Boosting Fan Experience with F1 Car Image Analysis

Group 02: Muthukumaran Samiayyan (A0027474A), Samin Batra (A0262665X), Varun Sharma (A0262674X), Prasanna Govindarajan (A0262732H), Nguyen Minh Hieu (A0262807B)

GitHub Link: https://github.com/appliedmlGRP2/F1CarClassificationDL

Abstract

This project aims to enhance the Formula 1 fan experience by developing a deep learning image classification solution to identify teams during live races. The solution includes collecting and labeling car images, training and evaluating CNN-based neural networks, and deploying the highest accuracy model to provide real-time team and driver information to audiences.

1. Introduction

Formula 1 is an immensely popular and highly competitive sport featuring advanced high-performance cars designed by 10 competing teams. These technologically sophisticated machines can achieve incredible speeds and have captivated fans around the world, reigniting their passion for the sport. However, for newcomers to Formula 1, the learning curve can be overwhelming. With various aspects of the sport to understand, new fans may have difficulty distinguishing between the cars and comprehending the importance of teams and their strategies.

To address this challenge and improve fan engagement, this project seeks to develop an image classification solution that can identify F1 teams in real-time during live races. By leveraging deep learning techniques and integrating the solution into web applications, fans will be able to access historical and real-time race details of the cars and teams they are interested in.

This will lead to a more immersive and enjoyable experience, enabling newcomers to better understand the intricacies of the sport and feel more connected to the teams and drivers they follow. Ultimately, this technology has the potential to transform the way fans engage with Formula 1 and expand the sport's global reach.

2. Literature Review

As part of our research on the related work in the domain of our project topic, we explore a research paper [1] that demonstrates the application of transfer learning and ensemble techniques in race car detection and classification.

In this research paper, the authors present a comprehensive solution for classifying NASCAR race cars in real-time using a combination of pre-trained deep learning models and ensemble learning techniques. The primary goal is to identify and categorize race cars based on their car number in images taken during races. The authors divide their approach into several stages, starting with the use of the MobileNetSSD model for detecting cars and creating bounding boxes around them. This step helps filter out irrelevant images and isolates the target objects (race cars) for further analysis.

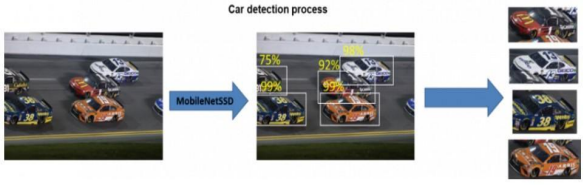

Figure 1: Example of object detection process for race cars

Next, the researchers repurpose pre-trained models, specifically VGG16, VGG19, InceptionV3, and InceptionResNetV2, for their specific use case. They employ transfer learning to leverage the knowledge gained from large datasets during the initial training of these models. By fine-tuning the models and training them on both color and grayscale images of race cars, the authors reduce training time and maintain accuracy even with a smaller dataset. The use of grayscale images helps make the models more resilient to changes in car designs, as they must focus more on the shape of features like the car number.

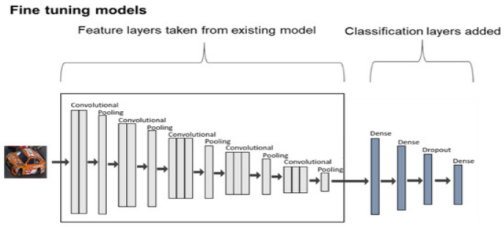

Figure 2: Illustration of pre-trained models fine-tuning for image classification

Ensemble learning is then used to improve the predictive performance of the models. The researchers employ a simple form of stacking, averaging the scores from each of the eight models (four pre-trained models for both color and grayscale images) to generate the final ensemble prediction. They find that the accuracy of the ensemble model is 81%, and when all eight models agree (which occurs 60% of the time), the accuracy reaches 96%. This allows them to prioritize quality over quantity, discarding images with less agreement among the models and focusing on those with higher accuracy.

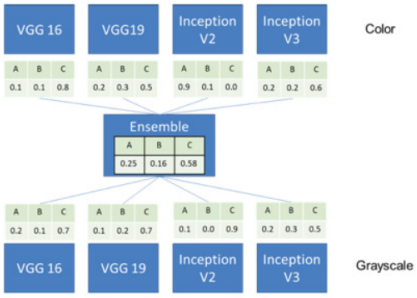

Figure 3: Illustration of ensemble method using multiple pre-trained models

The research paper demonstrates the power of AI and machine learning in solving complex image classification problems, specifically in the context of NASCAR races. By using transfer learning, fine-tuning, and ensemble learning, the authors create an accurate and efficient system that can be implemented for future races. Furthermore, the study highlights the potential benefits of using ensemble techniques to improve the performance of deep learning models and showcases the practicality of applying these methods to real-world scenarios.

Relating to the current topic of interest, this research paper serves as a valuable reference for leveraging pre-trained deep learning models and ensemble learning techniques in similar image classification tasks. The methodology and insights presented in the paper can be adapted and applied to other domains, thereby expanding the scope of AI and machine learning applications.

3. Data Collection

In the data collection phase of our project, we obtained a F1 car image dataset from Kaggle [2] that featured eight distinct car classes (teams) within Formula 1 racing. Each class contained a varying number of images, as detailed below:

- AlphaTauri: 123 images
- Ferrari: 374 images
- McLaren: 372 images
- Mercedes: 324 images
- Racing Point: 290 images
- Red Bull: 340 images
- Renault: 323 images
- Williams: 340 images

Upon closer examination, we find that some images were misclassified, while others did not actually depict the cars (e.g., podium images or Formula 1 memorabilia). To address these issues, we carefully curated the dataset by removing irrelevant images. The resulting cleaned dataset is then utilized for subsequent stages of processing.

4. Methodology

4.1 Data Augmentation & Preprocessing

For our project, a diverse dataset with multiple variations of images is essential for the model to generalize well and recognize objects in different orientations, scales, and brightness levels. If the dataset contains only a single image of the car in the same position, the algorithm will struggle to recognize the car from the images when the car is slightly rotated, positioned differently, or with varying brightness level.

To address this issue, image augmentation transformations are used to augment the original images and generate more diverse training examples. By applying a series of transformations, such as horizontal flips, rotations, scaling, brightness adjustments, and Gaussian blur we are able to create a large diversified image dataset from the original dataset. These transformations ensure that the model is exposed to a wide variety of images during training, helping it to learn robust features and improve its ability to recognize race cars in different conditions and scenarios.

Therefore, as a next step, data augmentation is applied to increase the dataset's size and improve model performance. The augmentation process involves applying a series of transformations as detailed below to the original images. The following sequence is defined for data augmentation:

1. Horizontal flips with a 50% probability i.e. to induce randomness, the image would not flip for 50% of the cases
2. Rotation of images by -45 to 45 degrees
3. Scaling of images by 50% to 150%
4. Changing the brightness by 50% to 150%
5. Applying Gaussian blur with sigma between 0 and 2

The number of augmentations per image is calculated to reach the desired count of 1,200 images per class.

In the below images, we can see the transformations applied on 'Ferrari' as well as 'Red Bull' input 3 times.

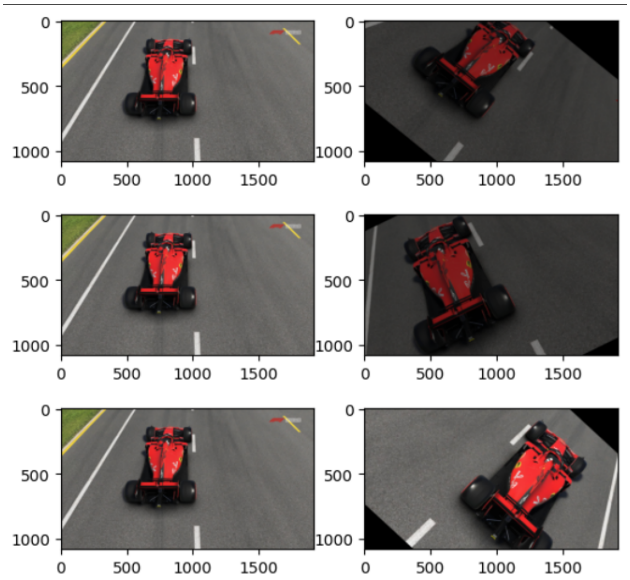

Figure 4: Examples of data augmentation for images

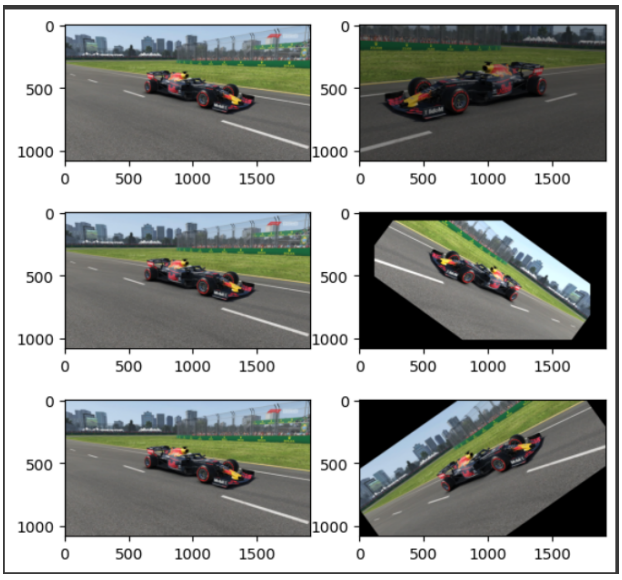

Figure 5: Examples of data augmentation for images

In terms of preprocessing, we apply two transformations to the images:

1. Resizing the images to 224x224 pixels
2. Converting the images to PyTorch tensors

As a final step, the truncated images are removed and all images are converted to the RGB format or the models to be trained upon.

4.2 Train-Test Split

The augmented dataset is of a larger size (approximately 1200 images per class) is split into training and validation.

The splitting ratio is 80% for the training set and 20% for the validation set. The models are then trained using this training dataset with a batch size of 32 for training. The validation dataset is then used to validate the model utilizing accuracy as a metric for the eight classes.

4.3 Baseline Model

We first train a simple convolutional neural network (SimpleCNN) for image classification that serves as our baseline model. By using this baseline model, we establish a performance benchmark that we can compare with more complex models. Here is the network architecture:

1. Input Layer: an input layer with 3 input channels (RGB), to be used as input to the first hidden layer

2. Hidden Layers:

- A convolutional layer with 64 output channels, kernel size of 3x3, stride of 1, and padding of 1
- ReLU activation function
- Max Pooling layer with kernel size of 2x2 and stride of 2
- A convolutional layer with 64 input channels, 128 output channels, kernel size of 3x3, stride of 1 and padding of 1
- ReLU activation function
- Max Pooling layer with kernel size of 2x2 and stride of 2
- Dropout layer with a dropout rate of 0.5
- Fully connected linear layer with 401,408 input features and 512 output features

3. Output Layer: a fully connected linear layer with 512 input features and 8 output features

Here are some other hyperparameters that we set to initialize model training:

1. Loss function: Cross-Entropy Loss
2. Optimizer: Adam
3. Learning rate: 0.001
4. Number of epochs: 10

4.4 Pre-trained Models

4.4.1 RESNET-18

The first pre-trained model that we explore is ResNet-18. ResNet stands for residual networks, which were

first introduced in 2015 by Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun in the paper 'Deep Residual Learning for Image Recognition' [3]. ResNet-18 is a residual network with 18 layers, including convolutional layers, batch normalization layers, ReLU activation layers and a fully connected linear output layer. ResNet-18 has been trained on ImageNet containing millions of images and thousands of classes. Moreover, the model utilizes residual connections to address the vanishing gradient problem, allowing it to learn effectively with 18 layers without performance degradation. Hence, ResNet-18 is preferred for tasks that require both deeper model architecture and better accuracy, without imposing too much computing power.

4.4.2 VGG-16

The second pre-trained model used is VGG-16. The model was first introduced in 2014 by Karen Simonyan and Andrew Zisserman in the paper 'Very Deep Convolutional Networks for Large-Scale Image Recognition' [4]. VGG-16 is a convolutional neural network with 16 layers, including 13 convolutional layers and 3 fully connected layers. VGG-16 has also been trained on the ImageNet dataset, making it a good candidate for transfer learning since we can leverage the knowledge the model already gained to apply to our use case. While VGG-16 only has fewer layers than ResNet-18, it is more computationally expensive because it uses 3 fully connected layers at the end of the network, and it also does not utilize any shortcuts (such as residual blocks). Thus, VGG-16 can be a good option for tasks where computational resources are not a significant constraint, and larger receptive fields are necessary to achieve desirable performance.

4.4.3 MOBILENET_V2

The last pre-trained model that we worked on is MobileNet_V2. The model was introduced in 2018 by Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen in the paper 'MobileNetV2: Inverted Residuals and Linear Bottlenecks' [5]. MobileNet_V2 is based on depth-wise separable convolutions to reduce the number of parameters. Depth-wise separable convolutions split computation into 2 steps: depthwise convolutions applying a single filter to each input channel, then pointwise convolutions combining the outputs using 1x1 convolutions. MobileNet_V2 also introduces the use of inverted residuals and linear bottlenecks to improve model efficiency without sacrificing accuracy. Therefore, this model is ideal for tasks with limited computational resources, such as mobile and embedded systems.

4.5 Transfer Learning by Fine-tuning Pre-trained Models

Two approaches can be used to fine-tune the pre-trained models. The first approach is by freezing all the layers in the pre-trained models and only training the top classification layer. The second approach is by training all the layers from the pre-trained models.

4.5.1 FINE-TUNING BY FREEZING ALL LAYERS EXCEPT TOP CLASSIFICATION LAYER

In this approach, we freeze all feature extraction layers of the pre-trained models to prevent their weights from being updated during training. We customize the output layer of the pre-trained models to match with the number of classes in our dataset and only train that layer. For this approach, we use Cross-Entropy loss function, Adam optimizer with a learning rate of 0.0001, and we train the model over 10 epochs.

4.5.2 FINE-TUNING USING ALL LAYERS

In this approach, we also modified the output layer of the pre-trained models to match with the number of classes in our dataset. However, we train all layers of the model, ensuring pre-trained parameters are also updated during training and allowing the pre-trained models to adapt to our specific task. For this approach, we use Cross-Entropy loss function, Adam optimizer with a learning rate of 0.0001, and we train the model over 10 epochs.

5. Results and Discussion

After model training, the trained model is used to make predictions on the validation dataset utilizing accuracy as our model evaluation metric. Below are the validation accuracy scores and confusion matrices for each model that we explored.

5.1 Baseline Model

Accuracy score: 61.92%

Confusion matrix:

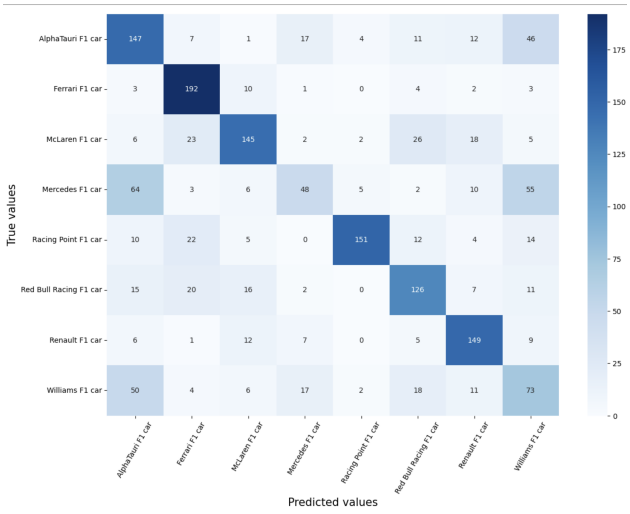

Figure 6: Confusion matrix for baseline model

As shown in the accuracy score and the confusion matrix of the baseline model, the performance of the image classification by the model is low. A significant number of the Mercedes F1 car images and the Williams F1 car images are classified as Alpha Tauri F1 car images. In addition, a significant number of Alpha Tauri F1 car images and Mercedes car images are incorrectly classified as Williams F1 cars. Therefore, the model is mainly not able to differentiate the classes between Alpha Tauri F1 cars, Mercedes cars and Williams F1 cars.

5.2 Pre-trained Models with Layers Freeze

5.2.1 RESNET-18

Accuracy score: 73.87%

Confusion matrix:

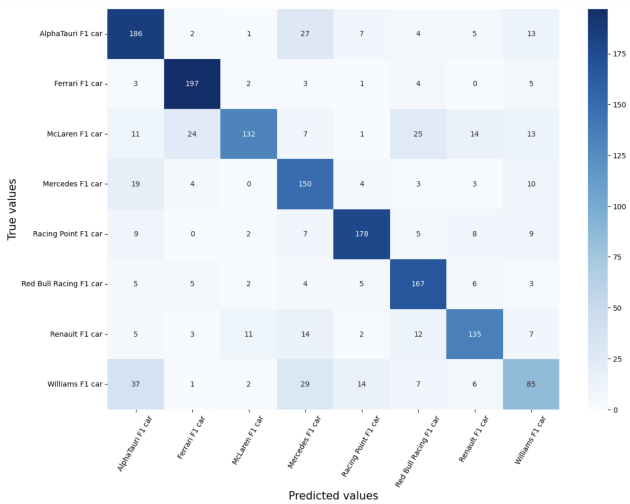

Figure 7: Confusion matrix for ResNet-18 with layers freeze

By fine-tuning only the top classification layer of the RESNET-18 model, we are able to improve the model from the baseline model. Nevertheless, as can be seen in the confusion matrix, most of the images that are being incorrectly classified are from the same Williams F1 car images, Alpha Tauri F1 cars and Mercedes F1 cars. In addition, there are also some images from Mclaren F1 car images that are being incorrectly classified as Ferrari F1 car images and Red Bull Racing F1 car images.

5.2.2 VGG-16

Accuracy score: 71.41%

Confusion matrix:

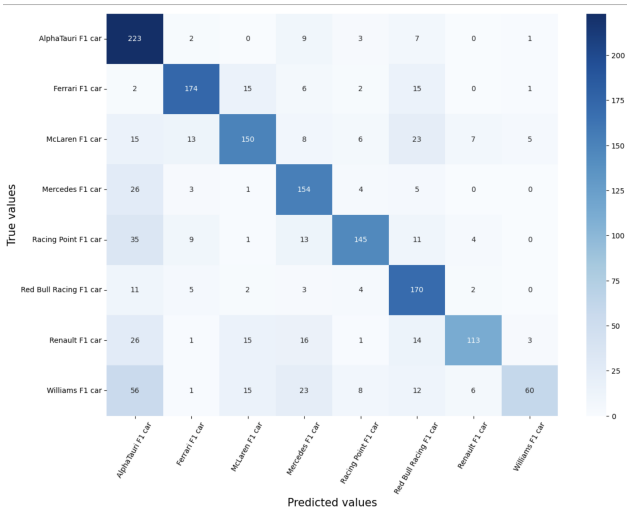

Figure 8: Confusion matrix for VGG-16 with layers freeze

By fine-tuning the top classification layer of the VGG-16 model, we obtain an accuracy that is significantly better than the baseline model. However, we get slightly lower performance compared to the fine-tuned top layer RESNET-18 model. As it can be seen in the confusion matrix shown above, where a significant amount of images from remaining classes are incorrectly classified as a Alpha Tauri F1 car image. Therefore, the model mostly confuses a lot of the other classes as a Alpha Tauri F1 car class.

5.2.3 MOBILENET_V2

Accuracy score: 79.70%

Confusion matrix

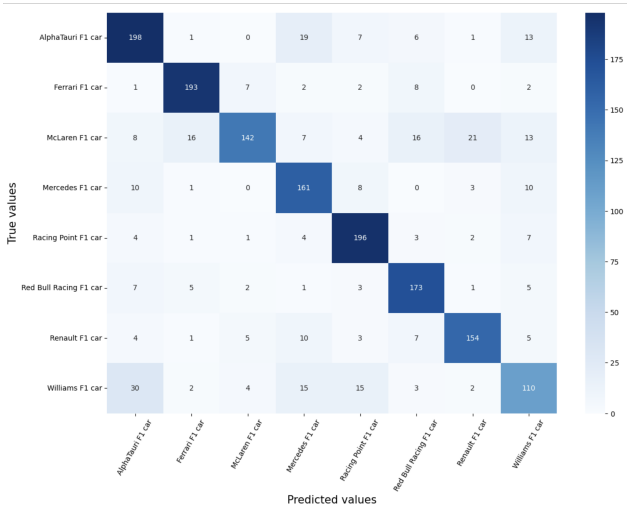

Figure 9: Confusion matrix for MobileNet_V2 with layers freeze

By fine-tuning the top classification layer of the MobileNet_V2 model, we obtain a model where its performance is significantly better than the baseline model and also better performance compared to the fine-tuned top layer VGG-16 and RESNET-18 models. However, the model mostly confuses Williams F1 car images with Alpha Tauri F1 car images. The model also confuses a lot of the Mclaren F1 images as other classes.

In the comparison of the 3 fine-tuned top layer pre-trained models - ResNet-18, VGG-16 and MobileNet_V2 - we observe that the MobileNet_V2 model outperforms the other two and demonstrates a significant improvement in performance over the baseline model.

5.3 Pre-trained Models with Full Fine-tuning

5.3.1 RESNET-18

Accuracy score: 95.50%

Confusion matrix:

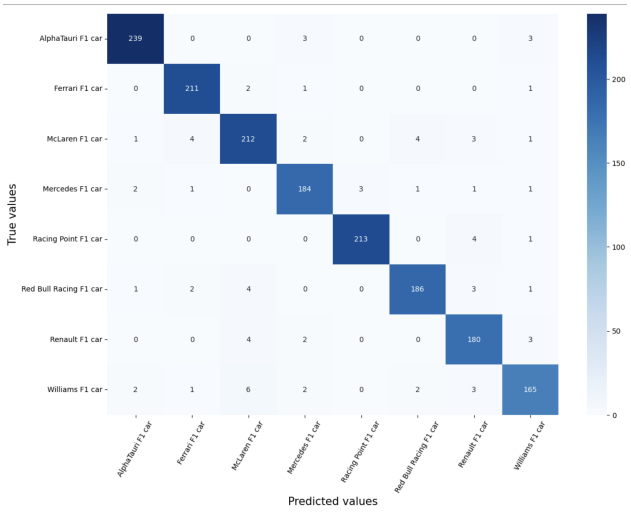

Figure 10: Confusion matrix for ResNet-18 with no layers freeze

By fine-tuning all the layers of the pre-trained RESNET-18 model, we are able to achieve a performance that is significantly better than the baseline model and all the fine-tuned top layer pre-trained models. The model is able to achieve good performance across all classes for image classification.

5.3.2 VGG-16

Accuracy score: 93.99%

Confusion matrix:

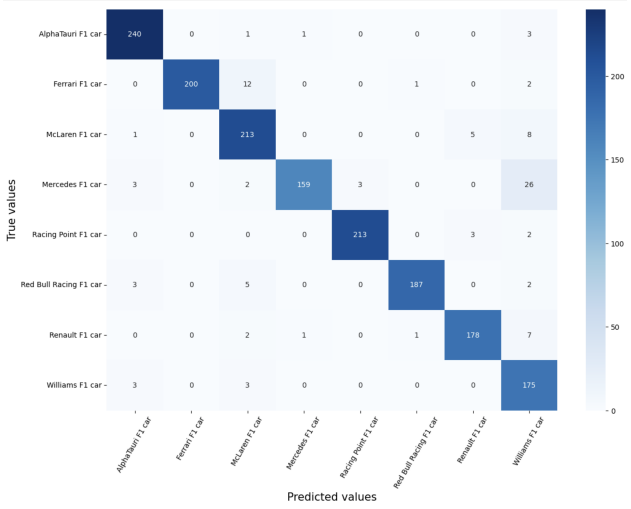

Figure 11: Confusion matrix for VGG-16 with no layers freeze

By fine-tuning all layers of the pre-trained VGG-16 model, we are able to achieve a performance that is better than the baseline model and all the fine-tuned top layer pre-trained models. However, the model performance is lower than the fine-tuned all layers RESNET-18 model.

The model mainly confuses Mercedes F1 car images as Williams F1 car images.

5.3.3 MOBILENET_V2

Accuracy score: 96.70%

Confusion matrix:

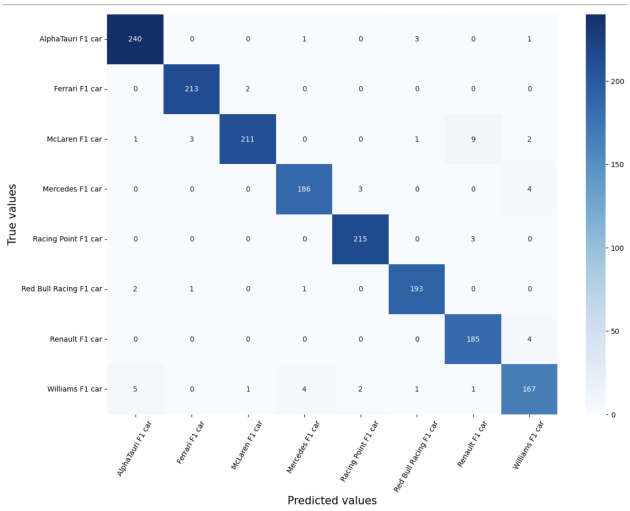

Figure 12: Confusion matrix for MobileNet_V2 with no layers freeze

By fine-tuning all the layers of the pre-trained MobileNet_V2 model, we are able to achieve the best performance across all the models tested thus far. With an accuracy of 96.70%, we are able to mostly correctly classify all classes of F1 car images.

6. Model Explainability

Having trained our models on the augmented dataset, we now attempt to explain why our model predicts a label for a particular image. For this purpose, we used LIME (Local Interpretable Model-Agnostic Explanations) technique to highlight areas of an image that a model does not use for prediction of the label. We input the image for a 2023 Mercedes AMG Petronas F1 car and examine the areas of the image that the model does not use for prediction.

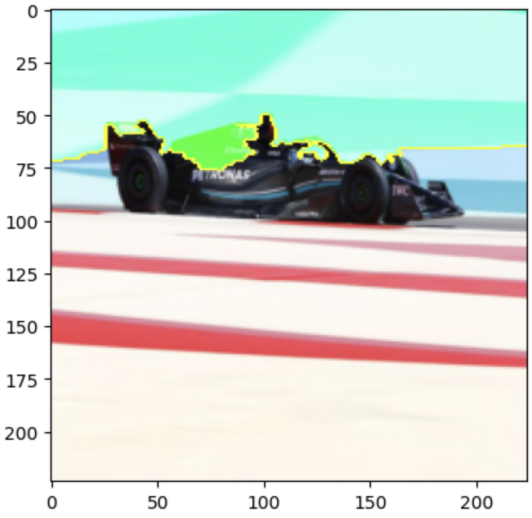

Figure 13: Areas of image highlighted by LIME

The above image highlights the part of the track above the car, which is not used by the model for classification.

7. Web Application Implementation

We implement a web application proof-of-concept (POC) to showcase the predictive abilities of our model. We use Flask and Jinja to deploy the web application, and the code can be found in the GitHub repository that we share along with our paper and presentation. The web application has a simple landing page where the user uploads the photo of a F1 car that he/she wants our model to classify. Once the user feeds the image into the application, the application preprocesses the image, loads the model and feeds the preprocessed image into the model. The output generated by the model is returned as a list of logits, which are converted to probabilities, and finally, the class with the largest probability is returned. We then display that class name in the web application. Below are a few screenshots from our web application.

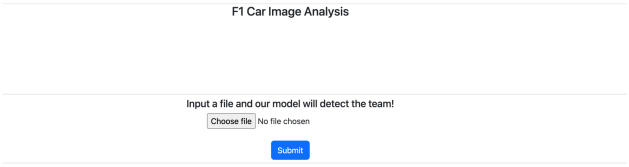

Figure 14: Landing page of the web application

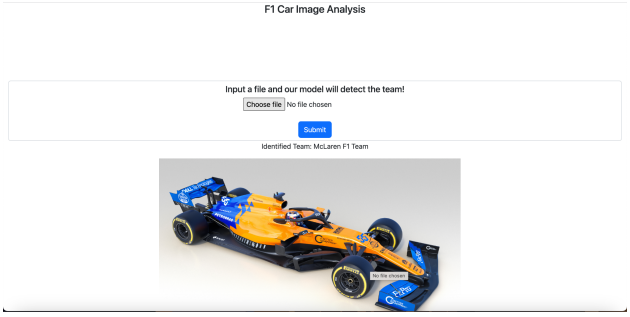

Figure 15: Testing with image of a 2020 Mclaren F1 Car

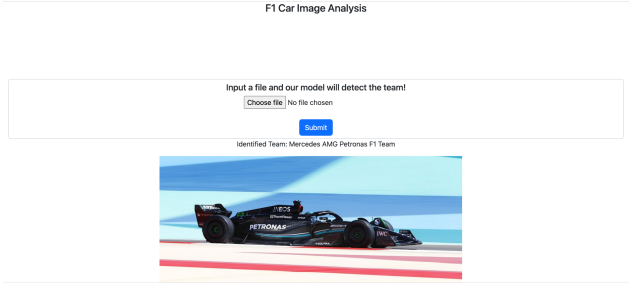

Figure 16: Testing with image of a 2023 Mercedes F1 Car

8. Conclusion and Future Work

In this project, we attempt Formula 1 cars image classification using different deep learning architectures. This can help new fans understand the teams and their origins and have a smoother experience while watching the race.

We approach the problem using a labeled dataset containing 8 distinct car classes (teams) within Formula 1 racing.

It is concluded that MobileNet_V2 with full fine-tuning leads to faster convergence and better prediction results compared to the other models. Significantly, the above conclusion is not final, as some other models could also be tested. Better results could be achieved by using proper resources and overcoming limitations.

Nonetheless, the deduction obtained from this research is very substantial, and proper inferences can be obtained to decide upon the best model.

In the future, traditional models and other deep convolutional neural networks can be implemented and compared. In addition, other large-scale vehicle datasets (Boxcars, BRCars, etc.) can be utilized to evaluate the diversity of models for the fine-grained classification of vehicles.

Another approach that can be explored is to use a bounding box object detector. Specifically, PASCAL VOC file format (an XML file format used by Image Net) can be used to annotate the images. After converting the PASCAL VOC primitive dataset to a TFRecord file (format optimized for tensorflow), an object detector model such as MobileNet SSD can be used to detect the cars and label them.

Acknowledgments

We would like to acknowledge the National University of Singapore for providing us with the necessary learnings and resources to ensure that we complete this project with adequate analysis. We would especially like to thank the NUS School of Computing for these resources.

References

Bhardwaj, S., Gupta, A., & Infanti, C. (2019). *Image Classification of Race Cars. WWT.* https://www.wwt.com/wwt-research/image-classification-of-race-cars [1]

Formula One Cars. Kaggle. Retrieved April 15, 2023, from https://www.kaggle.com/datasets/vesuvius13/formula-one-cars?select=Formula+One+Cars [2]

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. *arXiv (Cornell University).* https://doi.org/10.48550/arxiv.1512.03385 [3]

Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *Computer Vision and Pattern Recognition.* http://export.arxiv.org/pdf/1409.1556 [4]

Sandler, M., Howard, A. W., Zhu, M., Zhmoginov, A., & Chen, L. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. *arXiv (Cornell University*). https://doi.org/10.1109/cvpr.2018.00474 [5]