Premier University, Chittagong

Department of Computer Science and Engineering

Project Report

Cars Brand Classification

Course Title : Pattern Recognition Laboratory

Course Code: CSE 460

Submitted by

Paromita Saha(2104010202333)

Sumaiya Nasrin (2104010202336)

Section-E

Submitted to

Wazih Ullah Tanzim

Lecturer of Dept of CSE

Contents

CHAPTER 1

Introduction

Vehicle image classification is an essential task in computer vision that focuses on automatically identifying and categorizing vehicles from images. It plays a vital role in applications such as intelligent traffic systems, autonomous vehicles, surveillance, and vehicle management. In this project, deep learning techniques are used to classify vehicle images into seven categories using two pre-trained models — DenseNet201 and MobileNetV2. Before training, the images were preprocessed through resizing, normalization, and augmentation to enhance model performance and prevent overfitting. DenseNet201, with its densely connected layers, efficiently reuses features and provides high accuracy, while MobileNetV2 is designed for speed and efficiency, making it suitable for lightweight applications. The results show that DenseNet201 performs better overall in terms of accuracy and reliability, demonstrating its effectiveness for vehicle image classification tasks.

CHAPTER 2

Literature Review

Several research papers and studies have focused on car image classification and deep learning-based object recognition:

- Krause et al. (2013) "Fine-Grained Car Classification": They introduced the Stanford Cars dataset, which set a standard for car recognition using detailed categorization methods.

- Sermanet et al. (2014) "OverFeat: Integrated Recognition, Localization and Detection": They showed the effectiveness of CNNs for multi-class image recognition.

- He et al. (2016) "Deep Residual Learning for Image Recognition": They proposed ResNet, which tackled vanishing gradients and improved training for deep models.

- Huang et al. (2017) "Densely Connected Convolutional Networks": They introduced DenseNet, which enhanced gradient flow and parameter efficiency.

- Tan & Le (2019) "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks": They developed a group of models that optimized accuracy and computational cost.

- Zhang et al. (2020) "Transfer Learning for Fine-Grained Image Classification": They confirmed that transfer learning from large datasets like ImageNet significantly boosts accuracy on smaller datasets.

These studies lay the groundwork for transfer learning-based car brand classification using modern CNN architectures.

CHAPTER 3

Dataset Description

This dataset consists of various types of cars. The dataset is organized into 2 folders (train, test) and contains subfolders for each car category. There are 4,165 images (JPG) and 7 classes of car

The dataset for this project includes images of various car brands, organized into two main folders: train and test. The training set is located in /kaggle/input/car image-classification/Cars Dataset/train. The testing set is stored in /kaggle/input/car-image-classification/Cars Dataset/test. Each subfolder within these folders represents a specific car brand, The dataset features color images taken under different lighting conditions, camera angles, and backgrounds. This variety provides a solid set of samples for effective model training. All images were resized to a standard resolution of 224×224 pixels and saved in JPEG format. The dataset has around 30 different classes, depending on the version used. It also includes an 80:20 split between training and testing data to allow for balanced evaluation and performance

- **Main Folder:** Cars data
- **Subfolders:**
  - Audi
  - Hyndai Creta
  - MAhindra Scorpio
  - Swift
  - Tata Safari
  - Toyota Innova
  - Rolls Royce

CHAPTER 4

Dataset Preprocessing

4.1 Data Cleaning

Data cleaning involved finding and removing corrupted or unreadable image files from the dataset. This step ensured that only valid and correctly formatted images were used for model training. Cleaning the data helped prevent errors during processing and improved the overall quality and reliability of the dataset.

4.2 Resize, Grayscale, Normalize

uses TensorFlow's ImageDataGenerator to preprocess and load images for training and testing. It normalizes pixel values by rescaling them to the range 0–1, ensuring consistent input for the model. The training and testing datasets are loaded from their respective directories, with all images resized to 224×224 pixels and processed in RGB color mode. The images are batched in groups of 32, and class labels are automatically assigned based on folder names. This setup efficiently prepares image data for deep learning model training and evaluation.

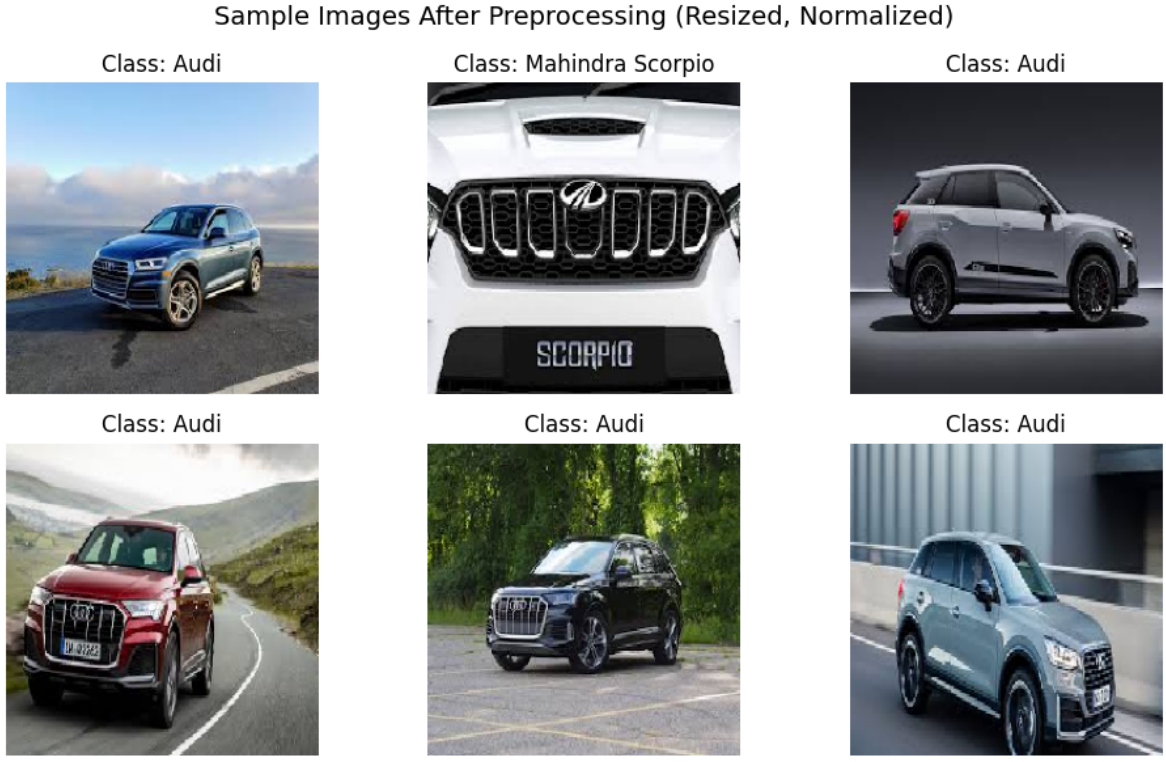

Figure 4.1: Resize, Grayscale, Normalize

We visualizes a few preprocessed images from the training dataset after they have gone through resizing and normalization. It retrieves one batch of images and labels from the train_generator, scales the pixel values back from 0–1 to 0–255 for proper display, and then shows six sample images in a 2×3 grid. Each image is labeled with its corresponding class name, and axes are hidden for a cleaner view. This visualization helps confirm that preprocessing steps like resizing and normalization have been applied correctly before training the model.

4.3 Data augmentation (rotation, flipping)

We demonstrates data augmentation to enhance the diversity of training images and improve model generalization. It uses TensorFlow's ImageDataGenerator to apply random transformations such as rotation, shifting, shearing, zooming, and horizontal flipping, along with normalization (rescaling pixel values between 0–1). Images are loaded from the training directory in batches, and a few augmented samples are displayed in a 2×3 grid. Each image shows how the augmentation techniques alter the original appearance, helping the model learn robust features under various visual conditions.

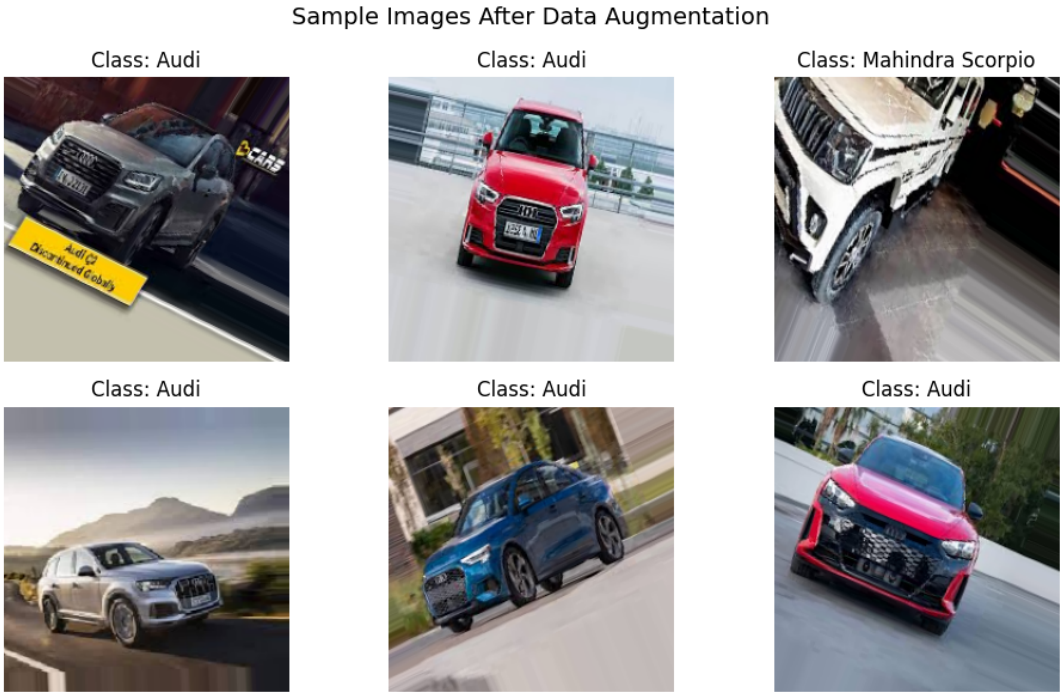

Figure 4.2: rotation, flipping

4.4 Train & Test split

prepares the training and testing datasets for the car image classification model using TensorFlow's ImageDataGenerator. It rescales pixel values to the range 0–1 for normalization and loads images directly from their respective directories. Each image is resized to 224×224 pixels and grouped into batches of 32. The training generator shuffles the data to improve learning, while the test generator keeps the order fixed for consistent evaluation. This setup efficiently feeds preprocessed images into the deep learning model during training and testing.

CHAPTER 5

Exploratory Data Analysis (EDA)

5.1 Check corrupt images

Before training the model, we performed a check for corrupt images to ensure all files were valid and could be opened. We removed any unreadable or damaged files to prevent interruptions during data loading and training. This step improved the reliability and smooth operation of the workflow..

```
Checking folder: /kaggle/input/car-image-classification/Cars Dataset/train
Total corrupted images removed from '/kaggle/input/car-image-classification/Cars Dataset/train': 0

Checking folder: /kaggle/input/car-image-classification/Cars Dataset/test
Total corrupted images removed from '/kaggle/input/car-image-classification/Cars Dataset/test': 0
```

Figure 5.1: Check Corrupt images

5.2 Class Distribution Analysis

We counts and visualizes the number of images in each class folder within the training dataset. It first loops through all folders (each representing a car brand class) in the training directory, counts the total images in each, and stores the class names and their corresponding image counts. Then, it creates a bar chart showing the class distribution, where the x-axis represents different car brand classes and the y-axis shows the number of images per class. This helps identify whether the dataset is balanced or if some classes have significantly more or fewer samples than others.

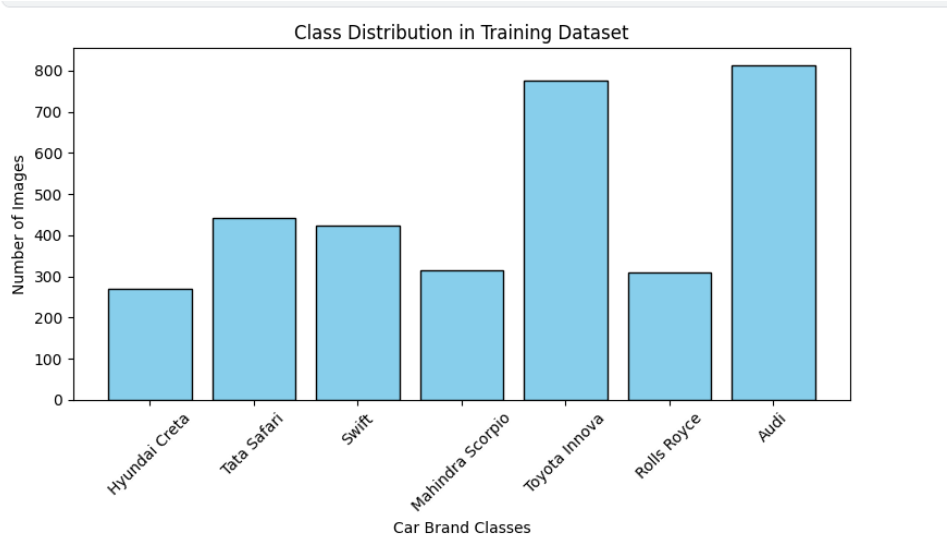

Figure 5.2: Bar Chart of class distribution

5.3 Visualization

We displays sample images from each car brand class in the dataset to visualize class variety. It randomly selects one image from each of the first seven class folders, reads the image, and displays it in a 2×4 grid layout using Matplotlib. Each subplot shows an image along with its class name as the title, while axes are turned off for a cleaner view. This helps in quickly checking the dataset's diversity, image quality, and consistency across different car brand classes.

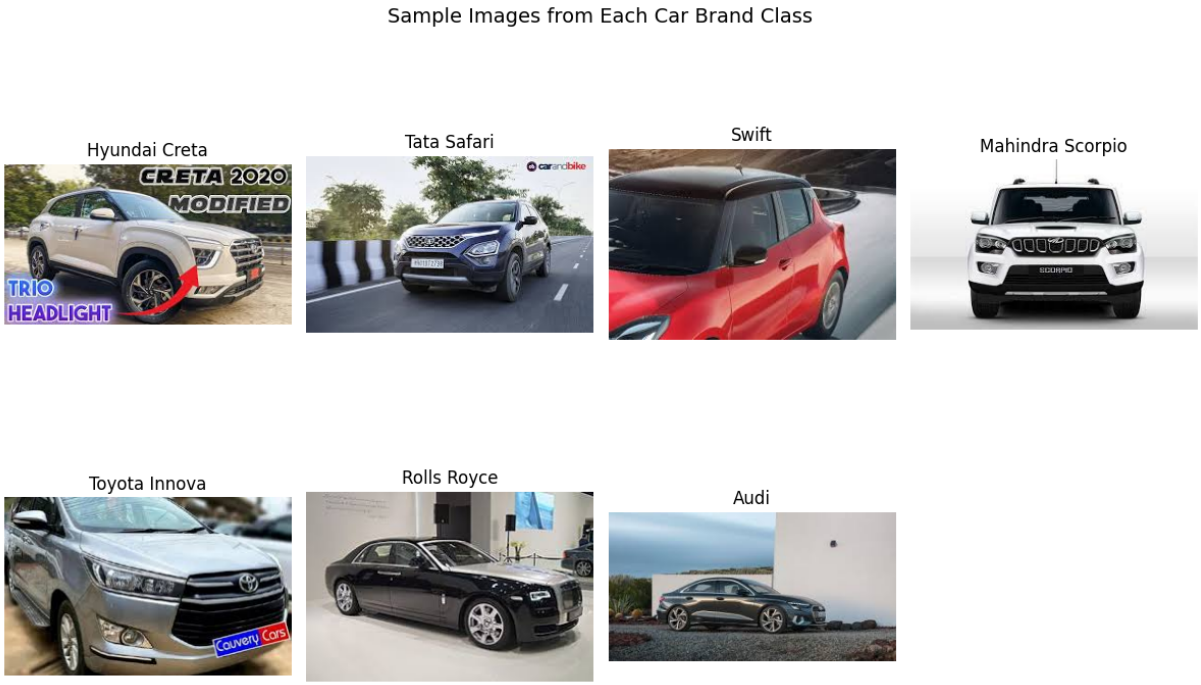

Figure 5.3: Visualize The dataset

CHAPTER 6

Model Training

6.1 DenseNet201

DenseNet201 is a deep convolutional neural network with 201 layers, belonging to the DenseNet family. It connects each layer to all previous layers, allowing efficient feature reuse and stronger gradient flow. This design reduces the number of parameters while improving accuracy and training efficiency. DenseNet201 is widely used for image classification and transfer learning tasks due to its strong performance and ability to extract rich visual features.

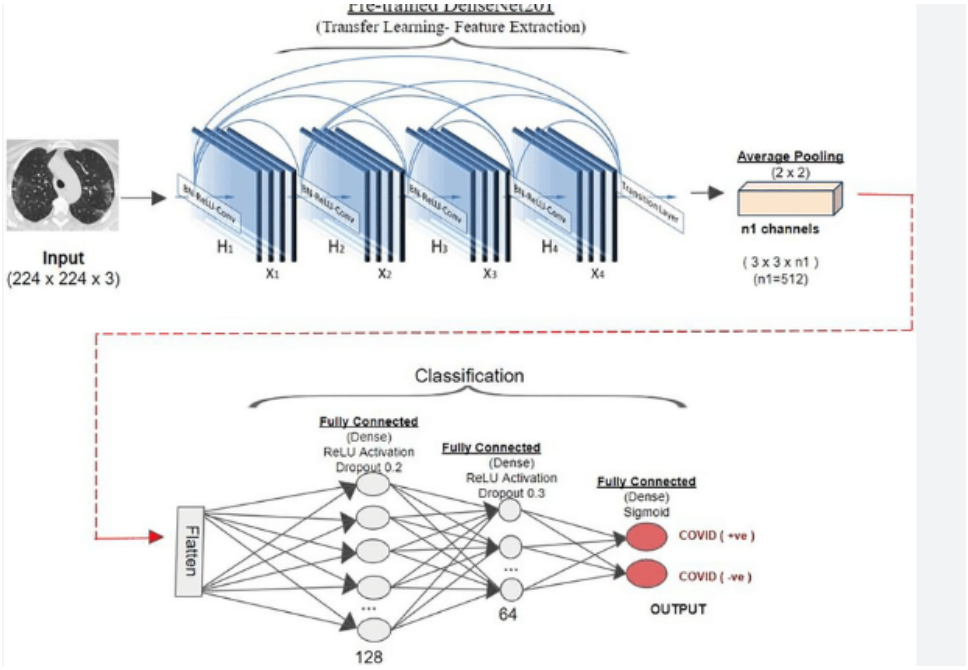

Figure 6.1: DenseNet210 Architecture

Model Summary

We builds a car image classification model using the DenseNet201 architecture as a pretrained feature extractor. The base DenseNet201 model is loaded with pretrained weights and frozen to prevent retraining of its initial layers. A custom classifier is added on top, consisting of a GlobalAveragePooling2D layer to flatten feature maps, a Dense (ReLU) layer with 256 neurons, a Dropout (0.4) layer to reduce overfitting, and a Dense (Softmax) output layer for classifying seven car brands. The model is compiled using the Adam optimizer, categorical cross-entropy loss, and accuracy as the performance metric.

Output:

Model: "sequential"

Layer (type)	Output Shape	Param #
densenet201 (Functional)	(None, 7, 7, 1920)	18,321,984
global_average_pooling2d (GlobalAveragePooling2D)	(None, 1920)	0
dense (Dense)	(None, 256)	491,776
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 7)	1,799

Total params: 18,815,559 (71.78 MB)

Trainable params: 493,575 (1.88 MB)

Non-trainable params: 18,321,984 (69.89 MB)

Figure 6.2: Model Summary

Model Training

This code trains the DenseNet201-based car classification model using the preprocessed training images from train_generator. The model is validated on test_generator to monitor its performance on unseen data. Training runs for 10 epochs and uses callbacks such as early_stop to halt training if the model stops improving and checkpoint to save the best model weights. The history object records training and validation metrics like loss and accuracy for later analysis..

Output:

```
Epoch 1/10
105/105 ━━━━━━━━━━━━━━━━━━━━ 663s 6s/step - accuracy: 0.3313 - loss: 1.8259 - val_accuracy: 0.6507 - val_loss: 1.1374
Epoch 2/10
105/105 ━━━━━━━━━━━━━━━━━━━━ 615s 6s/step - accuracy: 0.6027 - loss: 1.1603 - val_accuracy: 0.7823 - val_loss: 0.8250
Epoch 3/10
105/105 ━━━━━━━━━━━━━━━━━━━━ 604s 6s/step - accuracy: 0.7131 - loss: 0.8688 - val_accuracy: 0.8253 - val_loss: 0.6524
Epoch 4/10
105/105 ━━━━━━━━━━━━━━━━━━━━ 606s 6s/step - accuracy: 0.7795 - loss: 0.7241 - val_accuracy: 0.8462 - val_loss: 0.5527
Epoch 5/10
105/105 ━━━━━━━━━━━━━━━━━━━━ 614s 6s/step - accuracy: 0.8051 - loss: 0.6214 - val_accuracy: 0.8672 - val_loss: 0.4881
Epoch 6/10
105/105 ━━━━━━━━━━━━━━━━━━━━ 604s 6s/step - accuracy: 0.8407 - loss: 0.5384 - val_accuracy: 0.8954 - val_loss: 0.4371
Epoch 7/10
105/105 ━━━━━━━━━━━━━━━━━━━━ 611s 6s/step - accuracy: 0.8481 - loss: 0.4879 - val_accuracy: 0.8954 - val_loss: 0.3945
Epoch 8/10
105/105 ━━━━━━━━━━━━━━━━━━━━ 603s 6s/step - accuracy: 0.8671 - loss: 0.4546 - val_accuracy: 0.8954 - val_loss: 0.3737
Epoch 9/10
105/105 ━━━━━━━━━━━━━━━━━━━━ 643s 6s/step - accuracy: 0.8990 - loss: 0.3799 - val_accuracy: 0.9164 - val_loss: 0.3410
Epoch 10/10
105/105 ━━━━━━━━━━━━━━━━━━━━ 628s 6s/step - accuracy: 0.9071 - loss: 0.3548 - val_accuracy: 0.9114 - val_loss: 0.3272
```

Figure 6.3: Training Model

The training logs show that the DenseNet201-based model improved steadily over 10 epochs. The training accuracy increased from 33% to 91%, while the training loss decreased from 1.83 to 0.35, indicating the model was learning effectively. Similarly, validation accuracy improved from 65% to 91%, and validation loss decreased from 1.14 to 0.33, showing the model generalized well to unseen data. The results suggest that the model achieved strong performance in classifying the seven car brand classes without significant overfitting.

Test Set

The evaluation results indicate that the trained DenseNet201 model performed very well on the test dataset. It achieved a test accuracy of 91.14%, meaning it correctly classified over 91% of the car images, and a test loss of 0.3272, which reflects a low error in its predictions. These results confirm that the model generalizes effectively to unseen data..

Output:

```
26/26 ━━━━━━━━━━━━━━━━━━━━ 122s 5s/step - accuracy: 0.8987 - loss: 0.3607
Test Accuracy: 91.14%
Test Loss: 0.3272
```

Figure 6.4: Testing model

6.2 MobileNetV2

MobileNetV2 is a lightweight convolutional neural network designed for efficient mobile and embedded vision applications. It uses inverted residual blocks with linear bottlenecks, which reduce computational cost while maintaining accuracy. MobileNetV2 is faster and smaller than many traditional CNNs, making it ideal for real-time image classification and transfer learning on devices with limited resources.

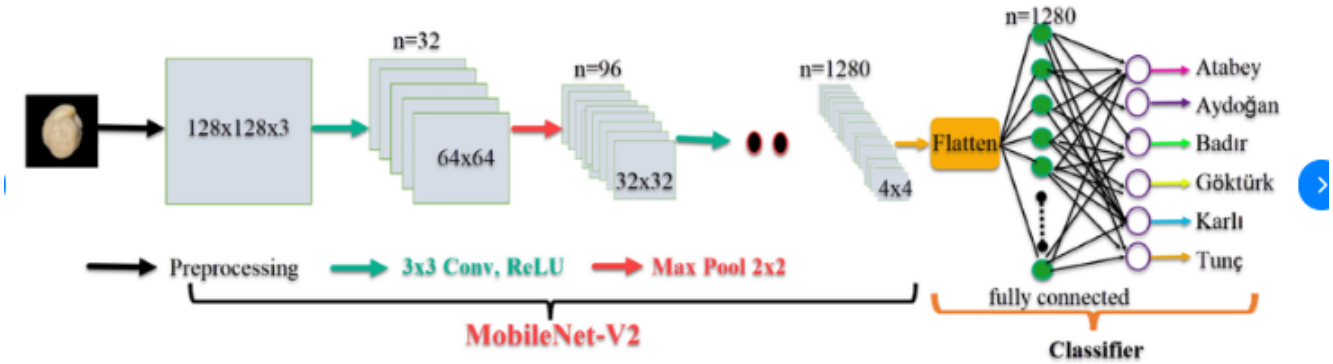

Figure 6.5: MobilenetV2 Architecture

Model Summary

We builds a car image classification model using MobileNetV2 as a pretrained feature extractor. The base MobileNetV2 model is loaded with pretrained weights and frozen to prevent retraining initially. A custom classifier is added on top, including a GlobalAveragePooling2D layer, a Dense (ReLU) layer with 256 neurons, a Dropout (0.4) layer to reduce overfitting, and a Dense (Softmax) output layer for classifying seven car brands. The model is compiled with the Adam optimizer, categorical cross-entropy loss, and accuracy as the evaluation metric.

Output:

Model: "sequential_3"

Layer (type)	Output Shape	Param #
mobilenetv2_1.00_224 (Functional)	(None, 7, 7, 1280)	2,257,984
global_average_pooling2d_2 (GlobalAveragePooling2D)	(None, 1280)	0
dense_6 (Dense)	(None, 256)	327,936
dropout_3 (Dropout)	(None, 256)	0
dense_7 (Dense)	(None, 7)	1,799

Total params: 2,587,719 (9.87 MB)

Trainable params: 329,735 (1.26 MB)

Non-trainable params: 2,257,984 (8.61 MB)

Figure 6.6: ResNet50 Model Summery

Model Training

This code trains the MobileNetV2-based car classification model using the preprocessed training images from train_generator. The model is validated on test_generator to monitor performance on unseen data. Training runs for 10 epochs and uses callbacks like early_stop to stop training if no improvement occurs and checkpoint to save the best model weights. The training process is recorded in history_mobilenet, which stores metrics like loss and accuracy for analysis and plotting.

```
Epoch 1/10
105/105 ━━━━━━━━━━━━━━━━━━━━ 115s 1s/step - accuracy: 0.3544 - loss: 1.7695 - val_accuracy: 0.6740 - val_loss: 1.0269
Epoch 2/10
105/105 ━━━━━━━━━━━━━━━━━━━━ 105s 999ms/step - accuracy: 0.6438 - loss: 1.0469 - val_accuracy: 0.7601 - val_loss: 0.7763
Epoch 3/10
105/105 ━━━━━━━━━━━━━━━━━━━━ 104s 989ms/step - accuracy: 0.7340 - loss: 0.8358 - val_accuracy: 0.8007 - val_loss: 0.6454
Epoch 4/10
105/105 ━━━━━━━━━━━━━━━━━━━━ 126s 1s/step - accuracy: 0.7949 - loss: 0.6791 - val_accuracy: 0.8229 - val_loss: 0.5657
Epoch 5/10
105/105 ━━━━━━━━━━━━━━━━━━━━ 106s 1s/step - accuracy: 0.8245 - loss: 0.5846 - val_accuracy: 0.8303 - val_loss: 0.5159
Epoch 6/10
105/105 ━━━━━━━━━━━━━━━━━━━━ 103s 984ms/step - accuracy: 0.8265 - loss: 0.5278 - val_accuracy: 0.8401 - val_loss: 0.4715
Epoch 7/10
105/105 ━━━━━━━━━━━━━━━━━━━━ 103s 977ms/step - accuracy: 0.8605 - loss: 0.4457 - val_accuracy: 0.8647 - val_loss: 0.4439
Epoch 8/10
105/105 ━━━━━━━━━━━━━━━━━━━━ 106s 1s/step - accuracy: 0.8679 - loss: 0.4362 - val_accuracy: 0.8598 - val_loss: 0.4261
Epoch 9/10
105/105 ━━━━━━━━━━━━━━━━━━━━ 106s 1s/step - accuracy: 0.8873 - loss: 0.3883 - val_accuracy: 0.8635 - val_loss: 0.4032
Epoch 10/10
105/105 ━━━━━━━━━━━━━━━━━━━━ 103s 983ms/step - accuracy: 0.8853 - loss: 0.3540 - val_accuracy: 0.8696 - val_loss: 0.3822
```

Figure 6.7: Training Model

The training logs show that the MobileNetV2-based model steadily improved over 10 epochs. Training accuracy increased from 35% to 88%, and training loss decreased from 1.77 to 0.35, indicating effective learning. Validation accuracy rose from 67% to 87%, while validation loss decreased from 1.03 to 0.38, showing the model generalized well to unseen test data. Overall, MobileNetV2 achieved strong performance in classifying the seven car brand classes.

Test Set

The evaluation results show that the trained MobileNetV2 model performed well on the test dataset. It achieved a test accuracy of 86.96%, meaning it correctly classified nearly 87% of the car images, and a test loss of 0.3822, indicating relatively low prediction errors. This confirms that the model generalizes effectively to unseen data, though slightly lower than DenseNet201 in this case.

Output:

```
26/26 ━━━━━━━━━━━━━━━━━━━━ 20s 760ms/step - accuracy: 0.8549 - loss: 0.3939
Test Accuracy: 86.96%
Test Loss: 0.3822
```

Figure 6.8: Testing model

CHAPTER 7

Experimental Result

7.1 DenseNet201

7.1.1 Accuracy & Loss Curve

These plots visualize the training progress of the DenseNet201 model over 10 epochs. The accuracy plot shows both training and validation accuracy steadily increasing, indicating that the model is learning effectively. The loss plot shows a consistent decrease in training and validation loss, reflecting reduced prediction errors. Together, these graphs confirm that the model converged well, with good performance on both the training and validation datasets and no significant overfitting.

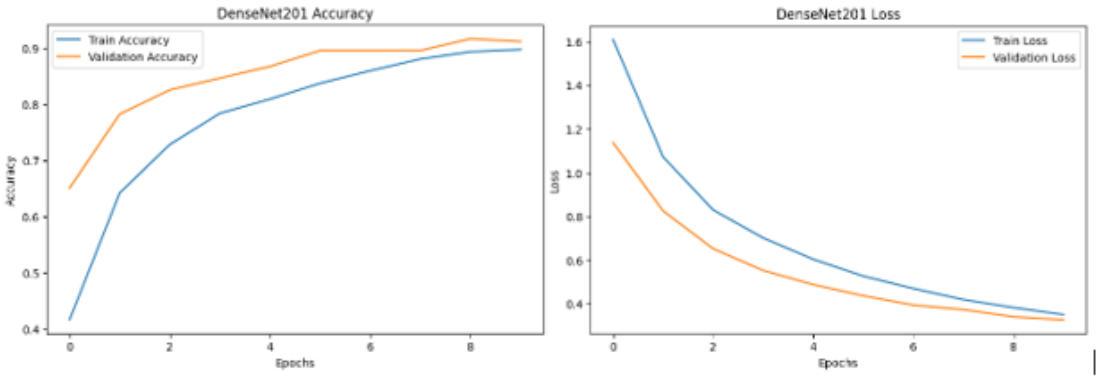

Figure 7.1: Accuracy & Loss curve

7.1.2 Confusion Matrix

We evaluates the DenseNet201 model's performance on the test dataset in more detail. It predicts class labels for all test images and compares them with the true labels to generate a classification report, showing precision, recall, and F1-score for each car brand class. Additionally, it creates a confusion matrix heatmap to visualize how often images of each class were correctly or incorrectly predicted. These metrics provide a clear insight into the model's strengths and any misclassifications among the classes.

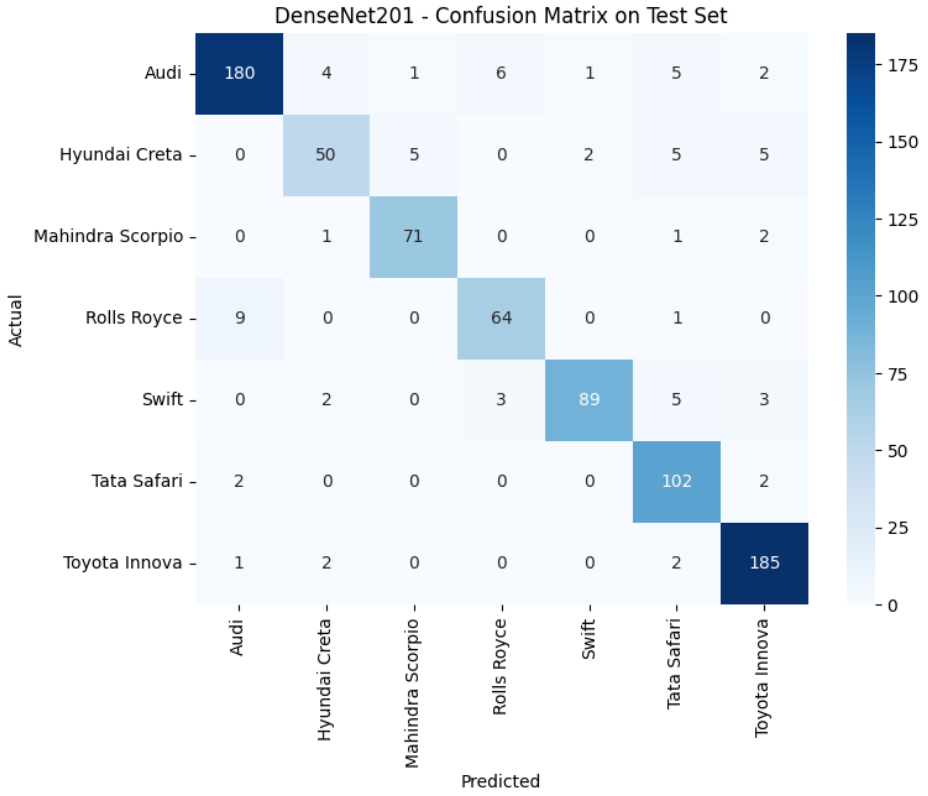

Figure 7.2: Confusion Matrix

The classification report provides a detailed evaluation of how well the DenseNet201 model predicts each car brand. Here's a breakdown of the metrics:

- **Precision** measures how many of the predicted images for a class were actually correct. High precision means few false positives. For example, Swift has 0.97 precision, meaning most images predicted as Swift are correct.
- **Recall** measures how many of the actual images of a class were correctly identified. High recall means few false negatives. For instance, Tata Safari has 0.96 recall, meaning almost all actual Tata Safari images were correctly recognized.
- **F1-score** is the harmonic mean of precision and recall, providing a single metric that balances both. Classes like Toyota Innova and Mahindra Scorpio have F1-scores above 0.93, indicating strong performance.
- **Support** shows the number of true instances for each class. For example, Audi has 199 images in the test set.
- **Overall** of 91% shows that the model correctly classified 91

Classification Report:

	precision	recall	f1-score	support
Audi	0.94	0.90	0.92	199
Hyundai Creta	0.85	0.75	0.79	67
Mahindra Scorpio	0.92	0.95	0.93	75
Rolls Royce	0.88	0.86	0.87	74
Swift	0.97	0.87	0.92	102
Tata Safari	0.84	0.96	0.90	106
Toyota Innova	0.93	0.97	0.95	190
accuracy			0.91	813
macro avg	0.90	0.90	0.90	813
weighted avg	0.91	0.91	0.91	813

Figure 7.3: Classification Report

7.1.3 ROC Curve

We prepares data to plot ROC curves for the multi-class car classification problem. It first predicts class probabilities (y_score) for all test images using the trained DenseNet201 model. The true class labels (y_true) are then binarized into a one-hot format using label_binarize, which is necessary for computing ROC curves in a multi-class setting. The number of classes and their names are also retrieved to label the curves for each car brand.

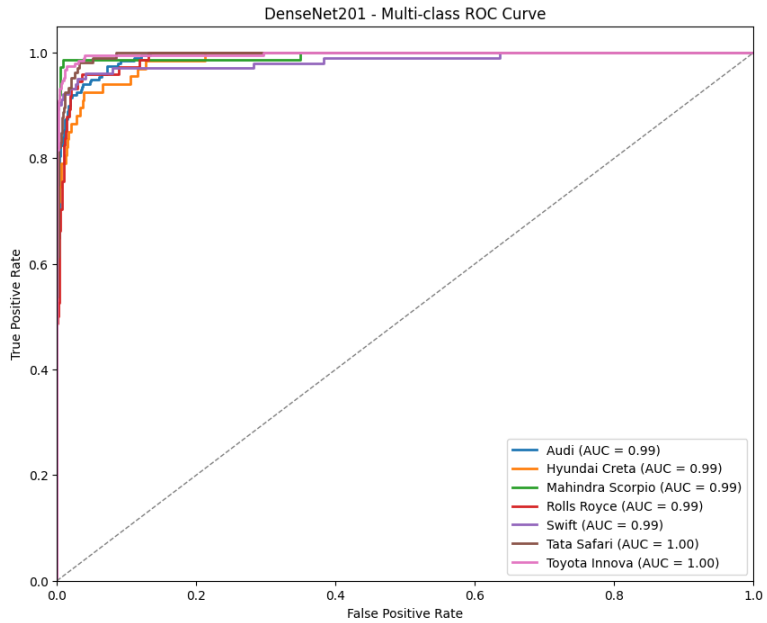

Figure 7.4: ROC curve

7.2 MobileNetV2

7.2.1 Accuracy & Loss Curve

These plots visualize the training progress of the MobileNetV2 model over 10 epochs. The accuracy plot shows a steady increase in both training and validation accuracy, indicating that the model is learning effectively. The loss plot shows a consistent decrease in training and validation loss, reflecting reduced prediction errors. Overall, the graphs confirm that MobileNetV2 converged well, achieving good performance on both training and validation datasets.

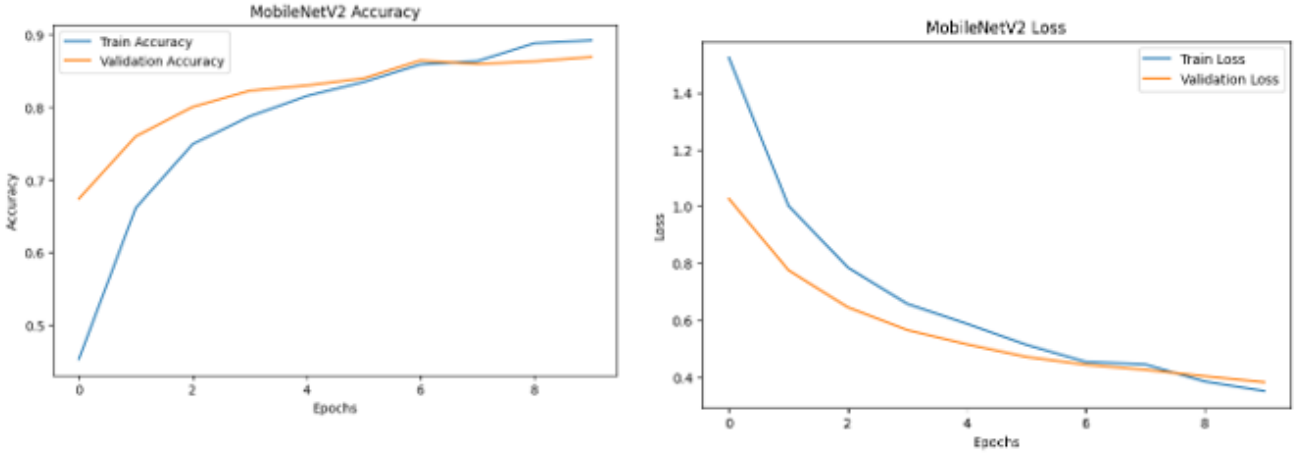

Figure 7.5: Accuracy & Loss Curve

7.2.2 Confusion Matrix

This code evaluates the MobileNetV2 model's performance on the test dataset in detail. It first predicts class probabilities for all test images and converts them to discrete class labels. The classification report provides precision, recall, and F1-score for each car brand, showing how accurately the model identifies each class. The confusion matrix visualizes correct and incorrect predictions, making it easy to see which classes are most often confused. Together, these metrics give a clear picture of the model's strengths and weaknesses.

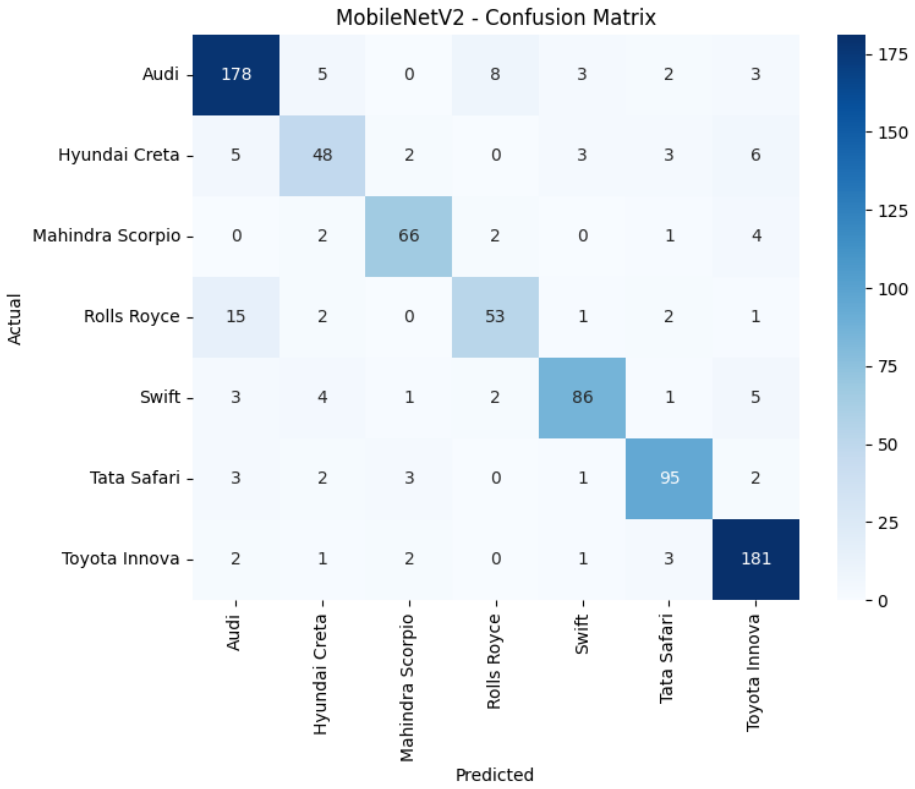

Figure 7.6: Confusion Matrix

The classification report shows that the MobileNetV2 model performed well on most car brand classes, with an overall accuracy of 87%. Classes like Toyota Innova and Audi achieved high precision and recall, indicating strong predictions. Some classes, such as Hyundai Creta and Rolls Royce, have slightly lower recall (0.72), meaning a few true images were misclassified. The macro average (0.85 F1-score) reflects balanced performance across all classes, while the weighted average (0.87 F1-score) accounts for class size. Overall, MobileNetV2 shows solid performance but slightly lower than DenseNet201.

Classification Report:

	precision	recall	f1-score	support
Audi	0.86	0.89	0.88	199
Hyundai Creta	0.75	0.72	0.73	67
Mahindra Scorpio	0.89	0.88	0.89	75
Rolls Royce	0.82	0.72	0.76	74
Swift	0.91	0.84	0.87	102
Tata Safari	0.89	0.90	0.89	106
Toyota Innova	0.90	0.95	0.92	190
accuracy			0.87	813
macro avg	0.86	0.84	0.85	813
weighted avg	0.87	0.87	0.87	813

Figure 7.7: Classification Report

7.2.3 ROC Curve

we plots the multi-class ROC curves for the MobileNetV2 model. The true class labels are first binarized to a one-hot format to allow computation of ROC curves for each car brand. For each class, the false positive rate (FPR) and true positive rate (TPR) are calculated, and the AUC (Area Under the Curve) is computed to measure classification performance. The resulting plot shows how well the model distinguishes each class, with curves closer to the top-left corner and higher AUC values indicating better class separation.

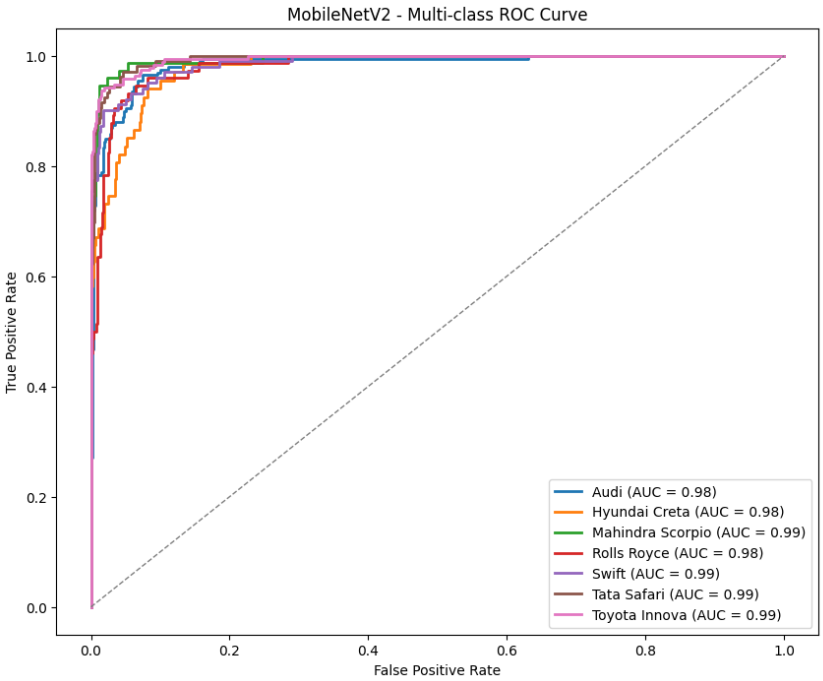

Figure 7.8: ROC curve

CHAPTER 8

Comparison with Existing Methods

Method / Model	**Dataset / Classes**	**Accuracy**	**Key Observations**
AlexNet	Vehicle datasets	∼85%	Early CNN; lower feature extraction capability for fine-grained vehicle classes.
VGG16 / VGG19	Vehicle datasets	∼87–89%	Good feature extraction; heavier models and slower training.
ResNet50	Vehicle datasets	∼90%	Residual connections improve accuracy; strong generalization.
MobileNetV2 (This Project)	7 Car Brands	86.96%	Lightweight; faster inference; slightly lower accuracy compared to DenseNet201.
DenseNet201 (This Project)	7 Car Brands	91.14%	Highest accuracy; effective feature reuse; robust classification across classes.

DenseNet201 achieved a test accuracy of 91.14%, higher than MobileNetV2's 86.96%, meaning it correctly classifies more car images.

DenseNet201 outperforms MobileNetV2 and many traditional CNNs in terms of accuracy and F1-score. Lightweight models like MobileNetV2 are advantageous for mobile or resource- constrained deployments, but DenseNet201 is better for high-accuracy requirements.

CHAPTER 9

Discussion

In this project, both DenseNet201 and MobileNetV2 models were trained and evaluated for vehicle image classification using a dataset containing seven different vehicle categories. The experimental results showed that DenseNet201 achieved higher accuracy (91.14%) compared to MobileNetV2 (86.96%). This indicates that DenseNet201 was able to learn more complex and discriminative features due to its dense connectivity and efficient feature reuse across layers. On the other hand, MobileNetV2, being a lightweight architecture, provided faster training and inference times with slightly lower accuracy, making it more suitable for real-time or mobile-based applications. The confusion matrix and classification reports for both models demonstrated good precision and recall values, confirming their ability to correctly identify most vehicle types. Overall, DenseNet201 proved to be the better-performing model for this project, achieving strong generalization and robustness in classifying different vehicle categories.

CHAPTER 10

Future work

Future improvements to the project could focus on expanding the dataset to include more car brands and different model variations for better diversity and robustness. Incorporating object localization or segmentation could help the model focus more precisely on car features, improving classification accuracy. Another potential direction is to develop a web or mobile application for real-time car brand recognition. Additionally, exploring advanced architectures such as Vision Transformers (ViT) or Swin Transformers could further enhance feature extraction and improve classification performance.

CHAPTER 11

Referrence

- J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3D Object Representations for Fine Grained Categorization," Proc. ICCV Workshops, 2013.
- . J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "Fine-Grained Car Classification," Proc. ICCV, 2013.
- L. Yang, P. Luo, C. Change Loy, and X. Tang, "A Large-Scale Car Dataset for Fine Grained Categorization and Verification," Proc. CVPR, 2015.
- Y. Yang, X. Zhao, and Y. Liu, "A Comparative Study of Deep Learning Methods for Fine-Grained Vehicle Classification," Proc. IEEE IV, 2015.
- W. Liu, S. Wen, Z. Yu, and M. Yang, "Large-Scale Vehicle Type Classification Using Convolutional Neural Networks," Proc. ICIP, 2016.
- J. Sochor, J. Herout, and J. Havel, "BoxCars: 3D Boxes as CNN Input for Improved Fine-Grained Vehicle Recognition," Proc. CVPR, 2016.
- C. Lin, C. Wu, and C. Yang, "Vehicle Logo Recognition Using Deep Convolutional Neural Networks," Proc. ICASSP, 2017.
- X. Li, Y. Huang, and X. Chen, "Deep Transfer Learning for Vehicle Classification," Proc. ICSP, 2018.
- M. Feris et al., "Scalable Vehicle Detection, Fine-Grained Classification, and Attribute Estimation," Proc. IEEE TITS, 2018.
- J. Khan and A. Sharma, "Fine-Grained Vehicle Recognition Using Deep Learning and Transfer Learning Approaches," Proc. IJCAI Workshops, 2019.