NAME : SALIM HARUNA
COURSE: DATS 6203
FINAL PROJECT INDIVIDUAL REPORT
Automobile Image Classification

1.0 Introduction

As technology advances, we continue to find labors that can be automated and in return free up humans to work on things that cannot be automated. With the power of machine learning and computer vision, the classification of images is one of those labors.

In the automobile and adjacent industries, much of the communication between customers and companies has been redirected online as of recently. In these industries, such as insurance, dealerships, and resale, customers are often required to upload images of their vehicles to the companies' websites. In order to confirm the vehicle matches the description given by the customer, a checkpoint must be established. With the overwhelming number of images being uploaded on a regular basis, this is a job that must be automated to keep up with demand. By adopting our automobile image classification system, companies can save thousands of human hours annually and thus reduce business expenses.

2.0 Work Done

Most of the work I did was on data preprocessing and some basic level of training the model with vgg 19. The data processing was a bit intense due to processing power bottlenecks which made me think of more efficient ways to process the data and save to file for training.Below are the work i did:

2.1 Data Preprocessing

1. Exported the meta data from matlab to excel.
2. Loaded the meta data into a pandas dataframe
3. From the metadata using the file path, I loaded the images using cv2 to generate the image convolution.
4. The loaded images were resized into 24 X 24 X 3
5. Next thing was to crop the images based on the values of x1, x2, y1 and y2 provided in the meta data .

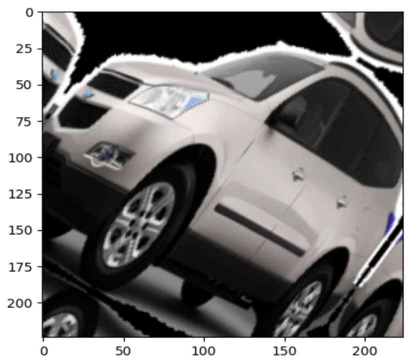
6. I also noticed that since we have so many images, running a loop for 40,000 times to preprocess the data might not be efficient and might fail sometimes due to overloading of the GPU memory.There I decided to preprocess all the train, test and augmented images and save to file. The images were saved in a .npy numpy file format. Although the numpy files were quite large, they loaded in seconds.Therefore during training, the numpy data was loaded and applied for training. The numpy files can be found here.
7. The train data set was split into the 80:20 pareto ratio rule.80% for training and 20% for validation.

2.2 Model Training

1. Defined sequential model
2. Loaded the pretrained vgg19 model using tensorflow keras
3. Flattened the inputs
4. Added early stopping and model checkpoints
5. Trained the model by fitting and saving the best model on each epoch.

3.0 Results

3.1 Augmentation Results:



3.2 Training Results

The best metrics obtained using the VGG19 model are as follows:

1. Hamming loss : 0.654
2. Accuracy Score : 0.40
3. Cohen Kappa Score : 0.3425
4. F1-macro Average : 0.41
5. Last loss value : 0.941

To achieve the metrics above during training, we used the following hyper parameters

1. Activation : Softmax
2. Weights : Imagenet
3. Epochs : 5
4. Batch Size : 64
5. Optimizer : Adam

4.0 Summary & Conclusion

Convolutional neural networks are very good at detecting features in images. I have learned the need to pre process your data appropriately before training.Also to further improve the results, one of the best options will be to improve the image quality because some of the images were small therefore when it resizes it becomes blurry making it difficult for the network to make any meaningful meaning.

5.0 Code

All the code was written in python. I used a repository system, segmenting each function into py files making the code clean and reusable.

Lines of code written : 80 Lines
Percentage of code from internet : 16.25%

6.0 References

1. https://ai.stanford.edu/~jkrause/cars/car_dataset.html
2. https://iq.opengenus.org/vgg19-architecture/