# RBE 577: Homework 2 Report

Azzam Shaikh
Oct. 14, 2024

## Introduction

The purpose of this assignment is to finetune a pretrained ResNet model to classify images of road vehicles. This network would provide autonomous vehicles the ability to classify surrounding vehicles using on-board vision systems.

The dataset used for this project can be found via Kaggle (link). The dataset contains 1800 images of various vehicles and is split into a train dataset with 1400 images, a validation dataset with 200 images, and a test dataset with 200 images.

## Methods

The assignment implementation will cover three areas: (1) data loading and augmentation, (2) model development and hyperparameters, and (3) the training and testing scheme. Each of these sections and their implementation will be discussed in this section

(1) Data Loading and Augmentation

In order to load the dataset into PyTorch, a custom dataset class was developed, VehicleClassificationDataset, which inherits from the torch.utils.data.Dataset class. The class loops through the dataset directory and generates a list of all the samples in the dataset with their defined image path, the classification (for train and validation only), and which directory they are located in. The __getitem__ function gets overridden to obtain to obtain the image from the sample list and apply a transform to the image. The specific transforms applied to the images are the following:

- A random horizontal flip
- A random vertical flip
- A random grayscale
- A random rotation up to 45 degrees

These random transformations are an implementation of data augmentation. This ensures that the images loaded to the model vary and help the model predict any object in any scene. After this augmentation, additional transformations include the resizing and cropping of the images to 244x244 pixels, and the scaling and normalization of images, to adhere to the processes used in the original training of the pretrained weights.

Once an object of the class is created, all the images in the dataset can be extracted into separate training, validation, and testing datasets. From these individual datasets, a DataLoader object of these datasets can be created. These objects are iterable objects that will be used in the training loop. All the dataloaders consist of a batch size of 32. Additionally, the training dataloader is shuffled.

Figure 1 below shows a batch of the training data.

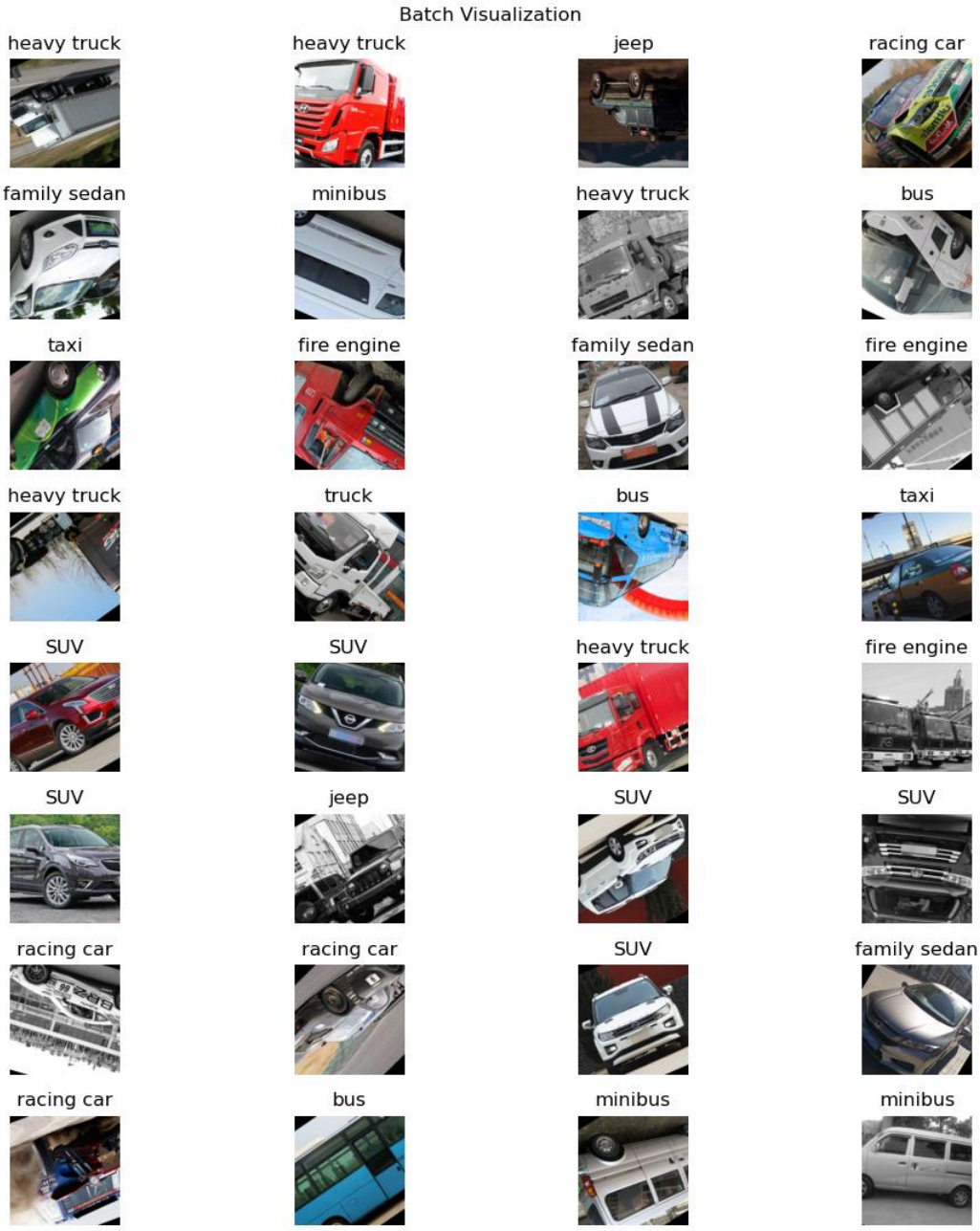



Figure 1: Batch of training data

(2) Model Development and Hyperparameters

For the development of the model, a pretrained ResNet50 was utilized for this assignment. In the default architecture, the model contained 1000 classifiers. For our case, since there are only 10 classes, the classification of the layer of the model was replaced with a new Linear layer with 10 outputs.

For this assignment, in order to finetune the model, three implementations will be compared, which are described as follows:

1. Freezing all the layers and training only the head
2. Freezing all the layers except layer 4 and the head
3. Train the entire model

The various implementations will enable a proper comparison of which approach results in the best classification capabilities.

For each model, cross entropy will be utilized for the loss function and an Adam optimizer will be utilized with a learning rate of 0.001 and a weight decay of 0.01. Furthermore, a learning rate scheduler will be utilized to decrease the learning rate by a factor of 10 every 3 epochs where the loss plateaus. This ensures that the model will be further fine tuned during every plateau. Additionally, early stopping was implemented to ensure that, once the model was trained sufficiently, if the validation loss does not improve after 5 epochs, the training will stop, and the best model will be reloaded. The training will occur with 50 epochs; however, not all 50 epochs will be utilized depending on the early stopping function.

(3) Training and Testing Scheme

For each of the three models, the training loop is implemented as follows. First, the model, along with the optimizer, loss function, schedulers, and data loaders, will get passed to the training function. In the training function, for the defined number of epochs, a train step will occur where the entire train data is passed through the model for training, and then a validation step will occur where the entire validation data is passed through the model for evaluation. Once complete, the early stopping function checks whether the training needs to stop or not. If the training continues, the scheduler decides whether to lower the learning rate or not. The next epoch and training cycle will then repeat itself. During the training loop, the training and validation loss and accuracy will be computed and logged via Tensorboard.

After the training is complete, the best model will be reloaded and the model will be used to make predictions on the entire test dataset. The predictions of the first 10 images of the

dataset will be visualized and evaluated. Since there is no ground truth data, the predictions will be manually evaluated.

Once complete, the model will be saved.

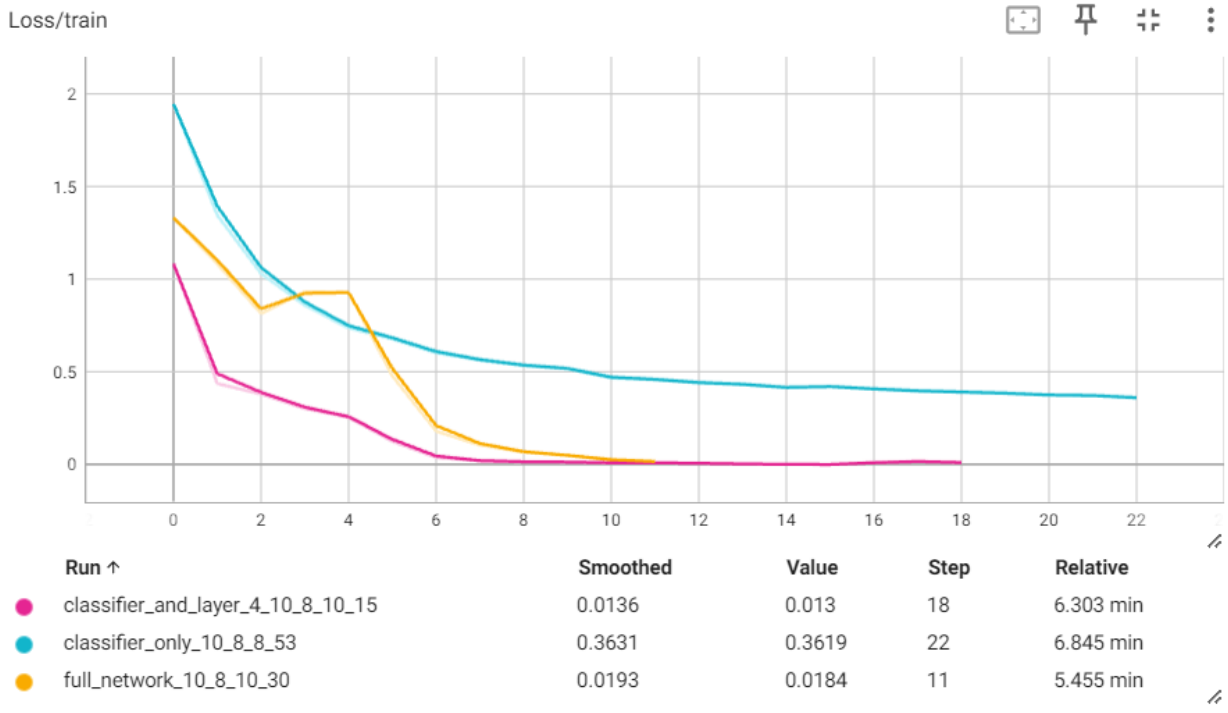
## Results and Discussion

After developing the three models and the overall training and testing schemes, the models could be tested and compared. As mentioned, the model is ran with 50 epochs but will stop, if needed, as decided by the early stopping.

The following plots depicted the results of the training and validation datasets by the three models. The three models and their line colors on the plots are:

1. Freezing all the layers and training only the head - Yellow
2. Freezing all the layers except layer 4 and the head - Purple
3. Train the entire model - Orange

Figure 2 and 3 visualizes the loss plots of the three models during training and validation, respectively.




| Run | Smoothed | Value | Step | Relative |
|---|---|---|---|---|
| classifier_and_layer_4_10_8_10_15 | 0.0136 | 0.013 | 18 | 6.303 min |
| classifier_only_10_8_8_53 | 0.3631 | 0.3619 | 22 | 6.845 min |
| full_network_10_8_10_30 | 0.0193 | 0.0184 | 11 | 5.455 min |

Figure 2: Model loss plots during training

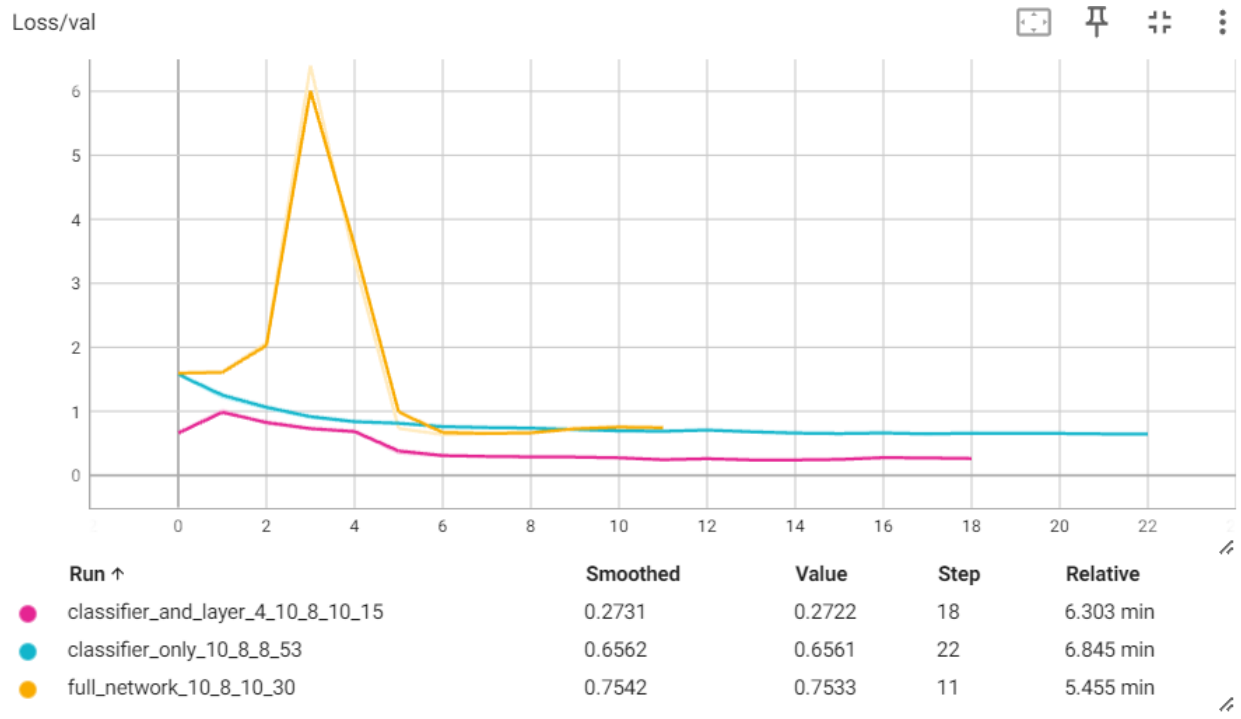



Figure 3: Model loss plots during validation

Figure 4 and 5 visualizes the accuracy plots of the three models during training and validation, respectively.

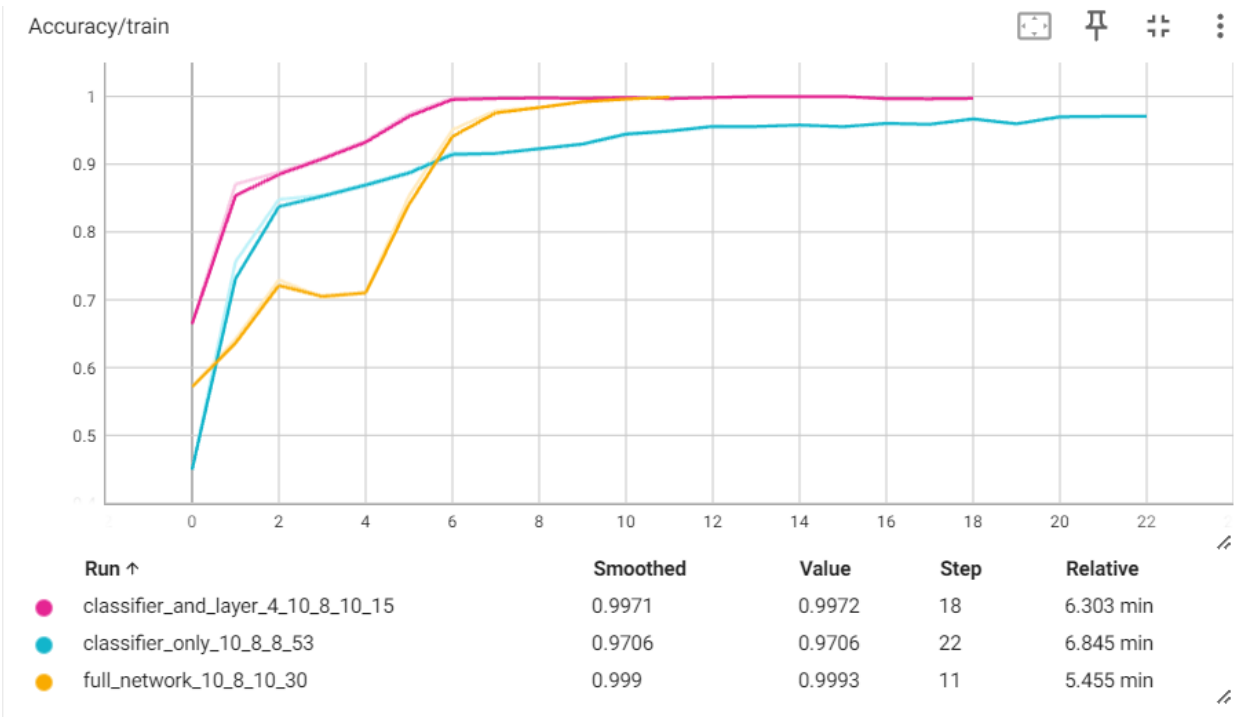



Figure 4: Model accuracy plots during training

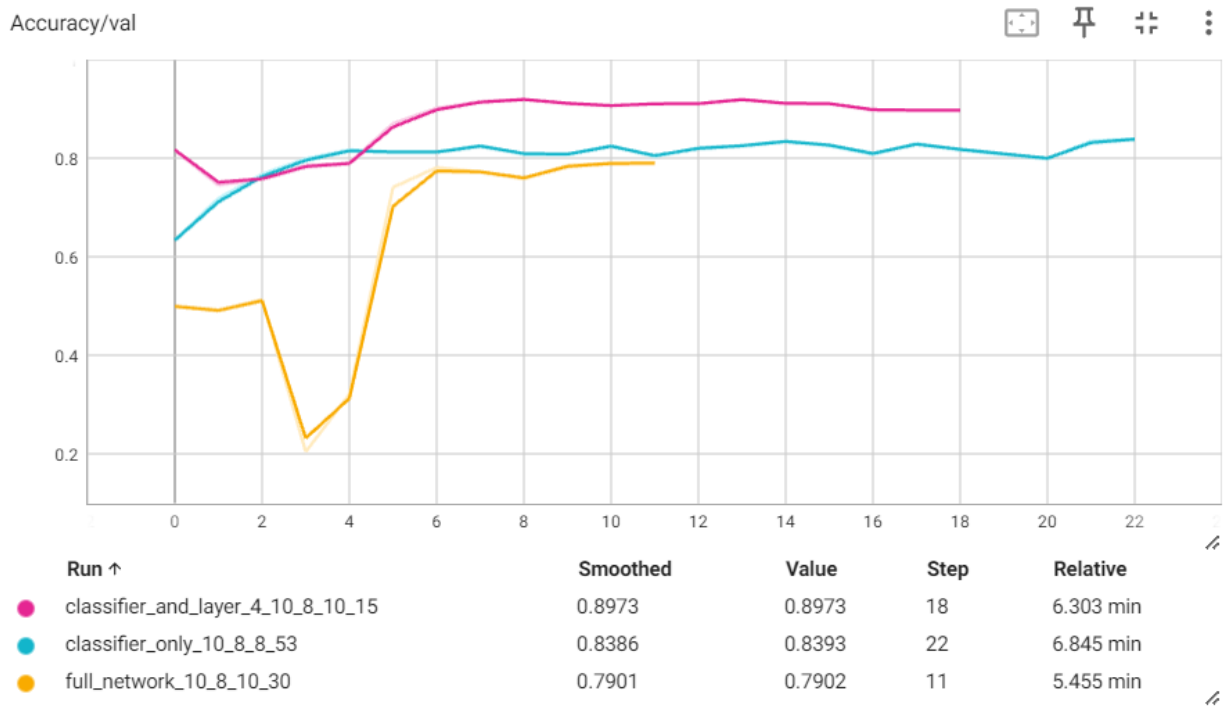



Figure 5: Model accuracy plots during validation

Based on the Figures 2-5, it can be seen that the model that trained layer 4 and the classifier (model 2) had the highest validation accuracy. The model that trained only the classifier (model 1) was the second most accurate, while the model that trained the whole network (model 3) was last. Interestingly, it can be seen that training the classifier took the most epochs before the early stopping function was initiated. Additionally, it can be noted that training of model 3, aka finetuning the whole network, lead to really high training accuracy but low validation accuracy; hinting that the model was being overtrained.

The huge spike that is seen by both model 3 is most likely due to a learning rate change applied by the scheduler.

Figure 4 visualizes the predictions for test dataset using the model that trains only the classifier.

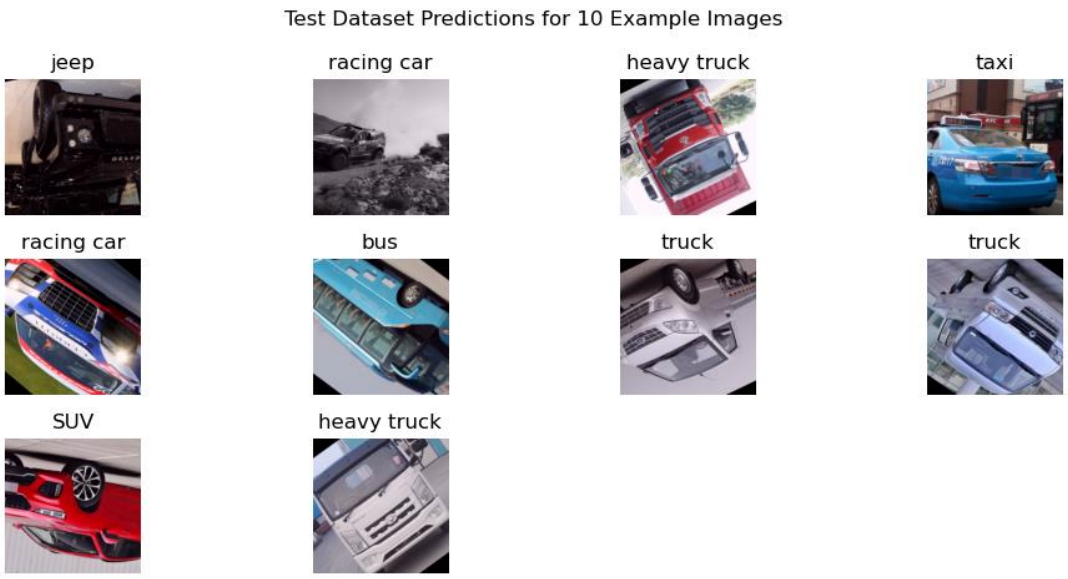



Figure 4: Predictions from model 1

Based on Figure 4, it can be seen that there is one incorrect classification:

1. Row 3, Column 2 – Should be a truck

Figure 5 visualizes the predictions for test dataset using the model that trains only layer 4 and the classifier.


Figure 5: Predictions from model 2

Based on Figure 5, it can be seen that there is one incorrect classification:

1. Row 1, Column 4 – Should be a taxi

Figure 6 visualizes the predictions for test dataset using the model that trains the whole network.


Figure 6: Predictions from model 3

Based on Figure 6, it can be seen that there is one incorrect classification:

1. Row 2, Column 2 – Should be a bus

In all test cases, there was at least one incorrect prediction. This is not an entirely accurate method of evaluating the predictions as the sample size is extremely small. Since there are no labels present for the test dataset, it is not a feasible evaluation. Thus, the usage of the validation accuracies is appropriate for selecting the best model.

Overall, the best approach toward finetuning is to not only train the classifier, but also a portion of the network. Training the whole network leads to overtraining, as described above. By training only a portion of the network, the original network is preserved while allowing some of the model to generalize the finer details of the vehicle dataset.