Vehicle Image Classification Report

Dataset

https://www.kaggle.com/datasets/mohamedmaher5/vehicle-classification/data

This dataset is designed for vehicle classification tasks and contains a total of 5,600 images distributed across seven categories. Each category represents a different type of vehicle.

Preprocessing

Before training any models, some of the images had to be removed from the dataset as they were corrupted or causing errors while running.

The images were pre-processed using tf.keras.applications.mobilenet_v2.preprocess_input(). This function rescales the image pixel values from the normal range of [0, 255] (original RGB image) to the range [-1, 1]. This is important because it prepares the image data in a way that is optimal for the MobileNetV2 model, which was originally trained on the ImageNet dataset.

Then the data augmentation was performed to artificially increase the diversity of the training dataset by applying various random transformations to the existing images. This helps the model become more robust and generalize better to new, unseen data, which is crucial in preventing overfitting.

Experiments

- Experiment 1:
  - Parameters: Dropout=0.2, LR=0.0001, Dense=None, Augmentation=Yes.
  - Final Results: Train Acc: 92.74%, Val Acc: 97.98%, Train Loss: 0.25, Val Loss: 0.11.
  - Observations: This was the base model with frozen MobileNetV2, initial dropout, and learning rate. It showed a strong performance on validation accuracy from the start, which indicates the effectiveness of the pre-trained model.
- Experiment 2:
  - Parameters: Dropout=0.4, LR=0.0005, Dense=None, Augmentation=Yes.
  - Final Results: Train Acc: 93.89%, Val Acc: 98.54%, Train Loss: 0.17, Val Loss: 0.05.
  - Observations: Compared to Experiment 1, this experiment used a higher dropout rate and a higher learning rate. This led to an improvement in both validation accuracy and a reduction in validation loss, indicating that these changes were beneficial for performance.
- Experiment 3:
  - Parameters: Unfreeze=20, Dropout=0.3, LR=0.00005, Dense=None, Augmentation=Yes.

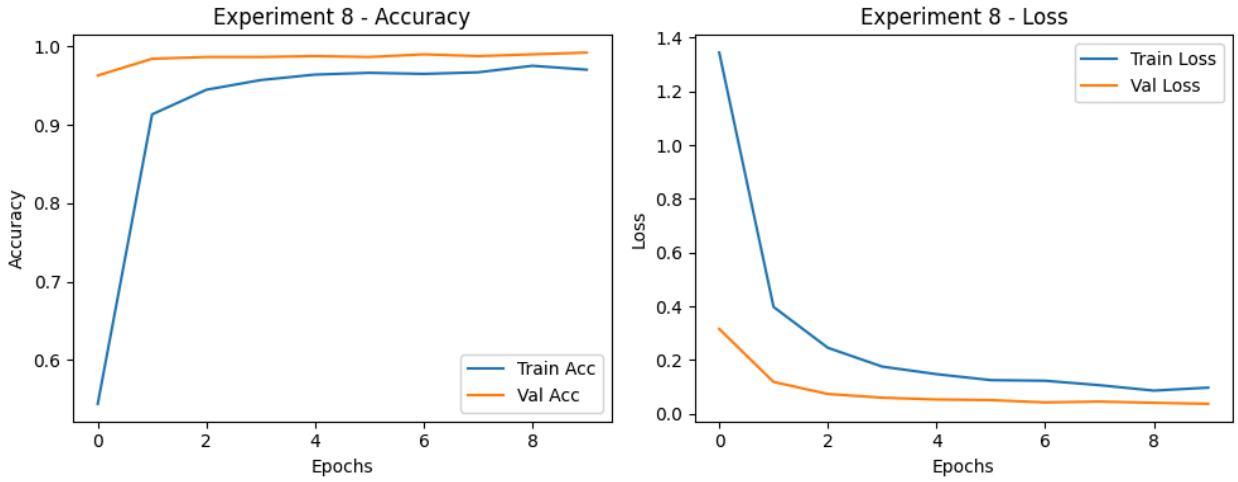
    - Final Results: Train Acc: 97.28%, Val Acc: 98.43%, Train Loss: 0.08, Val Loss: 0.05.
    - Observations: The last 20 layers of the base model were unfrozen for fine-tuning, and the learning rate was decreased significantly compared to Experiment 2. While the validation accuracy saw a slight decrease compared to Experiment 2, the training accuracy improved substantially and the training loss decreased, suggesting better learning and reduced underfitting on the training data due to the fine-tuning.
- Experiment 4:
    - Parameters: Unfreeze=40, Dropout=0.25, LR=0.00001, Dense=None, Augmentation=Yes.
    - Final Results: Train Acc: 96.41%, Val Acc: 98.88%, Train Loss: 0.12, Val Loss: 0.05.
    - Observations: This experiment unfroze more layers (40 layers) than Experiment 3 and used an even lower learning rate. It achieved a higher validation accuracy compared to the previous experiments, suggesting that unfreezing more layers with a carefully selected, very low learning rate further improved the model's ability to learn specific features from the dataset.
- Experiment 5:
    - Parameters: Unfreeze=80, DO=0.30, LR=5e-6, LS=0.1.
    - Results: Train Acc: 95.08%, Val Acc: 98.65%, Train Loss: 0.19, Val Loss: 0.06.
    - Observations: This experiment unfroze a larger portion of the base model (80 layers) and used an even lower learning rate. The validation accuracy was slightly lower than Experiment 4, and the training accuracy was also a bit lower, suggesting that unfreezing too many layers with such a low learning rate might not yield optimal results for this specific setup compared to unfreezing fewer layers.
- Experiment 6:
    - Parameters: Unfreeze=80, FreezeBN, Dropout=0.25, LR=0.000010.
    - Final Results: Train Acc: 98.29%, Val Acc: 98.54%, Train Loss: 0.06, Val Loss: 0.04.
    - Observations: This experiment also unfroze 80 layers, but froze the Batch Normalization layers and used a slightly different dropout and learning rate. The validation accuracy was very high but still smaller than Experiment 4. However, the training accuracy was significantly improved, suggesting that freezing Batch Normalization layers during fine-tuning contributed to more stable and effective training.

- Experiment 7:
    - Parameters: Unfreeze=50, AdamW wd=0.000100, Head L2=0.000100, Dropout=0.25, LR=0.000010.
    - Final Results: Train Acc: 96.63%, Val Acc: 98.99%, Train Loss: 0.11, Val Loss: 0.04.

    - Observations: This experiment used AdamW optimizer with weight decay and L2 regularization on the head's dense layer, along with unfreezing 50 layers. It achieved a very high validation accuracy, surpassing Experiment 6, and maintained good training accuracy. This indicates that the combined use of AdamW and regularization helped in improving generalization and preventing overfitting.
- Experiment 8 (Best Results):
    - Parameters: Unfreeze=40, Dense128, Dropout=0.30/0.20, LR=0.000010.
    - Final Results: Train Acc: 97.04%, Val Acc: 99.21%, Train Loss: 0.10, Val Loss: 0.04.
    - Observations: This experiment achieved the highest validation accuracy of all experiments so far by unfreezing 40 layers and adding a bottleneck Dense layer with 128 neurons, followed by a second dropout. The combination proved to be very effective in boosting performance and minimizing loss.
- Experiment 9:
    - Parameters: SGD+CosineDecayRestarts, Unfreeze=80, Dropout=0.25, LR_init=0.000010.
    - Final Results: Train Acc: 91.35%, Val Acc: 98.09%, Train Loss: 0.40, Val Loss: 0.20.
    - Observations: This experiment switched to the SGD optimizer with a Cosine Decay Restarts learning rate schedule, unfreezing 80 layers. The validation accuracy was lower than some of the Adam-based experiments, and both training accuracy and loss indicate that the model trained less effectively, possibly due to the optimizer choice or learning rate schedule not being as well-suited for this specific setup.
- Experiment 10:
    - Parameters: Unfreeze=60, FreezeBN, GAP-DO=0.30, Dense256+BN+DO=0.20, RMSprop LR=0.000015, ClipNorm=1.0.
    - Final Results: Train Acc: 99.03%, Val Acc: 99.10%, Train Loss: 0.03, Val Loss: 0.04.
    - Observations: This combination of parameters yielded extremely high training accuracy and good validation accuracy, very close to the best, and showed excellent loss values, demonstrating the benefits of fine-tuning a slightly deeper part of the backbone with careful head design and an RMSprop optimizer.

Visualizations

Best Model -



A table containing details of parameter testing and tuning -

Iteration	Parameters	Training and Test Accuracy
1	Dropout=0.2, LR=0.0001, Dense=None, Augmentati...	Train = 92.74% and Test = 97.98%
2	Dropout=0.4, LR=0.0005, Dense=None, Augmentati...	Train = 93.89% and Test = 98.54%
3	Unfreeze=20, Dropout=0.3, LR=0.00005, Dense=No...	Train = 97.28% and Test = 98.43%
4	Unfreeze=40, Dropout=0.25, LR=0.00001, Dense=N...	Train = 96.41% and Test = 98.88%
5	Unfreeze=80, DO=0.30, LR=5e-6, LS=0.1	Train = 95.08% and Test = 98.65%
6	Unfreeze=80, FreezeBN, Dropout=0.25, LR=0.000010	Train = 98.29% and Test = 98.54%
7	Unfreeze=50, AdamW wd=0.000100, Head L2=0.0001...	Train = 96.63% and Test = 98.99%
8	Unfreeze=40, Dense128, Dropout=0.30/0.20, LR=0...	Train = 97.04% and Test = 99.21%
9	SGD+CosineDecayRestarts, Unfreeze=80, Dropout=...	Train = 91.35% and Test = 98.09%
10	Unfreeze=60, FreezeBN, GAP-DO=0.30, Dense256+B...	Train = 99.03% and Test = 99.1%

The remaining visualizations / graphs can be found in the notebook.

How to run

The .ipynb notebook file is very easy to run. You just need to download the file and upload to Google Colab and everything should be populated and be able to run with the “Run all” button.