FUNDAMENTALS OF DEEP LEARNING

GROUP PROJECT

Shashank Gollapalli | Nandini Gantayat | Sai Sandesh Nagarur
IESEG SCHOOL OF MANAGEMENT

INTRODUCTION

The objective of this project is to classify vehicles based on images taken at accident sites and train various models to predict the vehicle type in the images and compare their performances. Initially, we used the multi-layer perceptron networks and then proceeded to Convolutional Neural Networks. We have also performed data augmentation and employed transfer learning to improve our model's accuracy.

Dataset

The 'Vehicles-in-accidents' dataset contains images of 4 different types of vehicles: car, bus, truck, and van. Each image is of size 224x224 and has 3 color channels (RGB). The dataset contains a total of 2800 images, with 700 images per class. We will split the dataset into training (70%) and validation (30%) sets.

Pre-processing

The images in the dataset were normalized by dividing the pixel values by 255 and were brought to the range [0,1]. The dataset was then split into training and validation sets

MLP Networks

We will start by training three different MLP networks, with different architectures. We will use the Keras library to create our MLP models. We will use the 'adam' optimizer and the categorical_crossentropy loss function. The input shape of the MLP models will be (224, 224, 3), which corresponds to the size of our images.

The architecture of the MLP networks used is shown as below:

Network	Layers	Neurons per layer
Model 1	3	64, 32, 16
Model 2	4	128, 64, 32, 16
Model 3	5	256, 128, 64, 32, 16

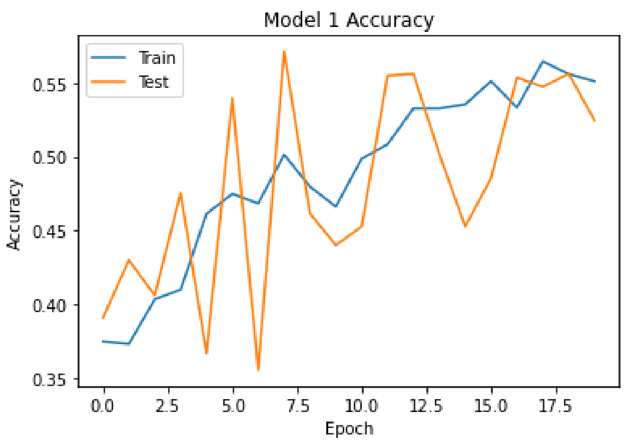
The number of parameters estimated for each model is shown in the table below:

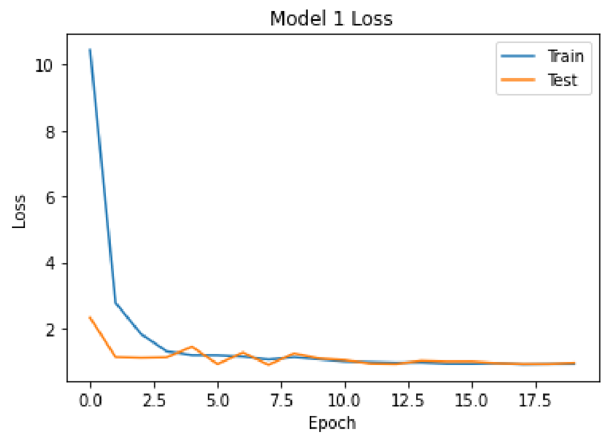
Network	Number of Parameters
Model 1	3,213,507
Model 2	6,433,091
Model 3	12,888,643

The models were trained for 20 epochs, with a batch size of 32 and were then evaluated on the validation set. The accuracy of each Model is shown in the table below followed by the accuracy and loss curves for each model.

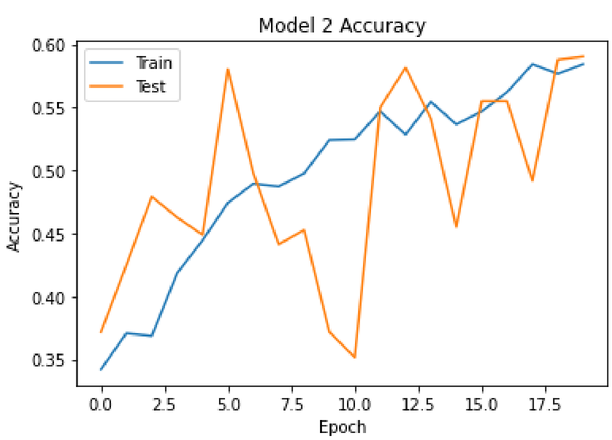
Network	Testing Accuracy	Loss
Model 1	52.46%	0.95
Model 2	59.03%	0.88
Model 3	60.68%	0.86

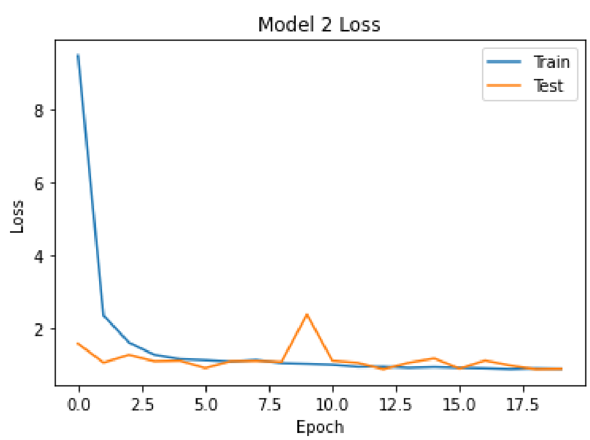
For model 1, the accuracy on the training set increased over the epochs, starting from 0.37 in the first epoch and reaching 0.55 in the last epoch. However, the validation accuracy fluctuated and was only at 0.54 in the last epoch. This indicates that the model may be overfitting to the training data and not generalizing well to new data. Similarly, the loss decreased over the epochs, while the validation loss showed a relatively inconsistent trend, further suggesting overfitting. Therefore, it is necessary to consider other approaches to improve the model's generalization ability, such as adjusting the model architecture.





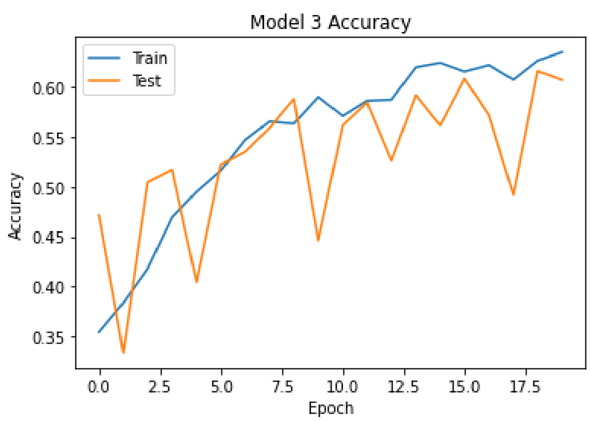
For model 2, we can see that the training accuracy improved from 0.34 in the first epoch to 0.58 in the last epoch. Similarly, the validation accuracy improved from 0.37 in the first epoch to 0.59 in the last epoch. The training loss decreased from 9.47 in the first epoch to 0.88 in the last epoch, while the validation loss decreased from 1.26 in the first epoch to 0.88 in the last epoch. This indicates that the model learned to classify the input data better as training progressed. The model was able to learn the features in the input data and classify it into the correct output class with good accuracy and low loss.

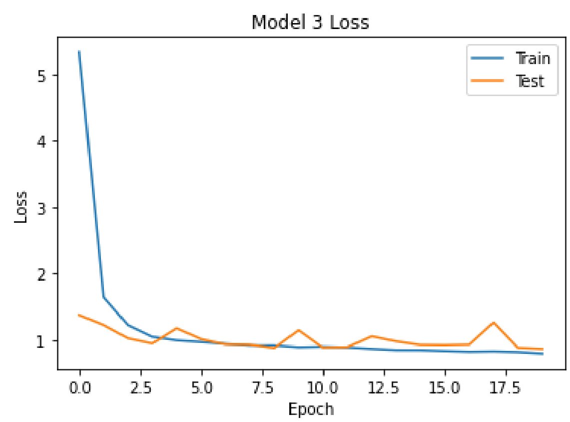




For model 3, the accuracy on the training set started at 0.35 and increased to 0.63 by the end of the training. The model's accuracy on the validation set started at 0.47 and increased to 0.60 by the end of the training. However, the validation accuracy showed some fluctuation during training, indicating that the model may be overfitting to the training data. The same trend was observed for the loss, with the validation loss showing an inconsistent trend relative to the training loss. As the training accuracy continued to improve while the validation accuracy did not, this suggests overfitting..

Based on the results above, we should select Model 2 as it was able to reduce loss, improve accuracy and did not see instances of over-fitting.





<u>CONVOLUTIONAL NEURAL NETWORK (CNN)</u>

We trained 5 different CNN models, the accuracies and total number of parameters of which are shown below:

CNN	Testing Accuracy	Loss	Number of Total Parameters
Traditional CNN	71.04%	1.82	25,236,691
Traditional CNN without Pooling	69.91%	1.41	100,936,915
CNN3 without Padding	70.79%	1.69	24,784,083
CNN4 with GlobalAveragePooling	63.84%	0.78	12,019
CNN5 with Dropout	46.64%	1.25	5,315

Traditional CNN Model

The first CNN model is a traditional CNN model follows the standard architecture pattern of alternating convolutional and pooling layers followed by one or more fully connected layers.

The first layer of the model is a 2D convolutional layer with 16 filters of size 3 x 3 and a 'relu' activation function. The 'padding' parameter is set to 'same', which means the output feature map has the same spatial dimensions as the input.

The second layer is also a 2D convolutional layer with 16 filters of size 3 x 3 and a 'relu' activation function.

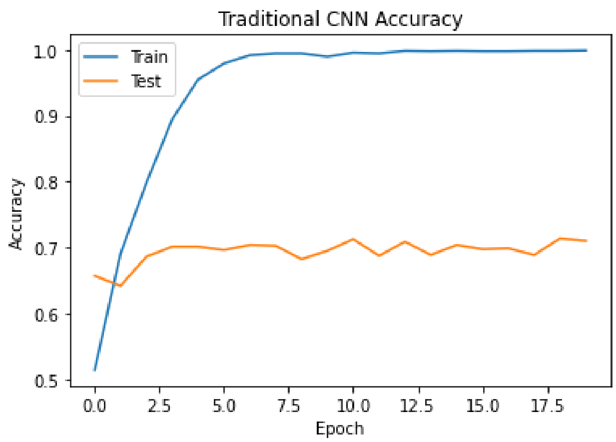
The third layer is a 2D max pooling layer with a pool size of 2 x 2.

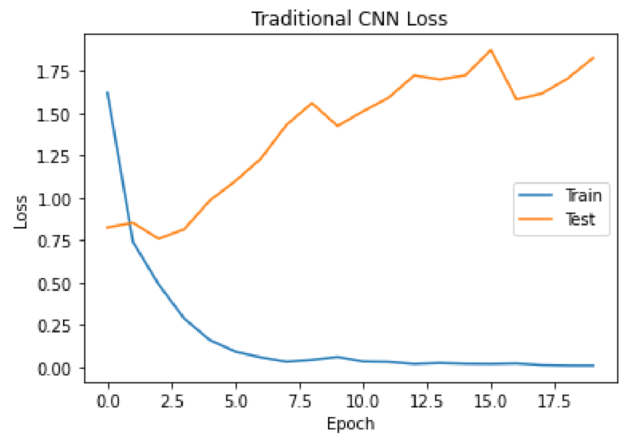
The fourth layer is a flatten layer that flattens the output of the previous layer to a 1D vector.

The fifth layer is a dense layer with 128 neurons and a 'relu' activation function.

The last layer is a dense layer with the number of neurons equal to the number of output classes (in this case, 3) and a 'softmax' activation function.

The accuracy and loss plots show that the model performance on the training set improved with each epoch, which indicates that the model is learning from the training data. However, the model's accuracy on the validation set did not improve significantly after the first few epochs, which indicates that the model may have overfit the training data.

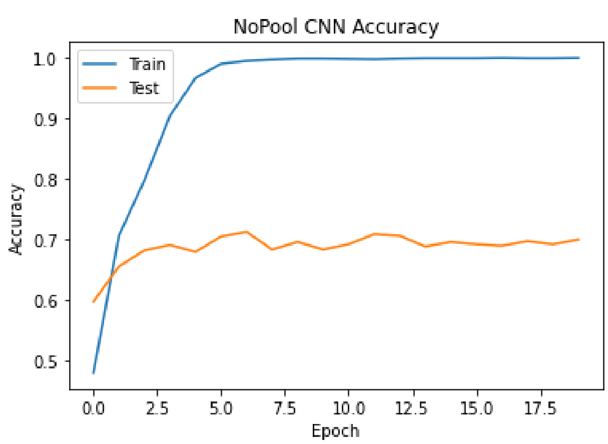


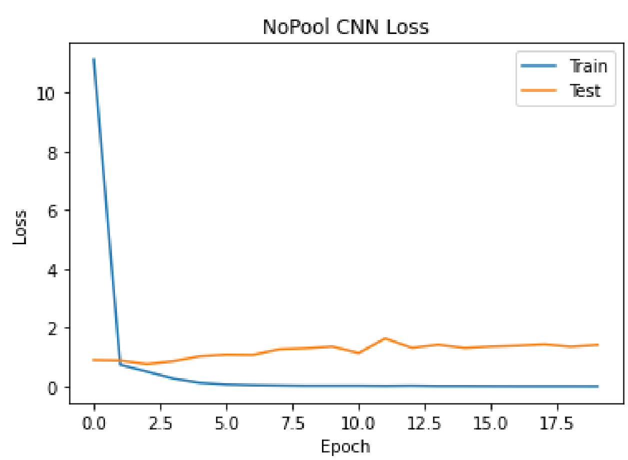


Traditional CNN without Pooling

This architecture is similar to the previous one, but it does not use any pooling layers. Instead, it relies on the convolutional layers to down sample the input image and extract features.

Comparing the results with the Traditional CNN model, the accuracy is slightly lower, but the loss is also lower. This indicates that the Traditional CNN without Pooling might have better generalization performance but might be prone to overfitting, as it has a much larger number of parameters (over 100 million) compared to the Traditional CNN model (about 25 million).

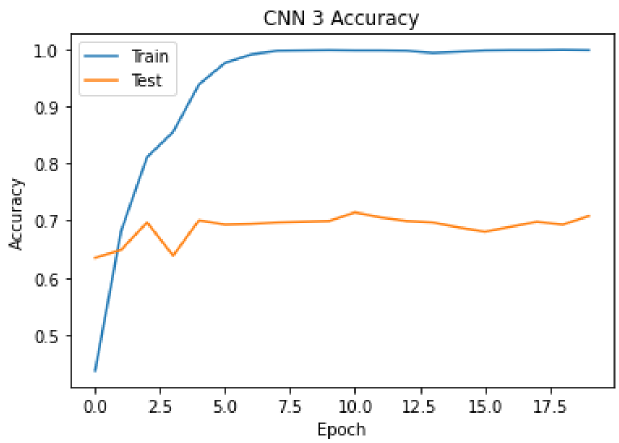


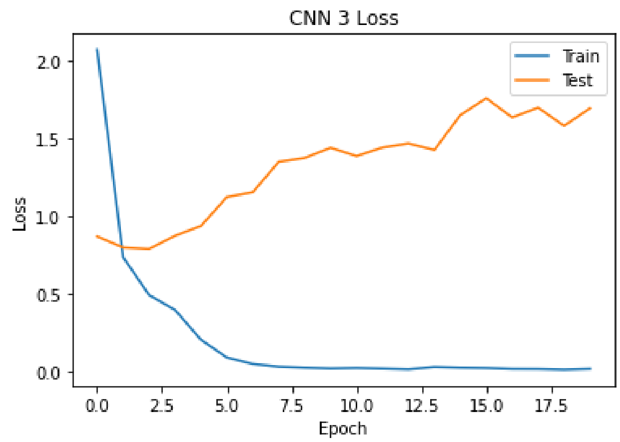


CNN3 without Padding

The architecture of this model consists of two convolutional layers with 16 filters each, followed by a MaxPooling2D layer, flattening, and two dense layers.

Compared to the previous models, this model has a better performance with a higher accuracy and lower loss.

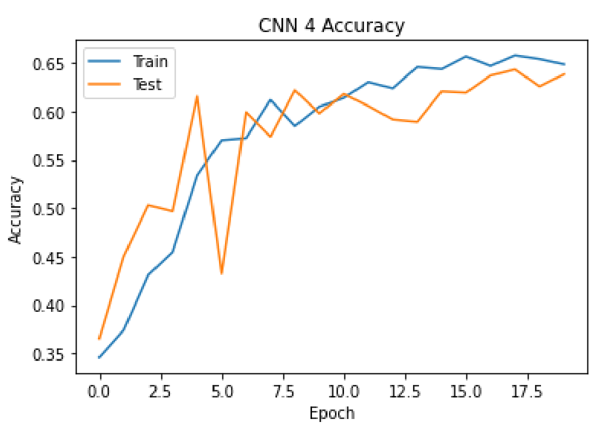


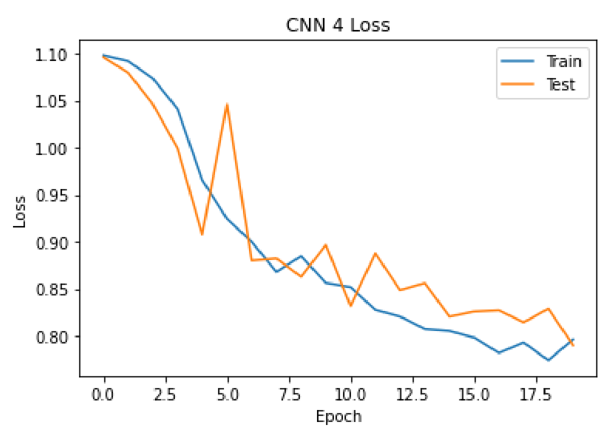


CNN4 with GlobalAveragePooling

The architecture consists of 4 convolutional layers, 2 max-pooling layers, and 1 Global Average Pooling layer, followed by two fully connected layers.

The accuracy of the training set increases as the number of epochs increases, but the accuracy of the validation set reaches a plateau after around 20 epochs, which may indicate overfitting. The loss of the training set decreases over time, but the loss of the validation set reaches a plateau after around 10 epochs.

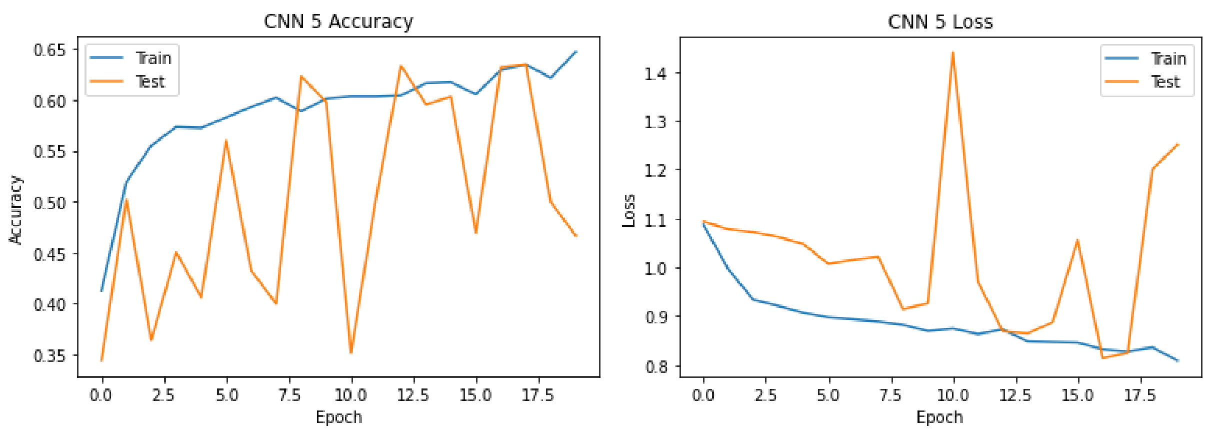




CNN5 with DropOut

The architecture of this CNN model consists of two convolutional layers with 16 and 32 filters respectively. Between the convolutional layers, there are three dropout layers with a rate of 0.25 to prevent overfitting. The first dropout layer is applied after the first convolutional layer, and the other two are applied after the global average pooling layer and batch normalization layer respectively. After the dropout layers, there is a global average pooling layer, which calculates the mean of the feature maps of each filter. This reduces the number of parameters and helps to avoid overfitting. There is a batch normalization layer to normalize the output of the global average pooling layer followed by a Dense layer.

This model is not performing very well, as it has a relatively high loss and low accuracy. It is likely that the model is overfitting the training data and not generalizing well to new data.



CNN Conclusion

Traditional CNN is the most suitable choice among the given models because it has the highest testing accuracy (71.04%) and a reasonable loss value (1.82). A high testing accuracy indicates that the model can accurately classify images into their respective classes, which is the primary goal of the image classification task.

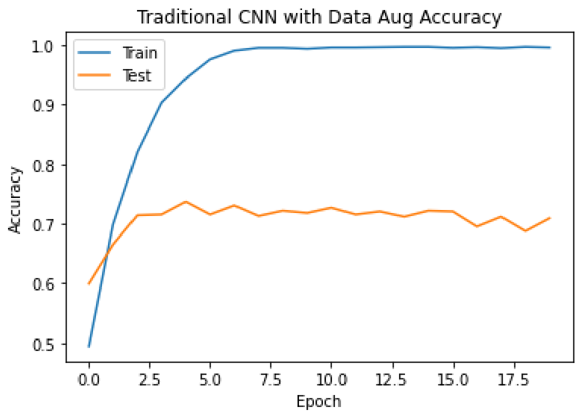
Additionally, Traditional CNN has a moderate number of total parameters (25,236,691), which means it strikes a good balance between model complexity and computational efficiency. Having too many parameters can lead to overfitting, while having too few parameters can result in underfitting, which can negatively affect the model's performance.

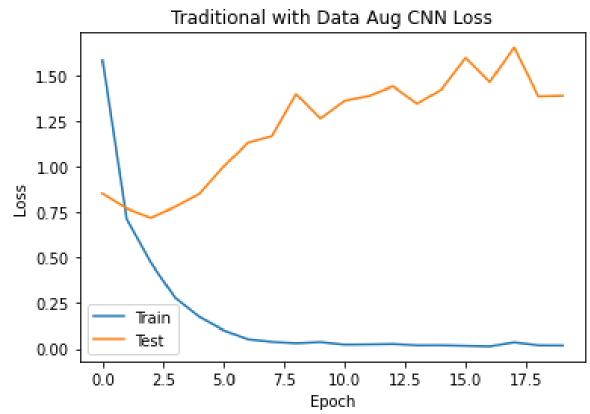
Comparison of CNN and MLP

Model	Testing Accuracy	Loss
Traditional CNN	71.04%	1.82
MLP Model 3	60.68%	0.86

DATA AUGMENTATION

Data Augmentation was done on the traditional CNN model which had the best accuracy. Following is the output we received:





Testing Accuracy: 73.45%

Testing Loss: 0.61

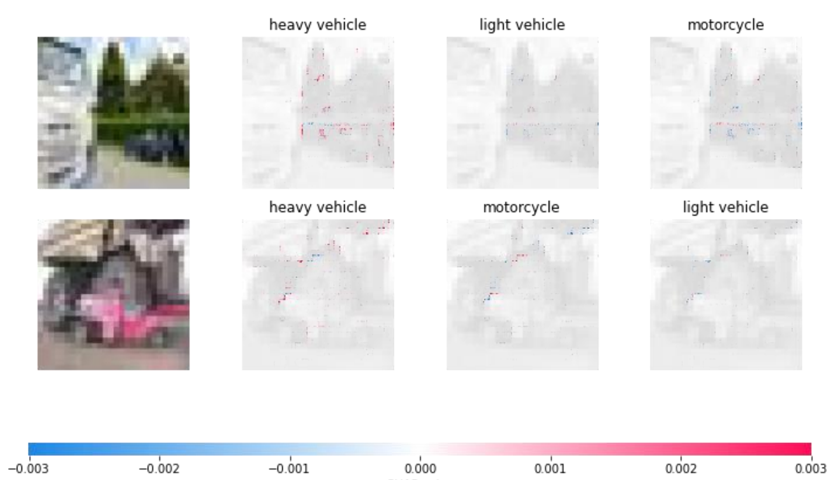
Data Augmentation resulted in increase in the test accuracy and decrease in test loss.

INTERPRETABILITY

SHAP

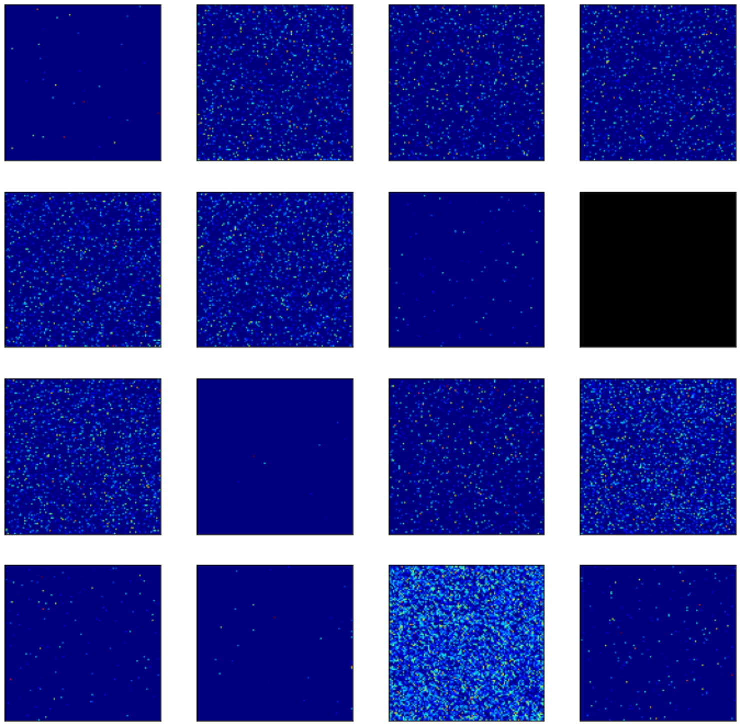
This method helps in understanding in to what extent each feature is contributing to the final prediction made by the model.

The output as seen below, shows original images and the class which these image belongs to. The red colored heatmap is higher in heavy vehicle for the first image which implies that the randomly selected image belongs to light vehicle. For the 2nd image we see that magnitude of heat colored is higher in heavy vehicle class.



Layer-wise Relevance Propagation

The output of LRP is a relevance map that highlights the important regions of the input that contribute most to the model's prediction.



TRANSFER LEARNING MODEL

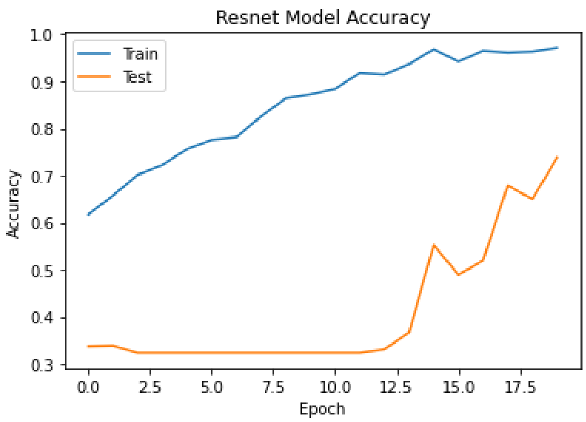
We trained 3 different Transfer Learning models on the best performing CNN model to further better the results. The accuracies and total number of parameters are shown below:

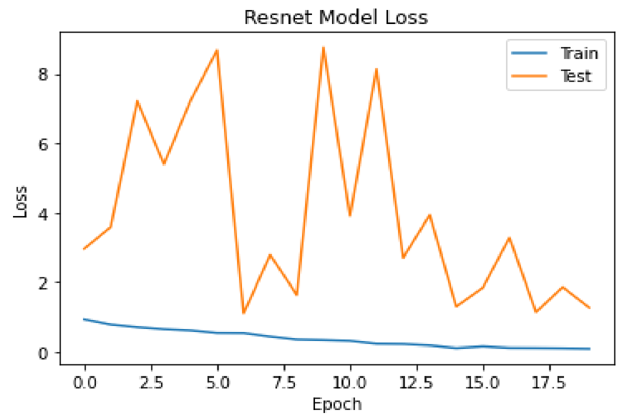
TLM	Testing Accuracy	Loss
ResNet	73.83	1.26
Inception	70.16	1.16
VGG	67.42	0.79

TLM	Number of Parameters
ResNet	23,903,907
Inception	22,102,595
VGG	14,793,315

RESNET MODEL

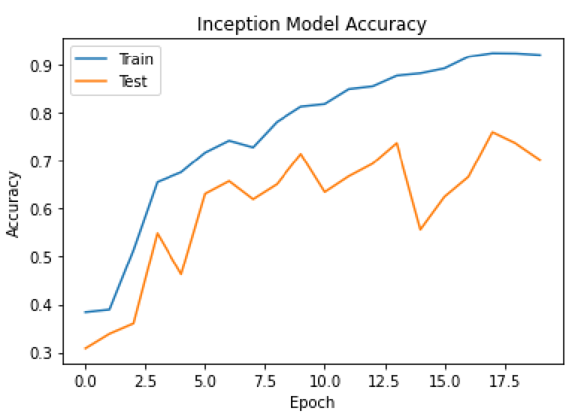
The ResNet model is fit on the traditional CNN. This model is supposed to solve the problem of vanishing gradients as a pre-trained model with fixed weights is fit on top of the traditional CNN model. The accuracy has improved in comparison from 71.04% to 73.83% and the loss has decreased as well, but the graphs below show us that there is overfitting.

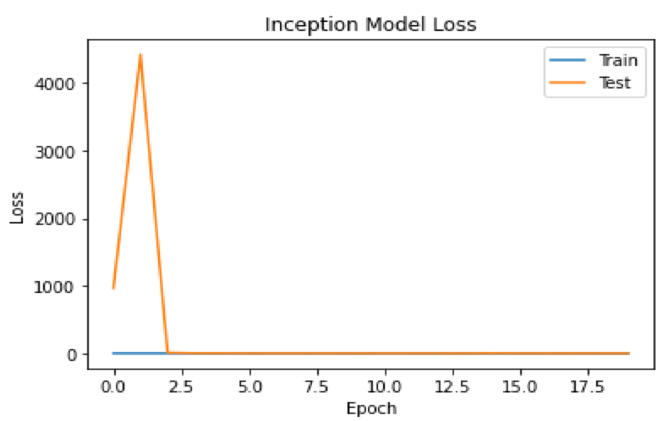




INCEPTION MODEL

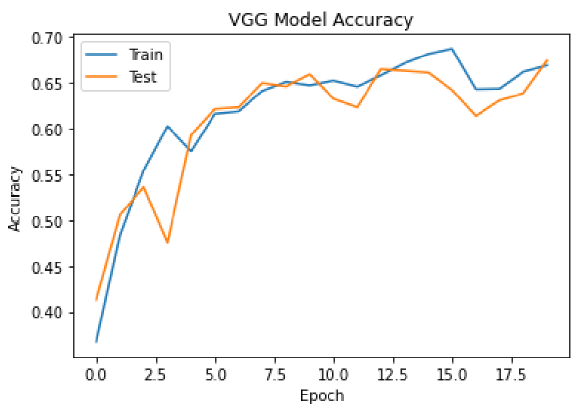
The inception model is fit on the CNN and this allows the network to capture features at different scales by applying a structure of multiple parallel convolutional layers. This method results in clear overfitting and in accuracy reduction but the loss has decreased.

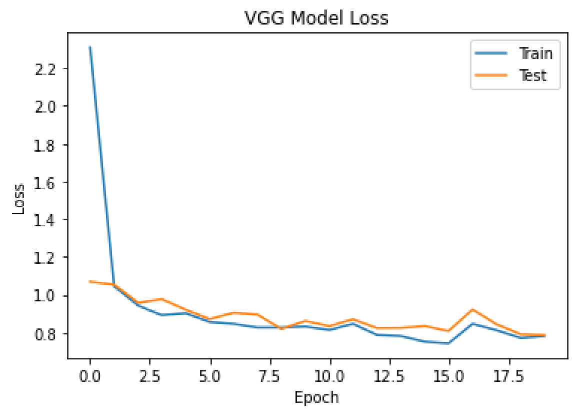




VGG

The VGG model is fit on the CNN and this produces the best results. We can see that the accuracy has reduced slightly to 67.42% but the gap between the train and test set is very low which means that there are is no over/under fitting and this is the most accurate model that we have produced. The loss is very low too.





TRANSFER LEARNING MODEL CONCLUSION

Though the ResNet model produces the best accuracy, the best performing model is the VGG model. There is a slight compromise in test accuracy but it is the most reliable model as it has the smallest variance between the train and test sets. It is also computationally the most efficient model with only 14,793,315 parameters to train whereas the other models have more than 22,000,000 parameters.

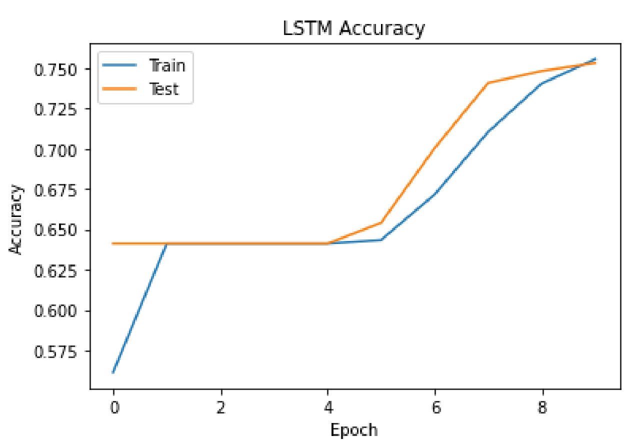
RECURRENT NEURAL NETWORK

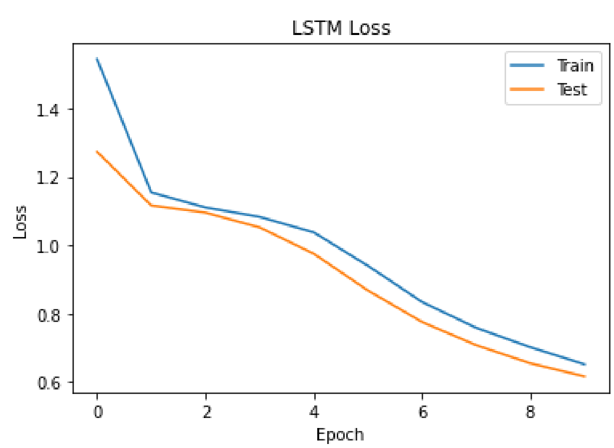
Three RNN models were run with the following parameters:

Network	Testing Accuracy	Loss	Number of Parameters
LTSM	65.84%	1.03	689,733
GRU	62.00%	1.91	677,573
Bi-Directional Simple RNN	61.25%	2.58	665,349

LSTM

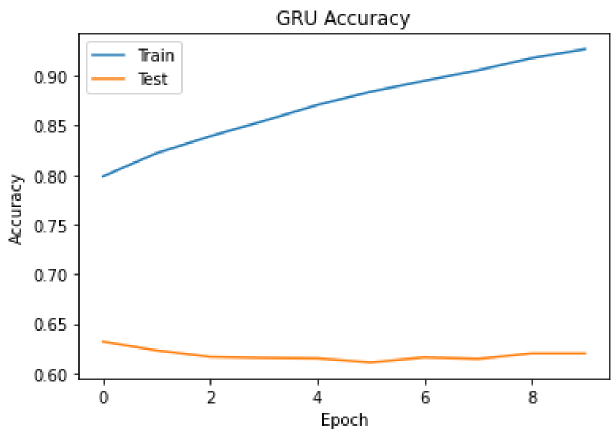
As seen in the graph below, there are no signs of under-fitting or over-fitting and the loss decreases over epochs showing that the model is able to generalize over new data.





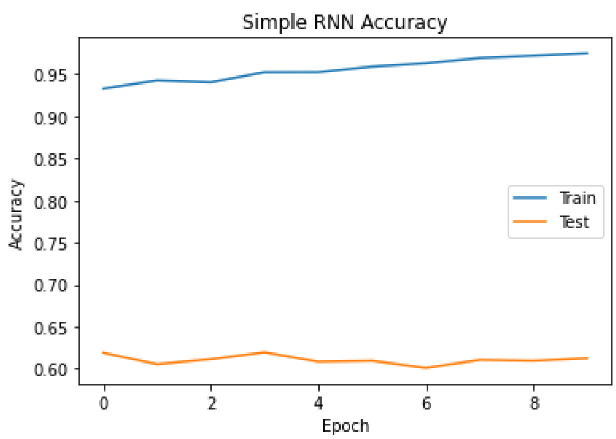
GRU

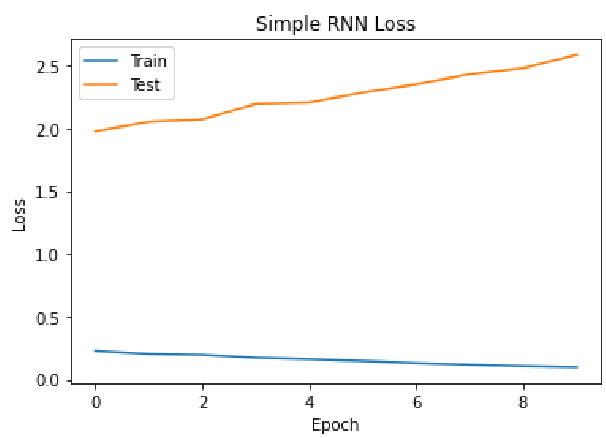
This model shows signs of over-fitting as there was a massive gap between train and validation accuracy. The loss was very high as compared to LSTM.



Bi-Directional Simple RNN

This model also had over-fitting and the loss is very high and accuracy too low as compared to LSTM model.





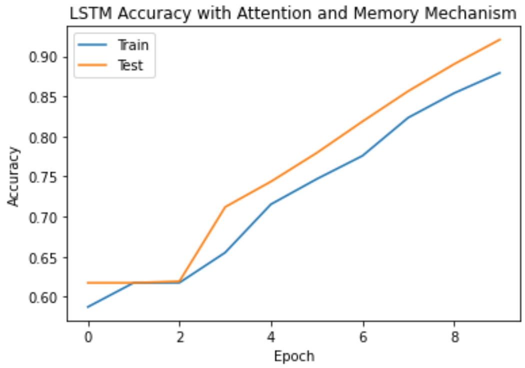
Hence, the LSTM was selected as the best performing model among the 3 RNN as it had the highest accuracy, lowest loss and did not show any signs of over/under fitting.

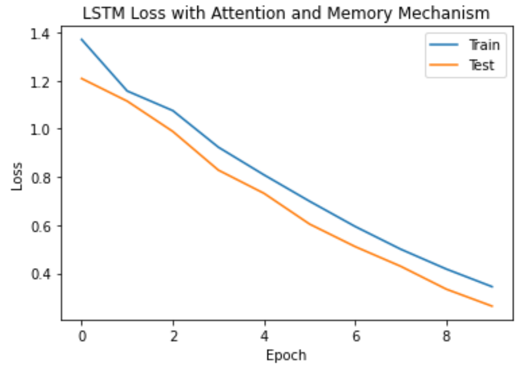
ATTENTION AND MEMORY MECHANISM

Attention and memory mechanism was applied on LSTM as it was our best performing RNN. The following is the comparison of LSTM pre and post applying attention and memory mechanism:

Network	Testing Accuracy	Loss	Number of Parameters
LTSM pre attention and memory	65.84	1.03	689,733
LTSM post attention and memory	63.49	1.30	709,938

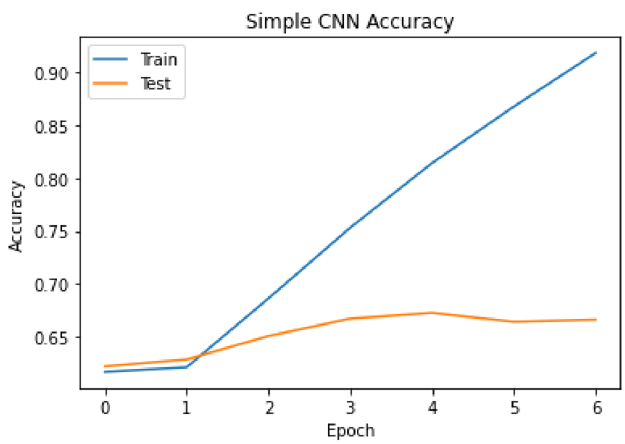
As seen above, there is an increase in the Number of parameters and loss for pre and post attention and memory mechanism. However, there is a decrease in accuracy and showed signs of underfitting in the model post applying attention and memory mechanism, which was absent in the model without attention and memory mechanism. Hence, we can conclude that the previous model is a better model as it has better accuracy, loss and shows no signs of under/over fitting.

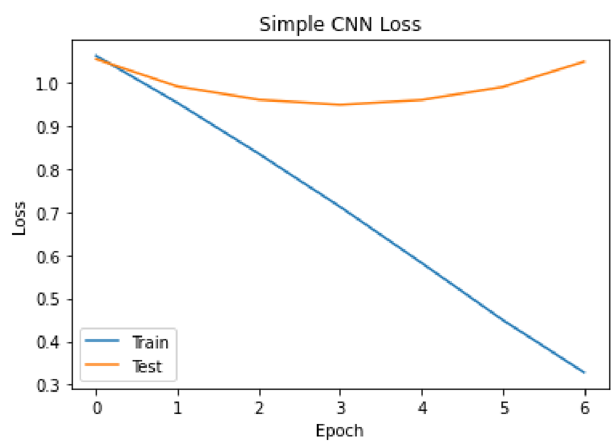




SIMPLE CNN

A simple CNN model was run on the same dataset for comparison. However, the model saw heavy over-fitting as noted in the graphs below:





The comparison between our best RNN model and a simple CNN model is as shown below:

Network	Testing Accuracy	Loss	Number of Parameters
LTSM RNN	65.84	1.03	689,733
Simple CNN	66.64	1.04	685,509

Although simple CNN has a higher accuracy, we cannot run this model as it has very high overfitting.