**Name: Martin Tran → ID: 28758544**

**Name: Ba Nguyen → ID: 27917223**

**Contribution: ‘EQUAL AMOUNT’**

**Introduction:**

The focus of this report is to explore the processes that were taken to complete the final neural network project. The final project involves developing a neural network that classifies different sorts of vehicles. The vehicles which will be classified by the network are motorbikes, bicycles, boats, buses, trains, cars and aeroplanes. So, the neural network will be split into two functions. The first part of the neural network is to function as simple image classification network. Simply, it takes an image and outputs what class it should belong to. The second part of the neural network is to function as an object detection network. In contrast to an image classification network, an object detection network labels where in the image the specific class is. For example, once the network is presented with an image of a vehicle, the network will pinpoint where in the image the vehicle is, by putting a box around the vehicle.

**Image Classification:**

*When developing an image classification network. One of the important things to think about is the structure of the model. In the field of neural networks, typically the two main structures which are used are ‘shallow’ and ‘deep’ networks. To explain simply, a ‘shallow’ network is a simple network which generally requires less computational power relative to a ‘deep’ network. The reason is because ‘shallow’ networks are generally smaller in size, typically usually only having one hidden layer and one fully connected layer. ‘Deep’ networks on the other hand have more hidden layers and fully connected layers. Both structures have their pros and cons, so the following pages will show the differences between ‘shallow’ and ‘deep’ networks and how they perform with the given dataset of vehicles.*

1. **Shallow Neural Network:**

For image classification using shallow neural network,

**Structure of Shallow Neural Network:**

**Convolutional Layers:**

Includes:

1. Convolutional Layer 1 ( Kernel Size = 3, Padding =0, Stride = 2)
2. Relu()
3. Convolutional Layer 2 ( Kernel Size = 3, Padding =0, Stride = 2)
4. Relu()
5. Convolutional Layer3 ( Kernel Size = 3, Padding =0, Stride = 2)
6. Relu()
7. Max Pooling Layer (Kernel Size =3, Stride =3)
8. The output matrix then be reshaped to 2D matrix of (number of images in the dataset, 1152)

After the convolutional layers is 2 linear layers:

1. Linear layer with input weight size of (1152, 128)
2. Relu()
3. Linear layer with input weight size of (128, 7)

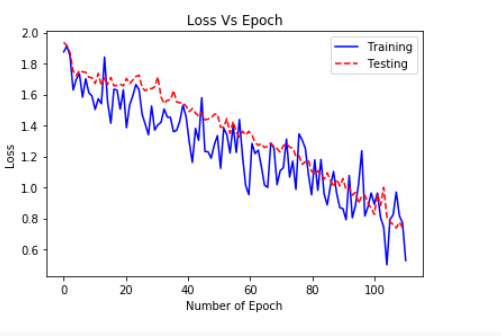
**With VOC dataset:**

**HYPER-PARAMETER:**

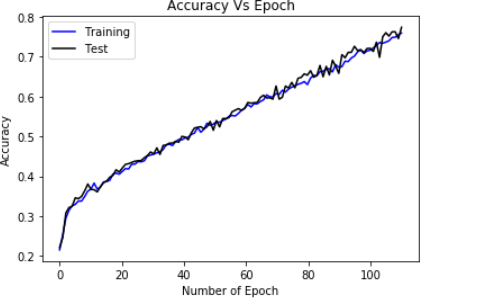
**Batch Size: 100**

**Learning Rate: 0.0001**

**Optimizer: ADAM**

***Plotting Training and TestLoss per Epoch:***

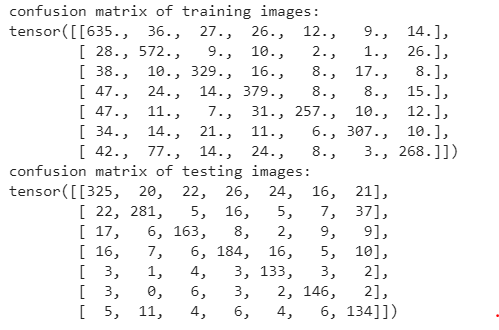
***Plotting Training,and Test Accuracy per Epoch:***



**Analysis:**

Just from the plots which are presented above, it seems as if that the following shallow network has performed extremely well with the provided dataset. Surprisingly, there is no indications of overfitting. Typically, with an overfitted network the loss of the testing data would diverge and increase rapidly. However in this case, the loss of the testing and training are decreasing at the same rate. What makes this result extremely surprising is that there were no overfitting counter measures included in this network. Even with no counter measures, the shallow network performed extremely well with the following dataset. However the only way to truly see if the network is ‘good’ or not is to test it with completely random images which will be shown later on.

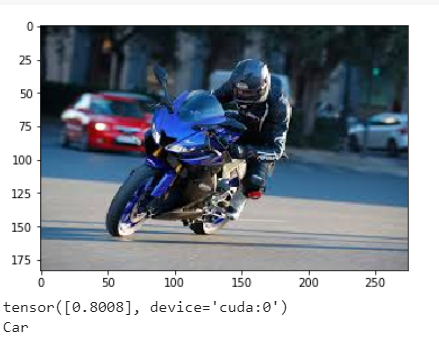
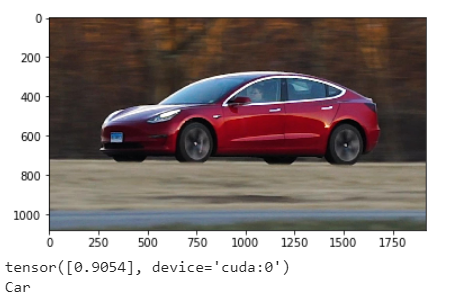
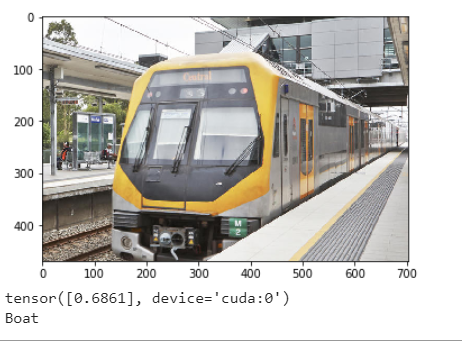
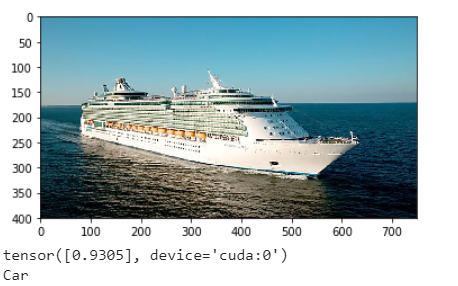
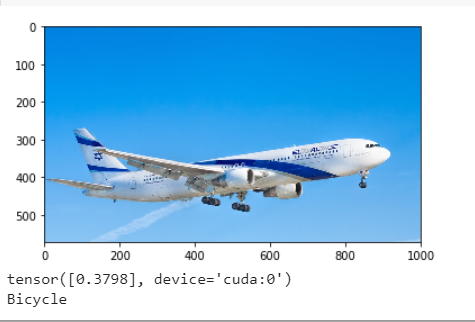
**Confusion Matrix:**

****

The model trained by shallow neural network provides a significantly large number of incorrect predictions in both training dataset and testing dataset. The confusion matrix given by the figure above contains 7 rows and 7 columns correspond to 7 expected labels and 7 predicted labels.

Despite the model predicted correctly in most cases but there are still, for example, 77 images of boats was predicted as aire planes in the training dataset, or 26 train pictures was predicted as car in testing dataset.

**Choosing 5 Random Images from the internet**

****

**Analysis:**

*From the images which are shown above, one can obviously tell that the network has performed pretty badly with the 5 randomly selected images. Labelling a boat as a ‘car’ and a ‘train’ as a ‘boat’.*

*Even though the network has performed extremely well with the provided test and training data, it seems when the network is presented with completely random images it doesn’t know how to make appropriate guesses. This just shows that the following network is bad at ‘generalisation’. It seems as if the network has grown too accustomed to the images presented in the dataset. So, one might say that the reason why the network is performing badly with the random images is because there wasn’t enough variation in the presented dataset. Another possible reason is that shallow network itself isn’t fit for ‘generalisation’, there aren’t enough layers to extract the finer features of each class. As with more features the network will be able to confidently classify classes with a higher accuracy.*

1. **Deep Neural Network:**

**Hyperparameters:**

**Batch Size: 100**

**Learning Rate: 0.0001**

**Optimizer: ADAM**

**Structure of Deep Neural Network:**

**Convolutional Block 1:**

Includes:

1. Convolutional Layer 1 ( Kernel Size = 3, Padding =1)
2. Relu()
3. Convolutional Layer 2 ( Kernel Size = 3, Padding =1)
4. Relu()
5. Max Pooling Layer (Kernel Size =2, Stride =2)

**Convolutional Block 2:**

Includes:

1. Convolutional Layer 1 ( Kernel Size = 3, Padding =1)
2. Relu()
3. Convolutional Layer 2 ( Kernel Size = 3, Padding =1)
4. Relu()
5. Max Pooling Layer (Kernel Size =2, Stride =2 )
6. Dropout Layer ( Probability = 0.05 )

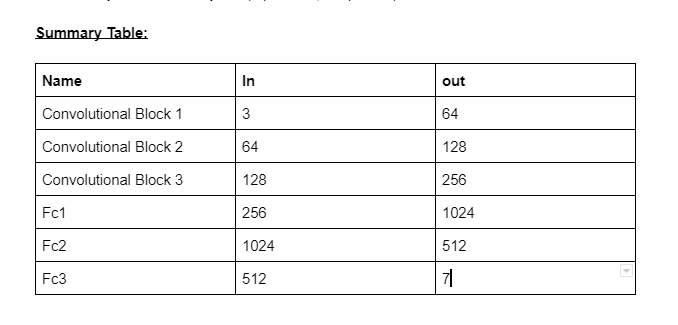
**Convolutional Block 3:**

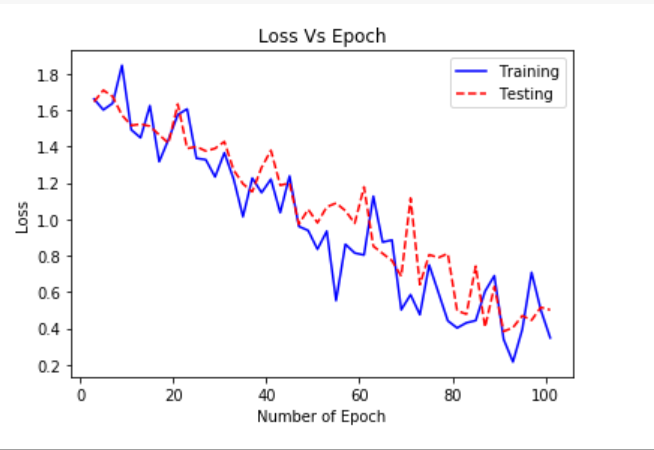
Includes:

1. Convolutional Layer 1 ( Kernel Size = 3, Padding =1)
2. Relu()
3. Convolutional Layer 2 ( Kernel Size = 3, Padding =1)
4. Max Pooling Layer (Kernel Size =2, Stride =2 )

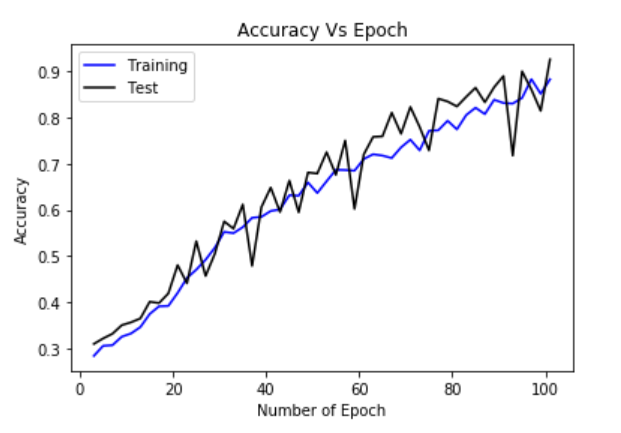
After the convolutional blocks is a:

1. GAP → Global Average Pooling Layer
2. Dropout Layer (Probability: 0.1)
3. Fully Connected Layer 1 ( Input: 256, Output:1024)
4. Relu()
5. Fully Connected Layer 2 ( Input: 1024, Output: 512)
6. Relu()
7. Dropout Layer (Probability: 0.1)
8. Fully Connected Layer 3 (Input: 512, Output: 7)



***Plotting Training and Test Loss per Epoch:*** 

***Plotting Training,and Test Accuracy per Epoch:***



**Analysis:**

From the plots which are provided above, one can definitely tell that the model has performed extremely well on this dataset. Just from inspection, there doesn’t seem to be any form of overfitting/overtraining. As the accuracy of the training dataset increases, so does the accuracy of the testing dataset. The main difference between these plots shown above and the ‘shallow’ network plots is it fluctuates more. The reason why there are fluctuations in the accuracy and loss is because an augmented data set has been included in this model. So, the network essentially switches datasets every second epoch, it will be training with both augmented training images and the stock training images. With more images the networks ability to generalise things becomes a lot better. Another main difference is the computation time, it took a lot longer for the network to learn with the presented dataset in comparison to the shallow network. One of the tradeoffs when choosing ‘deep’ and ‘shallow’ network is computational time. ‘Deep’ networks take longer to learn but they extract more features in comparison to a shallow network which learns faster but extracts less features.

**Type 5 Classified Images of each class:**

**Class 0: Car**

**Class 1: Aeroplane**

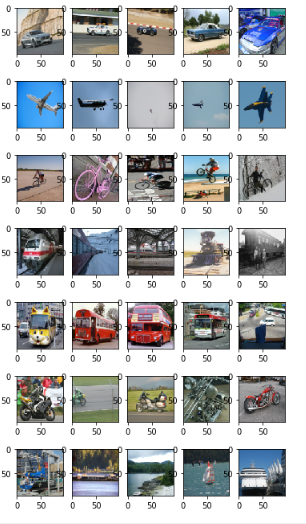
**Class 2: Bicycle**

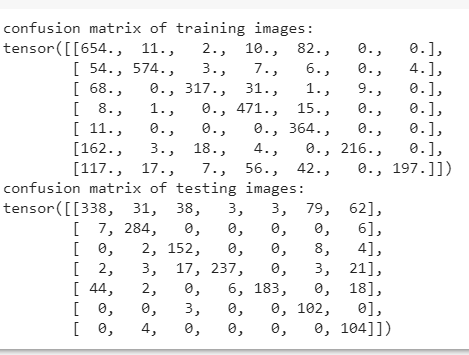
**Class 3: Train**

**Class 4: Bus**

**Class 5: Motorbike**

**Class 6: Boat**

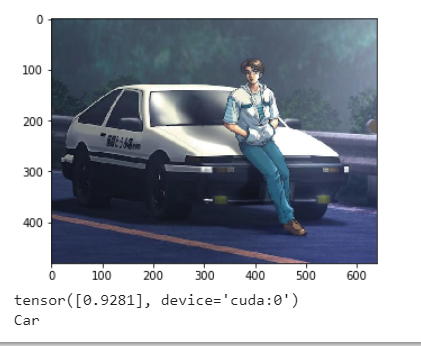


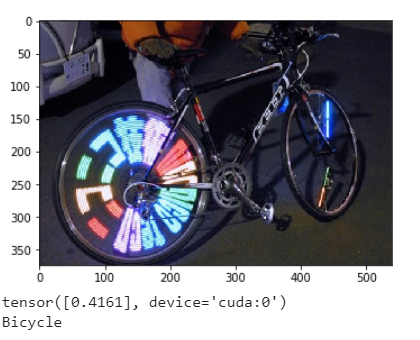
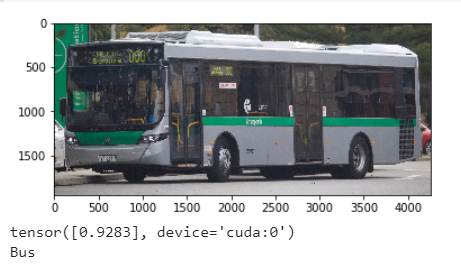
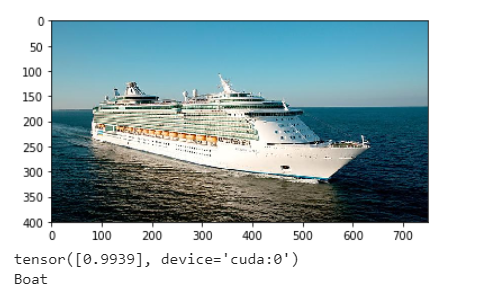


*From the results which are depicted above it seems that the network has performed extremely well on the testing and training dataset. Both achieving an accuracy of above 90%. Although the network seems to perform extremely well on the training and testing dataset, the images that have been chosen for this dataset may not be unique enough. This was particularly evident when analysing the shallow neural network. Even though the shallow network performed extremely well with the dataset it performed badly with randomly selected images.*

*So, the best way to check if the network has been properly trained and tested is to simply pull random images from the internet.*

**Choosing Random images on the internet to see how network performs:**





**Analysis:**

In comparison to the shallow network the deep neural network has performed significantly much better at generalising the randomly presented images. Obviously it isn’t 100% perfect, but overall it gets most of the images correct. One of the reasons why this model is better at generalising the images than the shallow network, is because with a deep neural network there are more layers. With more layers, network will be able to extract more unique features of each class. Thus making the network better at distinguishing the different classes.

**Summary**:

After exploring both the ‘shallow’ and ‘deep’ model one can confidently say that the ‘deep’ neural network performed the best out of the two. It was able to perfectly guess the images presented in the dataset, and it was also able to guess random images on the internet with a relatively high degree of accuracy. The shallow network, was also able to confidently guess the images in the dataset, but when presented with completely random images it performed extremely badly. One of the important takeaways from comparing the two models is understanding the trade off of computation time. Even though the shallow network was able to learn faster, it wasn’t able to appropriately generalise images. While the deep neural network took a lot longer to learn, at the end of the day it was able to confidently generalise the random images.

**Object Detection:**

*After successfully creating the image classification network. The next step was to somehow expand this network to something more useful and complex. One possible improvement is to include object detection. As well as identifying the class of the image, the model should also pinpoint where in the image that particular class is. So one way to show where the class is to simply add in a boundary box. The boundary box will be used as mark to show exactly where the class is.*

*There are multiple ways to include object detection to one’s network. One of the popular ways to include object detection is to use a pre-trained models such as ‘YOLO’ or ‘RNN’. The only issue with these sorts of models is that there are very computationally expensive. Typically, depending on the dataset and number of labels there are, the training process could take hours. If one had the time and computational power to run these models, object detection would be an extremely easy task.* ***So now the question is, is it possible to create an object detection network without relying on these models?***

*Yes, one way to include object detection in an image classification network is via the network visualization technique, ‘occlusion sensitivity’. Typically, this visualisation technique is used to show the significant features which are picked up in an image. But if used appropriately it can also be used to pinpoint classes in an image, which will be explained later on.*

***Occlusion Sensitivity Algorithm for object detection:***

*For any images that are correctly classified, a mask with a size of (45x45) will then scan the augmented image which was resized to 96x96 pixels RGB matrix with the stride of 35. The image uses a mask to calculate softmax score at every part it goes through. To do so, the mask image needs to be evaluated by deep neural network that was trained previously. At any point that has the softmax score above a threshold value of 0.85, a bounding box will be plotted. After the mask finished scanning, all the bounding boxes need to be combined together to produce a final bounding box. Note since the model works with 96x96 images, the bounding boxes that have been computed are relative to that 96x96 image. So the bounding box needs to be scaled so it matches the input image. After the bounding boxes have been scaled, the bounding boxes are then passed through a function which combines are bounding boxes into one big bounding box.*

*Another method to evaluate the coordinates of the transport is using the trained model to find at what point, the image has the object. To start, the 96x96 image is input to be scanned by a mask of 20x20 and stride of 2. The mask will be pass through the trained model to get the prediction label. Whenever the predicted label matches the label of the image then it will plot a square box on the image. Once the mask has already scanned every part of the image, the system will do the same way to get the exact coordinates of the transport.*

*By getting the label of every mask, the model is also able to get the location of multiple objects in the picture.*

***Results:***

