Name: Soumya Ranjan Nayak

PRN: 23070243063

DL_ASM: Image Augmentation Operations

```
# Import necessary libraries
import os
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.optimizers import Adam
from sklearn.metrics import classification_report

import tensorflow as tf
tf.get_logger().setLevel('ERROR')
```

```
import os
import random
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

# Path to the working directory where the dataset is saved
working_directory = '/kaggle/working/vehicle_images'

# Subdirectories in the working directory
subdirectories = ['Bike Images', 'Plane Image', 'Car Images']

# Set up a plot to display the images
plt.figure(figsize=(15, 5))

# Loop through the subdirectories
for i, subdir in enumerate(subdirectories):
    subdir_path = os.path.join(working_directory, subdir)

    # Get a list of all image files in the current subdirectory
    files = os.listdir(subdir_path)

    # Select a random image from the subdirectory
    random_image = random.choice(files)

    # Load and display the selected image
    img = mpimg.imread(os.path.join(subdir_path, random_image))
    plt.subplot(1, len(subdirectories), i+1)
    plt.imshow(img)
    plt.title(f'{subdir} - {random_image}')
    plt.axis('off')  # Hide axes for better visualization

# Show the images
plt.show()
```

Bike Images - Bike (1066).jpeg

Plane Image - airplane157.jpg

Car Images - 1233.jpg

```
import os

# Base dataset path (update based on your dataset name in Kaggle)
base_path = '/kaggle/input/multi-vehicle-image-car-plane-bike/Images'

# Check subdirectories
categories = os.listdir(base_path)
print("Categories:", categories)

# Explore one category
sample_category = categories[0]
sample_path = os.path.join(base_path, sample_category)
print(f"Images in {sample_category}:", os.listdir(sample_path)[:5])

# Visualize a sample image
from tensorflow.keras.preprocessing.image import load_img
```

```
import matplotlib.pyplot as plt

sample_image_path = os.path.join(sample_path, os.listdir(sample_path)[0])
img = load_img(sample_image_path)
plt.imshow(img)
plt.title(f"Sample Image - {sample_category}")
plt.axis('off')
plt.show()
```

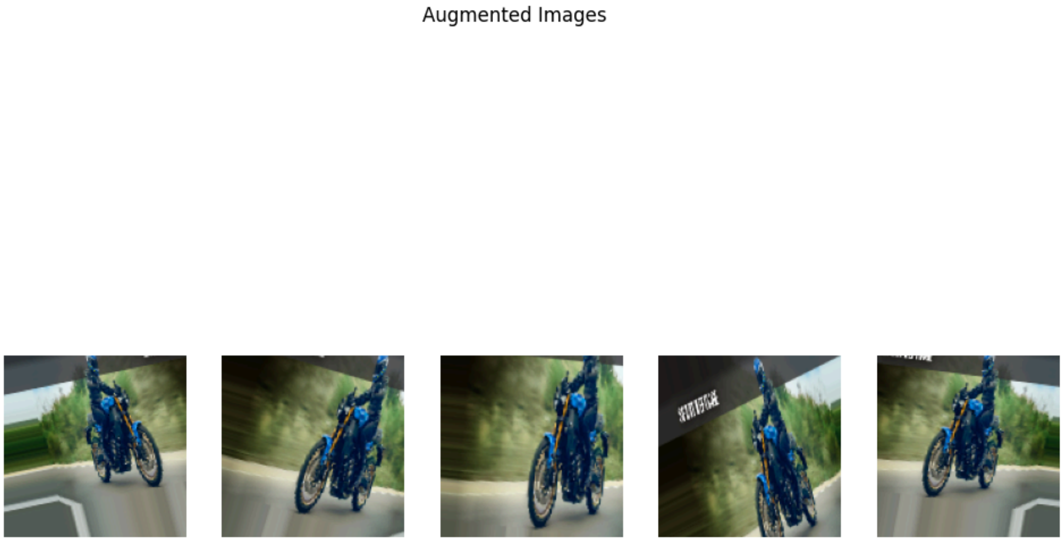
```
Categories: ['Bike Images', 'Plane Image', 'Car Images']
Images in Bike Images: ['Bike (6).jpg', 'bike image17.png', 'bike3 image01 (1).png', 'Bike (648).jpeg', 'bike3 image21 (1).jpeg']
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

img_height, img_width = 128, 128
batch_size = 32

datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)

# Training set
train_generator = datagen.flow_from_directory(
    base_path,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'
)

# Validation set
validation_generator = datagen.flow_from_directory(
    base_path,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation'
)
```

```
Found 3281 images belonging to 3 classes.
Found 819 images belonging to 3 classes.
```

```
import os
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array, array_to_img
import matplotlib.pyplot as plt

# Base path to dataset
base_path = '/kaggle/input/multi-vehicle-image-car-plane-bike/Images'

print("Dataset structure:")
for root, dirs, _ in os.walk(base_path):
    print("Directory:", root)
    print("Subdirectories:", dirs)
    print("-" * 50)

sample_image_path = os.path.join(base_path, 'Bike Images', 'Bike (6).jpg')

try:
    img = load_img(sample_image_path, target_size=(128, 128))
    img_array = img_to_array(img)
    img_array = img_array.reshape((1,) + img_array.shape)

    # Step 3: Define ImageDataGenerator with augmentations
    datagen = ImageDataGenerator(
        rescale=1.0 / 255,
        rotation_range=20,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True
    )

    # Step 4: Generate and visualize augmented images
    print("Displaying augmented images:")
    i = 0
    plt.figure(figsize=(12, 8))
    for batch in datagen.flow(img_array, batch_size=1):
        plt.subplot(1, 5, i + 1)
        augmented_image = array_to_img(batch[0])  # Convert back to image format
        plt.imshow(augmented_image)
        plt.axis('off')
        i += 1
        if i >= 5:
            break
    plt.suptitle("Augmented Images")
    plt.show()

except FileNotFoundError as e:
    print("File not found:", e)
    print("image not exists in the dataset.")

folders = ['Bike Images', 'Car Images', 'Plane Image']
for folder in folders:
    folder_path = os.path.join(base_path, folder)
    print(f"Contents of {folder}: {len(os.listdir(folder_path))} files")
```

```
Dataset structure:
Directory: /kaggle/input/multi-vehicle-image-car-plane-bike/Images
Subdirectories: ['Bike Images', 'Plane Image', 'Car Images']
--------------------------------------------------
Directory: /kaggle/input/multi-vehicle-image-car-plane-bike/Images/Bike Images
Subdirectories: []
--------------------------------------------------
Directory: /kaggle/input/multi-vehicle-image-car-plane-bike/Images/Plane Image
Subdirectories: []
--------------------------------------------------
Directory: /kaggle/input/multi-vehicle-image-car-plane-bike/Images/Car Images
Subdirectories: []
--------------------------------------------------
Displaying augmented images:
```



```
Contents of Bike Images: 1367 files
Contents of Car Images: 1367 files
Contents of Plane Image: 1367 files
```

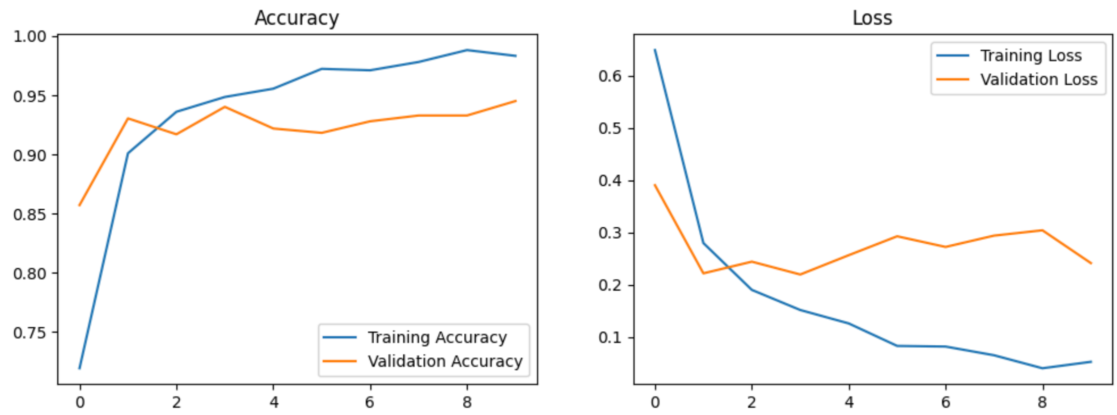
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

# Define the CNN model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(img_height, img_width, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(len(categories), activation='softmax')  # Output layer (one node per class)
])

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

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

```
/opt/conda/lib/python3.10/site-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ conv2d (Conv2D)                 │ (None, 126, 126, 32)   │           896 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d (MaxPooling2D)    │ (None, 63, 63, 32)     │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_1 (Conv2D)               │ (None, 61, 61, 64)     │        18,496 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_1 (MaxPooling2D)  │ (None, 30, 30, 64)     │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_2 (Conv2D)               │ (None, 28, 28, 128)    │        73,856 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_2 (MaxPooling2D)  │ (None, 14, 14, 128)    │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ flatten (Flatten)               │ (None, 25088)          │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                   │ (None, 128)            │     3,211,392 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout (Dropout)               │ (None, 128)            │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_1 (Dense)                 │ (None, 3)              │           387 │
└─────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 3,305,027 (12.61 MB)
 Trainable params: 3,305,027 (12.61 MB)
```

```
# Train the model
history = model.fit(
    train_generator,
    validation_data=validation_generator,
    epochs=10  # Adjust based on your dataset size and requirements
)
```

```
Epoch 1/10
/opt/conda/lib/python3.10/site-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super()
  self._warn_if_super_not_called()
 29/103 ━━━━━━━━━━━━━━━━━━━━ 41s 565ms/step - accuracy: 0.3897 - loss: 1.1696/opt/conda/lib/python3.10/site-packages/PIL/Image.py:1056: UserWarning: Palette
  warnings.warn(
103/103 ━━━━━━━━━━━━━━━━━━━━ 69s 640ms/step - accuracy: 0.5601 - loss: 0.9048 - val_accuracy: 0.8571 - val_loss: 0.3903
Epoch 2/10
103/103 ━━━━━━━━━━━━━━━━━━━━ 64s 608ms/step - accuracy: 0.8858 - loss: 0.3165 - val_accuracy: 0.9304 - val_loss: 0.2217
Epoch 3/10
103/103 ━━━━━━━━━━━━━━━━━━━━ 64s 613ms/step - accuracy: 0.9286 - loss: 0.2022 - val_accuracy: 0.9170 - val_loss: 0.2440
Epoch 4/10
103/103 ━━━━━━━━━━━━━━━━━━━━ 62s 591ms/step - accuracy: 0.9535 - loss: 0.1426 - val_accuracy: 0.9402 - val_loss: 0.2194
Epoch 5/10
103/103 ━━━━━━━━━━━━━━━━━━━━ 63s 601ms/step - accuracy: 0.9536 - loss: 0.1320 - val_accuracy: 0.9219 - val_loss: 0.2563
Epoch 6/10
103/103 ━━━━━━━━━━━━━━━━━━━━ 63s 601ms/step - accuracy: 0.9726 - loss: 0.0844 - val_accuracy: 0.9182 - val_loss: 0.2926
Epoch 7/10
103/103 ━━━━━━━━━━━━━━━━━━━━ 62s 595ms/step - accuracy: 0.9673 - loss: 0.0831 - val_accuracy: 0.9280 - val_loss: 0.2721
Epoch 8/10
103/103 ━━━━━━━━━━━━━━━━━━━━ 63s 599ms/step - accuracy: 0.9790 - loss: 0.0609 - val_accuracy: 0.9328 - val_loss: 0.2938
Epoch 9/10
103/103 ━━━━━━━━━━━━━━━━━━━━ 63s 601ms/step - accuracy: 0.9859 - loss: 0.0441 - val_accuracy: 0.9328 - val_loss: 0.3040
Epoch 10/10
103/103 ━━━━━━━━━━━━━━━━━━━━ 63s 600ms/step - accuracy: 0.9876 - loss: 0.0445 - val_accuracy: 0.9451 - val_loss: 0.2412
```

```
# Visualize training history
plt.figure(figsize=(12, 4))

# Plot accuracy
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.legend()
plt.title('Accuracy')

# Plot loss
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend()
plt.title('Loss')

plt.show()
```



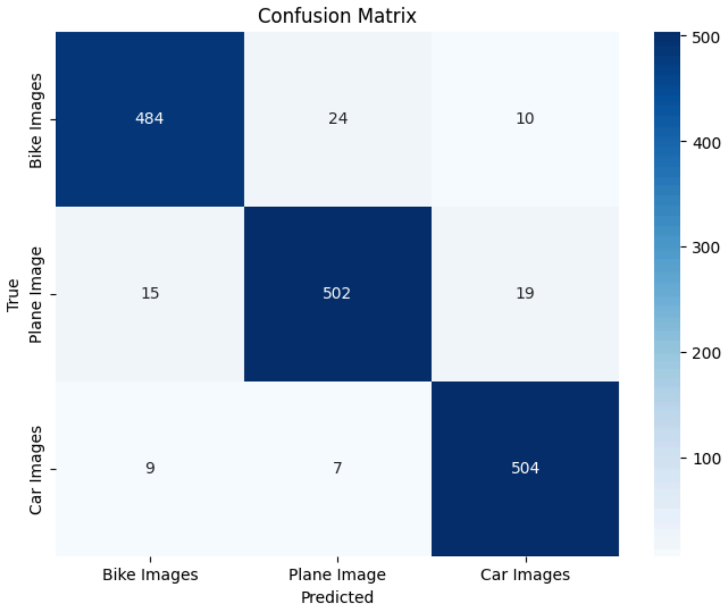
```
loss, accuracy = model.evaluate(validation_generator, verbose=0)
print(f"Validation Accuracy: {accuracy * 100:.2f}%")
```

```
Validation Accuracy: 94.51%
```

```
# Save the model
model.save('/kaggle/working/vehicle_classification_model.h5')
print("Model saved successfully!")
```

```
Model saved successfully!
```

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

y_true = []
y_pred = []

num_batches = 50

# Step 2: Loop through the validation data generator
for i, (images, labels) in enumerate(validation_generator):
    if i >= num_batches:
        break

    predictions = model.predict(images, verbose=0)
    predicted_labels = np.argmax(predictions, axis=1)

    y_true.extend(np.argmax(labels, axis=1))
    y_pred.extend(predicted_labels)

cm = confusion_matrix(y_true, y_pred)

plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=categories, yticklabels=categories)
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()

accuracy = np.sum(np.array(y_true) == np.array(y_pred)) / len(y_true)
print(f"Subset Accuracy: {accuracy * 100:.2f}%")
```



```
Subset Accuracy: 94.66%
```

```
num_batches = 10  # Limiting the number of batches
y_true = []
y_pred = []

for i, (images, labels) in enumerate(validation_generator):
    if i >= num_batches:
        break
    predictions = model.predict(images, verbose=0)
    predicted_labels = np.argmax(predictions, axis=1)
    y_true.extend(np.argmax(labels, axis=1))
    y_pred.extend(predicted_labels)

report = classification_report(y_true, y_pred, target_names=validation_generator.class_indices.keys())
print(report)
```

```
              precision    recall  f1-score   support

 Bike Images       0.94      0.93      0.93       100
  Car Images       0.95      0.92      0.94       115
 Plane Image       0.93      0.97      0.95       105

    accuracy                           0.94       320
   macro avg       0.94      0.94      0.94       320
weighted avg       0.94      0.94      0.94       320
```

```
import os
import shutil

# Base path to the dataset (Kaggle input directory)
base_path = '/kaggle/input/multi-vehicle-image-car-plane-bike/Images'

# Working directory to copy the dataset to (for easier access)
working_directory = '/kaggle/working/vehicle_images'

# Create a new directory to store the images if it doesn't exist
if not os.path.exists(working_directory):
    os.makedirs(working_directory)

# Subdirectories in the dataset
subdirectories = ['Bike Images', 'Plane Image', 'Car Images']

# Loop through the subdirectories and copy images to the working directory
for subdir in subdirectories:
    subdir_path = os.path.join(base_path, subdir)
    target_subdir = os.path.join(working_directory, subdir)

    # Create subdirectory inside the working directory
    if not os.path.exists(target_subdir):
        os.makedirs(target_subdir)

    # List all files in the current subdirectory
    files = os.listdir(subdir_path)

    # Copy all images from the dataset subdirectory to the target subdirectory
    for file in files:
        source_file = os.path.join(subdir_path, file)
        target_file = os.path.join(target_subdir, file)

        # Copy the file to the new directory
        shutil.copy(source_file, target_file)

print(f"Dataset copied successfully to {working_directory}")
```

```
Dataset copied successfully to /kaggle/working/vehicle_images
```

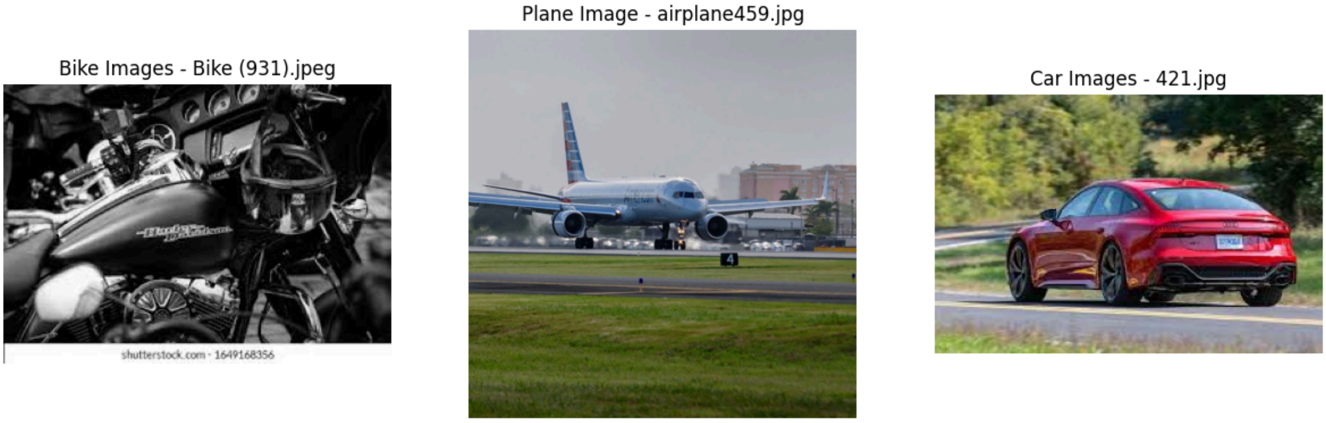
```
# List the files in the new working directory
for subdir in subdirectories:
    subdir_path = os.path.join(working_directory, subdir)
    files = os.listdir(subdir_path)
    print(f"Images in {subdir}:")
    for file in files:
        print(os.path.join(subdir_path, file))
    print("\n")
```

```
Images in Bike Images:
/kaggle/working/vehicle_images/Bike Images/Bike (965).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (1384).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (1389).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (1343).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (510).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (554).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (1448).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (726).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (652).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (615).jpeg
/kaggle/working/vehicle_images/Bike Images/bike image03.png
/kaggle/working/vehicle_images/Bike Images/Bike (31).jpg
/kaggle/working/vehicle_images/Bike Images/Bike (1346).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (84).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (830).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (698).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (1000).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (278).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (1211).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (1047).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (209).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (1370).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (875).jpeg
/kaggle/working/vehicle_images/Bike Images/bike1 image06.png
/kaggle/working/vehicle_images/Bike Images/Bike (21).png
/kaggle/working/vehicle_images/Bike Images/Bike (948).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (1375).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (203).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (206).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (8).jpg
/kaggle/working/vehicle_images/Bike Images/bike3 image05 (2).png
/kaggle/working/vehicle_images/Bike Images/Bike (605).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (1165).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (281).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (496).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (814).jpeg
/kaggle/working/vehicle_images/Bike Images/bike image06.png
/kaggle/working/vehicle_images/Bike Images/Bike (95).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (94).jpg
/kaggle/working/vehicle_images/Bike Images/Bike (301).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (313).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (1226).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (242).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (329).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (900).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (689).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (1107).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (1424).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (949).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (833).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (83).jpg
/kaggle/working/vehicle_images/Bike Images/Bike (845).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (1012).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (30).jpg
/kaggle/working/vehicle_images/Bike Images/Bike (664).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (989).jpeg
/kaggle/working/vehicle_images/Bike Images/Bike (858).jpeg
```

```
import os
import random
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

# Path to the working directory where the dataset is saved
working_directory = '/kaggle/working/vehicle_images'

# Subdirectories in the working directory
subdirectories = ['Bike Images', 'Plane Image', 'Car Images']

# Function to display a random image based on user input
def display_random_image_from_category(category):
    # Map the user input to the correct subdirectory
    if category not in ['Bike', 'Plane', 'Car']:
        print("Invalid category! Please choose from Bike, Plane, or Car.")
        return

    subdir = category + ' Images' if category == 'Bike' else category + ' Image'
    subdir_path = os.path.join(working_directory, subdir)

    # Get a list of all image files in the current subdirectory
    files = os.listdir(subdir_path)

    # Select a random image from the subdirectory
    random_image = random.choice(files)

    # Load and display the selected image
    img = mpimg.imread(os.path.join(subdir_path, random_image))
    plt.figure(figsize=(5, 5))
    plt.imshow(img)
    plt.title(f'{category} - {random_image}')
    plt.axis('off')  # Hide axes for better visualization
    plt.show()

# Ask the user for input to choose a category
user_input = input("Enter category (Bike, Plane, Car): ").capitalize()

# Call the function to display the random image based on user input
display_random_image_from_category(user_input)
```

```
Enter category (Bike, Plane, Car):  Car Image
Invalid category! Please choose from Bike, Plane, or Car.
```

```
import os
import random
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

# Path to the working directory where the dataset is saved
working_directory = '/kaggle/working/vehicle_images'

# Subdirectories in the working directory
subdirectories = ['Bike Images', 'Plane Image', 'Car Images']

# Set up a plot to display the images
plt.figure(figsize=(15, 5))

# Loop through the subdirectories
for i, subdir in enumerate(subdirectories):
    subdir_path = os.path.join(working_directory, subdir)

    # Get a list of all image files in the current subdirectory
    files = os.listdir(subdir_path)

    # Select a random image from the subdirectory
    random_image = random.choice(files)

    # Load and display the selected image
    img = mpimg.imread(os.path.join(subdir_path, random_image))
    plt.subplot(1, len(subdirectories), i+1)
    plt.imshow(img)
    plt.title(f'{subdir} - {random_image}')
    plt.axis('off')  # Hide axes for better visualization

# Show the images
plt.show()
```


Bike Images - Bike (931).jpeg
shutterstock.com · 1649168356
Plane Image - airplane459.jpg
Car Images - 421.jpg

```
import os
import random
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing import image

def predict_random_images(model, base_path, image_count=3):
    """
    Predicts random images from the dataset and displays them with predictions.

    Args:
    - model: Trained model to make predictions.
    - base_path: The base path where the images are stored (car, bike, plane).
    - image_count: Number of images to predict and display.
    """
    # Directories for the categories
    car_dir = os.path.join(base_path, 'car')
    bike_dir = os.path.join(base_path, 'bike')
    plane_dir = os.path.join(base_path, 'plane')

    # Get random images from each folder
    car_images = random.sample(os.listdir(car_dir), image_count)
    bike_images = random.sample(os.listdir(bike_dir), image_count)
    plane_images = random.sample(os.listdir(plane_dir), image_count)
    # Combine the images into one list
    image_paths = []
    image_paths.extend([os.path.join(car_dir, img) for img in car_images])
    image_paths.extend([os.path.join(bike_dir, img) for img in bike_images])
    image_paths.extend([os.path.join(plane_dir, img) for img in plane_images])

    # Shuffle the image paths
    random.shuffle(image_paths)

    # Define class labels based on your dataset
    class_labels = ['car', 'bike', 'plane']

    # Create a figure to display images
    plt.figure(figsize=(15, 5))  # Adjust size as needed

    for i, img_path in enumerate(image_paths[:image_count]):
        # Load and preprocess image
        img = image.load_img(img_path, target_size=(120, 120))
        img_array = image.img_to_array(img) / 255.0
        img_array = np.expand_dims(img_array, axis=0)  # Expand dims for batch size of 1

        # Make prediction
        prediction = model.predict(img_array)
        predicted_class = np.argmax(prediction)

        # Plot each image with prediction
        plt.subplot(1, image_count, i + 1)
        plt.imshow(img)
        plt.title(f"Predicted: {class_labels[predicted_class]}")
        plt.axis('off')

    # Show the entire row of images
    plt.tight_layout()
    plt.show()

# Example usage:
base_path = '/kaggle/input/multi-vehicle-image-car-plane-bike/Images'  # Path to your dataset
predict_random_images(model, base_path)
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[56], line 63
     61 # Example usage:
     62 base_path = '/kaggle/input/multi-vehicle-image-car-plane-bike/Images'  # Path to your dataset
---> 63 predict_random_images(model, base_path)

Cell In[56], line 22, in predict_random_images(model, base_path, image_count)
     19 plane_dir = os.path.join(base_path, 'plane')
     21 # Get random images from each folder
---> 22 car_images = random.sample(os.listdir(car_dir), image_count)
     23 bike_images = random.sample(os.listdir(bike_dir), image_count)
     24 plane_images = random.sample(os.listdir(plane_dir), image_count)

FileNotFoundError: [Errno 2] No such file or directory: '/kaggle/input/multi-vehicle-image-car-plane-bike/Images/car'
```

```
import os

# Specify the base directory path
base_directory_path = '/kaggle/input/multi-vehicle-image-car-plane-bike/Images'

# List of subdirectories
subdirectories = ['Bike Images', 'Plane Image', 'Car Images']

# Loop through the subdirectories and list the images
for subdir in subdirectories:
    subdir_path = os.path.join(base_directory_path, subdir)  # Get the full path of the subdirectory
    image_files = os.listdir(subdir_path)  # List files in the subdirectory

    # Create the full image paths
    image_paths = [os.path.join(subdir_path, image_file) for image_file in image_files]

    # Print the image paths
    print(f"Images in {subdir}:")
    for path in image_paths:
        print(path)
    print("\n")
```

```
Images in Bike Images:
/kaggle/input/multi-vehicle-image-car-plane-bike/Images/Bike Images/Bike (6).jpg
/kaggle/input/multi-vehicle-image-car-plane-bike/Images/Bike Images/bike image17.png
/kaggle/input/multi-vehicle-image-car-plane-bike/Images/Bike Images/bike3 image01 (1).png
/kaggle/input/multi-vehicle-image-car-plane-bike/Images/Bike Images/Bike (648).jpeg
/kaggle/input/multi-vehicle-image-car-plane-bike/Images/Bike Images/bike3 image21 (1).jpeg
/kaggle/input/multi-vehicle-image-car-plane-bike/Images/Bike Images/Bike (1194).jpeg
/kaggle/input/multi-vehicle-image-car-plane-bike/Images/Bike Images/Bike (498).jpeg
/kaggle/input/multi-vehicle-image-car-plane-bike/Images/Bike Images/bike image16.png
/kaggle/input/multi-vehicle-image-car-plane-bike/Images/Bike Images/Bike (587).jpeg
/kaggle/input/multi-vehicle-image-car-plane-bike/Images/Bike Images/Bike (101).jpeg
/kaggle/input/multi-vehicle-image-car-plane-bike/Images/Bike Images/Bike (607).jpeg
/kaggle/input/multi-vehicle-image-car-plane-bike/Images/Bike Images/bike3 image13.png
/kaggle/input/multi-vehicle-image-car-plane-bike/Images/Bike Images/Bike (608).jpeg
/kaggle/input/multi-vehicle-image-car-plane-bike/Images/Bike Images/Bike (878).jpeg
/kaggle/input/multi-vehicle-image-car-plane-bike/Images/Bike Images/Bike (560).jpeg
/kaggle/input/multi-vehicle-image-car-plane-bike/Images/Bike Images/Bike (137).jpg
/kaggle/input/multi-vehicle-image-car-plane-bike/Images/Bike Images/Bike (54).jpeg
```