGG16.
- To learn how deep convolutional neural networks (CNNs) extract hierarchical image features.
- To implement the VGG16 model using a deep learning framework such as TensorFlow or Keras.
- To perform transfer learning using a pre-trained VGG16 model on a custom dataset.
- To evaluate model performance using metrics such as accuracy and loss.

THEORY:

VGG- Network is a convolutional neural network model proposed by K. Simonyan and A. Zisserman in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition" [1]. This architecture achieved top-5 test accuracy of 92.7% in ImageNet, which has over 14 million images belonging to 1000 classes.
It is one of the famous architectures in the deep learning field. Replacing large kernel-sized filters with 11 and 5 in the first and second layer respectively showed the improvement over AlexNet architecture, with multiple 3×3 kernel-sized filters one after another. It was trained for weeks and was using NVIDIA Titan Black GPU's.

VGG16 Architecture

- The input to the convolution neural network is a fixed-size 224 × 224 RGB image. The only preprocessing it does is subtracting the mean RGB values, which are computed on the training dataset, from each pixel.
- Then the image is running through a stack of convolutional (Conv.) layers, where there are filters with a very small receptive field that is 3 × 3, which is the smallest size to capture the notion of left/right, up/down, and center part.
- In one of the configurations, it also utilizes 1 × 1 convolution filters, which can be observed as a linear transformation of the input channels followed by non-linearity. The convolutional strides are fixed to 1 pixel; the spatial padding of convolutional layer input is such that the spatial resolution is maintained after convolution, that is the padding is 1 pixel for 3 × 3 Conv. layers.
- Then the Spatial pooling is carried out by five max-pooling layers, 16 which follow some of the Conv. layers but not all the Conv. layers are followed by max-pooling. This Max-pooling is performed over a 2 × 2-pixel window, with stride 2.



- The architecture contains a stack of convolutional layers which have a different depth in different architectures which are followed by three Fully-Connected (FC) layers: the first two FC have 4096 channels each and the third FC performs 1000-way classification and thus contains 1000 channels that is one for each class.
- The final layer is the soft-max layer. The configuration of the fully connected layers is similar in all networks.
- All of the hidden layers are equipped with rectification (ReLU) non-linearity. Also, here one of the networks contains Local Response Normalization (LRN), such normalization does not improve the performance on the trained dataset, but usage of that leads to increased memory consumption and computation time.

Architecture Summary:

- Input to the model is a fixed size 224×224224×224 RGB image
- Pre-processing is subtracting the training set RGB value mean from each pixel
- Convolutional layers 17
  - Stride fixed to 1 pixel
  - padding is 1 pixel for 3×33×3
- Spatial pooling layers
  - This layer doesn’t count to the depth of the network by convention
  - Spatial pooling is done using max-pooling layers
  - window size is 2×22×2
  - Stride fixed to 2
  - Convnets used 5 max-pooling layers

- Fully-connected layers:
  - 1st: 4096 (ReLU).
  - 2nd: 4096 (ReLU).
  - 3rd: 1000 (Softmax).

Advantages:

1. **High Accuracy:** Performs exceptionally well for image classification tasks.
2. **Simple Architecture:** Uses only 3×3 convolutions and 2×2 pooling layers, making the design consistent and easy to implement.
3. **Transfer Learning:** Pre-trained VGG16 models can be easily used for various applications with limited data.
4. **Feature Extraction Power:** Deep layers effectively capture spatial and visual hierarchies of images.

Disadvantages:

1. **High Computational Cost:** Requires significant memory and computation power due to a large number of parameters (~138 million).
2. **Training Time:** Takes a long time to train from scratch.
3. **Redundant Parameters:** Large fully connected layers contribute heavily to model size and redundancy.
4. **Not Efficient for Real-Time Applications:** Due to its depth and complexity, it is less suitable for real-time or mobile-based applications.

Applications:

1. **Image Classification:** Classifying objects in large-scale datasets like ImageNet.
2. **Feature Extraction:** Used to extract deep visual features for other machine learning tasks.
3. **Object Detection and Recognition:** Serves as a backbone network in models like Faster R-CNN.
4. **Medical Image Analysis:** Used for disease detection from MRI, CT scans, or X-rays.
5. **Transfer Learning:** Fine-tuned for custom datasets (e.g., plant disease detection, animal classification, facial recognition).

PROGRAM CODE:

Vgg Implementation

```
%pip install tensorflow

import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import Sequential
from tensorflow.keras.utils import plot_model
from keras.layers import Conv2D, MaxPool2D, Dropout, Dense, Input, concatenate,
GlobalAveragePooling2D, AveragePooling2D, Flatten
from keras.models import Model

# Preparing the dataset
(train_images, train_labels), (test_images, test_labels) = keras.datasets.cifar10.load_data()
CLASS_NAMES= ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse','ship','truck']

# Preparing the dataset
validation_images, validation_labels = train_images[:5000], train_labels [:5000]
train_images, train_labels , train_images [5000:], train_labels [5000:]

# Building tensorflow datasets.
train_ds = tf.data.Dataset.from_tensor_slices((train_images, train_labels))
test_ds = tf.data.Dataset.from_tensor_slices((test_images, test_labels))
validation_ds = tf.data.Dataset.from_tensor_slices((validation_images, validation_labels))
def process_images (image, label):
   # Normalize images to have a mean of 0 and standard deviation of 1
   image = tf.image.per_image_standardization (image)
   # Resize images from 32x32 to 274x274
   image = tf.image.resize(image, (224,224))
   label = tf.one_hot(label, depth=10)  # One-hot encode the labels
   label = tf.squeeze(label, axis=0) # Remove the extra dimension
   return image, label

train_ds_size = tf.data.experimental.cardinality (train_ds).numpy()
test_ds_size = tf.data.experimental.cardinality(test_ds).numpy()
validation_ds_size = tf.data.experimental.cardinality (validation_ds).numpy()
train_ds = (train_ds
         .map(process_images)
         .shuffle(buffer_size=train_ds_size)
         .batch(batch_size=32, drop_remainder=True))
test_ds = (test_ds
        .map(process_images)
        .shuffle(buffer_size=train_ds_size)
        .batch(batch_size=32, drop_remainder=True))
validation_ds = (validation_ds
            .map(process_images)
            .shuffle(buffer_size=train_ds_size)
            .batch(batch_size=32, drop_remainder=True))

model = Sequential()
model.add(Conv2D(input_shape=(224,224,3),filters=64, kernel_size=(3,3), padding="same",
activation="relu"))
model.add(Conv2D (filters=64, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPool2D (pool_size=(2,2), strides=(2,2)))
model.add(Conv2D(filters=128, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D (filters=128, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPool2D (pool_size=(2,2), strides=(2,2)))
model.add(Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D (filters=256, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D (filters=256, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPool2D (pool_size=(2,2), strides=(2,2)))
model.add(Conv2D (filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D (filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPool2D(pool_size=(2,2), strides=(2,2)))
model.add(Conv2D (filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D (filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D (filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPool2D(pool_size=(2,2), strides=(2,2), name='vgg16'))
model.add(Flatten (name='flatten'))
model.add(Dense (256, activation= 'relu', name='fc1'))
model.add(Dense (128, activation='relu', name='fc2'))
model.add(Dense(10, activation='softmax', name='output'))

# Compiling the Model
model.compile(optimizer='adam', loss=keras.losses.categorical_crossentropy)
# Checking Model Summary
model.summary()

pip install pydot
plot_model(model, to_file="my_model.png", show_shapes=True, show_layer_names=True)

#model testing
history = model.fit(train_ds, epochs=10, validation_data=validation_ds,validation_freq=1)

#Model Evaluartion
model.evaluate(test_x, test_y)
```

Vgg Transfer Learning:

```
import tensorflow as tf
print(tf.__version__)

from google.colab import drive
```

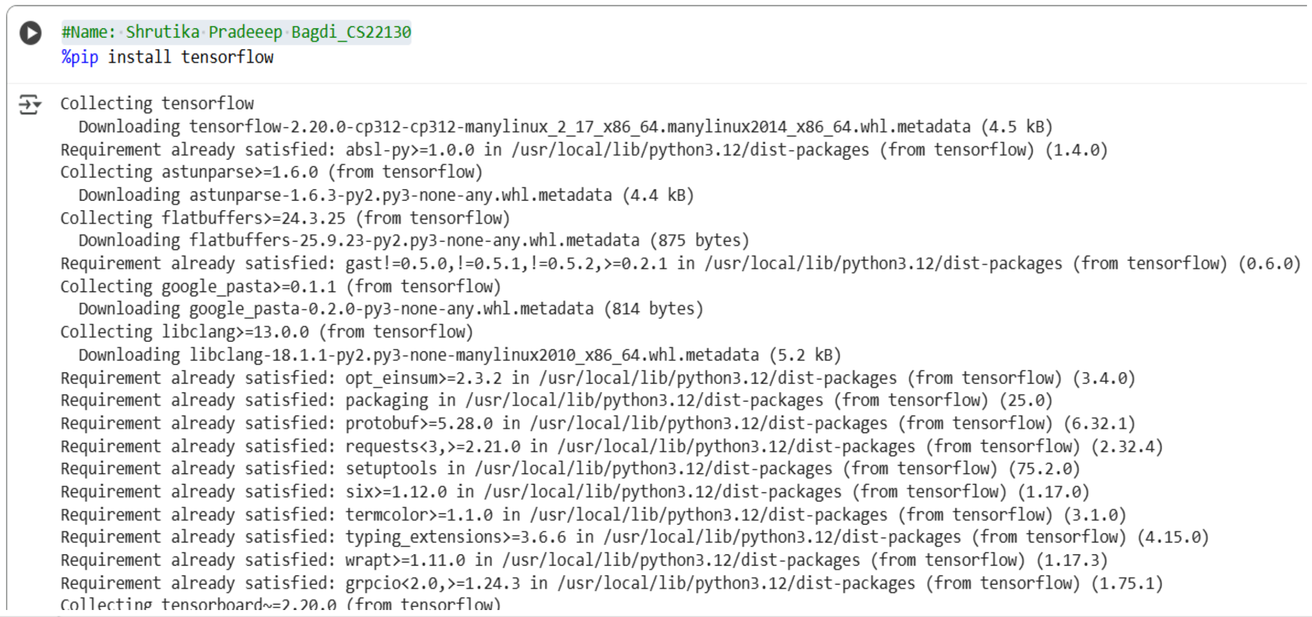
```
drive.mount("/content/drive")

# import the libraries as shown below
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.vgg19 import VGG19
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img
from tensorflow.keras.models import Sequential
import numpy as np
from glob import glob

# re-size all the images to this
IMAGE_SIZE = [224, 224]
#train_path = 'Datasets/train'
#valid_path = 'Datasets/test'
train_path = '/content/drive/MyDrive/Deep Learning/Cotton Disease/train'
valid_path = '/content/drive/MyDrive/Deep Learning/Cotton Disease/test'


# Import the VGG16 library as shown below and add preprocessing layer to the front # Here we will be
using imagenet weights
vgg16 = VGG16 (input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)
# don't train existing weights
for layer in vgg16.layers:
   layer.trainable = False

# useful for getting number of output classes
folders = glob('/content/drive/MyDrive/Deep Learning/Cotton Disease/train/*')

# our layers you can add more if you want
x = Flatten()(vgg16.output)
len(folders)
prediction = Dense(len(folders), activation='softmax')(x)
# create a model object
model = Model(inputs=vgg16.input, outputs=prediction)
# view the structure of the model
model.summary()

# tell the model what cost and optimization method to use
model.compile(
  optimizer='adam',
  loss='categorical_crossentropy',
  metrics=['accuracy']
)

# Use the Image Data Generator to import the images from the dataset from
tensorflow.keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator (rescale = 1./255,
shear_range = 0.2,
zoom_range = 0.2, horizontal_flip = True)
test_datagen = ImageDataGenerator (rescale = 1./255)

# Make sure you provide the same target size as initialied for the image size
training_set = train_datagen.flow_from_directory('/content/drive/MyDrive/Deep Learning/Cotton
Disease/train', target_size = (224, 224), batch_size = 32, class_mode = 'categorical')

test_set = test_datagen.flow_from_directory('/content/drive/MyDrive/Deep Learning/Cotton
Disease/test', target_size = (224, 224), batch_size = 32, class_mode = 'categorical')

# fit the model
# Run the cell. It will take some time to execute
r = model.fit(training_set,validation_data=test_set,epochs=1,steps_per_epoch=len (training_set),
validation_steps=len(test_set))

import matplotlib.pyplot as plt
# plot the loss
plt.plot(r.history['loss'], label='train loss')
plt.plot(r.history['val_loss'], label='val loss')
plt.legend()
plt.show()
plt.savefig('LossVal_loss')
# plot the accuracy
plt.plot(r.history['accuracy'], label='train acc')
plt.plot(r.history['val_accuracy'], label='val acc')
plt.legend()
plt.show()
plt.savefig('AccVal_acc')

from tensorflow.keras.models import load_model
model.save('model_vgg16.h5')

y_pred = model.predict(test_set)
```

y_pred

import numpy as np
y_pred = np.argmax (y_pred, axis=1)
y_pred

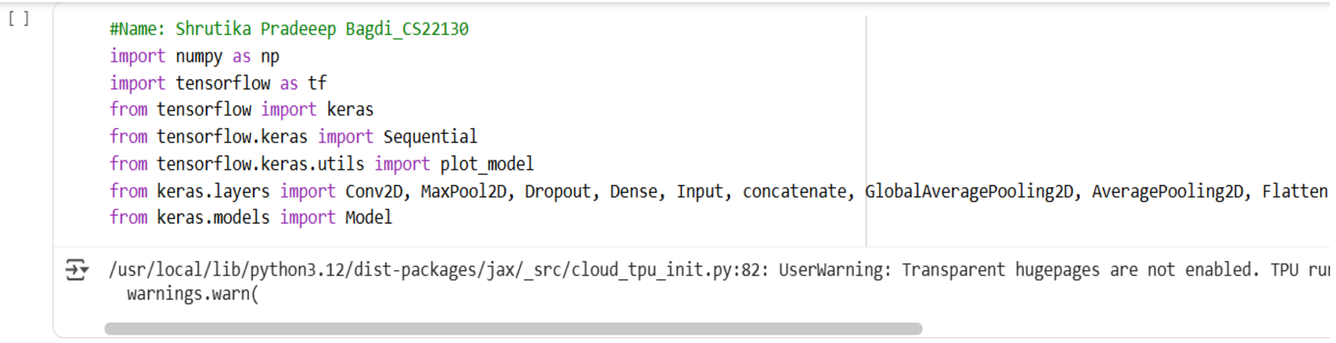
OUTPUT (SCREENSHOT):

Vgg Implementation:



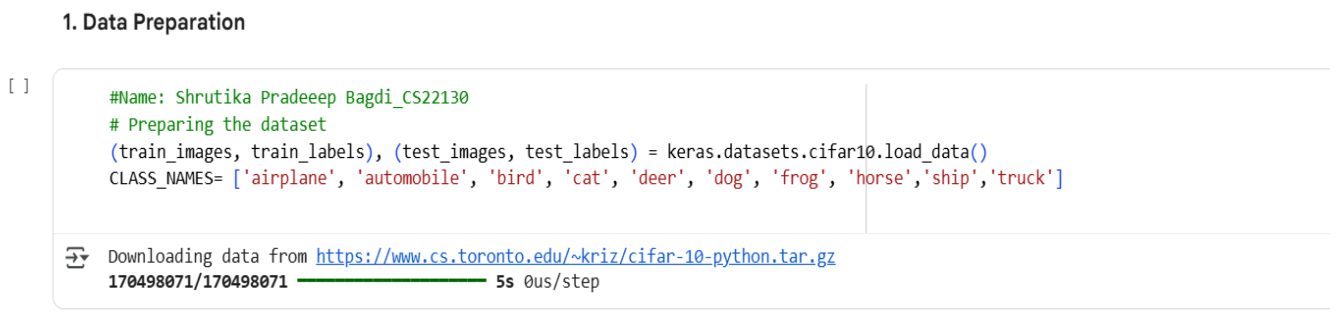
```
#Name: Shrutika Pradeeep Bagdi_CS22130
%pip install tensorflow
```

```
Collecting tensorflow
  Downloading tensorflow-2.20.0-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (4.5 kB)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (1.4.0)
Collecting astunparse>=1.6.0 (from tensorflow)
  Downloading astunparse-1.6.3-py2.py3-none-any.whl.metadata (4.4 kB)
Collecting flatbuffers>=24.3.25 (from tensorflow)
  Downloading flatbuffers-25.9.23-py2.py3-none-any.whl.metadata (875 bytes)
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) (0.6.0)
Collecting google_pasta>=0.1.1 (from tensorflow)
  Downloading google_pasta-0.2.0-py3-none-any.whl.metadata (814 bytes)
Collecting libclang>=13.0.0 (from tensorflow)
  Downloading libclang-18.1.1-py2.py3-none-manylinux2010_x86_64.whl.metadata (5.2 kB)
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>=5.28.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (6.32.1)
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) (1.75.1)
Collecting tensorboard~=2.20.0 (from tensorflow)
```



```
#Name: Shrutika Pradeeep Bagdi_CS22130
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import Sequential
from tensorflow.keras.utils import plot_model
from keras.layers import Conv2D, MaxPool2D, Dropout, Dense, Input, concatenate, GlobalAveragePooling2D, AveragePooling2D, Flatten
from keras.models import Model
```

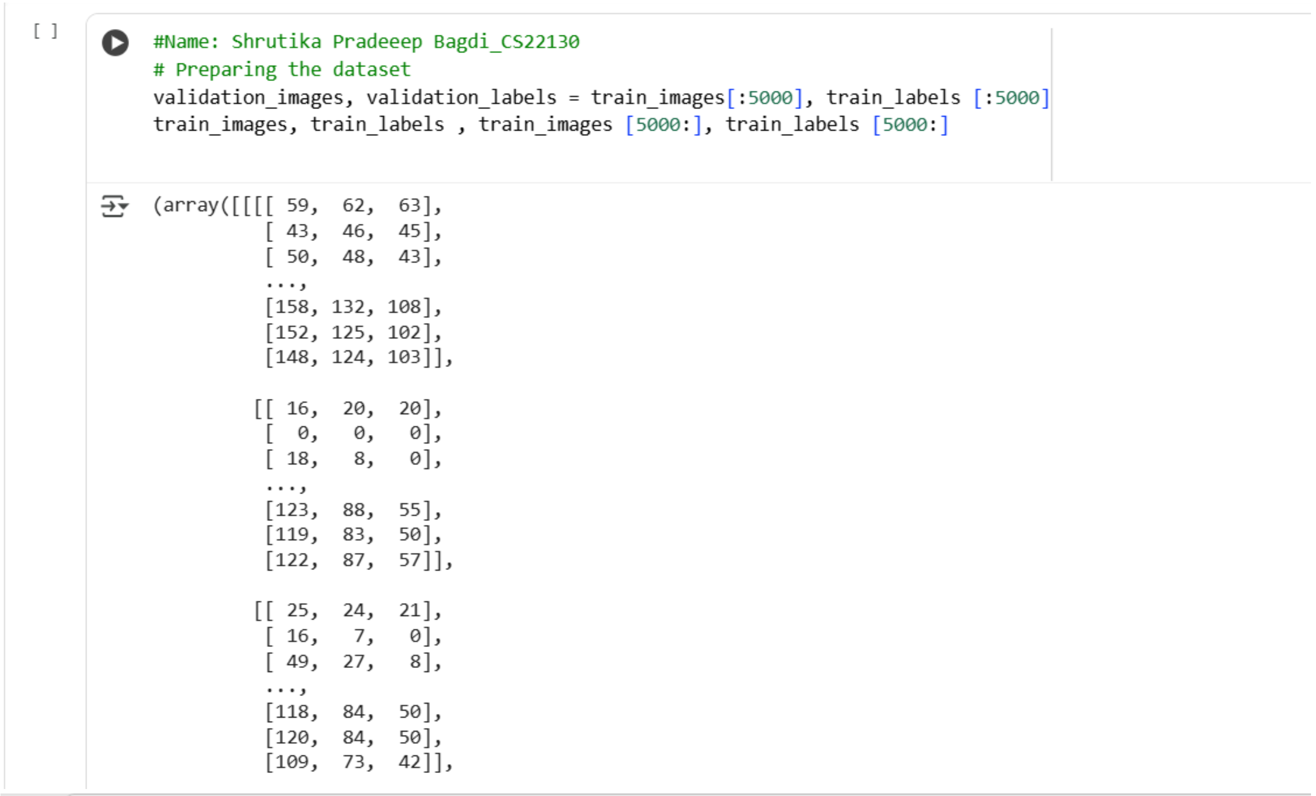
```
/usr/local/lib/python3.12/dist-packages/jax/_src/cloud_tpu_init.py:82: UserWarning: Transparent hugepages are not enabled. TPU ru
  warnings.warn(
```



1. Data Preparation

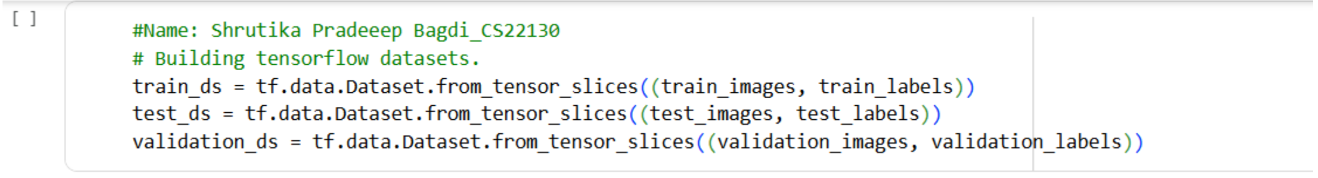
```
#Name: Shrutika Pradeeep Bagdi_CS22130
# Preparing the dataset
(train_images, train_labels), (test_images, test_labels) = keras.datasets.cifar10.load_data()
CLASS_NAMES= ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse','ship','truck']
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 ━━━━━━━━━━━━━━━━━━━━ 5s 0us/step
```

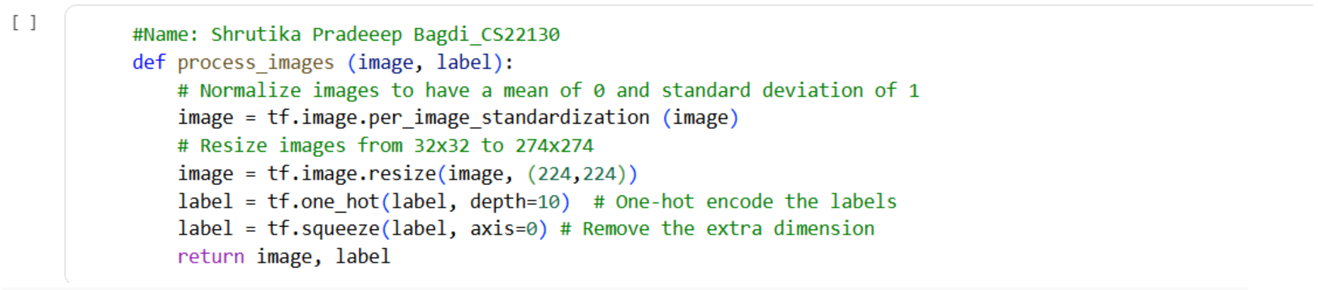


```
[ ]
#Name: Shrutika Pradeeep Bagdi_CS22130
# Preparing the dataset
validation_images, validation_labels = train_images[:5000], train_labels [:5000]
train_images, train_labels , train_images [5000:], train_labels [5000:]
```

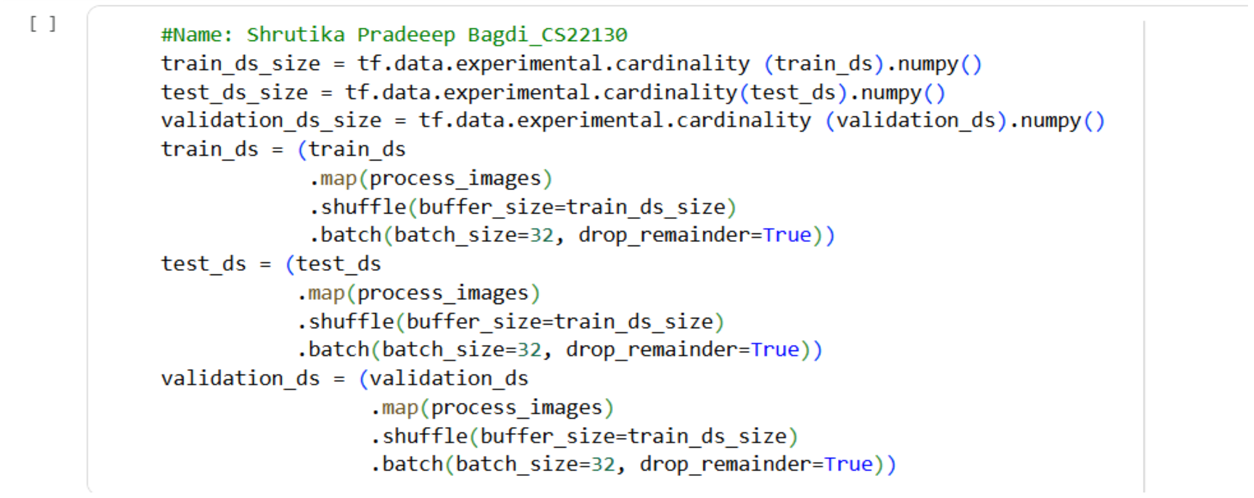
```
(array([[[[ 59,  62,  63],
          [ 43,  46,  45],
          [ 50,  48,  43],
          ...,
          [158, 132, 108],
          [152, 125, 102],
          [148, 124, 103]],

         [[ 16,  20,  20],
          [  0,   0,   0],
          [ 18,   8,   0],
          ...,
          [123,  88,  55],
          [119,  83,  50],
          [122,  87,  57]],

         [[ 25,  24,  21],
          [ 16,   7,   0],
          [ 49,  27,   8],
          ...,
          [118,  84,  50],
          [120,  84,  50],
          [109,  73,  42]],
```



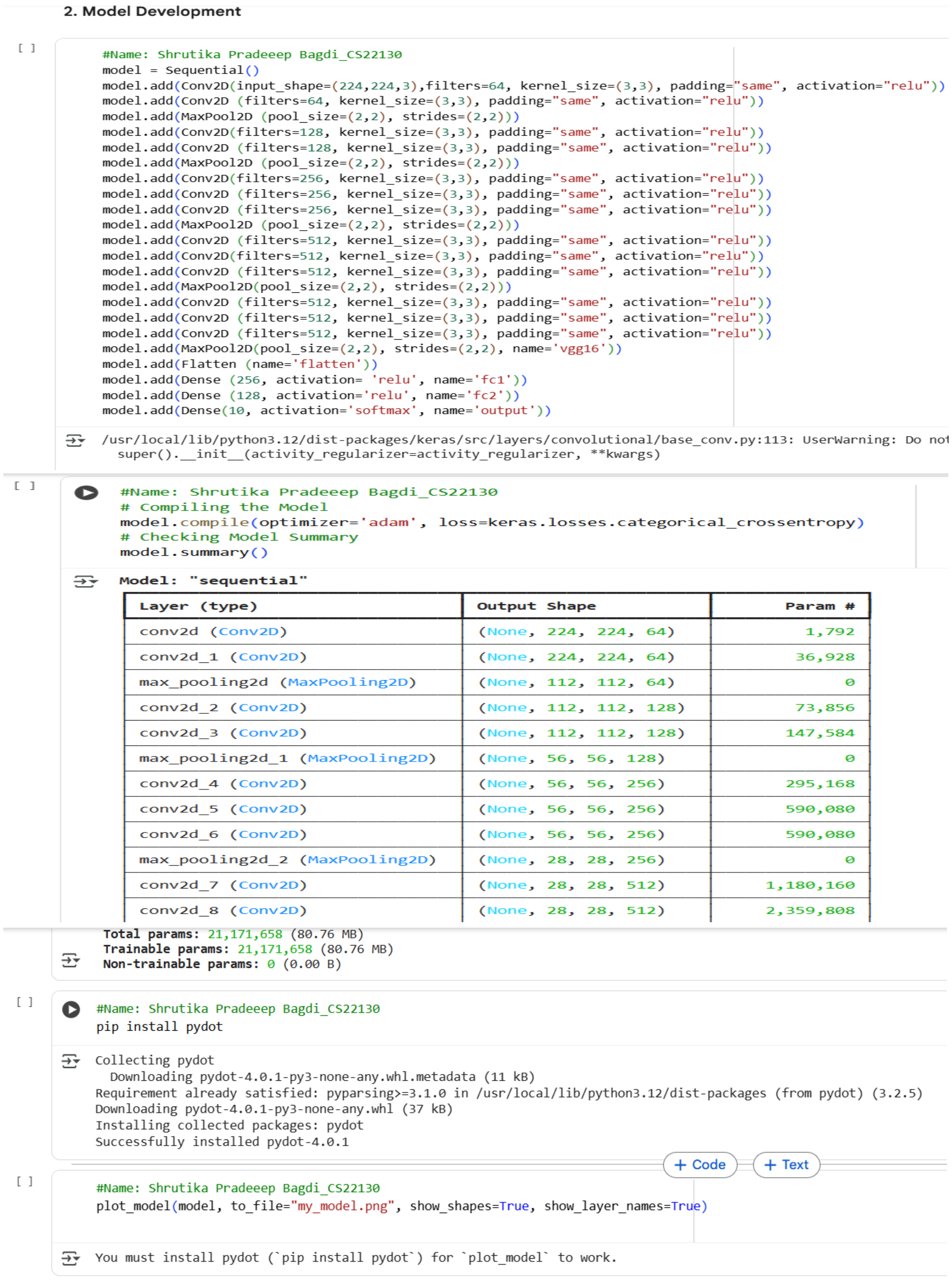
```
[ ]
#Name: Shrutika Pradeeep Bagdi_CS22130
# Building tensorflow datasets.
train_ds = tf.data.Dataset.from_tensor_slices((train_images, train_labels))
test_ds = tf.data.Dataset.from_tensor_slices((test_images, test_labels))
validation_ds = tf.data.Dataset.from_tensor_slices((validation_images, validation_labels))
```



```
[ ]
#Name: Shrutika Pradeeep Bagdi_CS22130
def process_images (image, label):
    # Normalize images to have a mean of 0 and standard deviation of 1
    image = tf.image.per_image_standardization (image)
    # Resize images from 32x32 to 274x274
    image = tf.image.resize(image, (224,224))
    label = tf.one_hot(label, depth=10)  # One-hot encode the labels
    label = tf.squeeze(label, axis=0) # Remove the extra dimension
    return image, label
```



```
[ ]
#Name: Shrutika Pradeeep Bagdi_CS22130
train_ds_size = tf.data.experimental.cardinality (train_ds).numpy()
test_ds_size = tf.data.experimental.cardinality(test_ds).numpy()
validation_ds_size = tf.data.experimental.cardinality (validation_ds).numpy()
train_ds = (train_ds
             .map(process_images)
             .shuffle(buffer_size=train_ds_size)
             .batch(batch_size=32, drop_remainder=True))
test_ds = (test_ds
            .map(process_images)
            .shuffle(buffer_size=train_ds_size)
            .batch(batch_size=32, drop_remainder=True))
validation_ds = (validation_ds
                  .map(process_images)
                  .shuffle(buffer_size=train_ds_size)
                  .batch(batch_size=32, drop_remainder=True))
```



2. Model Development

```
[ ]
#Name: Shrutika Pradeeep Bagdi_CS22130
model = Sequential()
model.add(Conv2D(input_shape=(224,224,3),filters=64, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D (filters=64, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPool2D (pool_size=(2,2), strides=(2,2)))
model.add(Conv2D(filters=128, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D (filters=128, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPool2D (pool_size=(2,2), strides=(2,2)))
model.add(Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D (filters=256, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D (filters=256, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPool2D (pool_size=(2,2), strides=(2,2)))
model.add(Conv2D (filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D (filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPool2D(pool_size=(2,2), strides=(2,2)))
model.add(Conv2D (filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D (filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D (filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPool2D(pool_size=(2,2), strides=(2,2), name='vgg16'))
model.add(Flatten (name='flatten'))
model.add(Dense (256, activation= 'relu', name='fc1'))
model.add(Dense (128, activation='relu', name='fc2'))
model.add(Dense(10, activation='softmax', name='output'))
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/convolutional/base_conv.py:113: UserWarning: Do no
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
[ ]
#Name: Shrutika Pradeeep Bagdi_CS22130
# Compiling the Model
model.compile(optimizer='adam', loss=keras.losses.categorical_crossentropy)
# Checking Model Summary
model.summary()
```

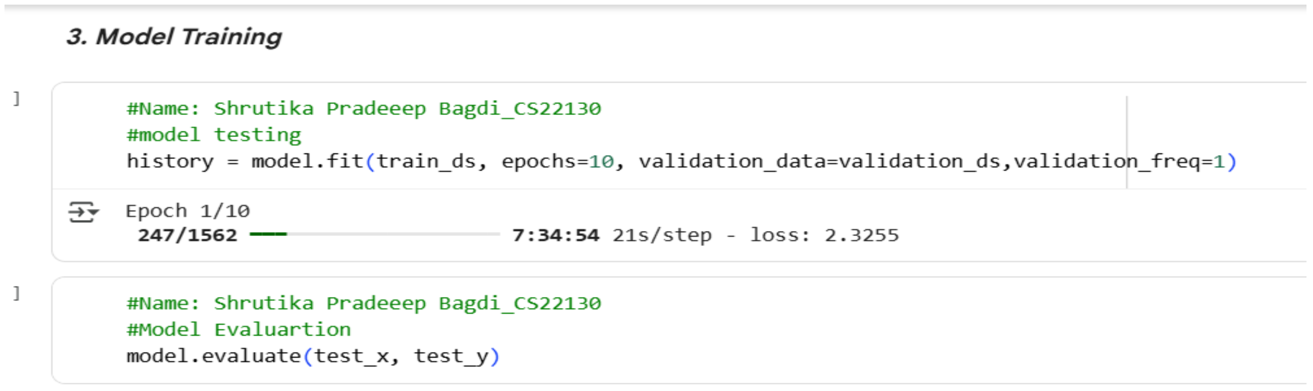
```
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                            ┃ Output Shape                ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ conv2d (Conv2D)                         │ (None, 224, 224, 64)        │         1,792 │
│ conv2d_1 (Conv2D)                       │ (None, 224, 224, 64)        │        36,928 │
│ max_pooling2d (MaxPooling2D)            │ (None, 112, 112, 64)        │             0 │
│ conv2d_2 (Conv2D)                       │ (None, 112, 112, 128)       │        73,856 │
│ conv2d_3 (Conv2D)                       │ (None, 112, 112, 128)       │       147,584 │
│ max_pooling2d_1 (MaxPooling2D)          │ (None, 56, 56, 128)         │             0 │
│ conv2d_4 (Conv2D)                       │ (None, 56, 56, 256)         │       295,168 │
│ conv2d_5 (Conv2D)                       │ (None, 56, 56, 256)         │       590,080 │
│ conv2d_6 (Conv2D)                       │ (None, 56, 56, 256)         │       590,080 │
│ max_pooling2d_2 (MaxPooling2D)          │ (None, 28, 28, 256)         │             0 │
│ conv2d_7 (Conv2D)                       │ (None, 28, 28, 512)         │     1,180,160 │
│ conv2d_8 (Conv2D)                       │ (None, 28, 28, 512)         │     2,359,808 │
 Total params: 21,171,658 (80.76 MB)
 Trainable params: 21,171,658 (80.76 MB)
 Non-trainable params: 0 (0.00 B)
```

```
[ ]
#Name: Shrutika Pradeeep Bagdi_CS22130
pip install pydot
```

```
Collecting pydot
  Downloading pydot-4.0.1-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: pyparsing>=3.1.0 in /usr/local/lib/python3.12/dist-packages (from pydot) (3.2.5)
Downloading pydot-4.0.1-py3-none-any.whl (37 kB)
Installing collected packages: pydot
Successfully installed pydot-4.0.1
```

```
[ ]
#Name: Shrutika Pradeeep Bagdi_CS22130
plot_model(model, to_file="my_model.png", show_shapes=True, show_layer_names=True)
```

```
You must install pydot (`pip install pydot`) for `plot_model` to work.
```



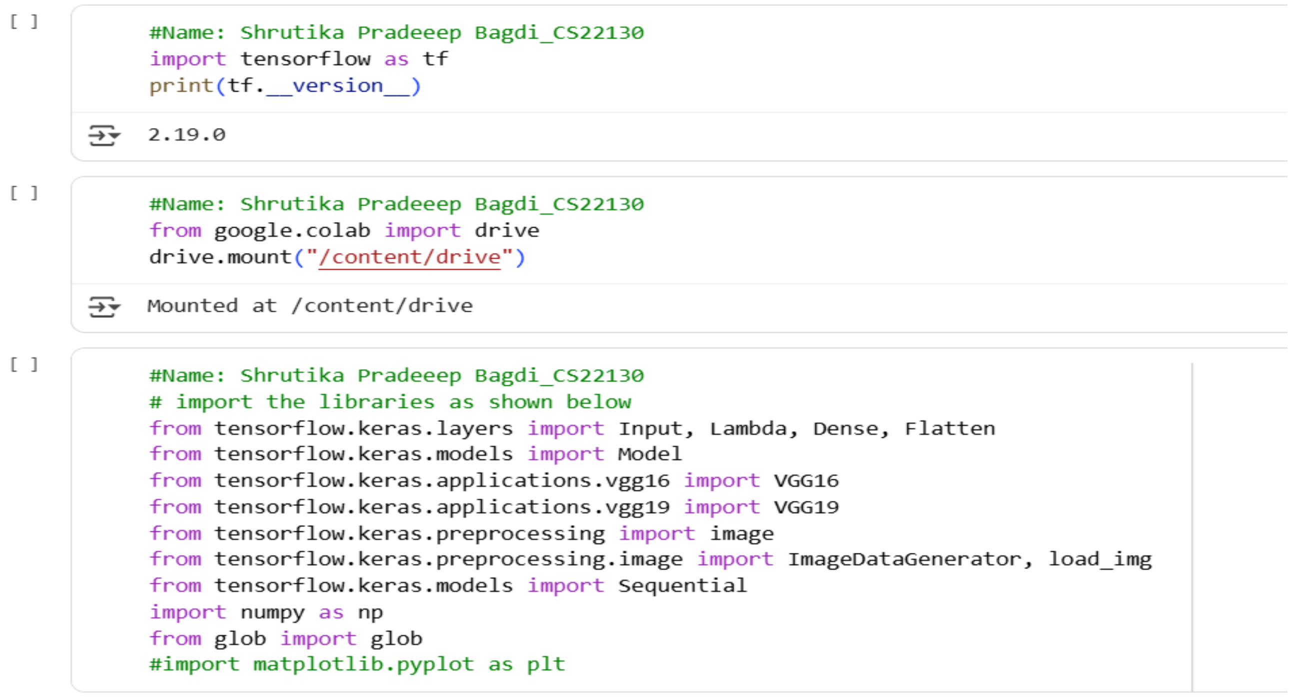
3. Model Training

```
]
#Name: Shrutika Pradeeep Bagdi_CS22130
#model testing
history = model.fit(train_ds, epochs=10, validation_data=validation_ds,validation_freq=1)
```

```
Epoch 1/10
 247/1562 ━━━                  7:34:54 21s/step - loss: 2.3255
```

```
]
#Name: Shrutika Pradeeep Bagdi_CS22130
#Model Evaluartion
model.evaluate(test_x, test_y)
```

VGG Transfer Learning



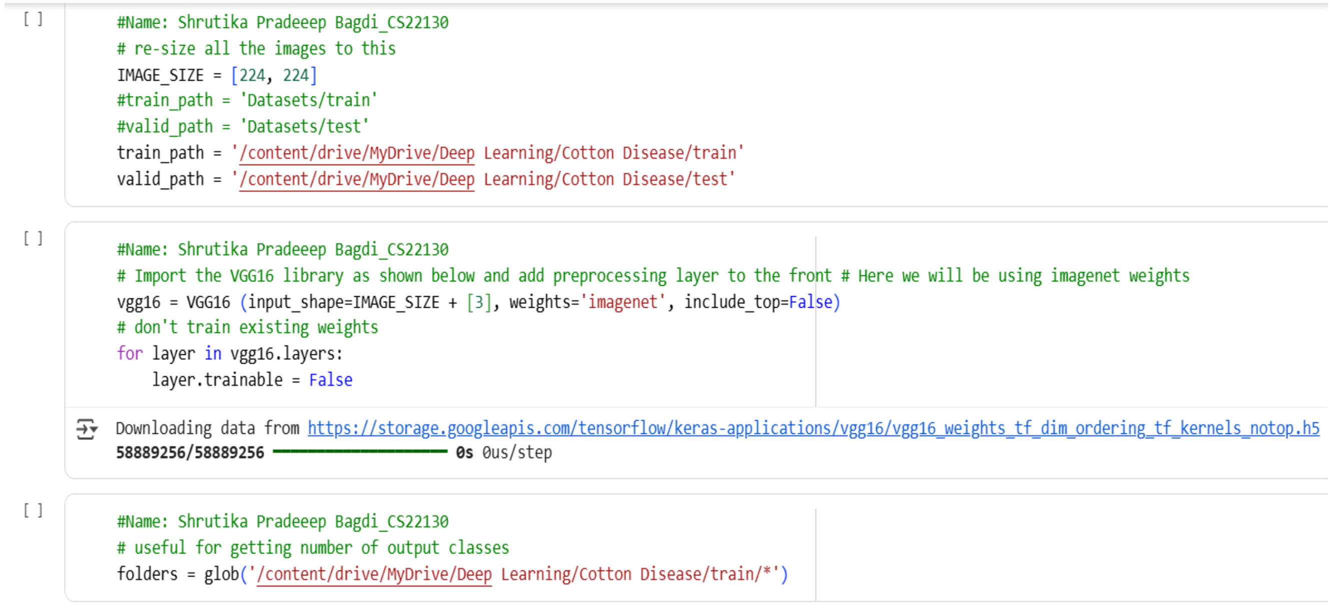
```
[ ]
#Name: Shrutika Pradeeep Bagdi_CS22130
import tensorflow as tf
print(tf.__version__)
```

```
2.19.0
```

```
[ ]
#Name: Shrutika Pradeeep Bagdi_CS22130
from google.colab import drive
drive.mount("/content/drive")
```

```
Mounted at /content/drive
```

```
[ ]
#Name: Shrutika Pradeeep Bagdi_CS22130
# import the libraries as shown below
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.vgg19 import VGG19
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img
from tensorflow.keras.models import Sequential
import numpy as np
from glob import glob
#import matplotlib.pyplot as plt
```

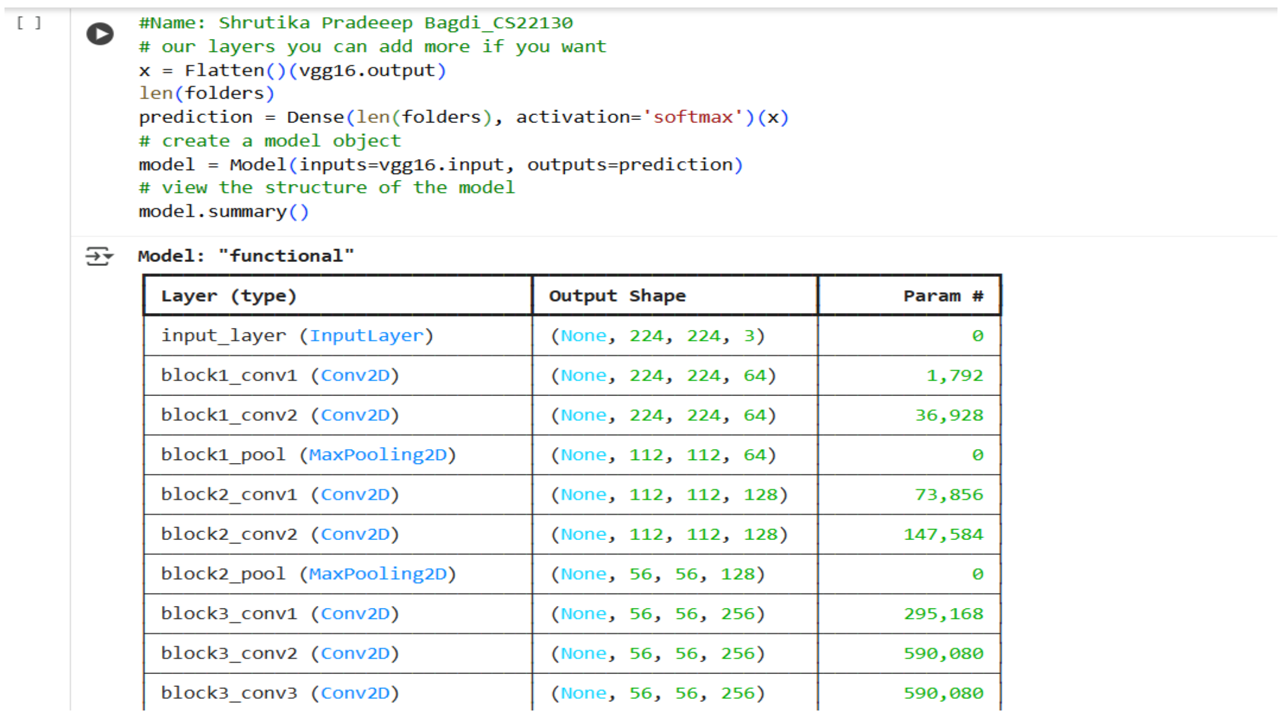


```
[ ]
#Name: Shrutika Pradeeep Bagdi_CS22130
# re-size all the images to this
IMAGE_SIZE = [224, 224]
#train_path = 'Datasets/train'
#valid_path = 'Datasets/test'
train_path = '/content/drive/MyDrive/Deep Learning/Cotton Disease/train'
valid_path = '/content/drive/MyDrive/Deep Learning/Cotton Disease/test'
```

```
[ ]
#Name: Shrutika Pradeeep Bagdi_CS22130
# Import the VGG16 library as shown below and add preprocessing layer to the front # Here we will be using imagenet weights
vgg16 = VGG16 (input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)
# don't train existing weights
for layer in vgg16.layers:
    layer.trainable = False
```

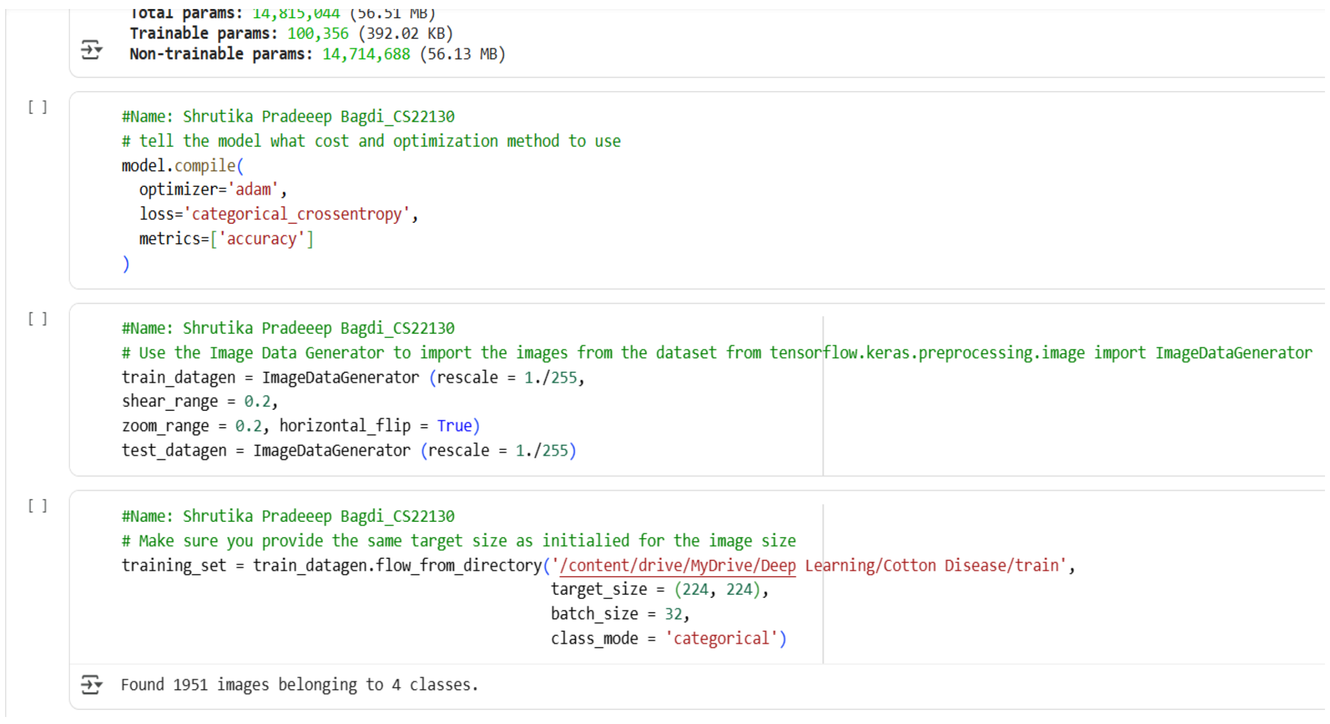
```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

```
[ ]
#Name: Shrutika Pradeeep Bagdi_CS22130
# useful for getting number of output classes
folders = glob('/content/drive/MyDrive/Deep Learning/Cotton Disease/train/*')
```



```
[ ]
#Name: Shrutika Pradeeep Bagdi_CS22130
# our layers you can add more if you want
x = Flatten()(vgg16.output)
len(folders)
prediction = Dense(len(folders), activation='softmax')(x)
# create a model object
model = Model(inputs=vgg16.input, outputs=prediction)
# view the structure of the model
model.summary()
```

```
Model: "functional"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ 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 │
```



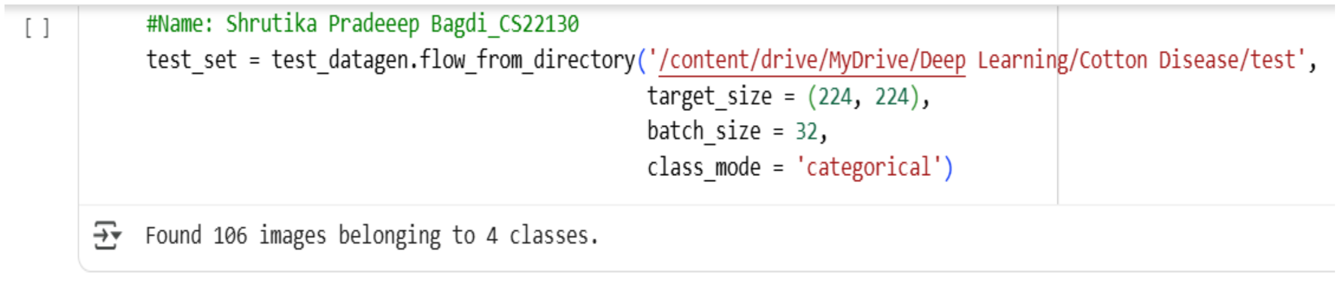
```
 Total params: 14,815,044 (56.51 MB)
 Trainable params: 100,356 (392.02 KB)
 Non-trainable params: 14,714,688 (56.13 MB)
```

```
[ ]
#Name: Shrutika Pradeeep Bagdi_CS22130
# tell the model what cost and optimization method to use
model.compile(
  optimizer='adam',
  loss='categorical_crossentropy',
  metrics=['accuracy']
)
```

```
[ ]
#Name: Shrutika Pradeeep Bagdi_CS22130
# Use the Image Data Generator to import the images from the dataset from tensorflow.keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator (rescale = 1./255,
shear_range = 0.2,
zoom_range = 0.2, horizontal_flip = True)
test_datagen = ImageDataGenerator (rescale = 1./255)
```

```
[ ]
#Name: Shrutika Pradeeep Bagdi_CS22130
# Make sure you provide the same target size as initialied for the image size
training_set = train_datagen.flow_from_directory('/content/drive/MyDrive/Deep Learning/Cotton Disease/train',
                                                 target_size = (224, 224),
                                                 batch_size = 32,
                                                 class_mode = 'categorical')
```

```
Found 1951 images belonging to 4 classes.
```



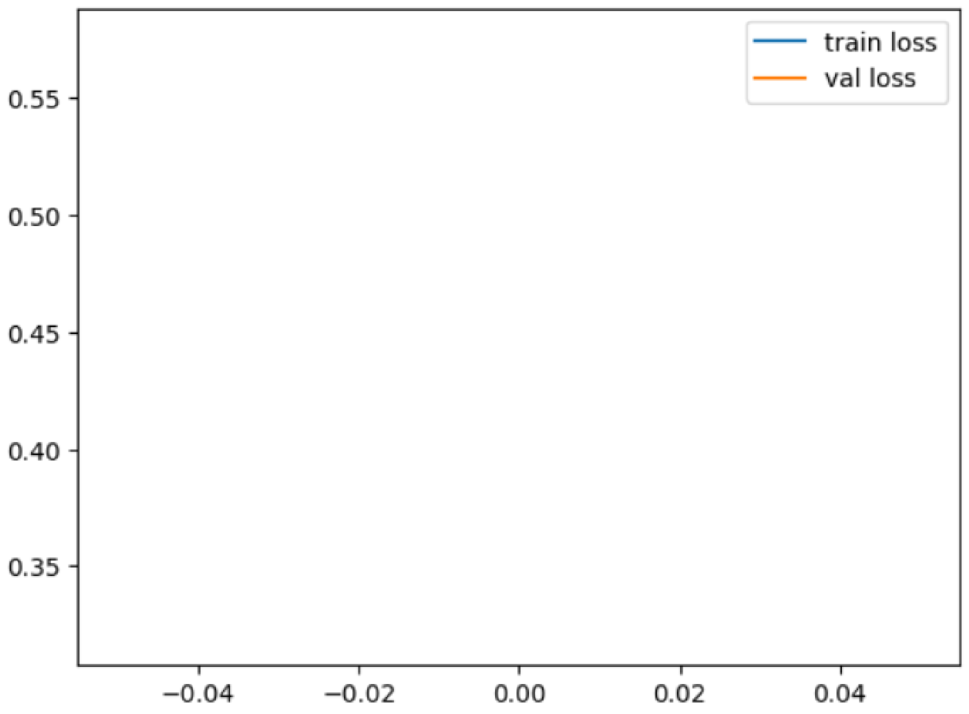
```
[ ]
#Name: Shrutika Pradeeep Bagdi_CS22130
test_set = test_datagen.flow_from_directory('/content/drive/MyDrive/Deep Learning/Cotton Disease/test',
                                            target_size = (224, 224),
                                            batch_size = 32,
                                            class_mode = 'categorical')
```

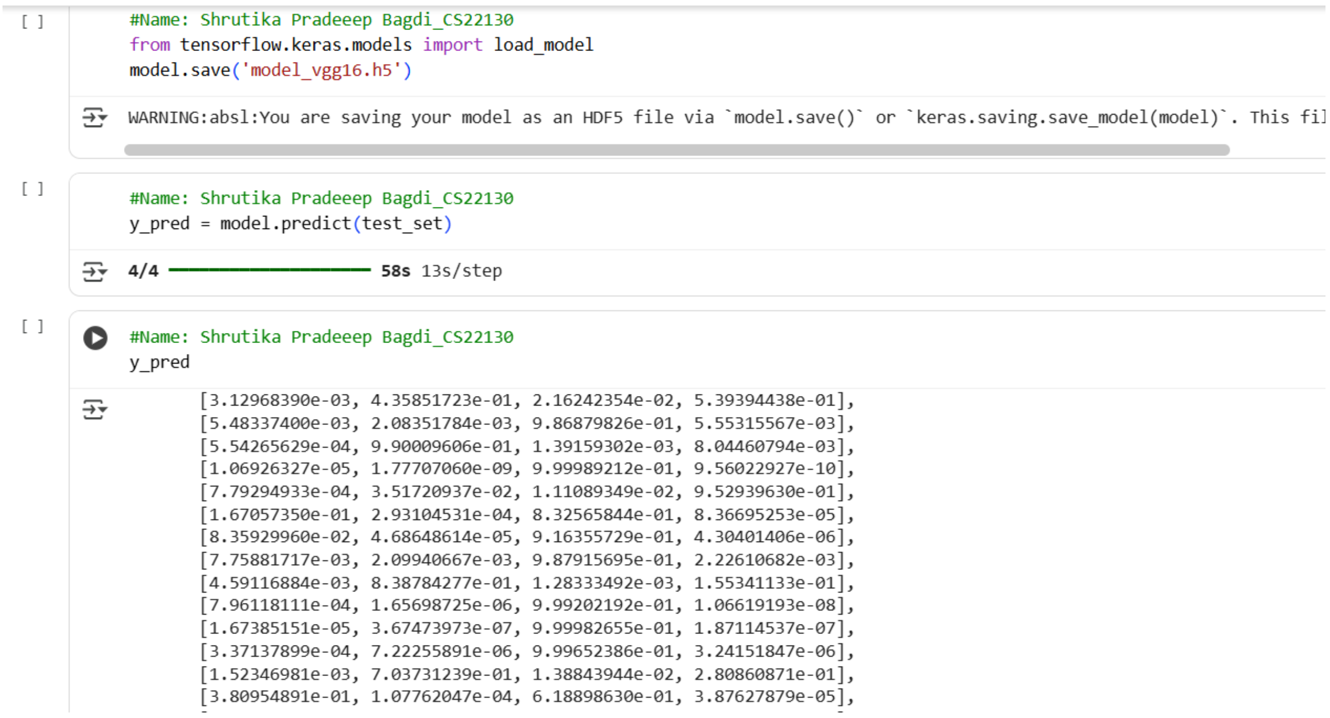
```
Found 106 images belonging to 4 classes.
```

```
[ ]  #Name: Shrutika Pradeeep Bagdi_CS22130
     # fit the model
     # Run the cell. It will take some time to execute
     r = model.fit(
     training_set,
     validation_data=test_set,
     epochs=1,
     steps_per_epoch=len (training_set),
     validation_steps=len(test_set)
     )
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your
  self._warn_if_super_not_called()
61/61 ━━━━━━━━━━━━━━━━━━━━ 1158s 19s/step - accuracy: 0.6619 - loss: 0.8809 - val_accuracy: 0.8774 - val_loss: 0.3203
```

```
[ ]  #Name: Shrutika Pradeeep Bagdi_CS22130
     import matplotlib.pyplot as plt
     # plot the loss
     plt.plot(r.history['loss'], label='train loss')
     plt.plot(r.history['val_loss'], label='val loss')
     plt.legend()
     plt.show()
     plt.savefig('LossVal_loss')
     # plot the accuracy
     plt.plot(r.history['accuracy'], label='train acc')
     plt.plot(r.history['val_accuracy'], label='val acc')
     plt.legend()
     plt.show()
     plt.savefig('AccVal_acc')
```

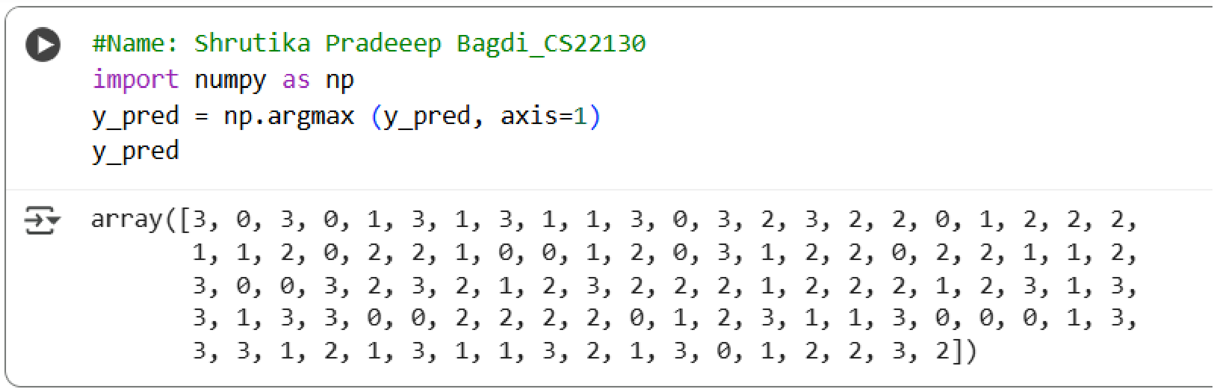




```
[ ]
#Name: Shrutika Pradeeep Bagdi_CS22130
from tensorflow.keras.models import load_model
model.save('model_vgg16.h5')

WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This fil
```

```
[ ]
#Name: Shrutika Pradeeep Bagdi_CS22130
y_pred = model.predict(test_set)

4/4 ━━━━━━━━━━━━━━━━━━━━ 58s 13s/step
```

```
[ ]
#Name: Shrutika Pradeeep Bagdi_CS22130
y_pred

        [3.12968390e-03, 4.35851723e-01, 2.16242354e-02, 5.39394438e-01],
        [5.48337400e-03, 2.08351784e-03, 9.86879826e-01, 5.55315567e-03],
        [5.54265629e-04, 9.90009606e-01, 1.39159302e-03, 8.04460794e-03],
        [1.06926327e-05, 1.77707060e-09, 9.99989212e-01, 9.56022927e-10],
        [7.79294933e-04, 3.51720937e-02, 1.11089349e-02, 9.52939630e-01],
        [1.67057350e-01, 2.93104531e-04, 8.32565844e-01, 8.36695253e-05],
        [8.35929960e-02, 4.68648614e-05, 9.16355729e-01, 4.30401406e-06],
        [7.75881717e-03, 2.09940667e-03, 9.87915695e-01, 2.22610682e-03],
        [4.59116884e-03, 8.38784277e-01, 1.28333492e-03, 1.55341133e-01],
        [7.96118111e-04, 1.65698725e-06, 9.99202192e-01, 1.06619193e-08],
        [1.67385151e-05, 3.67473973e-07, 9.99982655e-01, 1.87114537e-07],
        [3.37137899e-04, 7.22255891e-06, 9.99652386e-01, 3.24151847e-06],
        [1.52346981e-03, 7.03731239e-01, 1.38843944e-02, 2.80860871e-01],
        [3.80954891e-01, 1.07762047e-04, 6.18898630e-01, 3.87627879e-05],
```



```
[ ]
#Name: Shrutika Pradeeep Bagdi_CS22130
import numpy as np
y_pred = np.argmax (y_pred, axis=1)
y_pred

array([3, 0, 3, 0, 1, 3, 1, 3, 1, 1, 3, 0, 3, 2, 3, 2, 2, 0, 1, 2, 2, 2,
       1, 1, 2, 0, 2, 2, 1, 0, 0, 1, 2, 0, 3, 1, 2, 2, 0, 2, 2, 1, 1, 2,
       3, 0, 0, 3, 2, 3, 2, 1, 2, 3, 2, 2, 2, 1, 2, 2, 2, 1, 2, 3, 1, 3,
       3, 1, 3, 3, 0, 0, 2, 2, 2, 2, 0, 1, 2, 3, 1, 1, 3, 0, 0, 0, 1, 3,
       3, 3, 1, 2, 1, 3, 1, 1, 3, 2, 1, 3, 0, 1, 2, 2, 3, 2])
```

CONCLUSION:

____

____

____

DISCUSSION AND VIVA VOCE:

1. What is VGG16, and who developed it?
2. What is the main idea behind the VGG architecture?
3. How does VGG16 differ from other CNN architectures like AlexNet or ResNet?
4. Why does VGG16 use smaller (3×3) convolutional filters instead of larger ones?
5. What is the purpose of using multiple convolutional layers before a pooling layer?

6. What is the role of the ReLU activation function in VGG16?
7. What type of pooling is used in VGG16, and why?
8. What is the input image size used in VGG16?
9. How many trainable parameters are there approximately in VGG16?

REFERENCE:

- https://www.analyticsvidhya.com/blog/2021/06/build-vgg-net-from-scratch-with-python/
- https://www.kaggle.com/code/blurredmachine/vggnet-16-architecture-a-complete-guide
- https://keras.io/api/applications/vgg/
- https://www.mygreatlearning.com/blog/introduction-to-vgg16/