HU21CSEN0300296, T.BHAVYA SHREE

```
# @title HU21CSEN0300296, T.BHAVYA SHREE
from google.colab import drive
drive.mount('/content/drive')
```

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

loading the data

```
# @title loading the data
#loading the data
import numpy as np
import pandas as pd
import os
print(os.listdir("/content/drive/MyDrive/Dl"))
```

```
['test_dataset', 'validation_dataset', 'train_dataset', 'vgg_model.h5']
```

Importing required libraries

```
# @title Importing required libraries
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing import image
import os
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import VGG16
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.applications import InceptionV3

```

```
!pip install python-resize-image
```

```
Collecting python-resize-image
  Downloading python_resize_image-1.1.20-py2.py3-none-any.whl (8.4 kB)
Requirement already satisfied: Pillow>=5.1.0 in /usr/local/lib/python3.10/dist-packages (from python-resize-image) (9.4.0)
Requirement already satisfied: requests>=2.19.1 in /usr/local/lib/python3.10/dist-packages (from python-resize-image) (2.31.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.1->python-re
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.1->python-resize-image)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.1->python-resize-i
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.1->python-resize-i
Installing collected packages: python-resize-image
Successfully installed python-resize-image-1.1.20
```

```
train_ds = train_datagen.flow_from_directory(
    train_dataset,
    class_mode='categorical',
    shuffle=True,
    batch_size=32,
    color_mode='rgb',
    target_size=(224, 224),
    subset='training'
)
```

```
Found 18 images belonging to 3 classes.
```

Exploratory data analysis

```
# @title Exploratory data analysis
# Checking class distribution in the training dataset
class_counts = train_ds.classes
unique_classes, class_counts = np.unique(class_counts, return_counts=True)
class_distribution = dict(zip(train_ds.class_indices.keys(), class_counts))

print("Class Distribution:")
for cls, count in class_distribution.items():
    print(f"{cls}: {count} images")
```

```
Class Distribution:
drone: 6 images
flight: 6 images
helicopter: 6 images
```

```
# Displaying sample images from each class
plt.figure(figsize=(15, 10))
for i in range(len(train_ds.class_indices)):
    cls_name = list(train_ds.class_indices.keys())[i]
    cls_dir = os.path.join(train_dataset, cls_name)
    sample_img = os.listdir(cls_dir)[0]
    img_path = os.path.join(cls_dir, sample_img)

    plt.subplot(1, len(train_ds.class_indices), i+1)
    plt.imshow(image.load_img(img_path))
    plt.title(cls_name)
    plt.axis('off')
plt.show()
```

```
# Checking image dimensions
sample_img, _ = train_ds[0]
img_shape = sample_img[0].shape
print(f"Image Shape: {img_shape}")
```

```
Image Shape: (224, 224, 3)
```

```
# Visualizing augmented images
augmented_images = [train_ds[0][0][0] for _ in range(5)]

plt.figure(figsize=(15, 5))
for i, img in enumerate(augmented_images):
    plt.subplot(1, 5, i+1)
    plt.imshow(img)
    plt.axis('off')
plt.show()
```

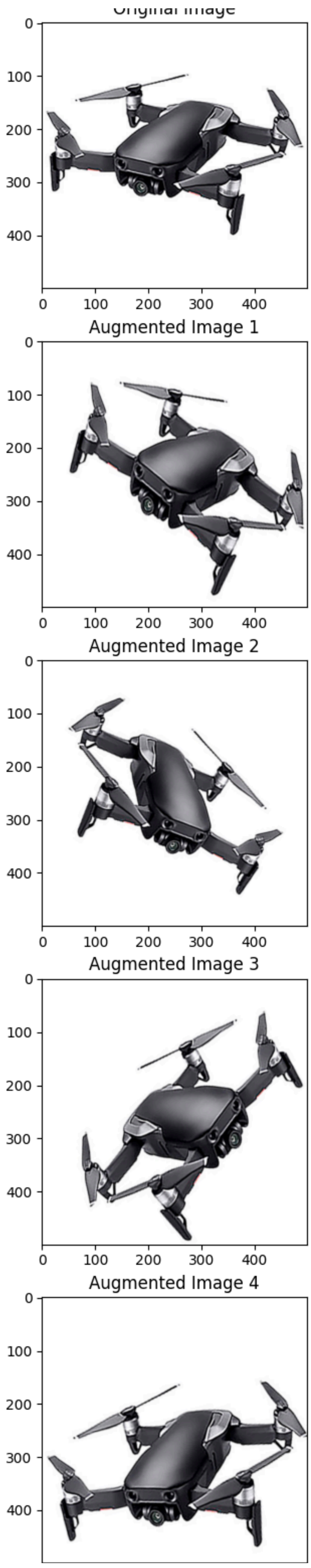
```
# path to the test dataset
test_dataset = "/content/drive/MyDrive/Dl/test_dataset/test_dataset/test_dataset"
class_name = list(os.listdir(test_dataset)[0])

# Initialize the ImageDataGenerator for testing (only rescaling)
test_datagen = ImageDataGenerator(rescale=1./255)

# Load Test Data
test_ds = test_datagen.flow_from_directory(
    test_dataset,
    class_mode='categorical',
    shuffle=False,
    batch_size=1,  # Adjust to the actual number of images in your test dataset
    color_mode='rgb',
    target_size=(224, 224)  # Adjust the target size for VGG16
)

# Get a batch of images for visualization
sample_images, _ = next(test_ds)

# Visualize a few sample images
plt.figure(figsize=(15, 10))
for i in range(len(sample_images)):
    plt.subplot(1, len(sample_images), i + 1)
    plt.imshow(sample_images[i])
    plt.title(f"Class: {class_name[i]}")
    plt.axis('off')

plt.show()
```

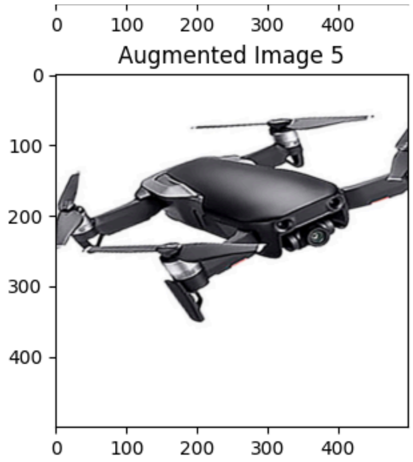
```
Found 6 images belonging to 3 classes.
```

```
from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator
import numpy as np
import matplotlib.pyplot as plt

# Path to your model and sample image
model_path = '/content/drive/MyDrive/Dl/your_model.h5'  # Replace with the path to your model
sample_image_path = '/content/drive/MyDrive/Dl/test_dataset/test_dataset/ship/677562.jpg'  # Replace with the path to your sample ima

# Load your model
from keras.models import load_model
model = load_model(model_path)

# Load the sample image
sample_image = image.load_img(sample_image_path)
img_array = image.img_to_array(sample_image)
img_array = np.expand_dims(img_array, axis=0)

# ImageDataGenerator
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'
)

# Generate augmented images
augmented_images = []

for batch in datagen.flow(img_array, batch_size=1):
    augmented_images.append(image.array_to_img(batch[0]))
    if len(augmented_images) >= 5:  # Generate 5 augmented images
        break

# Display original and augmented images
plt.figure(figsize=(27, 4 * (len(augmented_images) + 1)))

# Original image
plt.subplot(len(augmented_images) + 1, 6, 1)
plt.imshow(sample_image)
plt.title('Original Image')

# Augmented images on separate lines
for i, augmented_img in enumerate(augmented_images):
    plt.subplot(len(augmented_images) + 1, 6, 6 * (i + 1) + 1)
    plt.imshow(augmented_img)
    plt.title(f'Augmented Image {i + 1}')

# Prediction on original and augmented images
plt.subplot(len(augmented_images) + 1, 6, 6)
original_prediction = model.predict(img_array)
plt.text(0, 0.5, f'Prediction: {original_prediction}', fontsize=12, va='center')

for i, augmented_img in enumerate(augmented_images):
    plt.subplot(len(augmented_images) + 1, 6, 6 * (i + 1) + 6)
    augmented_img_array = image.img_to_array(augmented_img)
    augmented_img_array = np.expand_dims(augmented_img_array, axis=0)
    augmented_prediction = model.predict(augmented_img_array)
    plt.text(0, 0.5, f'Prediction: {augmented_prediction}', fontsize=12, va='center')

plt.show()

```

```
from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator
import numpy as np
import matplotlib.pyplot as plt

sample_image_path = '/content/drive/MyDrive/Dl/train_dataset/train_dataset/train_dataset/drone/drone 1.png'
sample_image = image.load_img(sample_image_path)
img_array = image.img_to_array(sample_image)
img_array = np.expand_dims(img_array, axis=0)

#ImageDataGenerator
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'
)

# Generate augmented images
augmented_images = []

for batch in datagen.flow(img_array, batch_size=1):
    augmented_images.append(image.array_to_img(batch[0]))
    if len(augmented_images) >= 5:  # Generate 5 augmented images
        break

#original and augmented images
plt.figure(figsize=(27, 4 * (len(augmented_images) + 1)))  # Adjust the height

#original image
plt.subplot(len(augmented_images) + 1, 6, 1)
plt.imshow(sample_image)
plt.title('Original Image')

```

```
# augmented image are then displayed on seperate lines for better visibility
for i, augmented_img in enumerate(augmented_images):
    plt.subplot(len(augmented_images) + 1, 6, 6 * (i + 1) + 1)
    plt.imshow(augmented_img)
    plt.title(f'Augmented Image {i + 1}')

#The augmentation includes random rotations, shifts, shearing, zooming, and horizontal flipping
plt.show()
```


Original Image
0
100
200
300
400
0
100
200
300
400
Augmented Image 1
Augmented Image 2
Augmented Image 3
Augmented Image 4

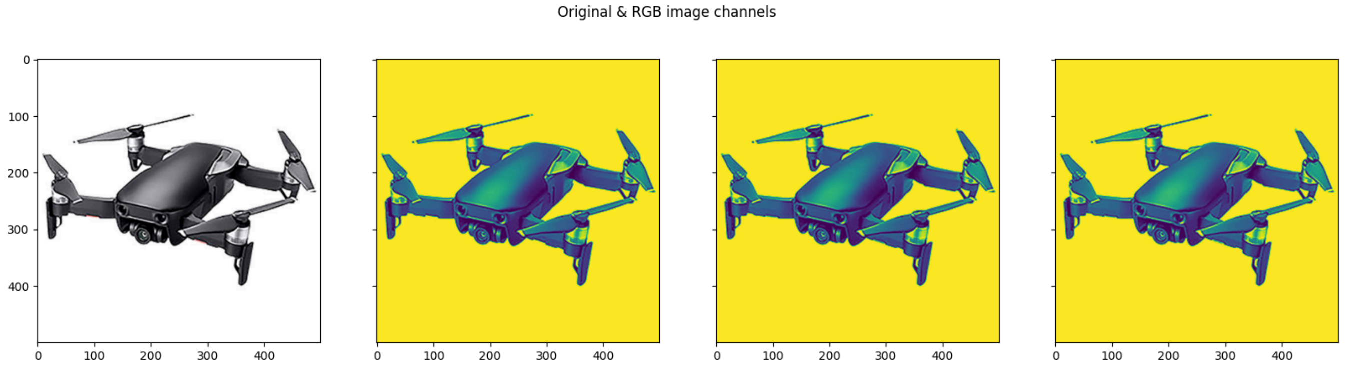


```python
from PIL import Image
image_path = '/content/drive/MyDrive/Dl/train_dataset/train_dataset/train_dataset/drone/drone 1.png'
pil_image = Image.open(image_path)

#PIL image to NumPy array
image_array = np.array(pil_image)

#plot of original ship image
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(image_array)
plt.title('Original Image')
plt.axis('off')

#The second plot with interpolation (has smoothing effect applied by the 'hanning' interpolation)
plt.figure(figsize=(5, 4))
plt.imshow(image_array, interpolation='hanning')
plt.axis('off')
plt.show()
```

```python
i, (im1, im2, im3, im4) = plt.subplots(1, 4, sharey=True)
i.set_figwidth(20)
```

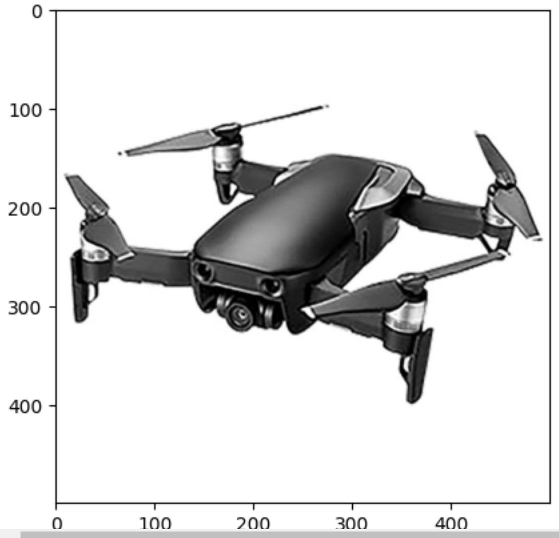
```

im1.imshow(image_array,interpolation='hanning')  #Original image
im2.imshow(image_array[:, : , 0],interpolation='hanning') #Red channel
im3.imshow(image_array[:, : , 1],interpolation='hanning') #Green channel
im4.imshow(image_array[:, : , 2],interpolation='hanning') #Blue channel
i.suptitle('Original & RGB image channels')
```

```
Text(0.5, 0.98, 'Original & RGB image channels')
```



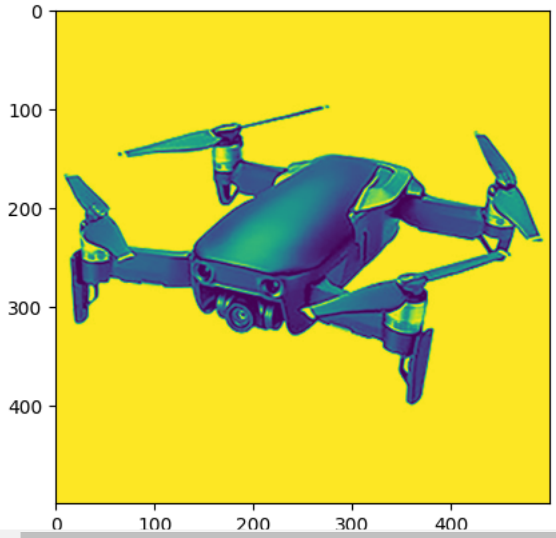
```
#convert the original RGB image to a grayscale image
# used to visualize the intensity (luminance) of an image without considering color information
from skimage import io
import skimage
gray_image = skimage.color.rgb2gray(image_array[:,:,:3])
plt.imshow(gray_image, cmap = 'gray',interpolation='hanning')
```

```
<matplotlib.image.AxesImage at 0x7bb87e0bf9d0>
```

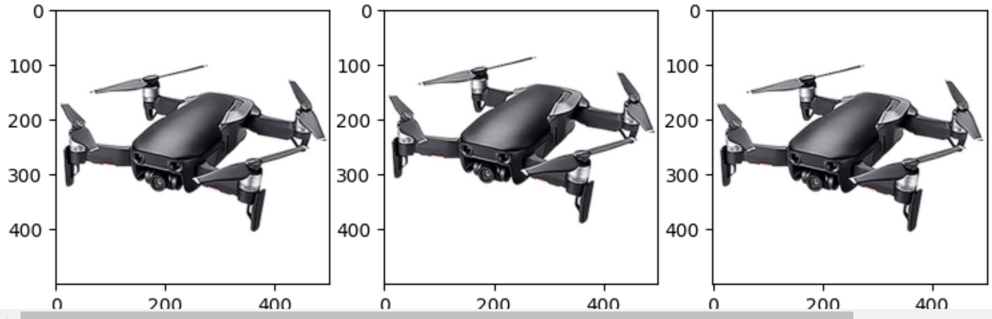


```
norm_image = (gray_image - np.min(gray_image)) / (np.max(gray_image) - np.min(gray_image))
plt.imshow(norm_image,interpolation='hanning')
```

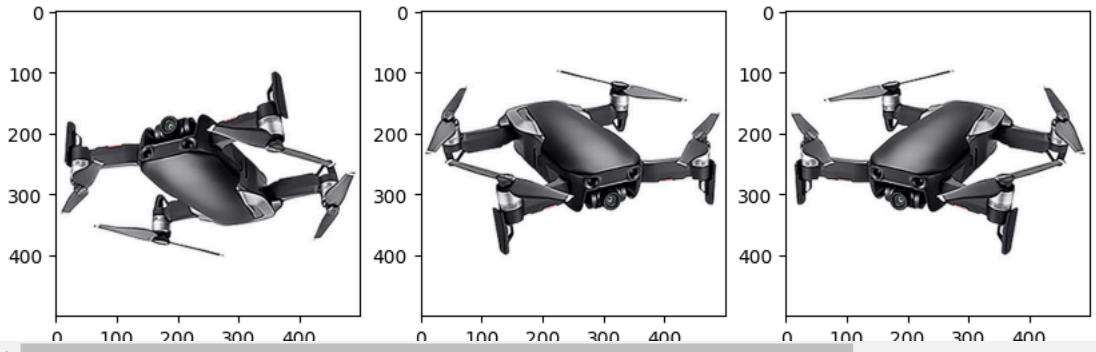
```
<matplotlib.image.AxesImage at 0x7bb878733820>
```



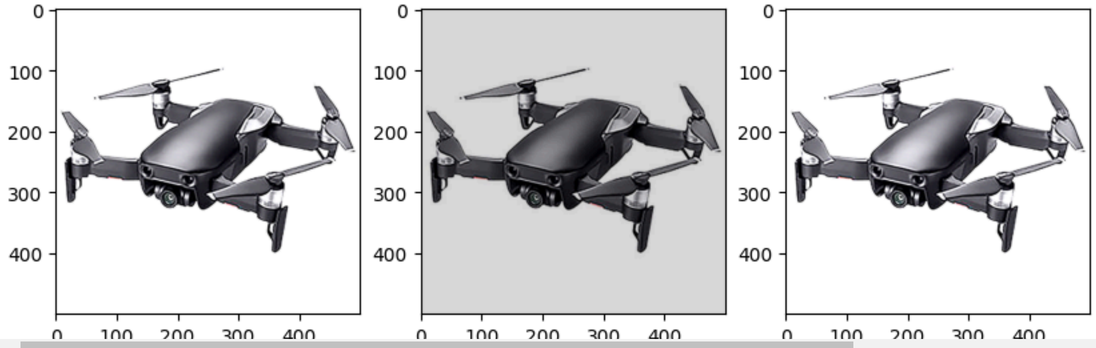
```
# Import libraries and use Keras' ImageDataGenerator for image data augmentation
from numpy import expand_dims
from keras.preprocessing.image import load_img
from keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt

# Expand dimension to one sample
samples = expand_dims(image_array, 0)

# Create image data augmentation generator
datagen = ImageDataGenerator(width_shift_range=[5, 10])

# Create an iterator
it = datagen.flow(samples, batch_size=1)
fig, im = plt.subplots(nrows=1, ncols=3, figsize=(9, 5))

# Generate batch of images
for i in range(3):
    # Convert to unsigned integers
    images = next(it)[0].astype('uint8')

    # Plot image
    im[i].imshow(images, interpolation='hanning')

plt.show()

```



```
datagen = ImageDataGenerator(horizontal_flip=True, vertical_flip=True)

# create an iterator
it = datagen.flow(samples, batch_size=1)
fig, im = plt.subplots(nrows=1, ncols=3, figsize=(10,10))

# generate batch of images
for i in range(3):

    # convert to unsigned integers
    images = next(it)[0].astype('uint8')

    # plot image
    im[i].imshow(images,interpolation='hanning')
```

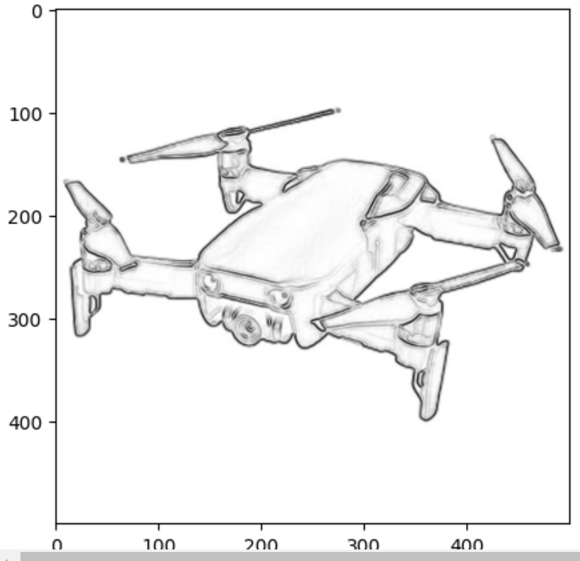


```
# ImageDataGenerator for brightness
datagen = ImageDataGenerator(brightness_range=[0.3,2.0])
# create an iterator
it = datagen.flow(samples, batch_size=1)
fig, im = plt.subplots(nrows=1, ncols=3, figsize=(10,10))

# generate batch of images
for i in range(3):

    # convert to unsigned integers
    images = next(it)[0].astype('uint8')

    # plot image
    im[i].imshow(images,interpolation='hanning')
```



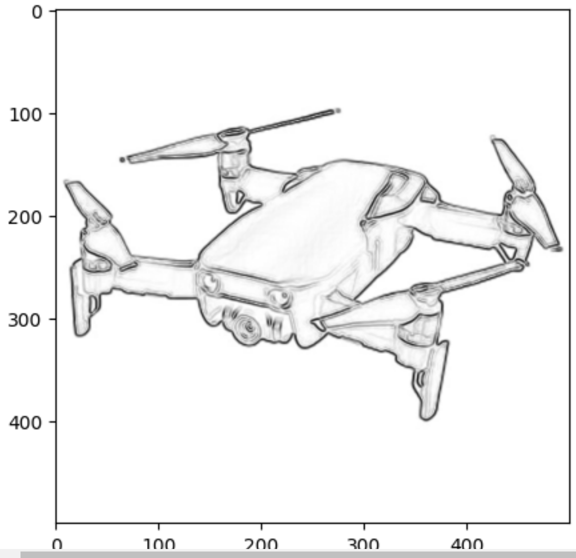
```
from skimage import io, filters, color, data
import os
plt.figure(figsize=(5,5))
img_sobel = filters.sobel(norm_image)
plt.imshow(1-img_sobel, cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x7bb877b28280>
```



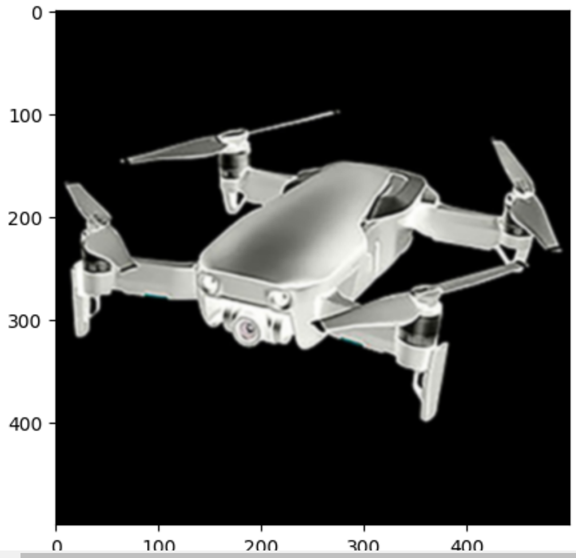
```
plt.figure(figsize=(5,5))
img_prewit = filters.prewitt(norm_image)
plt.imshow(1-img_prewit, cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x7bb8785372e0>
```

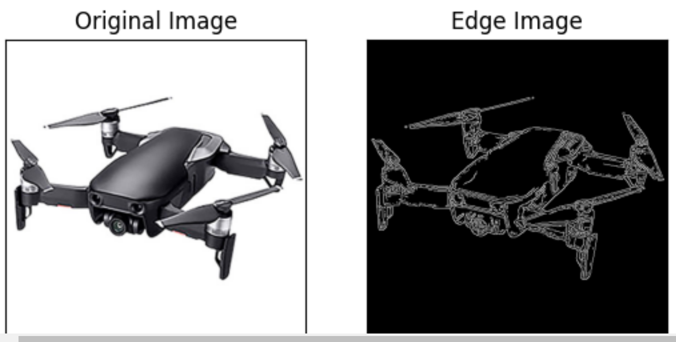


```
plt.figure(figsize=(5,5))
img_gau = filters.gaussian(image_array,channel_axis=-1)
plt.imshow(1-img_gau)
```

```
<matplotlib.image.AxesImage at 0x7bb8786d43d0>
```



```
#canny edge detection
import cv2 as cv
def hist(histimg):
    ycrcb = cv.cvtColor(image_array, cv.COLOR_RGB2YCR_CB)
    channels = cv.split(ycrcb)
    cv.equalizeHist(channels[0], channels[0])
    cv.merge(channels, ycrcb)
    cv.cvtColor(ycrcb, cv.COLOR_YCR_CB2RGB, histimg)
    return histimg

dehizing = hist(norm_image)

edges = cv.Canny(image_array,50,100)

plt.subplot(121),plt.imshow(image_array,cmap = 'gray')
plt.title('Original Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(edges,cmap = 'gray')
plt.title('Edge Image'), plt.xticks([]), plt.yticks([])

plt.show()
```



Path for traning,testing and validation datasets

```
# @title Path for traning,testing and validation datasets
train_dataset = "/content/drive/MyDrive/Dl/train_dataset/train_dataset/train_dataset"
test_dataset = "/content/drive/MyDrive/Dl/test_dataset/test_dataset/test_dataset"
validation_dataset = "/content/drive/MyDrive/Dl/validation_dataset/validation_dataset/validation_dataset"
```

Image preprocessing

```
# @title Image preprocessing
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    horizontal_flip=True,
    validation_split=0.1
)
```

loading data

```
# @title loading data
# Load Data
train_ds = train_datagen.flow_from_directory(
    train_dataset,
    class_mode='categorical',
    shuffle=True,
    batch_size=32,
    color_mode='rgb',
    target_size=(224, 224),  # Adjust the target size for VGG16
    subset='training'
)

valid_ds = train_datagen.flow_from_directory(
    validation_dataset,
    class_mode='categorical',
    shuffle=True,
    batch_size=32,
    color_mode='rgb',
    target_size=(224, 224),  # Adjust the target size for VGG16
    subset='validation'
)
```

```
Found 18 images belonging to 3 classes.
Found 0 images belonging to 3 classes.
```

VGG16 Model

```
# @title VGG16 Model
from tensorflow.keras.applications import VGG16
base_model = VGG16(include_top=False, input_shape=(224, 224, 3))  # Change the model to VGG16
base_model.trainable = False

# Model Architecture
model = Sequential([
    base_model,
    GlobalAveragePooling2D(),
    Dense(256, activation='relu', kernel_initializer='he_normal'),
    Dropout(0.4),
    Dense(len(train_ds.class_indices), activation='softmax')
], name="VGG16-TL")

model.summary()

```

```
# Compiling Model
model.compile(
    loss='categorical_crossentropy',
    optimizer=Adam(),
    metrics=['accuracy']
)
callbacks = [EarlyStopping(patience=3, restore_best_weights=True)]
history = model.fit(train_ds, validation_data=valid_ds, epochs=10, callbacks=callbacks)
```

```
Model: "VGG16-TL"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 7, 7, 512)         14714688

 global_average_pooling2d_6  (None, 512)               0
  (GlobalAveragePooling2D)

 dense_12 (Dense)            (None, 256)               131328

 dropout_6 (Dropout)         (None, 256)               0

 dense_13 (Dense)            (None, 3)                 771

=================================================================
Total params: 14846787 (56.64 MB)
Trainable params: 132099 (516.01 KB)
Non-trainable params: 14714688 (56.13 MB)
_________________________________________________________________
Epoch 1/10
1/1 [==============================] - ETA: 0s - loss: 1.2802 - accuracy: 0.2778WARNING:tensorflow:Early stopping conditioned on met
1/1 [==============================] - 12s 12s/step - loss: 1.2802 - accuracy: 0.2778
Epoch 2/10
1/1 [==============================] - ETA: 0s - loss: 1.2561 - accuracy: 0.2778WARNING:tensorflow:Early stopping conditioned on met
1/1 [==============================] - 12s 12s/step - loss: 1.2561 - accuracy: 0.2778
Epoch 3/10
1/1 [==============================] - ETA: 0s - loss: 1.1706 - accuracy: 0.3889WARNING:tensorflow:Early stopping conditioned on met
1/1 [==============================] - 11s 11s/step - loss: 1.1706 - accuracy: 0.3889
Epoch 4/10
1/1 [==============================] - ETA: 0s - loss: 1.1109 - accuracy: 0.3889WARNING:tensorflow:Early stopping conditioned on met
1/1 [==============================] - 12s 12s/step - loss: 1.1109 - accuracy: 0.3889
Epoch 5/10
1/1 [==============================] - ETA: 0s - loss: 0.8140 - accuracy: 0.7222WARNING:tensorflow:Early stopping conditioned on met
1/1 [==============================] - 12s 12s/step - loss: 0.8140 - accuracy: 0.7222
Epoch 6/10
1/1 [==============================] - ETA: 0s - loss: 1.1160 - accuracy: 0.4444WARNING:tensorflow:Early stopping conditioned on met
1/1 [==============================] - 10s 10s/step - loss: 1.1160 - accuracy: 0.4444
Epoch 7/10
1/1 [==============================] - ETA: 0s - loss: 0.8825 - accuracy: 0.7778WARNING:tensorflow:Early stopping conditioned on met
1/1 [==============================] - 12s 12s/step - loss: 0.8825 - accuracy: 0.7778
Epoch 8/10
1/1 [==============================] - ETA: 0s - loss: 0.8168 - accuracy: 0.6667WARNING:tensorflow:Early stopping conditioned on met
1/1 [==============================] - 12s 12s/step - loss: 0.8168 - accuracy: 0.6667
Epoch 9/10
1/1 [==============================] - ETA: 0s - loss: 0.7871 - accuracy: 0.5556WARNING:tensorflow:Early stopping conditioned on met
1/1 [==============================] - 10s 10s/step - loss: 0.7871 - accuracy: 0.5556
Epoch 10/10
1/1 [==============================] - ETA: 0s - loss: 0.9508 - accuracy: 0.6111WARNING:tensorflow:Early stopping conditioned on met
1/1 [==============================] - 12s 12s/step - loss: 0.9508 - accuracy: 0.6111
```

Testing accuray and loss of VGG16 Model

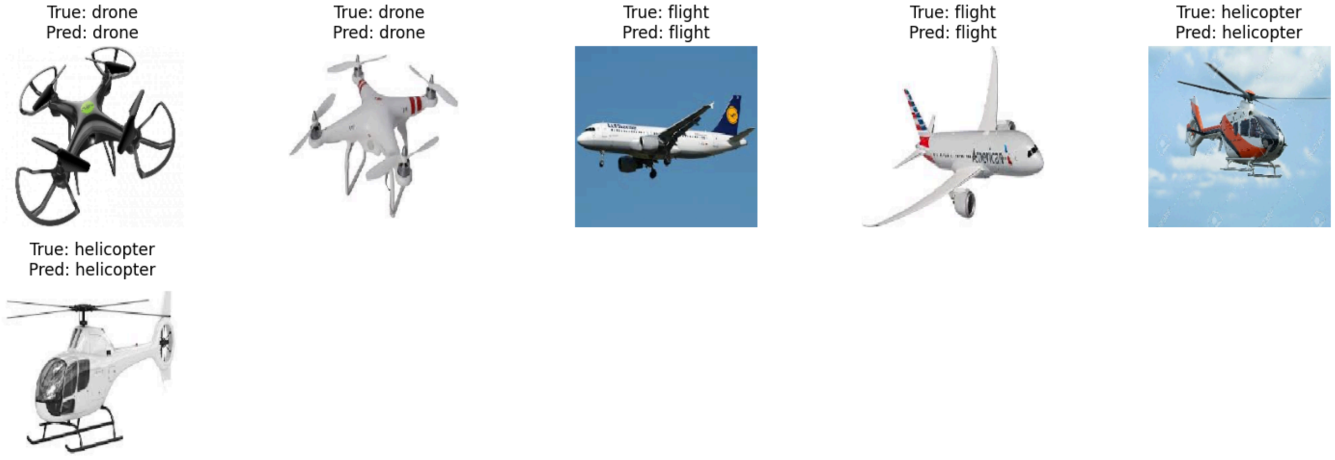
```
# @title Testing accuray and loss of VGG16 Model
# Load the test dataset
test_datagen = ImageDataGenerator(rescale=1./255)

test_ds = test_datagen.flow_from_directory(
    test_dataset,
    class_mode='categorical',
    shuffle=False,
    batch_size=32,
    color_mode='rgb',
    target_size=(224, 224)  # Adjust the target size for VGG16
)

# Evaluate the model on the test dataset
test_loss, test_accuracy = model.evaluate(test_ds)

print(f'Test Loss: {test_loss:.4f}')
print(f'Test Accuracy: {test_accuracy * 100:.2f}%')
```

```
Found 6 images belonging to 3 classes.
1/1 [==============================] - 3s 3s/step - loss: 0.8455 - accuracy: 1.0000
Test Loss: 0.8455
Test Accuracy: 100.00%
```

Testing the VGG16 Model

```python
# @title Testing the VGG16 Model
# Set the path to your test dataset
test_dataset = "/content/drive/MyDrive/Dl/test_dataset/test_dataset/test_dataset"

# Initialize the ImageDataGenerator for testing (only rescaling)
test_datagen = ImageDataGenerator(rescale=1./255)

# Load Test Data
test_ds = test_datagen.flow_from_directory(
    test_dataset,
    class_mode='categorical',
    shuffle=False,
    batch_size=1,  # Adjust to the actual number of images in your test dataset
    color_mode='rgb',
    target_size=(224, 224)  # Adjust the target size for VGG16
)

# Get class indices
class_indices = {v: k for k, v in test_ds.class_indices.items()}

# Visualize predictions on a few test images
plt.figure(figsize=(15, 10))
for i in range(len(test_ds)):
    plt.subplot(4, 5, i+1)
    img, true_label = test_ds[i]
    pred_probs = model.predict(img)
    pred_label = np.argmax(pred_probs, axis=1)[0]

    plt.imshow(img[0])
    plt.title(f"True: {class_indices[true_label[0].argmax()]}\nPred: {class_indices[pred_label]}")
    plt.axis('off')

plt.tight_layout()
plt.show()
```

```
Found 6 images belonging to 3 classes.
1/1 [==============================] - 1s 933ms/step
1/1 [==============================] - 0s 477ms/step
1/1 [==============================] - 0s 492ms/step
1/1 [==============================] - 0s 481ms/step
1/1 [==============================] - 0s 491ms/step
1/1 [==============================] - 0s 495ms/step
```



True: drone
Pred: drone

True: drone
Pred: drone

True: flight
Pred: flight

True: flight
Pred: flight

True: helicopter
Pred: helicopter

True: helicopter
Pred: helicopter

```python
from tensorflow.keras.applications import InceptionV3
# Set the paths to your dataset
train_dataset = "/content/drive/MyDrive/Dl/train_dataset/train_dataset/train_dataset"
test_dataset = "/content/drive/MyDrive/Dl/test_dataset/test_dataset/test_dataset"
validation_dataset = "/content/drive/MyDrive/Dl/validation_dataset/validation_dataset/validation_dataset"
```

InceptionV3 Model

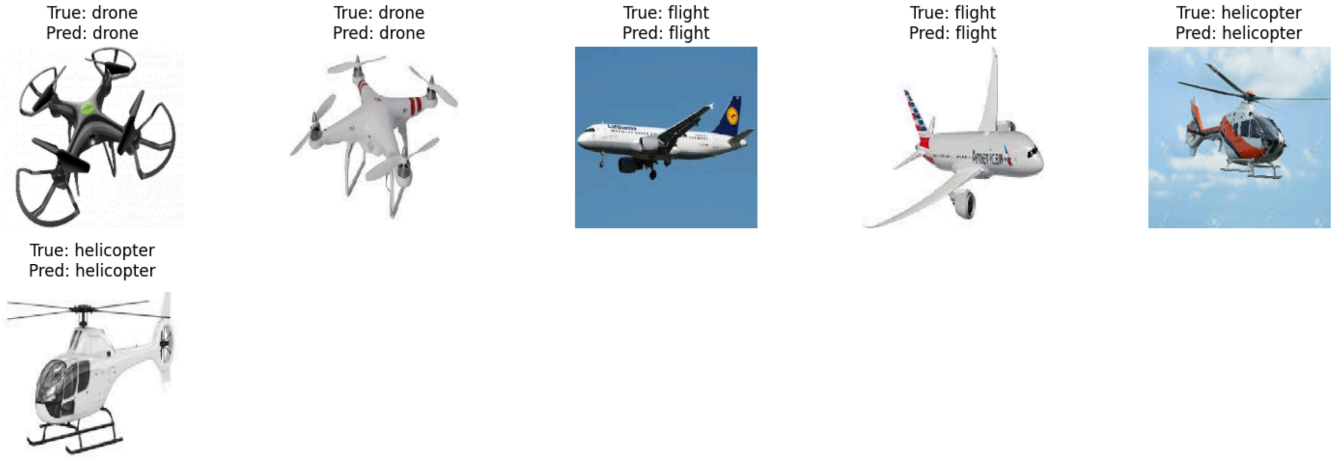
```
# @title InceptionV3 Model
# Load Pretrained InceptionV3 Model
base_model = InceptionV3(include_top=False, input_shape=(256, 256, 3))
base_model.trainable = False

# Model Architecture
model = Sequential([
    base_model,
    GlobalAveragePooling2D(),
    Dense(256, activation='relu', kernel_initializer='he_normal'),
    Dropout(0.4),
    Dense(len(train_ds.class_indices), activation='softmax')
], name="Inception-TL")

model.summary()

# Compile Model
model.compile(
    loss='categorical_crossentropy',
    optimizer=Adam(),
    metrics=['accuracy']
)

# Callbacks
callbacks = [EarlyStopping(patience=3, restore_best_weights=True)]

# Model Training
history = model.fit(train_ds, validation_data=valid_ds, epochs=10, callbacks=callbacks)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/inception_v3_weights_tf_dim_ordering
87910968/87910968 [==============================] - 1s 0us/step
Model: "Inception-TL"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inception_v3 (Functional)   (None, 6, 6, 2048)        21802784

 global_average_pooling2d_1  (None, 2048)              0
  (GlobalAveragePooling2D)

 dense_2 (Dense)             (None, 256)               524544

 dropout_1 (Dropout)         (None, 256)               0

 dense_3 (Dense)             (None, 3)                 771

=================================================================
Total params: 22328099 (85.17 MB)
Trainable params: 525315 (2.00 MB)
Non-trainable params: 21802784 (83.17 MB)
_________________________________________________________________
Epoch 1/10
1/1 [==============================] - ETA: 0s - loss: 1.4728 - accuracy: 0.2778WARNING:tensorflow:Early stopping conditioned on met
1/1 [==============================] - 14s 14s/step - loss: 1.4728 - accuracy: 0.2778
Epoch 2/10
1/1 [==============================] - ETA: 0s - loss: 0.6912 - accuracy: 0.7222WARNING:tensorflow:Early stopping conditioned on met
1/1 [==============================] - 4s 4s/step - loss: 0.6912 - accuracy: 0.7222
Epoch 3/10
1/1 [==============================] - ETA: 0s - loss: 0.3377 - accuracy: 0.8889WARNING:tensorflow:Early stopping conditioned on met
1/1 [==============================] - 3s 3s/step - loss: 0.3377 - accuracy: 0.8889
Epoch 4/10
1/1 [==============================] - ETA: 0s - loss: 0.2638 - accuracy: 0.8889WARNING:tensorflow:Early stopping conditioned on met
1/1 [==============================] - 4s 4s/step - loss: 0.2638 - accuracy: 0.8889
Epoch 5/10
1/1 [==============================] - ETA: 0s - loss: 0.1216 - accuracy: 0.9444WARNING:tensorflow:Early stopping conditioned on met
1/1 [==============================] - 4s 4s/step - loss: 0.1216 - accuracy: 0.9444
Epoch 6/10
1/1 [==============================] - ETA: 0s - loss: 0.1675 - accuracy: 0.9444WARNING:tensorflow:Early stopping conditioned on met
1/1 [==============================] - 3s 3s/step - loss: 0.1675 - accuracy: 0.9444
Epoch 7/10
1/1 [==============================] - ETA: 0s - loss: 0.0291 - accuracy: 1.0000WARNING:tensorflow:Early stopping conditioned on met
1/1 [==============================] - 4s 4s/step - loss: 0.0291 - accuracy: 1.0000
Epoch 8/10
1/1 [==============================] - ETA: 0s - loss: 0.0317 - accuracy: 1.0000WARNING:tensorflow:Early stopping conditioned on met
1/1 [==============================] - 5s 5s/step - loss: 0.0317 - accuracy: 1.0000
Epoch 9/10
1/1 [==============================] - ETA: 0s - loss: 0.0120 - accuracy: 1.0000WARNING:tensorflow:Early stopping conditioned on met
1/1 [==============================] - 3s 3s/step - loss: 0.0120 - accuracy: 1.0000
Epoch 10/10
1/1 [==============================] - ETA: 0s - loss: 0.0149 - accuracy: 1.0000WARNING:tensorflow:Early stopping conditioned on met
1/1 [==============================] - 3s 3s/step - loss: 0.0149 - accuracy: 1.0000
```

Testing the InceptionV3 Model

```
# @title Testing the InceptionV3 Model
# Set the path to your test dataset
test_dataset = "/content/drive/MyDrive/Dl/test_dataset/test_dataset/test_dataset"

# Initialize the ImageDataGenerator for testing (only rescaling)
test_datagen = ImageDataGenerator(rescale=1./255)

# Load Test Data
test_ds = test_datagen.flow_from_directory(
    test_dataset,
    class_mode='categorical',
    shuffle=False,
    batch_size=1,  # Adjust to the actual number of images in your test dataset
    color_mode='rgb',
    target_size=(256, 256)
)

# Evaluate the model on the test set
test_loss, test_accuracy = model.evaluate(test_ds)
print(f'InceptionV3 Test Loss: {inception_test_loss:.4f}')
print(f'InceptionV3 Test Accuracy: {inception_test_accuracy * 100:.2f}%')
# Get class indices
class_indices = {v: k for k, v in test_ds.class_indices.items()}

# Visualize predictions on a few test images
plt.figure(figsize=(15, 10))
for i in range(len(test_ds)):
    plt.subplot(4, 5, i+1)
    img, true_label = test_ds[i]
    pred_probs = model.predict(img)
    pred_label = np.argmax(pred_probs, axis=1)[0]

    plt.imshow(img[0])
    plt.title(f"True: {class_indices[true_label[0].argmax()]}\nPred: {class_indices[pred_label]}")
    plt.axis('off')

plt.tight_layout()
plt.show()
```

```
Found 6 images belonging to 3 classes.
6/6 [==============================] - 5s 755ms/step - loss: 0.8641 - accuracy: 1.0000
InceptionV3 Test Loss: 0.8455
InceptionV3 Test Accuracy: 100.00%
1/1 [==============================] - 1s 782ms/step
1/1 [==============================] - 1s 640ms/step
1/1 [==============================] - 1s 623ms/step
1/1 [==============================] - 1s 632ms/step
1/1 [==============================] - 1s 638ms/step
1/1 [==============================] - 1s 622ms/step
```



Comparing the VGG16 and InceptionV3

```
# @title Comparing the VGG16 and InceptionV3
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, classification_report
# Evaluate the VGG16 model on the test set
```

```
vgg_test_loss, vgg_test_accuracy = model.evaluate(test_ds)
print(f'VGG16 Test Loss: {vgg_test_loss:.4f}')
print(f'VGG16 Test Accuracy: {vgg_test_accuracy * 100:.2f}%')

# Evaluate the InceptionV3 model on the test set
inception_test_loss, inception_test_accuracy = model.evaluate(test_ds)
print(f'InceptionV3 Test Loss: {inception_test_loss:.4f}')
print(f'InceptionV3 Test Accuracy: {inception_test_accuracy * 100:.2f}%')
```

```
6/6 [==============================] - 4s 614ms/step - loss: 0.8455 - accuracy: 1.0000
VGG16 Test Loss: 0.8455
VGG16 Test Accuracy: 100.00%
6/6 [==============================] - 5s 837ms/step - loss: 0.8455 - accuracy: 1.0000
InceptionV3 Test Loss: 0.8455
InceptionV3 Test Accuracy: 100.00%
```

```
# VGG16 Model

# Load Pretrained VGG16 Model
base_model_vgg16 = VGG16(include_top=False, input_shape=(224, 224, 3))  # Change the model to VGG16
base_model_vgg16.trainable = False

# Model Architecture
model_vgg16 = Sequential([
    base_model_vgg16,
    GlobalAveragePooling2D(),
    Dense(256, activation='relu', kernel_initializer='he_normal'),
    Dropout(0.4),
    Dense(len(train_ds.class_indices), activation='softmax')
], name="VGG16-TL")

model_vgg16.summary()

# Compile Model
model_vgg16.compile(
    loss='categorical_crossentropy',
    optimizer=Adam(),
    metrics=['accuracy']
)

# Training for VGG16
history_vgg16 = model_vgg16.fit(train_ds, validation_data=valid_ds, epochs=10, callbacks=callbacks)

# InceptionV3 Model

# Load Pretrained InceptionV3 Model
base_model_inception = InceptionV3(include_top=False, input_shape=(256, 256, 3))
base_model_inception.trainable = False

# Model Architecture
model_inception = Sequential([
    base_model_inception,
    GlobalAveragePooling2D(),
    Dense(256, activation='relu', kernel_initializer='he_normal'),
    Dropout(0.4),
    Dense(len(train_ds.class_indices), activation='softmax')
], name="Inception-TL")

model_inception.summary()

# Compile Model
model_inception.compile(
    loss='categorical_crossentropy',
    optimizer=Adam(),
    metrics=['accuracy']
)

# Training for InceptionV3
history_inception = model_inception.fit(train_ds, validation_data=valid_ds, epochs=10, callbacks=callbacks)

# Evaluation

# Predict and evaluate on the training set for VGG16
train_accuracy_vgg16 = model_vgg16.evaluate(train_ds)[1]
print(f"Train Accuracy (VGG16): {train_accuracy_vgg16}")

# Predict and evaluate on the test set for VGG16
test_accuracy_vgg16 = model_vgg16.evaluate(test_ds)[1]
print(f"Test Accuracy (VGG16): {test_accuracy_vgg16}")

# Predict and evaluate on the training set for InceptionV3
train_accuracy_inception = model_inception.evaluate(train_ds)[1]
print(f"Train Accuracy (InceptionV3): {train_accuracy_inception}")
print(f"Train Accuracy (InceptionV3): {train_accuracy_inception}")

# Predict and evaluate on the test set for InceptionV3
test_accuracy_inception = model_inception.evaluate(test_ds)[1]
print(f"Test Accuracy (InceptionV3): {test_accuracy_inception}")

```

```
Model: "VGG16-TL"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 vgg16 (Functional)          (None, 7, 7, 512)         14714688  
                                                                 
 global_average_pooling2d_9  (None, 512)               0         
  (GlobalAveragePooling2D)                                       
                                                                 
 dense_18 (Dense)            (None, 256)               131328    
                                                                 
 dropout_9 (Dropout)         (None, 256)               0         
                                                                 
 dense_19 (Dense)            (None, 3)                 771       
                                                                 
=================================================================
Total params: 14846787 (56.64 MB)
Trainable params: 132099 (516.01 KB)
Non-trainable params: 14714688 (56.13 MB)
_________________________________________________________________
Epoch 1/10
1/1 [==============================] - ETA: 0s - loss: 1.1013 - accuracy: 0.2778WARNING:tensorflow:Early stopping conditioned on
1/1 [==============================] - 15s 15s/step - loss: 1.1013 - accuracy: 0.2778
Epoch 2/10
1/1 [==============================] - ETA: 0s - loss: 1.1338 - accuracy: 0.3889WARNING:tensorflow:Early stopping conditioned on
1/1 [==============================] - 10s 10s/step - loss: 1.1338 - accuracy: 0.3889
Epoch 3/10
1/1 [==============================] - ETA: 0s - loss: 0.8591 - accuracy: 0.6667WARNING:tensorflow:Early stopping conditioned on
1/1 [==============================] - 12s 12s/step - loss: 0.8591 - accuracy: 0.6667
Epoch 4/10
1/1 [==============================] - ETA: 0s - loss: 0.9937 - accuracy: 0.3333WARNING:tensorflow:Early stopping conditioned on
1/1 [==============================] - 10s 10s/step - loss: 0.9937 - accuracy: 0.3333
Epoch 5/10
1/1 [==============================] - ETA: 0s - loss: 0.9740 - accuracy: 0.5556WARNING:tensorflow:Early stopping conditioned on
1/1 [==============================] - 12s 12s/step - loss: 0.9740 - accuracy: 0.5556
Epoch 6/10
1/1 [==============================] - ETA: 0s - loss: 0.7966 - accuracy: 0.6111WARNING:tensorflow:Early stopping conditioned on
1/1 [==============================] - 10s 10s/step - loss: 0.7966 - accuracy: 0.6111
Epoch 7/10
1/1 [==============================] - ETA: 0s - loss: 0.8530 - accuracy: 0.5556WARNING:tensorflow:Early stopping conditioned on
1/1 [==============================] - 12s 12s/step - loss: 0.8530 - accuracy: 0.5556
Epoch 8/10
1/1 [==============================] - ETA: 0s - loss: 0.9500 - accuracy: 0.5556WARNING:tensorflow:Early stopping conditioned on
1/1 [==============================] - 11s 11s/step - loss: 0.9500 - accuracy: 0.5556
Epoch 9/10
1/1 [==============================] - ETA: 0s - loss: 0.9261 - accuracy: 0.5556WARNING:tensorflow:Early stopping conditioned on
1/1 [==============================] - 12s 12s/step - loss: 0.9261 - accuracy: 0.5556
Epoch 10/10
1/1 [==============================] - ETA: 0s - loss: 0.9792 - accuracy: 0.5556WARNING:tensorflow:Early stopping conditioned on
1/1 [==============================] - 11s 11s/step - loss: 0.9792 - accuracy: 0.5556
Model: "Inception-TL"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 inception_v3 (Functional)   (None, 6, 6, 2048)        21802784  
                                                                 
 global_average_pooling2d_1  (None, 2048)              0         
```

```
# Classification metrics and confusion matrix
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Function to plot training history
def plot_training_history(history, model_name):
    plt.figure(figsize=(12, 4))

    # Plot training & validation accuracy values
    plt.subplot(1, 2, 1)
    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
    plt.title(f'{model_name} Model Accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend(['Train', 'Validation'], loc='upper left')

    # Plot training & validation loss values
    plt.subplot(1, 2, 2)
```