CAB320 Transfer Learning Assignment Report

Group members:

- Alexander Kim n10794565
- Wei-Chung Lin n10664599

Introduction

This report contains description of investigation for a flower classifier using transfer learning on a neural network, MobileNetV2, trained on the ImageNet dataset. From the investigative process of developing a flower classifier, different parameters of the SGD optimizer such as learning rate and non-zero momentum were experimented and its effects on the model is presented. The findings from the investigation and development were then used to suggest recommendations to improve the model for future reference.

Description of Investigation

To build a flower classifier using transfer learning method of machine learning, a pre trained Convolutional Network (ConvNet) that has been initially trained on a much bigger dataset such as ImageNet can be used as a starting point. The model can then be trained again on a second smaller dataset called the target dataset to achieve the desired computer vision task. In this case, a neural network model, MobileNetV2 developed by Google is chosen for its lightweight nature allowing reliable image classification under computing constraints such as mobile devices.

To apply transfer learning to the chosen neural network model, we first obtain target dataset which will be trained on top of the pretrained model to recognise our desired custom use case which is to classify flowers. The target data of flowers have been conveniently provided where each of the five flower types are organised under its respectively named folder which allowed the dataset to be split as train and validation datasets easily. An image batch of 32 was then taken out of the validation set to use as test sets to test the model's ability to classify unseen images. The overall train, test, validation dataset split is 80%: 16.8%: 3.2%.

Next, MobileNetV2 base model can be downloaded and instantiated from TensorFlow where we have configured some parameters to not include the fully connected layer at the top of the network, set weights to 'imagenet', and set our custom input size of the images while the rest of the parameters were left to defaults. We then added input and output layers to the base model where the input layer rescales images to a consistent size while the output Dense layer classified the predictions into the five different classes. Finally, the model was compiled and trained using SGD optimizer with these initial parameters:

learning_rate = 0.01, momentum = 0.0, nesterov = False

Epochs of 20 were chosen for training as it achieved a good balance between time and performance given the size of the training dataset. Plotting the training and validation error and accuracies results in the following plot:

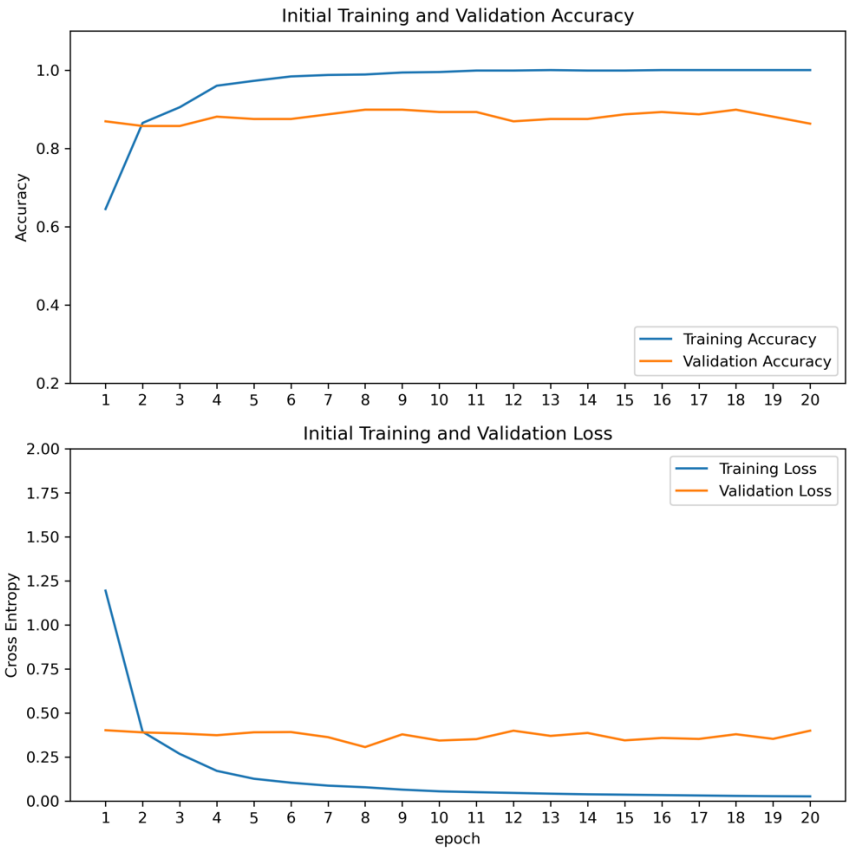

Figure 1: Initial training with the default learning rate and momentum

From the graph above in figure 1, the training accuracy quickly reached near 100% and plateaued from around 10 epochs onwards whereas the validation accuracy shows it plateaued since the beginning which indicates that the model has not learned from its transfer learning when it has been presented with unseen validation data. While training loss shows a quick and continuous decline to a low loss throughout, the validation loss has plateaued since beginning showing signs of an underfitting model indicating the model has not learned. It is apparent that the learning rate of 0.1 and momentum of 0 is not ideal. To determine the ideal values for learning and momentum, the different magnitudes of learning rates of 0.1, 0.01 and 0.001 was investigated first. The following three figures plots the effects of different learning rate magnitudes with momentum of 0.

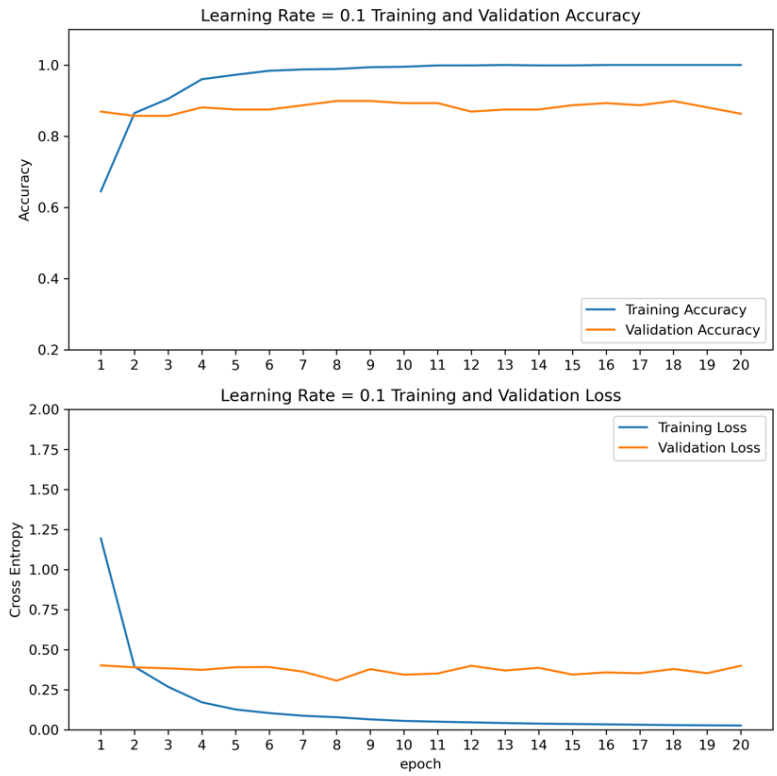

Figure 2: Training and validation loss and accuracy vs time with Learning rate = 0.1 and momentum = 0 (Same as figure 1)

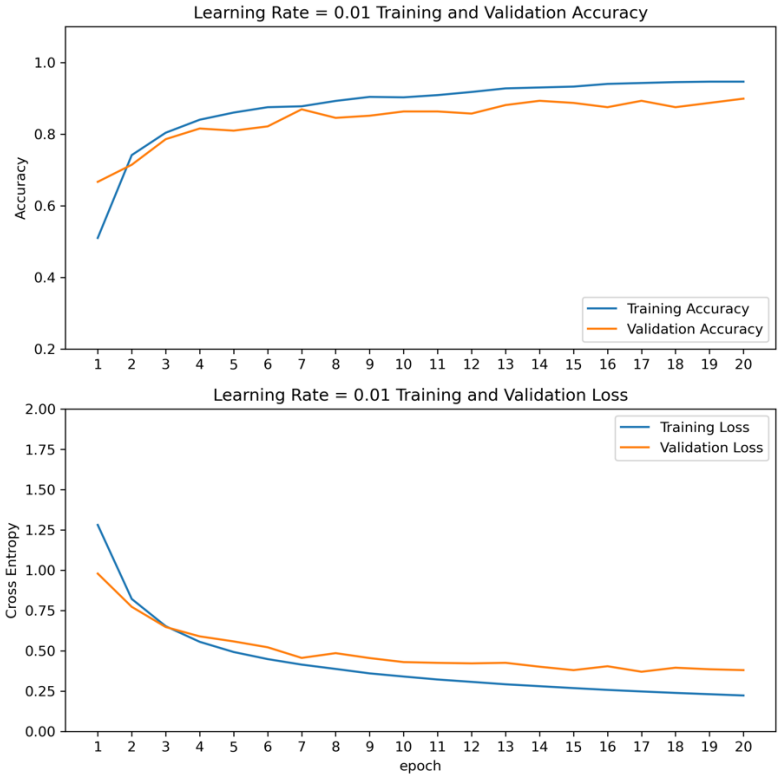

Figure 3: Training and validation loss and accuracy vs time with Learning rate = 0.01 and momentum = 0

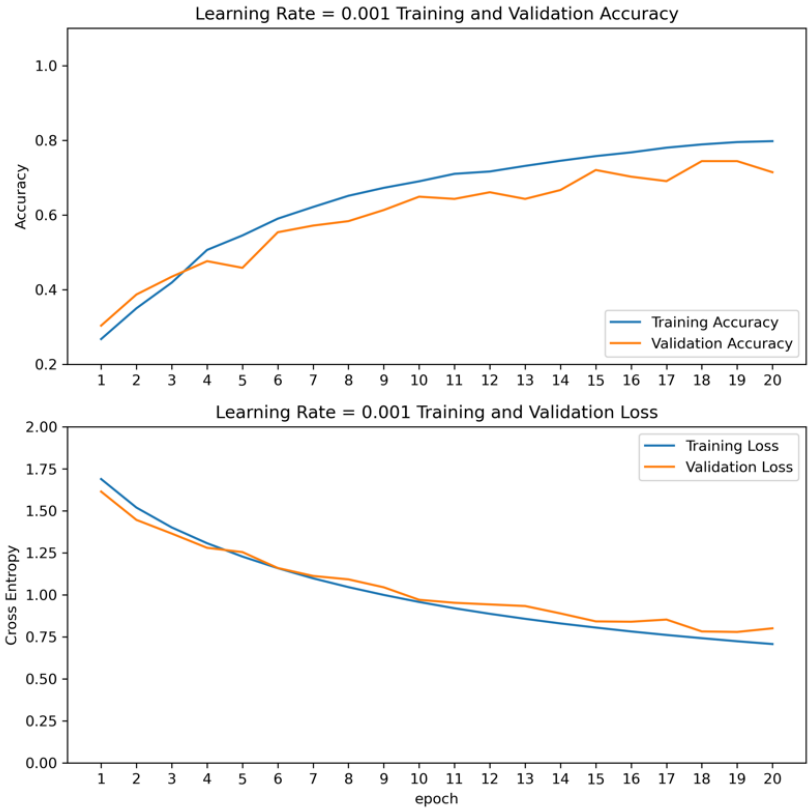

Figure 4: Training and validation loss and accuracy vs time with Learning rate = 0.001 and momentum = 0

When comparing the three magnitudes of the learning rates which were 0.1, 0.01 and 0.001, the learning rate of 0.1 has already been ruled out previously as the parameters are the same with the given default and hence the two figures, 1 and 2 are the same. The learning rate of 0.001 in figure 4 on the other hand illustrates a well-fitting loss model where the difference between training and validation loss are very minimal. However, the actual loss values are much higher where it seems that it is continuing to learn and improve its losses. The accuracy plot of both training and validation are also a good fit. However, the value of accuracy is much lower than desired although it indicates that it is still learning. It is apparent that the slowest learning rate of 0.001 needs much more epochs which can be a constraint.

This leaves the learning rate of 0.01 which is presented in figure 3. This learning rate shows a good fit of the loss plot between both training and validation data with a much lower loss value at the end of its time at 0.025 compared to the learning rate of 0.001. The accuracy plot also shows a good fit where the validation accuracy continues to climb higher than the previous, showing that the model has learned and that it is predicting relatively well compared to the learning rate of 0.1 from figure 1. With the validation accuracy achieving around 85% and loss of around 0.025 at the end of its time, learning rate of 0.01 was determined to be ideal. With the ideal learning rate determined the different values of momentum of 0.25, 0.5 and 0.75 was then explored. The following three figures plots the effects of different momentums but with learning rate of 0.01.

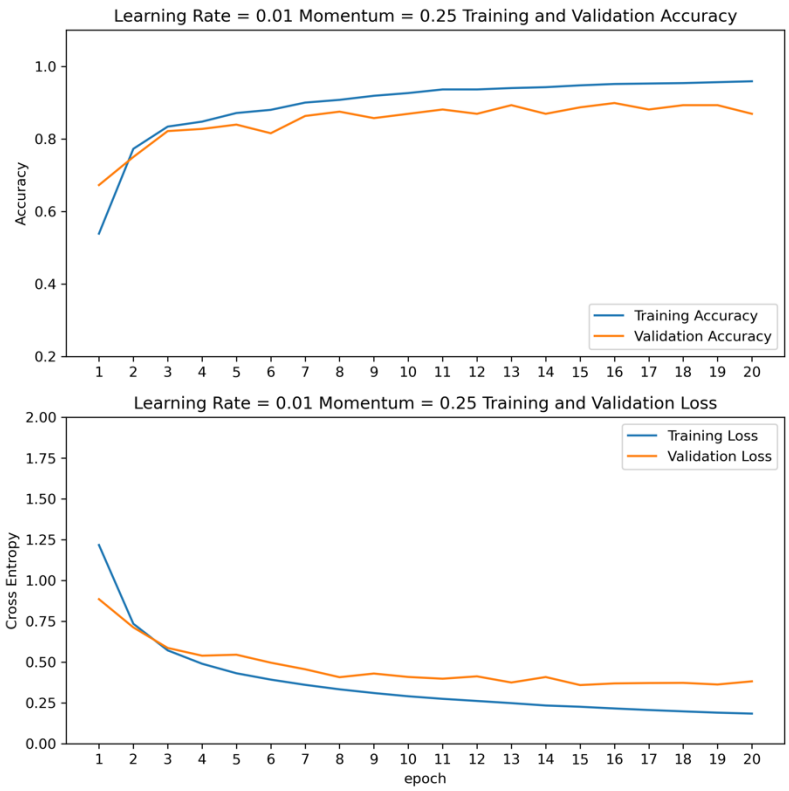

Figure 5: Training and validation loss and accuracy vs time with Learning rate = 0.01 and momentum = 0.25

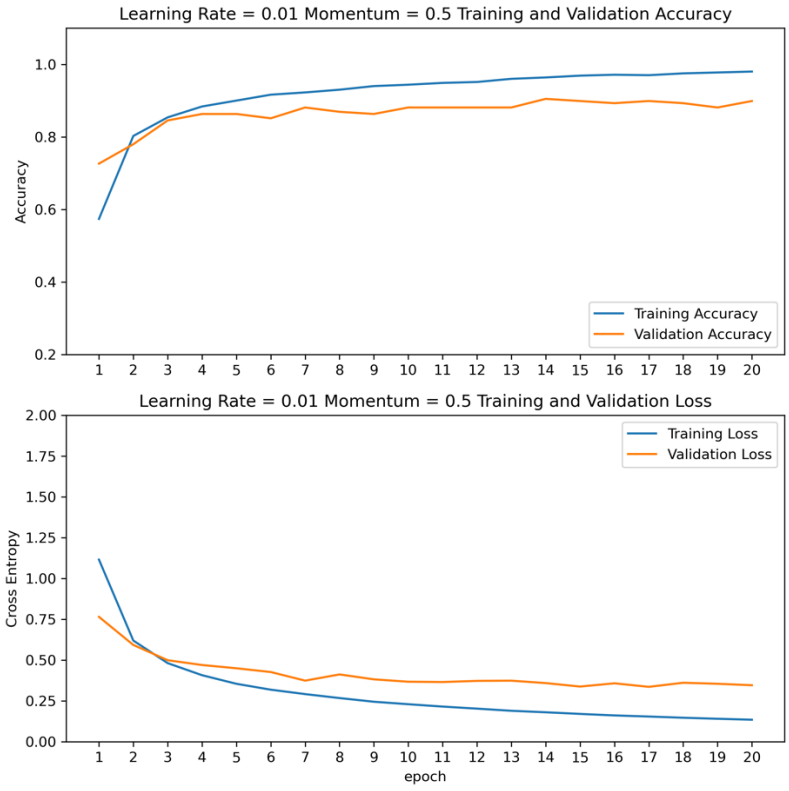

Figure 6: Training and validation loss and accuracy vs time with Learning rate = 0.01 and momentum = 0.5

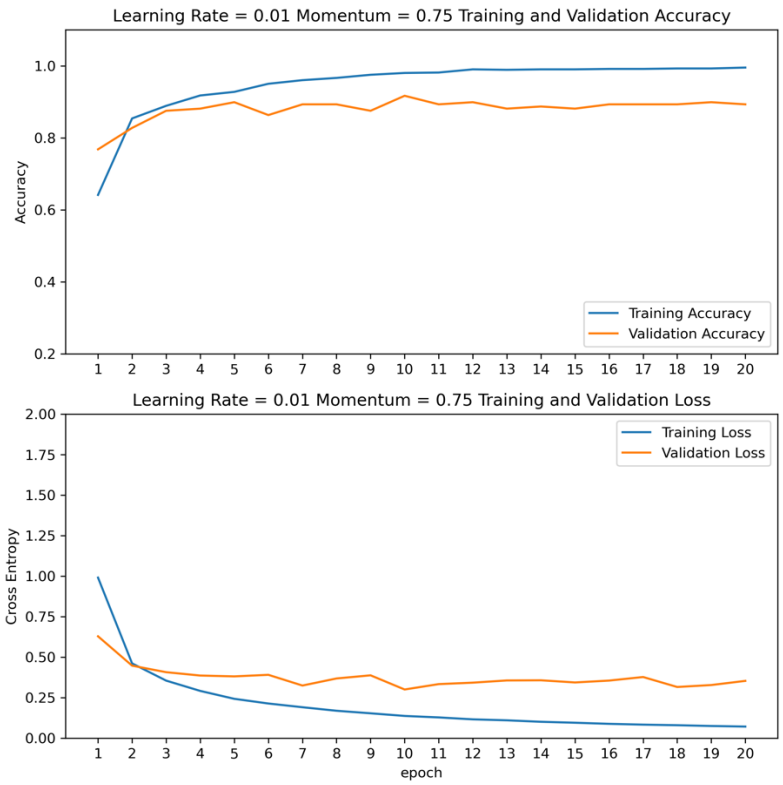

Figure 7: Training and validation loss and accuracy vs time with Learning rate = 0.01 and momentum = 0.75

Observing the experimental results above, it is evident that in all cases where momentum was used in the SGD Optimiser, the rate in which the validation loss converges toward the training loss was faster. This effect was more apparent as the value of the momentum increased. This observation aligns with the theory of momentum, which is to help accelerate the gradient vectors to overcome local minima. For the cases where momentum was 0.25 and 0.50, the model was best fitted at between 4 to 7 epochs. Whereas for the case where the momentum was at the highest of 0.75, the model was only best fitted between 3 to 4 epochs. The usage of momentum accelerates the learning process, yet excessive amount of momentum may cause the global minima to be missed. This was apparent in the highest momentum case of 0.75, where the validation loss seemed to oscillate around 0.50 after 4 epochs, which signalled that the model was unable to learn further and arguably showed signs of overfitting towards 20 epochs. Comparing the cases of 0.25 and 0.50, the difference between these two values of magnitude was negligible. It was noticed that the higher the momentum used, the accuracy of the model on the validation dataset appeared to be more stable, showing faster stabilisation with within a smaller number of epochs. However, it is arguable that only a small value of momentum is required to achieve a compromise between the speed of convergence and the guarantee of optimal global minimum.

It is recommended for the momentum value to be the lowest experimented value of 0.25, to help the SGD optimiser to overcome local minima, as well as to hold the integrity of best fitting the model.

With the most ideal learning rate to be determined at 0.01 and the momentum at 0.25 for the optimiser, the following predictions were made by the model on one batch of test data.

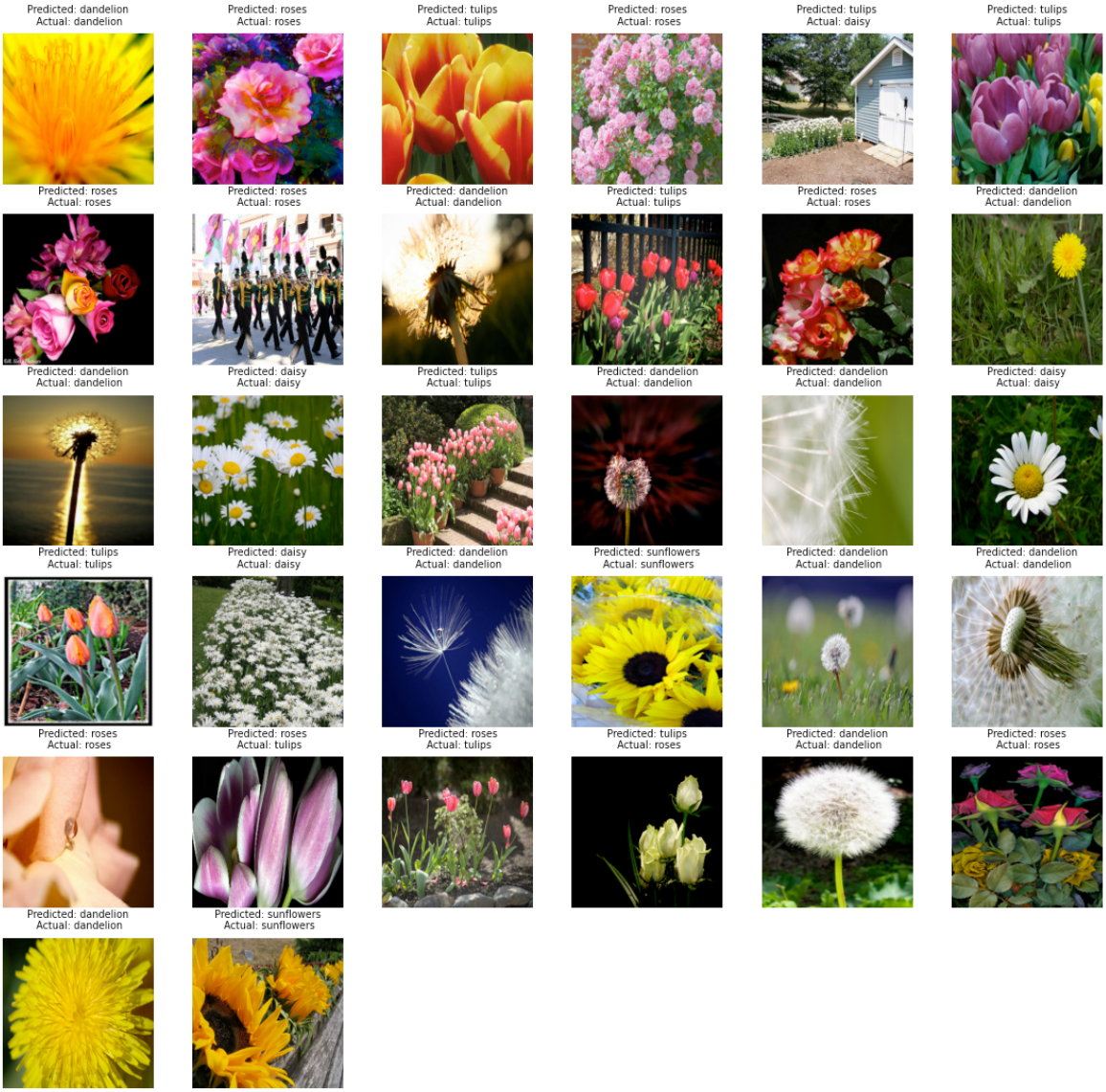

Figure 8: Model predictions on test data

From this set of 32 unseen images, the model was able to get 28 of 32 total images which is 87.5% accuracy.

Recommendation

As discussed in detail in the previous section, the most ideal parameters found for the learning rate and momentum for the SGD optimiser was deemed to be 0.01 and 0.25 respectively. Through experimenting with different magnitudes of the learning rate, learning rate of 0.1 showed ineffective training of the model while the learning rate of 0.001 was indeed a good model but it needed much more epochs to be trained effectively which is a limiting constraint. It was also discovered that a small momentum is advantageous in helping the SGD optimiser in finding the optimal minima, while an increased momentum may cause overshooting and not be ideal. At an epoch of 20, learning rate of 0.01 and

momentum of 0.25 is recommended for best results as evidenced by the accuracy of predictions presented in figure 8.

In addition to the optimised parameters for the SGD optimiser, the size of the training dataset should be drastically increased to lower the estimation variance of the model which would result in a much better predictive performance. Furthermore, fine tuning the model after the initial training would improve the fit of model allowing for more reliable predictions as well.