UNIVERSITY PARTNER

UNIVERSITY OF WOLVERHAMPTON

HERALD COLLEGE KATHMANDU

Image Classification of Car Brand Logos Using Convolutional Neural Networks (6CS012)

Student ID: 2329861

Student Name: Anurag Sharma

Group: L6C6G

Tutor: Mr. Shiv Kumar Yadav

Module Leader: Mr. Siman Giri

Cohort: 9

Submitted on: 5/13/2025

Abstract

This study investigates using Convolutional Neural Networks (CNNs) to classify car logos from photographs. The task is difficult because there are differences in logo design, lighting, and backdrop elements. To address this, a collection of photos from eight different car brands was used, with preprocessing techniques like as scaling, normalization, and data augmentation applied to prepare the images for training. The purpose was to assess how different deep learning algorithms performed in reliably recognizing and classifying logos.

Three models were created and tested. The first was a simple CNN architecture composed of a few convolutional layers and trained with the Adam optimizer. To avoid overfitting, the second model added extra layers and regularization techniques such as dropout, L2 regularization, and batch normalization to the baseline. Finally, the third model used transfer learning on the VGG16 architecture. Layers were frozen and adjusted to fit the dataset, allowing the model to benefit from features learned from a big dataset.

The results showed that both the deeper and transfer learning models outperformed the baseline in terms of accuracy and stability during training. The transfer learning model, in particular, achieved the highest accuracy with the lowest validation loss, proving its superiority in handling smaller datasets. The study demonstrates how various CNN architectures and training strategies affect performance, as well as how leveraging pre-trained models can greatly enhance outcomes for image classification tasks such as logo recognition.

Contents

1. Introduction 1

2. Dataset 2

3. Methodology 2

4. Experiments and Results 6

  4.1 Baseline vs. Deeper architecture 6

  4.2 Computational Efficiency 8

  4.3 Training with Different Optimizers. 8

  4.4 Challenges in Training 12

5. Fine–Tuning or Transfer Learning 12

6. Conclusion and Future Work 14

7. References 16

Table of Figures

Figure 1: Summary of Baseline Model 3
Figure 2: Summary of Deeper Model 4
Figure 3 Graphs of training and validation accuracy and loss 5
Figure 4: Classification Report 6
Figure 5: Graph of SGD and Adam 7
Figure 6 Classification report of SGD and Adam 7
Figure 7: Summary of Fine Tuning 9
Figure 8: Transfer Learning Training and Validation Curve 10
Figure 9: Transfer Learning Classification Report 10

1. Introduction

In the real world, recognizing car brand logos is a critical capability for technologies like autonomous traffic monitoring and vehicle tracking systems. However, this task is fraught with difficulties due to diverse logo designs, variable image quality, and environmental factors such as poor lighting and background noise. These challenges necessitate advanced solutions to ensure accurate identification under varying conditions.

This study uses Convolutional Neural Networks (CNNs) to address these challenges by classifying car brand logos in photos. CNNs are extremely effective for picture classification applications because they can extract spatial characteristics using feature maps, making them ideal for detecting distinct patterns in automobile logos.

A 2023 study by Saeed R. Saeed, Alyaa Hashem Mohammed, Shahad Khalid Khaleel, and Aqeel Ali Al-Hilali (Saeed, 2023)sheds light on deep learning for automotive logo identification. They employed a dataset of 15 automobile brand classes, initially with 400 photos per class, and eventually increased to 800 to resolve overfitting. Their technique combines two models ResNet50, which had been pretrained on ImageNet, was modified with four additional convolutional layers using ReLU activation, a dense layer with 0.5 dropout, and a sigmoid output for binary classification, while a VGGNet-based CNN with 16-19 layers of 3x3 convolutions and 2x2 max-pooling handled the main classification task.Their solution examined video frames to locate logos and license plates, extracted areas of interest (ROIs), and employed a fine-tuned VGGNet, attaining approximately 10% more average precision (AP) than the baseline VGGNet, showing greater accuracy and robustness.

The project's goal is to create and analyze three deep learning models a baseline CNN, a deeper CNN with regularization, and a transfer learning model to categorize vehicle brand logos from a public dataset, comparing their performance to determine the best effective method. This work will help to further computer vision applications in intelligent transport systems.

The link to project notebook: https://github.com/AnuragSharma8/6CS012/upload/main

2. Dataset

The dataset used for this project is publicly available in Kaggle (source: https://www.kaggle.com/datasets/volkandl/car-brand-logos) and was created by Volkan Özdemir. It contains a total of 2,913 images categorized into 8 car brand classes. The training set includes:

- 302 Hyundai,
- 301 Lexus,
- 317 Mazda,
- 342 Mercedes,
- 301 Opel,
- 314 Skoda,
- 306 Toyota,
- 330 Volkswagen

The test set consists of 50 images per class.

The resolutions in the dataset vary with a maximum resolution of 6000 width and 4000 heights and a minimum of 64*64.

Several preprocessing techniques were applied to prepare the data for training:

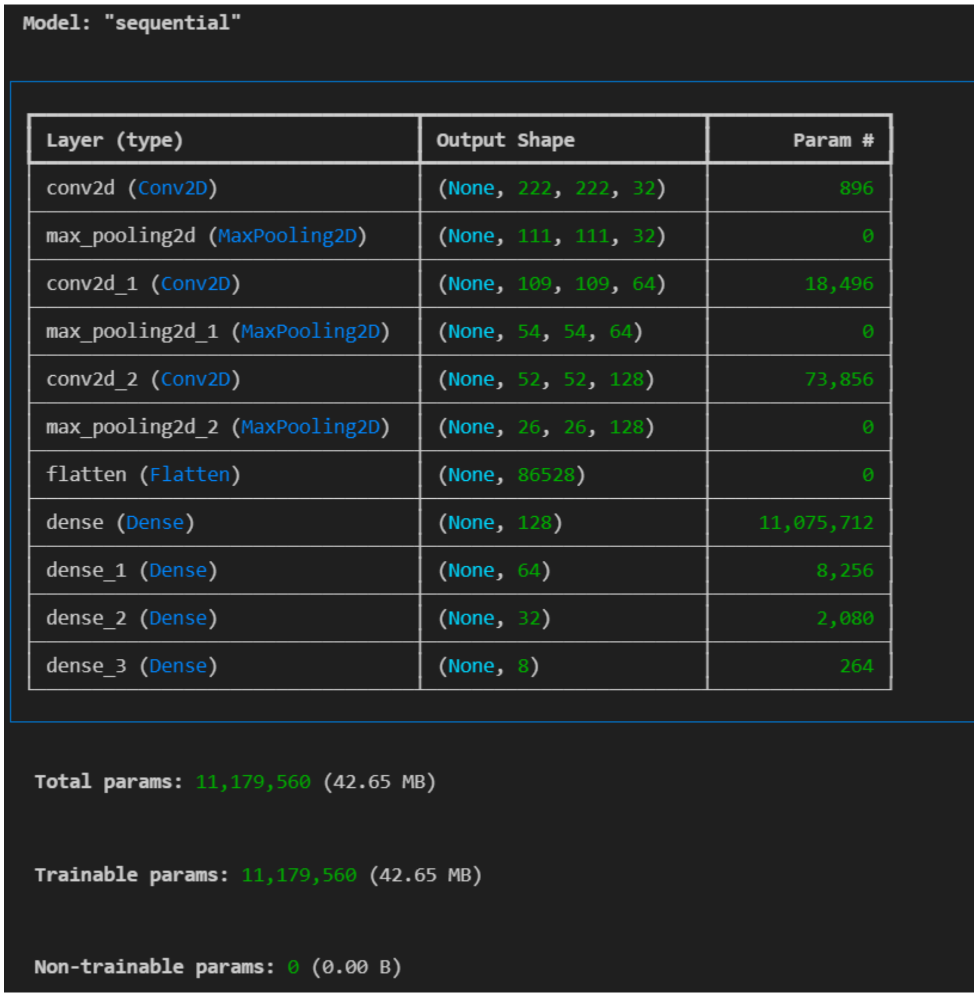
- Verifying the images to eradicate the corrupt images
- Data was augmented in the training sample to increase the data distribution
- Images were resized to 224*224 pixels
- All pixel values were normalized to a 0-1 range

3. Methodology

The project was conducted on three models and two optimizers to evaluate the performance of all the models.

- **Baseline model:** The basic model includes three convolutional layers with 32, 64, and 128 filters, each with a 3x3 kernel and ReLU activation, followed by max pooling layers to minimize spatial dimensions. A flatten layer turns the feature maps into a one-dimensional vector, which is then passed through dense layers of 512, 256, and 128 units using ReLU. The output layer comprises eight units and utilizes softmax activation to classify. There is no regularization performed to raw performance. The model employs the Adam optimizer (learning rate 0.001), sparse categorical cross-entropy, and is trained over 50 epochs using ModelCheckpoint and EarlyStopping.

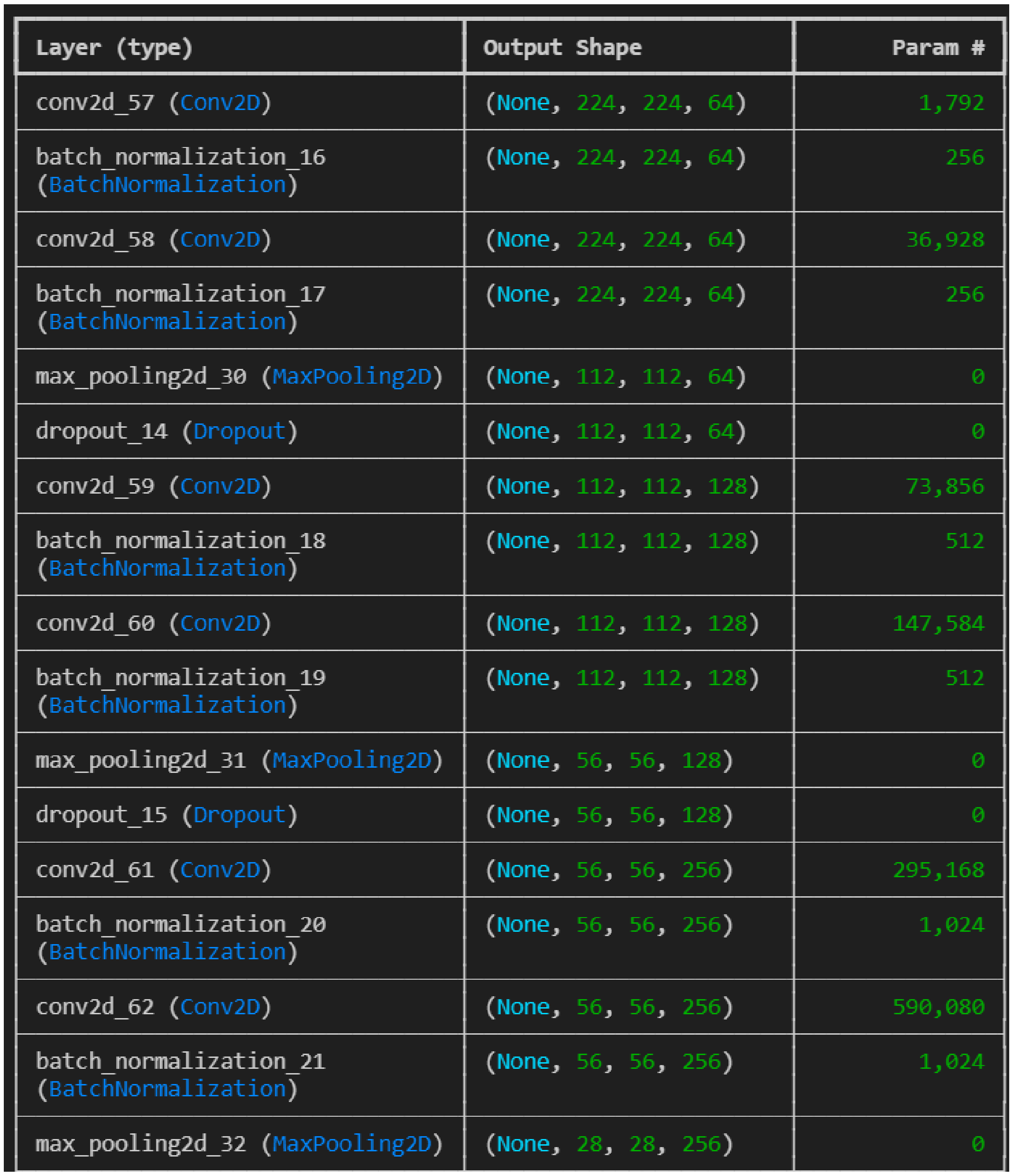
Below is the model summary with the layer-wise configuration and number of trainable parameters:



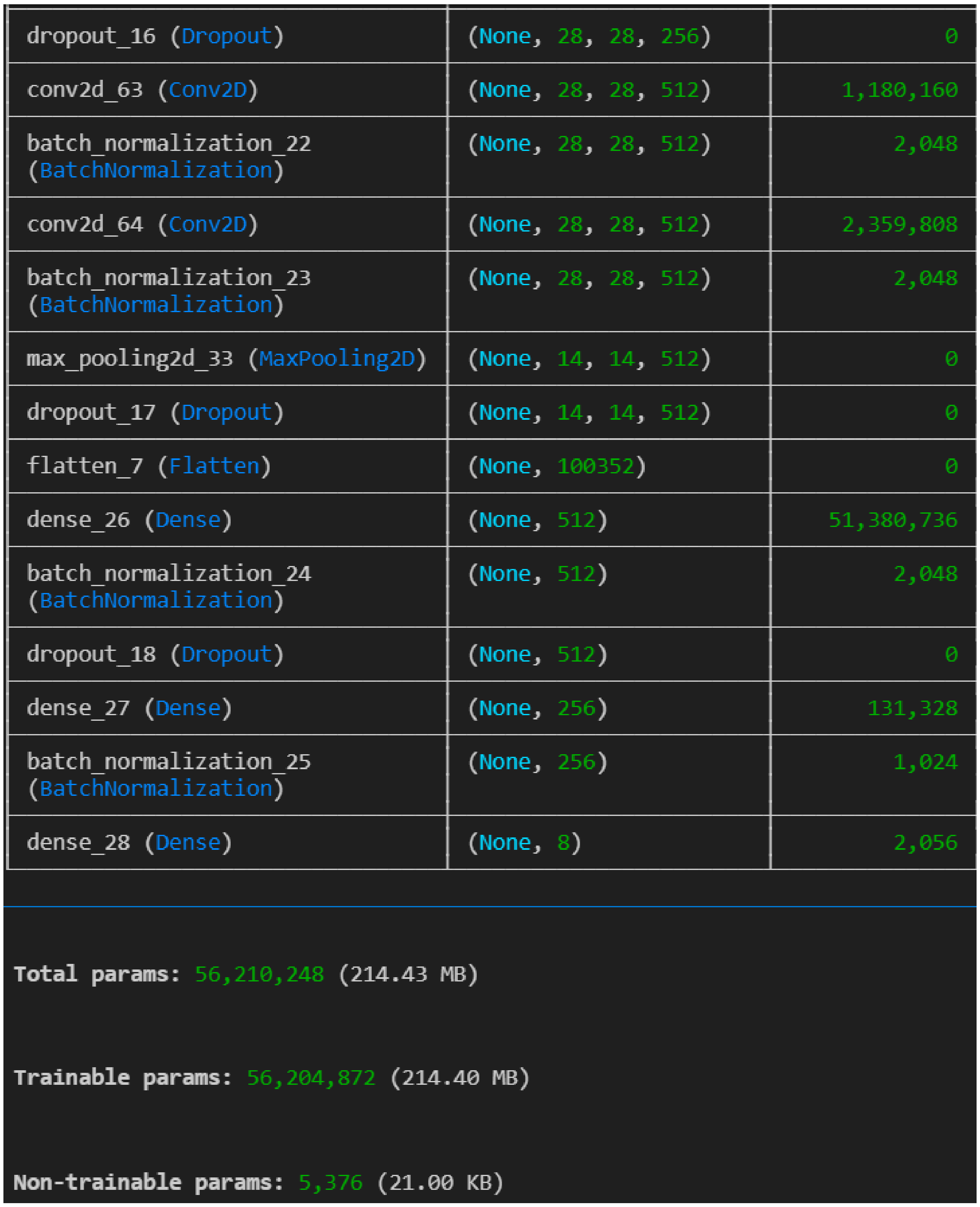
```
Model: "sequential"

┌──────────────────────────────────┬─────────────────────────┬──────────────┐
│ Layer (type)                     │ Output Shape            │      Param # │
├──────────────────────────────────┼─────────────────────────┼──────────────┤
│ conv2d (Conv2D)                  │ (None, 222, 222, 32)    │          896 │
│ max_pooling2d (MaxPooling2D)     │ (None, 111, 111, 32)    │            0 │
│ conv2d_1 (Conv2D)                │ (None, 109, 109, 64)    │       18,496 │
│ max_pooling2d_1 (MaxPooling2D)   │ (None, 54, 54, 64)      │            0 │
│ conv2d_2 (Conv2D)                │ (None, 52, 52, 128)     │       73,856 │
│ max_pooling2d_2 (MaxPooling2D)   │ (None, 26, 26, 128)     │            0 │
│ flatten (Flatten)                │ (None, 86528)           │            0 │
│ dense (Dense)                    │ (None, 128)             │   11,075,712 │
│ dense_1 (Dense)                  │ (None, 64)              │        8,256 │
│ dense_2 (Dense)                  │ (None, 32)              │        2,080 │
│ dense_3 (Dense)                  │ (None, 8)               │          264 │
└──────────────────────────────────┴─────────────────────────┴──────────────┘

 Total params: 11,179,560 (42.65 MB)

 Trainable params: 11,179,560 (42.65 MB)

 Non-trainable params: 0 (0.00 B)
```

Figure 1: Summary of Baseline model

- **Deeper model:** The deeper model builds on the baseline model by doubling the number of convolutional layers and incorporating regularization techniques such as l2, batch normalization, and dropout to increase model generalization. The maximum filter number also raised from 128 to 512. During training, the model was trained using two types of optimizers: Adam and Stochastic Gradient Descent (SGD). The separation allowed for a direct evaluation of how the optimizer affects the identical network architecture in terms of convergence rate and loss reduction. The deeper model, like the base model, utilizes the same loss function, but it adds uses ReduceLROnPlateau in callbacks to reduce learning rate when the "val_loss" measure stops improving. The model overview is provided below, along with the layer-wise configuration and the number of trainable parameters:



Layer (type)	Output Shape	Param #
conv2d_57 (Conv2D)	(None, 224, 224, 64)	1,792
batch_normalization_16 (BatchNormalization)	(None, 224, 224, 64)	256
conv2d_58 (Conv2D)	(None, 224, 224, 64)	36,928
batch_normalization_17 (BatchNormalization)	(None, 224, 224, 64)	256
max_pooling2d_30 (MaxPooling2D)	(None, 112, 112, 64)	0
dropout_14 (Dropout)	(None, 112, 112, 64)	0
conv2d_59 (Conv2D)	(None, 112, 112, 128)	73,856
batch_normalization_18 (BatchNormalization)	(None, 112, 112, 128)	512
conv2d_60 (Conv2D)	(None, 112, 112, 128)	147,584
batch_normalization_19 (BatchNormalization)	(None, 112, 112, 128)	512
max_pooling2d_31 (MaxPooling2D)	(None, 56, 56, 128)	0
dropout_15 (Dropout)	(None, 56, 56, 128)	0
conv2d_61 (Conv2D)	(None, 56, 56, 256)	295,168
batch_normalization_20 (BatchNormalization)	(None, 56, 56, 256)	1,024
conv2d_62 (Conv2D)	(None, 56, 56, 256)	590,080
batch_normalization_21 (BatchNormalization)	(None, 56, 56, 256)	1,024
max_pooling2d_32 (MaxPooling2D)	(None, 28, 28, 256)	0



```
│ dropout_16 (Dropout)               │ (None, 28, 28, 256)      │              0 │
│ conv2d_63 (Conv2D)                 │ (None, 28, 28, 512)      │      1,180,160 │
│ batch_normalization_22             │ (None, 28, 28, 512)      │          2,048 │
│ (BatchNormalization)               │                          │                │
│ conv2d_64 (Conv2D)                 │ (None, 28, 28, 512)      │      2,359,808 │
│ batch_normalization_23             │ (None, 28, 28, 512)      │          2,048 │
│ (BatchNormalization)               │                          │                │
│ max_pooling2d_33 (MaxPooling2D)    │ (None, 14, 14, 512)      │              0 │
│ dropout_17 (Dropout)               │ (None, 14, 14, 512)      │              0 │
│ flatten_7 (Flatten)                │ (None, 100352)           │              0 │
│ dense_26 (Dense)                   │ (None, 512)              │     51,380,736 │
│ batch_normalization_24             │ (None, 512)              │          2,048 │
│ (BatchNormalization)               │                          │                │
│ dropout_18 (Dropout)               │ (None, 512)              │              0 │
│ dense_27 (Dense)                   │ (None, 256)              │        131,328 │
│ batch_normalization_25             │ (None, 256)              │          1,024 │
│ (BatchNormalization)               │                          │                │
│ dense_28 (Dense)                   │ (None, 8)                │          2,056 │

Total params: 56,210,248 (214.43 MB)

Trainable params: 56,204,872 (214.40 MB)

Non-trainable params: 5,376 (21.00 KB)
```

Figure 2: Summary of Deeper Model

4. Experiments and Results

4.1 Baseline vs. Deeper architecture

The validation accuracy in the baseline model remained rather steady for the first four epochs before fluctuating from the fifth epoch. As training continued, the model began to overfit. This was demonstrated by the rising disparity between the training and validation accuracy and loss curves. Despite the overfitting, the baseline model showed smoother optimization, most likely because to its simpler structure and lower complexity, which aided convergence. However, its overall performance was limited, with mediocre grades in most classes.

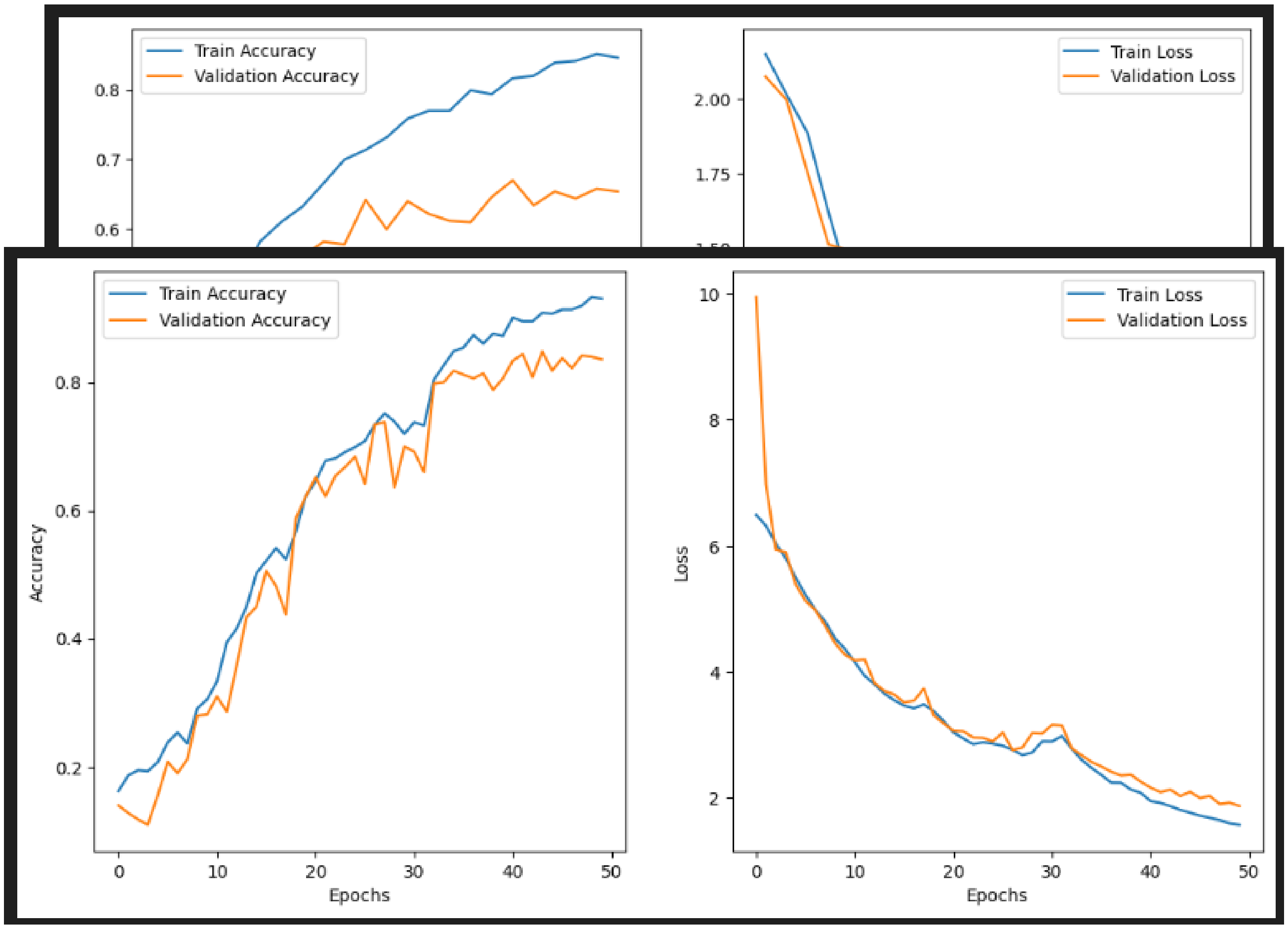
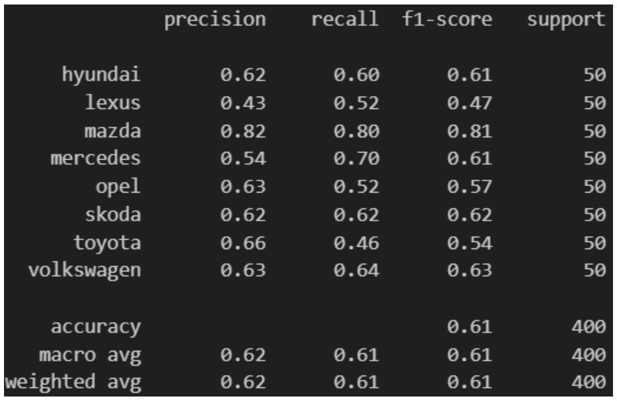
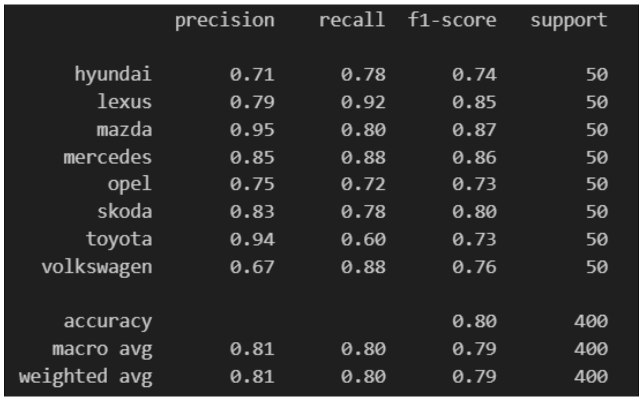


Figure 3: Graphs of training and validation accuracy and loss

The deeper model, trained with the Adam optimizer, did not suffer from over fitting, although its training and validation metrics fluctuated more during the epochs. Performance-wise, it exhibited clear improvements in practically every category. Precision improved for Hyundai (from 0.62 to 0.71) and Lexus (from 0.43 to 0.79). Toyota and Volkswagen both improved their recall scores, rising from 0.46 to 0.60 and 0.64 to 0.88, respectively. F1 scores followed a similar pattern, with Mazda climbing from 0.81 to 0.87 and Skoda from 0.62 to 0.80. Overall, the deeper model achieved 80% accuracy, up from 61% in the baseline, demonstrating the advantages of utilizing a more complicated architecture with more layers and filters.



```
              precision    recall  f1-score   support

     hyundai       0.62      0.60      0.61        50
       lexus       0.43      0.52      0.47        50
       mazda       0.82      0.80      0.81        50
    mercedes       0.54      0.70      0.61        50
        opel       0.63      0.52      0.57        50
       skoda       0.62      0.62      0.62        50
      toyota       0.66      0.46      0.54        50
  volkswagen       0.63      0.64      0.63        50

    accuracy                           0.61       400
   macro avg       0.62      0.61      0.61       400
weighted avg       0.62      0.61      0.61       400
```



```
              precision    recall  f1-score   support

     hyundai       0.71      0.78      0.74        50
       lexus       0.79      0.92      0.85        50
       mazda       0.95      0.80      0.87        50
    mercedes       0.85      0.88      0.86        50
        opel       0.75      0.72      0.73        50
       skoda       0.83      0.78      0.80        50
      toyota       0.94      0.60      0.73        50
  volkswagen       0.67      0.88      0.76        50

    accuracy                           0.80       400
   macro avg       0.81      0.80      0.79       400
weighted avg       0.81      0.80      0.79       400
```

Figure 4: Classification Report

4.2 Computational Efficiency

The deeper model outperforms the baseline model, but it is computationally more expensive. The baseline model took 21 minutes to complete training, while the deeper model took 45 minutes. The increase in time is primarily due to the deeper model's employment of more filters and layers, which increased the network's complexity. This demonstrates an obvious trade-off between accuracy and speed. To get the necessary results with several models, the project was trained on Google Colab with a T4 GPU.

4.3 Training with Different Optimizers.

The training accuracy in the SGD optimized model began about 0.15 and eventually increased to more than 0.75, while the validation accuracy began near 0.12 and barely reached about 0.61. This noticeable discrepancy between training and validation accuracy shows obvious overfitting. The training loss declined continuously, indicating stable optimization; however, the validation loss fluctuated, dropping from roughly 5.5 to 4.5, indicating unstable generalization.

The Adam optimized model's training and validation curves fluctuated slightly, but the difference between them remained tiny. Both training and validation accuracy increased above 0.80, while loss decreased below 2, indicating faster convergence and improved overall performance due to more consistent learning across epochs.

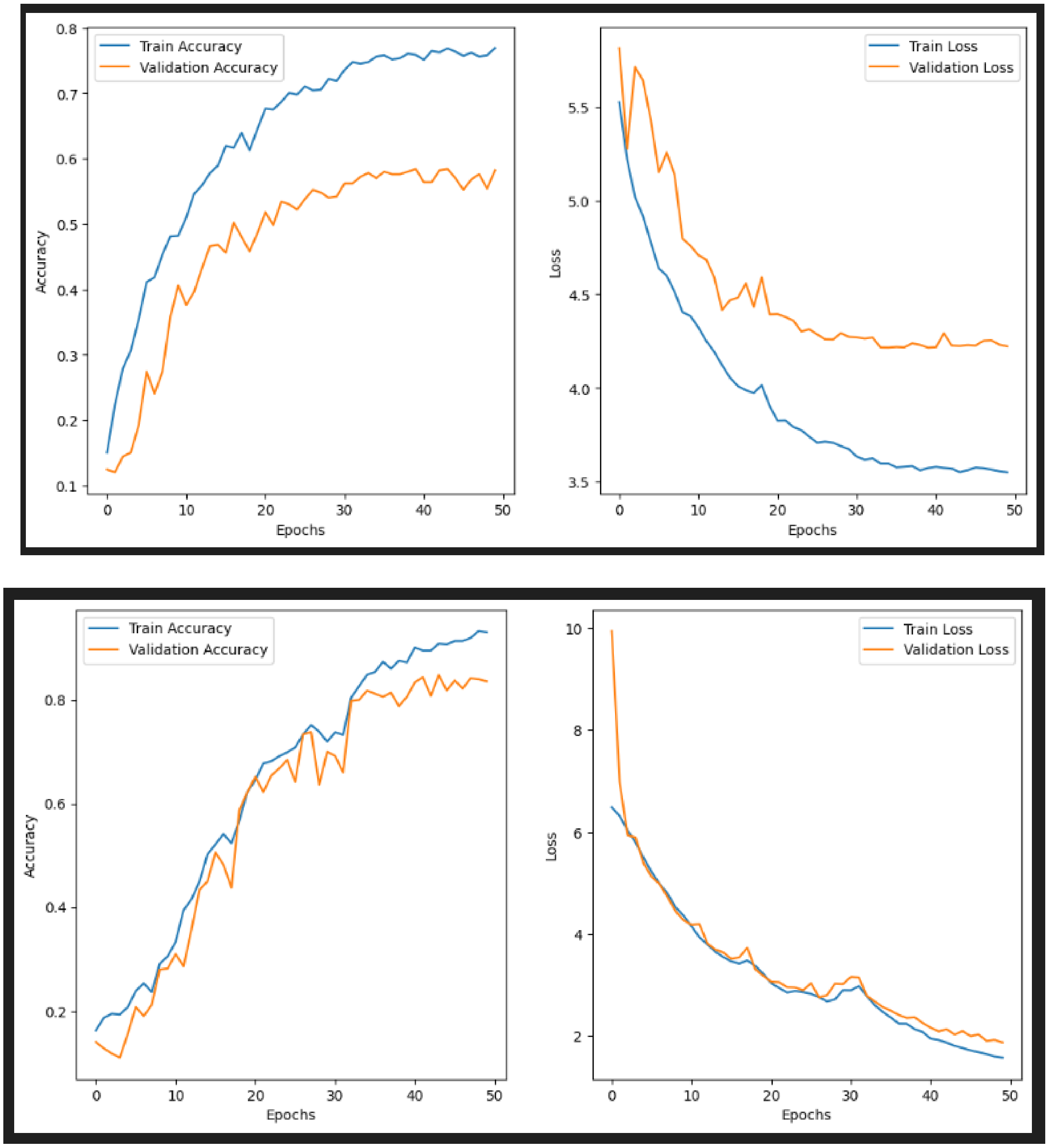
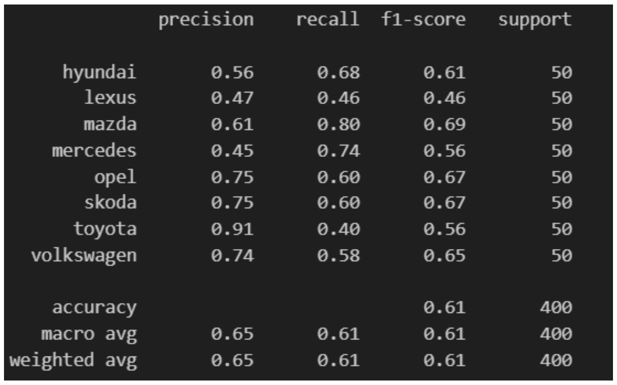


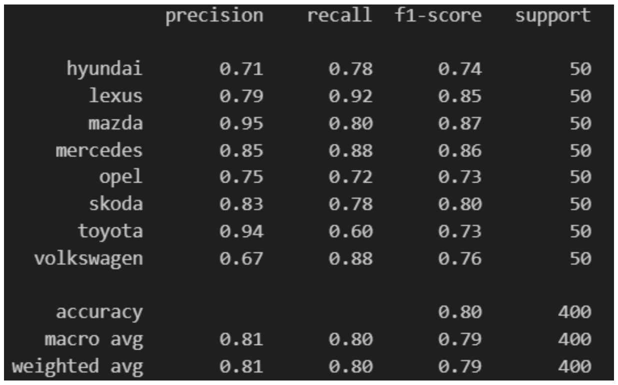
Figure 5: Graph of SGD and Adam

When analyzing the classification reports, the SGD-based model had an overall accuracy of 0.61, with the average precision, recall, and F1-scores ranging from 0.61-0.65. Some brands scored moderately, such as Toyota, which had high precision (0.91) but low recall (0.40), implying that it did not capture many genuine Toyota incidents. Hyundai, Opel, and Skoda had balanced F1 scores (about 0.67), but their recall was quite low, affecting overall robustness.

By comparison, the Adam-based model surpassed SGD, with an accuracy of 0.80. It performed admirably across most brands. Hyundai achieved an F1 score of 0.74, Lexus improved greatly to an F1 score of 0.85, and companies such as Mercedes, Mazda, and Skoda demonstrated high, consistent ratings. The categorization report clearly demonstrates that the Adam model was more accurate and missed fewer actual samples in all groups.



```
              precision    recall  f1-score   support

     hyundai       0.56      0.68      0.61        50
       lexus       0.47      0.46      0.46        50
       mazda       0.61      0.80      0.69        50
    mercedes       0.45      0.74      0.56        50
        opel       0.75      0.60      0.67        50
       skoda       0.75      0.60      0.67        50
      toyota       0.91      0.40      0.56        50
  volkswagen       0.74      0.58      0.65        50

    accuracy                           0.61       400
   macro avg       0.65      0.61      0.61       400
weighted avg       0.65      0.61      0.61       400
```



```
              precision    recall  f1-score   support

     hyundai       0.71      0.78      0.74        50
       lexus       0.79      0.92      0.85        50
       mazda       0.95      0.80      0.87        50
    mercedes       0.85      0.88      0.86        50
        opel       0.75      0.72      0.73        50
       skoda       0.83      0.78      0.80        50
      toyota       0.94      0.60      0.73        50
  volkswagen       0.67      0.88      0.76        50

    accuracy                           0.80       400
   macro avg       0.81      0.80      0.79       400
weighted avg       0.81      0.80      0.79       400
```

Figure 6: Classification report of SGD and Adam

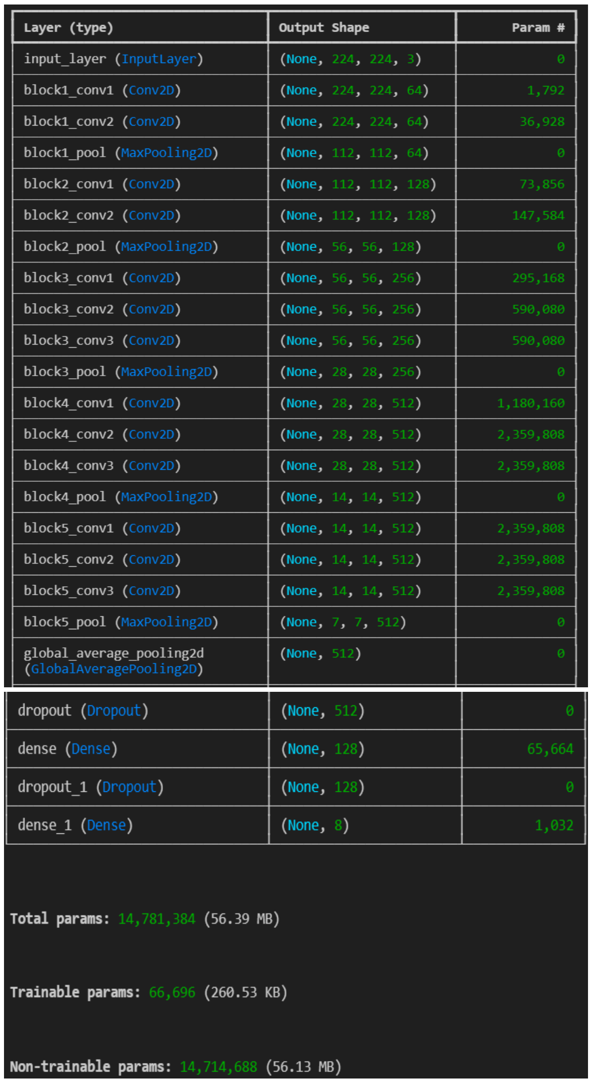
4.4 Challenges in Training

Several obstacles emerged during the training process. One significant difficulty was the dataset's small size, which caused the baseline model to overfit early on. To solve this issue, picture augmentation techniques were used to artificially enlarge training data and improve model generalization. However, the baseline model still failed to produce satisfactory results. As a result, a more comprehensive and in-depth model was built. This new architecture helped to reduce overfitting by including regularization and boosting the depth and number of filters.

Another major problem was the increased training time caused by the deeper model's extra complexity. The basic model trained in about 21 minutes, but the deeper model took much longer—45 minutes with the Adam optimizer and 50 minutes using SGD.

5. Fine–Tuning or Transfer Learning

The third model is based on the pre-trained VGG16 model, which was trained on the ImageNet dataset. The VGG16 model's layers were frozen to prevent weight updates, and the original output layer was removed. On top of the base model, additional layers such as GlobalAveragePooling, Dense, and Dropout were added, followed by a categorization layer for each of the eight car manufacturers. The model has the same configuration as the deeper model, including the Adam optimizer; however, the learning rate has been changed to 1e-5. During training, the last 20 layers were unfrozen and fine-tuned so that the model could capture higher-level features by changing their weights and learning the relevant ones. Below is the fine-tuning model summary:



```
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
│ input_layer (InputLayer)        │ (None, 224, 224, 3)    │             0 │
│ block1_conv1 (Conv2D)           │ (None, 224, 224, 64)   │         1,792 │
│ block1_conv2 (Conv2D)           │ (None, 224, 224, 64)   │        36,928 │
│ block1_pool (MaxPooling2D)      │ (None, 112, 112, 64)   │             0 │
│ block2_conv1 (Conv2D)           │ (None, 112, 112, 128)  │        73,856 │
│ block2_conv2 (Conv2D)           │ (None, 112, 112, 128)  │       147,584 │
│ block2_pool (MaxPooling2D)      │ (None, 56, 56, 128)    │             0 │
│ block3_conv1 (Conv2D)           │ (None, 56, 56, 256)    │       295,168 │
│ block3_conv2 (Conv2D)           │ (None, 56, 56, 256)    │       590,080 │
│ block3_conv3 (Conv2D)           │ (None, 56, 56, 256)    │       590,080 │
│ block3_pool (MaxPooling2D)      │ (None, 28, 28, 256)    │             0 │
│ block4_conv1 (Conv2D)           │ (None, 28, 28, 512)    │     1,180,160 │
│ block4_conv2 (Conv2D)           │ (None, 28, 28, 512)    │     2,359,808 │
│ block4_conv3 (Conv2D)           │ (None, 28, 28, 512)    │     2,359,808 │
│ block4_pool (MaxPooling2D)      │ (None, 14, 14, 512)    │             0 │
│ block5_conv1 (Conv2D)           │ (None, 14, 14, 512)    │     2,359,808 │
│ block5_conv2 (Conv2D)           │ (None, 14, 14, 512)    │     2,359,808 │
│ block5_conv3 (Conv2D)           │ (None, 14, 14, 512)    │     2,359,808 │
│ block5_pool (MaxPooling2D)      │ (None, 7, 7, 512)      │             0 │
│ global_average_pooling2d        │ (None, 512)            │             0 │
│ (GlobalAveragePooling2D)        │                        │               │
│ dropout (Dropout)               │ (None, 512)            │             0 │
│ dense (Dense)                   │ (None, 128)            │        65,664 │
│ dropout_1 (Dropout)             │ (None, 128)            │             0 │
│ dense_1 (Dense)                 │ (None, 8)              │         1,032 │

Total params: 14,781,384 (56.39 MB)

Trainable params: 66,696 (260.53 KB)

Non-trainable params: 14,714,688 (56.13 MB)
```

Figure 7: Summary of Fine-Tuning

In contrast to the other models, the transfer learning model demonstrated an impressive start with validation accuracy above 0.85, which remained stable throughout the training process and eventually reached over 0.90. The validation loss consistently decreased from a low value of 0.5 to below 0.4. As a result, the overall accuracy of this model improved to 0.86, which was the highest among all three models.

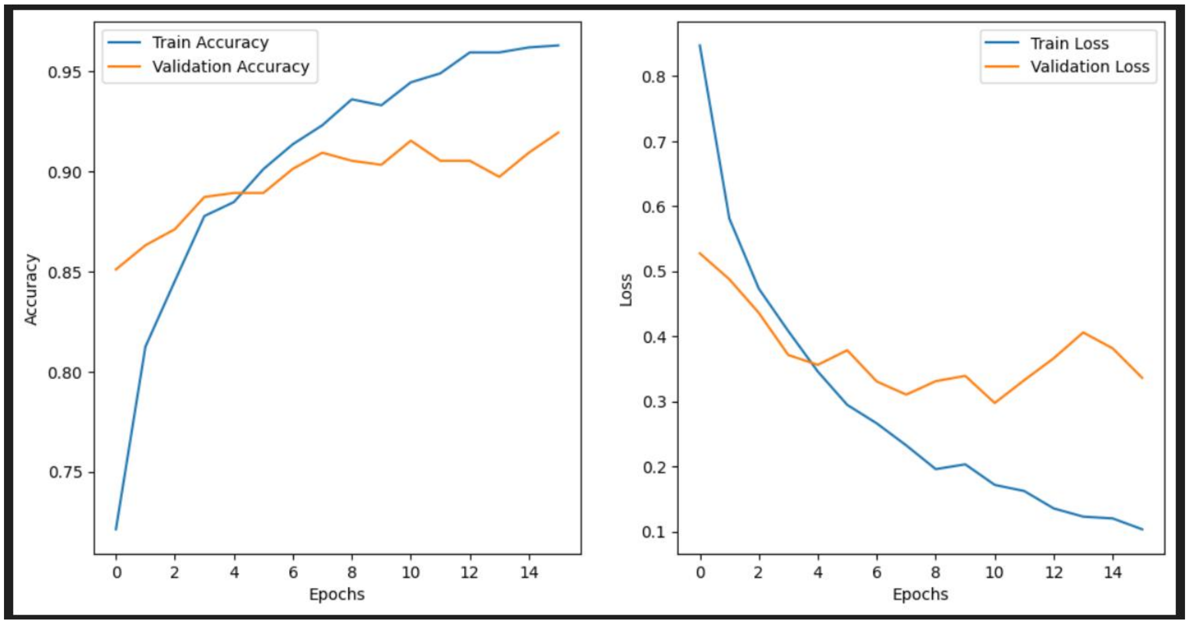
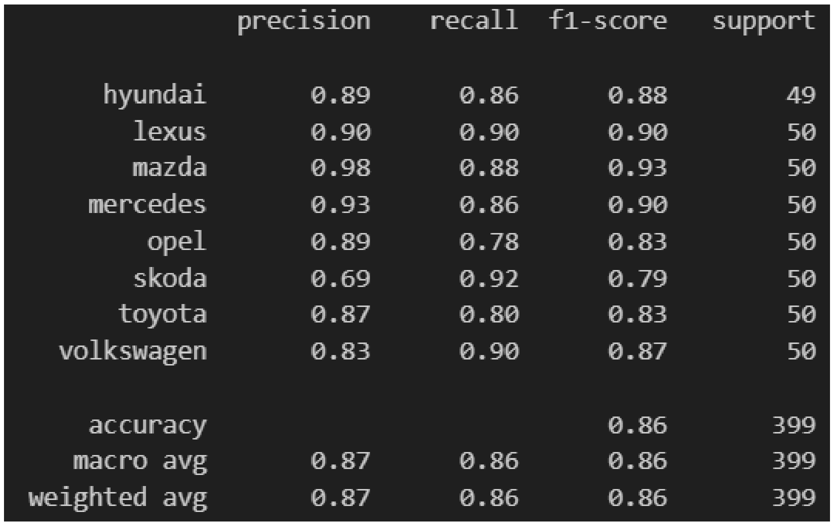


Figure 8: Transfer Learning Training and Validation Curve

The classification report also shows better performance over the precision, recall and f1 score compared to the rest of the two models.



```
              precision    recall  f1-score   support

     hyundai       0.89      0.86      0.88        49
       lexus       0.90      0.90      0.90        50
       mazda       0.98      0.88      0.93        50
    mercedes       0.93      0.86      0.90        50
        opel       0.89      0.78      0.83        50
       skoda       0.69      0.92      0.79        50
      toyota       0.87      0.80      0.83        50
  volkswagen       0.83      0.90      0.87        50

    accuracy                           0.86       399
   macro avg       0.87      0.86      0.86       399
weighted avg       0.87      0.86      0.86       399
```

Figure 9: Transfer Learning Classification Report

6. Conclusion and Future Work

The major findings show that the Adam optimized deeper model had the highest accuracy (0.80), followed by the SGD optimized model (0.61), and the baseline model (0.61). There is a trade-off between accuracy and computational expense, as the deeper model took about

45 minutes to train but performed significantly better on previously unseen data. On the other hand, the baseline model trained significantly more quickly but fared poorly. To boost performance, future work could include training with a larger dataset and studying more complex architectures like as ResNet.

7. References

Saeed, S. R. M. A. H. K. S. K. a. A.-H. A. A., 2023. Deep learning-based car logo recognition from video streams. *Journal of Advanced Computer Vision Studies*, 8(3), pp. 145-158.