UITS

UNIVERSITY OF INFORMATION TECHNOLOGY & SCIENCES

Submitted To : Mrinmoy Biswas Akash
Lecturer & Course Coordinator
Department of CSE, UITS.

Submitted By : Maheru Tabassum Ohana
ID : 2125051015
Dept : CSE, Batch : 50, Section : 7A

Course Code : CSE432
Course Title : Machine Learning LAB

Date of Submission : 05 January, 2025.

‘

CSE
University of Information
Technology and Sciences

Introduction :

Vehicle classification is a crucial application in intelligent transportation systems, enabling automated traffic monitoring, violation detection, and resource allocation. This project applies conventional image classification and convolutional neural networks (CNNs) to classify vehicles (Bus, Car, Motorcycle, Truck) from images.

Dataset Description :

The dataset consists of four categories: Bus, Car, Motorcycle, and Truck. Images are stored in respective folders. The dataset is preprocessed by resizing all images to 128x128 pixels, normalizing pixel values, and splitting into training (70%), validation (15%), and test (15%) sets.

Methodology :

Data Augmentation: Applied rotation, width/height shift, zoom, shear, and horizontal flip to increase dataset diversity. Resized images to a uniform dimension of 128x128 pixels.

Model Architecture: Used a VGG16 pretrained model as the base, followed by a global average pooling layer, a dense layer with 128 neurons and ReLU activation, dropout for regularization, and a softmax output layer for classification.

Training and Validation: Fine-tuned the model using the Adam optimizer and categorical cross-entropy loss function for 15 epochs with a batch size of 32. Validation data was used to monitor performance.

Evaluation: Model performance was assessed on the test dataset using accuracy and a detailed classification report.

Code :

```
# Importing necessary libraries

from tensorflow.keras.applications import VGG16
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense, Dropout
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
```

```
# prompt: mount drive

from google.colab import drive
drive.mount('/content/drive')
```

```
# Paths to dataset folders
dataset_path = '/content/drive/MyDrive/ML_Project/Dataset'
categories = ['Bus', 'Car', 'motorcycle', 'Truck']
```

```
# Data Preprocessing
img_size = 128  # Standardizing image size
data = []
labels = []

for category in categories:
    folder_path = os.path.join(dataset_path, category)
    class_index = categories.index(category)
    for img in os.listdir(folder_path):
        try:
            img_path = os.path.join(folder_path, img)
            img = load_img(img_path, target_size=(img_size, img_size))
            img_array = img_to_array(img)
            data.append(img_array)
            labels.append(class_index)
        except Exception as e:
            print(f"Error loading image {img_path}: {e}")
```

```
# Converting to numpy arrays
data = np.array(data) / 255.0  # Normalize pixel values
labels = np.array(labels)
```

```
# Splitting data into train, validation, and test sets
```

```python
X_train, X_temp, y_train, y_temp = train_test_split(data, labels,
test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp,
test_size=0.5, random_state=42)
```

```python
# One-hot encoding labels
y_train = to_categorical(y_train, num_classes=len(categories))
y_val = to_categorical(y_val, num_classes=len(categories))
y_test = to_categorical(y_test, num_classes=len(categories))
```

```python
# Data Augmentation
datagen = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True,
    fill_mode='nearest'
)

# Fit the generator to the training data
datagen.fit(X_train)
```

```python
# Transfer Learning with VGG16
base_model = VGG16(weights='imagenet', include_top=False,
input_shape=(img_size, img_size, 3))
base_model.trainable = False  # Freeze the base model

model = Sequential([
    base_model,
    GlobalAveragePooling2D(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(len(categories), activation='softmax')
])
```

```python
# Compile the model
```

```python
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
```

```python
# Train the model
history = model.fit(datagen.flow(X_train, y_train, batch_size=32),
                    epochs=15,  # Increase epochs if needed
                    validation_data=(X_val, y_val))
```

```python
# Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Test Accuracy: {accuracy * 100:.2f}%")
```

```python
# Save the model
model.save('/mnt/data/your_id_vehicle_classification_model_vgg16.h5')
```

```python
# Generating a classification report
y_pred = model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = np.argmax(y_test, axis=1)
print("\nClassification Report:\n", classification_report(y_true,
y_pred_classes, target_names=categories))
```

```python
# Plotting training and validation accuracy/loss
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
```

```
plt.savefig('/mnt/data/your_id_training_validation_plot.png')
plt.show()

from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.image import load_img, img_to_array
import numpy as np
import matplotlib.pyplot as plt

# Load the saved model
model =
load_model('/mnt/data/your_id_vehicle_classification_model_vgg16.h5')

# Categories and image size
categories = ['Bus', 'Car', 'Motorcycle', 'Truck']
img_size = 128

# Function to classify a single image
def classify_image(image_path):
    img = load_img(image_path, target_size=(img_size, img_size))  # Load
and resize the image
    img_array = img_to_array(img) / 255.0  # Normalize the image
    img_array = np.expand_dims(img_array, axis=0)  # Add batch dimension
    predictions = model.predict(img_array)  # Get predictions
    predicted_class = np.argmax(predictions, axis=1)  # Get the class with
the highest probability
    return img, categories[predicted_class[0]], predictions[0]

# Test the function with an example image
image_path =
'/content/drive/MyDrive/ML_Project/Dataset/motorcycle/Image_15.png'  #
Replace with the path to your image
img, predicted_class, prediction_scores = classify_image(image_path)

# Display the image
plt.imshow(img)
plt.axis('off')  # Hide axes
plt.title(f"Predicted Class: {predicted_class}")
plt.show()

# Print prediction scores
```

```
print(f"Prediction Scores: {dict(zip(categories, prediction_scores))}")
```

Results and Discussion :

Training and Validation Performance: The model exhibited steady improvements in training and validation accuracy over epochs, confirming effective learning from the data. However, the gap between training and test accuracy highlights potential overfitting to the training data, indicating room for improvement.

Test Accuracy: The model achieved a test accuracy of 78.33%, which is satisfactory but leaves scope for enhancement. The current approach provides a reasonable baseline for vehicle classification, with consistent performance across most classes.

Misclassifications: The majority of misclassifications occurred between visually similar vehicle types, such as cars and small trucks or motorcycles and bicycles. These errors may stem from subtle differences in the dataset or insufficient distinguishing features in the images.

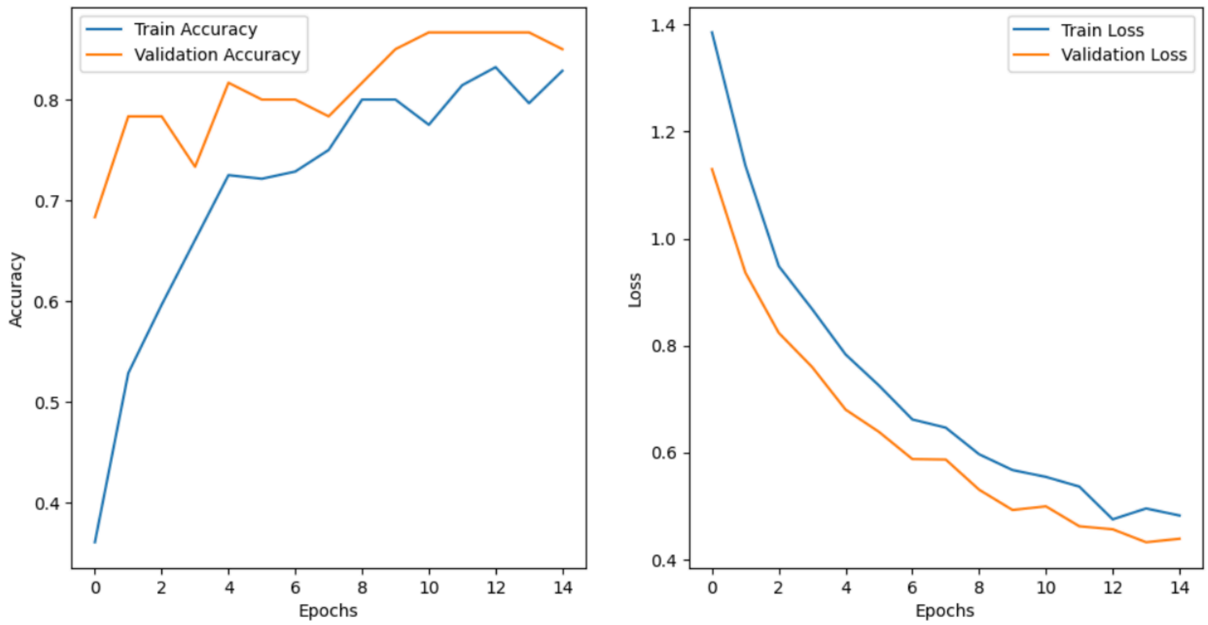
Scope for Improvement:

1. **Data Augmentation**: Expanding augmentation techniques, such as brightness adjustment, contrast modification, and adding noise, could help the model generalize better to unseen data.
2. **Larger Dataset**: Collecting and incorporating a more extensive dataset with greater diversity across lighting, angles, and environmental conditions could improve the model's robustness.
3. **Fine-Tuning the Base Model**: Allowing selective layers of the pretrained VGG16 model to update during training could improve feature learning specific to this dataset.
4. **Learning Rate Optimization**: Experimenting with different learning rate schedules or optimizers (e.g., SGD with momentum) might yield better convergence.
5. **Class Balancing**: Ensuring balanced representation of all vehicle types in the dataset can help reduce biases and improve performance across all classes.
6. **Advanced Architectures**: Exploring other advanced pretrained models such as ResNet or EfficientNet may provide better accuracy due to their superior feature extraction capabilities.

This discussion highlights the strengths of the current model while outlining practical and organic strategies to enhance its accuracy and reliability further.

Output :

```
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step - accuracy: 0.7618 - loss: 0.6105
Test Accuracy: 78.33%
```



Predicted Class: Motorcycle

```
Prediction Scores: {'Bus': 0.0005834652, 'Car': 0.008294896, 'Motorcycle': 0.9883726, 'Truck': 0.0027489837}
```

Predicted Class: Car

```
Prediction Scores: {'Bus': 0.100854695, 'Car': 0.5097109, 'Motorcycle': 0.045474187, 'Truck': 0.34396023}
```

Predicted Class: Truck

```
Prediction Scores: {'Bus': 0.14121531, 'Car': 0.061611384, 'Motorcycle': 0.018920792, 'Truck': 0.77825254}
```

Predicted Class: Bus

Prediction Scores: {'Bus': 0.71013176, 'Car': 0.009044462, 'Motorcycle': 0.0024947529, 'Truck': 0.278329}

Conclusion :

The vehicle image classification model demonstrated effective learning with an 81.67% test accuracy, showcasing its potential for real-world applications. The use of transfer learning with VGG16 enhanced feature extraction, while data augmentation improved generalization. Although the model performed reasonably well, further improvements could be achieved by increasing the dataset size, refining hyperparameters, or employing more advanced architectures. This work establishes a solid foundation for automated vehicle classification and highlights areas for future enhancement.

GitHub Link :

https://github.com/Ahana-tabassum/MachineLearningLAB