

Neural Networks and Deep Learning Assignment 3

Coursework Report

Name: Kweku E. Acquaye

This report summarises the accompanying python Jupyter notebook in which a neural network (NN) model was built, trained, evaluated, and plotted using modern data science methods.

Dataset

The dataset for this coursework is the Fashion-MNIST dataset (Xiao et al, 2017), a 10-class dataset consisting of 60,000 greyscale clothing item images for training, and a further 10,000 images for evaluation/testing.

The Model

Stem

The model stem consists of a single linear layer taking as input vectorised patches of 28 x 28 pixel images.

Backbone

The NN comprised a backbone of three 2-unit blocks of fully-connected (FC) hidden multilayer perceptrons (MLPs). Each layer is followed by an activation function that performs a non-linearity operation on the matrices before passing them on to the next layer.

Classifier

The model classifier implements Softmax regression as a single 10-neuron FC linear layer.

Training Hyperparameters

1. **Optimisation:** Subsequent to building the model, the following weighted sum non-linear activation functions were trialled - Stochastic Gradient Descent (SGD), Rectified Linear Unit (ReLU), Leaky ReLU (LeakyReLU), and adaptive moment estimation (Adam) (Kingma and Ba, 2014). Adam was found to deliver the best results for this model.
2. **Loss:** For quantifying the difference between model output and expected output during training, Cross Entropy Loss (CrossEntropyLoss) was used throughout trials.
3. **Weight decay:** Values between 0.05 and 0.00005 were experimented with, and the optimum found to be 0.0001.
4. **Learning rate:** Values between 0.5 and 0.0005 were experimented with, and the optimum found to be 0.001.

5. **Batch size:** Batch sizes of 100, 128, 200, 256, 384, 500, and 512 were trialled before settling on 256 as optimum batch size.
6. **Epochs:** Trials between 15 and 45 at intervals of 5 epochs were observed with respect to the output plots as well as the performance metrics, and it was found that model performance did not improve after 30 epochs.
7. **Hidden layers:** Several trial and error runs were performed with changing the number of hidden layer blocks as well as the number of neurons/perceptrons in each layer. The architecture reported in the notebook, and schematically represented below, was found to provide the best accuracy scores.

Figure 1: Schematic Representation of Model Architecture

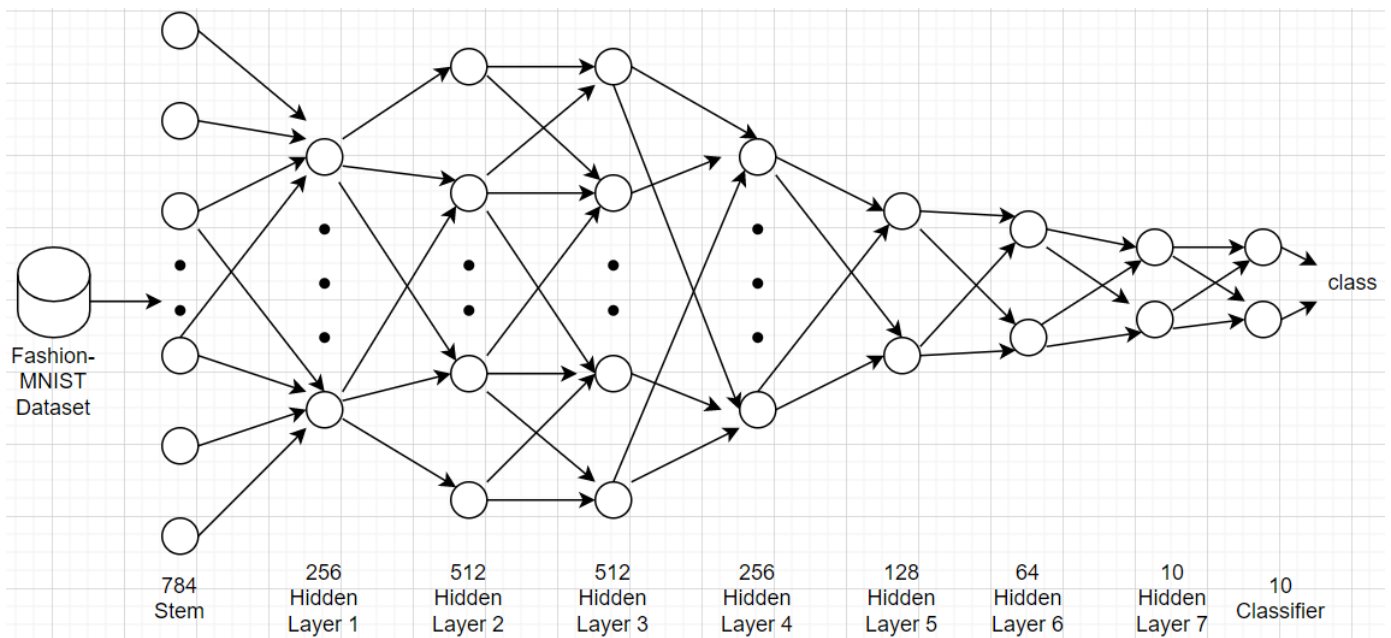


Figure 2: Screenshot of Model training, testing and loss Plot

✓ 29m [15] # Implementing model training and testing
num_epochs = 30
train_ch3(net, train_iter, test_iter, loss, num_epochs, optimizer)

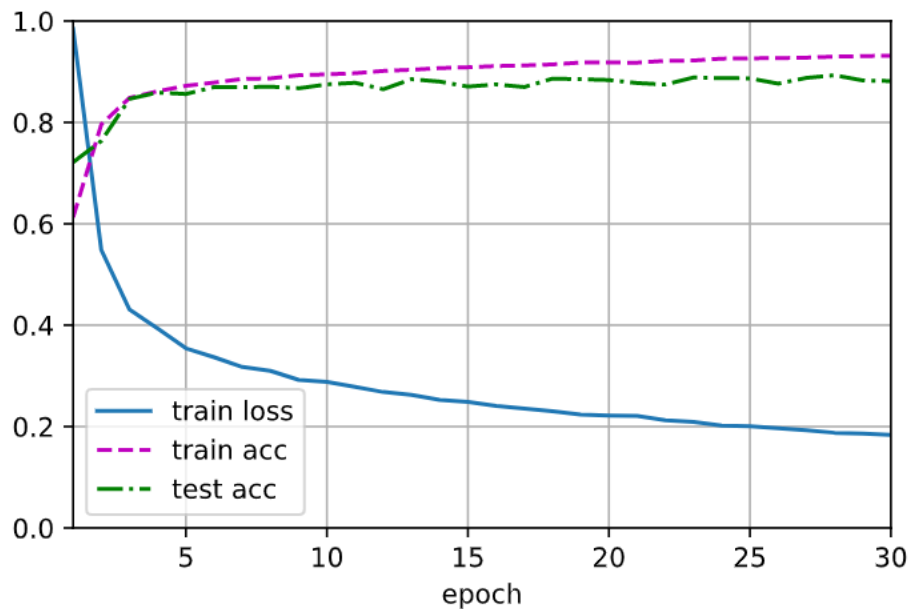



Figure 3: Screenshot of Model Performance Metrics

✓ 37s  # Outputting training accuracy
training_acc = evaluate_accuracy(net, train_iter)
print("Model training accuracy is", training_acc)

Model training accuracy is 0.9298333333333333

✓ 6s [17] # Outputting test accuracy
testing_acc = evaluate_accuracy(net, test_iter)
print("Model testing accuracy is", testing_acc)

Model testing accuracy is 0.8816

References

1. Kingma, D.P. and Ba, J. (2014), *Adam: A method for stochastic optimization*, arXiv preprint arXiv:1412.6980. [online]: <https://arxiv.org/abs/1412.6980>
2. Zhang, A., Lipton, Z.C., Li, M. and Smola, A.J., 2021. Dive into deep learning. arXiv preprint arXiv:2106.11342. [online]: <http://d2l.ai/>