ECS6P9U/P
Neural Networks & Deep Learning Project Report

Task Introduction

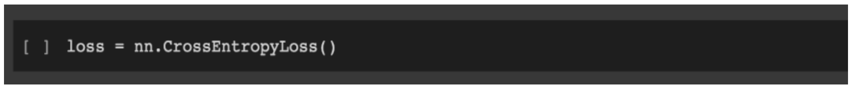
This project details the coursework for ECS6P9U/P: Neural Networks & Deep Learning 2021/22. The problem involves constructing a neural network that can correctly classify 10 different categories of clothing from the Fashion-MNIST classification dataset, using the PyTorch python library. This report will discuss tasks 4 and 5 of the project, which detail the Model's training function, the curves for the loss, training and test accuracies, and the overall test accuracy of the final model produced. The accompanying Jupyter notebook submitted with this report provides a more thorough description of the code for the produced model.

Model Training Methods & Hyperparameters

To train the model, several hyperparameters were used to define how the model should be trained and optimised. Some of the tests used to determine which hyperparameters yield the best model accuracy are detailed below.

Loss Function

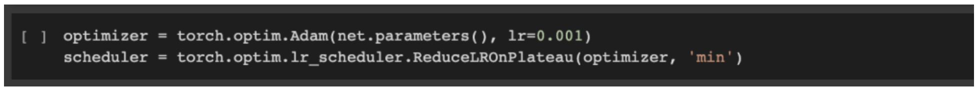
As our model uses a SoftMax regression classifier to produce the final model classifications, the loss function used in the training of our model is the Cross Entropy Loss function. The code used to implement this is shown below.



```
[ ]  loss = nn.CrossEntropyLoss()
```

Learning Rate Scheduler

A learning rate scheduler was used to vary the learning rate over time for the model. The learning rate scheduler chosen was PyTorch's "RedcuceLROnPlateau" scheduler [1], which reduces the learning rate by a set factor when there are no significant changes detected in the test accuracy of the model after an epoch cycle. This results in our chosen optimiser using a smaller and smaller learning rate, the closer the model gets to its test accuracy convergence. When compared to a model not using a scheduler, an accuracy increase of $\sim 0.5 - 1\%$ can be observed. In all cases, an initial learning rate of 0.001 was chosen for the optimisers used. Although the optimisers evaluated throughout our model's creation adapt their own learning rate during training (i.e., Adam, RAdam) they are limited to a minimum learning rate, which is defined in the optimiser's definition as the global learning rate. This global learning rate is the learning rate that the scheduler will alter over time, and its initial value is shown below in the definition of the optimiser and scheduler within the code.



```
[ ]  optimizer = torch.optim.Adam(net.parameters(), lr=0.001)
     scheduler = torch.optim.lr_scheduler.ReduceLROnPlateau(optimizer, 'min')
```

Varying backbone size (adding more blocks)

The backbone was specified in the project outline to contain N blocks, that each consist of 2 MLPs, with transpose steps occurring before both MLPs. This allows for the easy implementation of additional blocks, by simply copying the structure of the initial block, and adding the necessary layer inputs and outputs for the additional linear layers introduced by each block.

For the tests conducted below, the optimiser used was the Adam optimiser (with learning rate = 0.001), and the Cross Entropy loss function was used for each. 60 Epochs were used to test the different outcomes of each case, and the resultant training accuracy, testing accuracy and loss values are shown in the table below.

Number of Blocks	Epochs	Training Accuracy	Test Accuracy	Loss
1	60	0.904	0.884	0.158
2	60	0.901	0.853	0.231
3	60	0.892	0.814	0.345

Note that despite additional blocks having a significant impact on the training accuracy, no significant changes in the test accuracy were observed, which could suggest that the addition of more than one block to the neural network may be encouraging the model to overfit the training data.

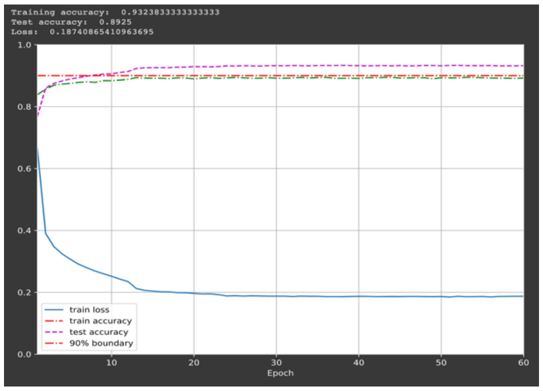
Testing Different Optimisers

Adaptive Moment Estimation (Adam) and the Rectified Adaptive Moment Estimation (RAdam) were the two main optimisers used to test the model. Stochastic Gradient Descent (SGD) was also tested in early stages but failed to produce a stable model. Briefly noting the differences in the candidate optimisers:

- Adam is an optimization algorithm that converges much faster than SGD by altering the learning rate up to the globally defined learning rate in our code, allowing us to achieve a higher accuracy in a shorter amount of time [2].
- RAdam is a variation of the Adam optimiser that attempts to correct learning rate variance early on in model training, up to the globally defined learning rate [3].

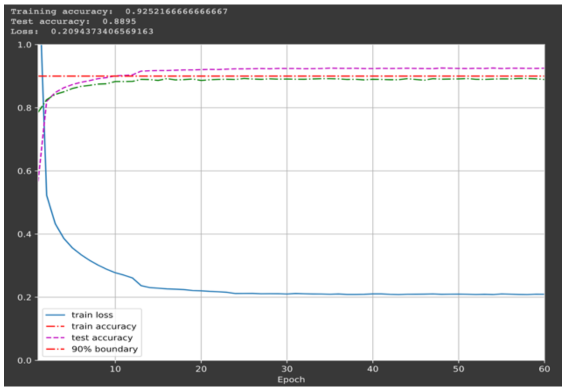
While testing the model, one block was used in the model for all tests, and the only hyperparameters changed between each test were the optimiser used, and the learning rate. The number of epochs were constrained to 60 in order to obtain results in a reasonable amount of time.

Optimiser	Epochs	Training Accuracy	Test Accuracy	Loss
Adam	60	0.932	0.892	0.18
Rectified Adam	60	0.925	0.889	0.209



```
Training accuracy:  0.9323833333333333
Test accuracy:  0.8925
Loss:  0.18740865410963695
```

1. Adam Train/Test Accuracy and Loss Curves



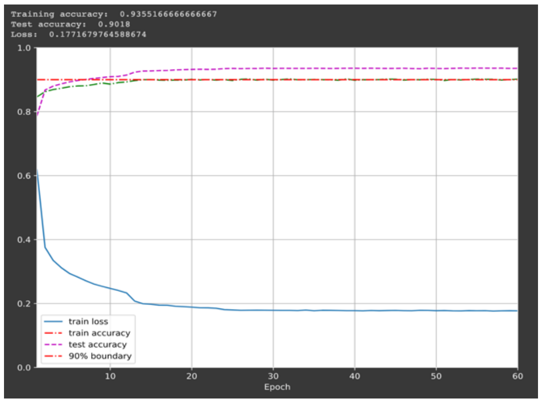
```
Training accuracy:  0.9252166666666667
Test accuracy:  0.8895
Loss:  0.2094373406569163
```

2. Rectified Adam Train/Test Accuracy and Loss Curves

TESTING DIFFERENT METHODS FOR MODEL WEIGHTS INITIALISATION

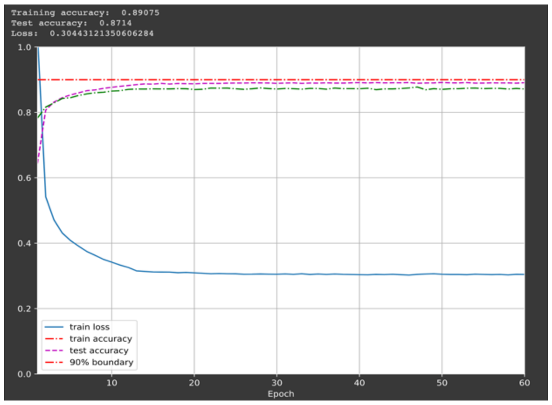
The final main test conducted on the model was the initialisation method for the model weights. In these tests, several initial weight initialisation schemes were used. These include Xavier Uniform Initialisation, Normal Distribution Initialisation and Kaiming Uniform Initialisation. Each of these initialisations were tested on the model, where a single block was used. Each test also used an Adam optimiser, selected for its good performance in previous tests. From these tests, the Xavier Uniform initialiser outperformed both the Normal distribution initialiser, as well as the Kaiming Uniform initialiser. Kaiming initialisation followed close behind but did not manage to break the 90% accuracy score for the test data. The Normal distribution initialiser was the worst performing initialiser by far, producing an accuracy of < 88%, and a much higher final loss than both the Xavier and Kaiming initialisers. The results are presented below.

Weights Initialiser	Epochs	Training Accuracy	Test Accuracy	Loss
Xavier Uniform	60	0.936	0.902	0.177
Normal	60	0.891	0.871	0.304
Kaiming Uniform	60	0.930	0.892	0.19



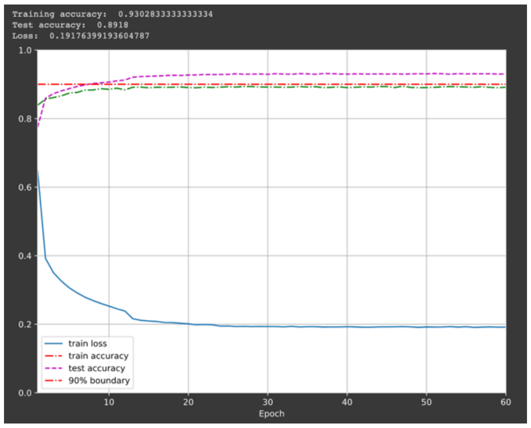
```
Training accuracy:  0.9355166666666667
Test accuracy:  0.9018
Loss:  0.1771679764588674
```

3. Xavier Uniform Train/Test Accuracy and Loss Curves



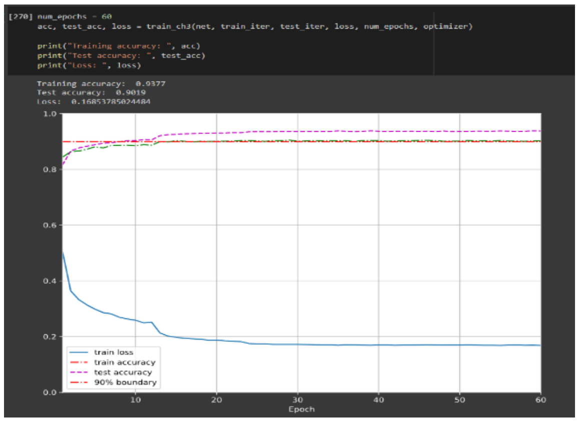
```
Training accuracy:  0.89075
Test accuracy:  0.8714
Loss:  0.30443121350606284
```

4. Normal Train/Test Accuracy and Loss Curves



```
Training accuracy:  0.9302833333333334
Test accuracy:  0.8918
Loss:  0.19176399193604787
```

5. Kaiming Uniform Train/Test Accuracy and Loss Curves



```
[270] num_epochs = 60
      acc, test_acc, loss = train_ch3(net, train_iter, test_iter, loss, num_epochs, optimizer)

      print("Training accuracy: ", acc)
      print("Test accuracy: ", test_acc)
      print("Loss: ", loss)

      Training accuracy:  0.9377
      Test accuracy:  0.9019
      Loss:  0.16853785024484
```

6. Final Model Train/Test Accuracy and Loss Curves.

Final Model Accuracy and Results

After deciding upon the most appropriate Optimiser, Initialiser and Block size, the most accurate model that could be constructed within the time given was trained using the following hyperparameters and training methods:

- A single backbone block.
- An Adam Optimiser.
- A Learning Rate Optimization Scheduler.
- Xavier Uniform Weights initialisation.

This yielded a final model test accuracy of 0.902, a training accuracy of 0.938, and a loss of 0.16. The training accuracy, test accuracy and loss curves can all be seen above for this model, in figure 6.

BIBLIOGRAPHY

[1] T. Contributors, “REDUCE LR ON PLATEAU,” PyTorch, 2019. [Online]. Available: https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.ReduceLROnPlateau.html. [Accessed 2022 March 19].

[2] N. Chan, “ADAM in 2019 — What’s the next ADAM optimizer,” 2019. [Online]. Available: https://towardsdatascience.com/adam-in-2019-whats-the-next-adam-optimizer-e9b4a924b34f#:~:text=Adam%20is%20great%2C%20it's%20much,2019%20were%20still%20using%20SGD.. [Accessed 18 March 2022].

[3] G. Tanner, “RAdam - Rectified Adam,” 2021. [Online]. Available: https://ml-explained.com/blog/radam-explained. [Accessed 2022 March 28].