UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Homework 1: Supervised Deep Learning

Neural Networks and Deep Learning

Chelsea Owen
Student ID: 2006036
September 10, 2022

In this homework the task was to solve two supervised tasks using trained Neural Networks. The first was a regression task of approximating a function and the second was a classification problem with images from the Fashion-MNIST data set. In order to find the best model in both cases a random search and cross validation was used. The PyTorch framework is used throughout.

1 Introduction

The two tasks of this homework were implementing Supervised Learning. This method is defined by using the labels of input data to train algorithms, which learn by minimising the difference between the predicted value and the true value.

The regression task was to approximate a function, given some input $x \rightarrow y = f(x)$ so that the output of the network then is such that $network(x) \approx f(x)$. The network is a fully connected feed-forward network with two hidden layers.

The second task is a multi-class classification problem with mutually exclusive classes, where there are ten possible labels of the Fashion-MNIST clothing images in the form 0,1,2,3,4,5,6,7,8,9. The network in this case is a Convolutional Neural Network (CNN) with two convolutional layers and then two fully connected layers.

To find the best model, a random search of the hyper-parameters, namely regularisers, optimisers and number of hidden units, was implemented. The best model in each case was selected as the one with the minimum average validation loss over the train-

ing epochs. This was decided as the deciding measure of accuracy as low average implies quicker convergence as well as better performance. Cross validation was then carried out to find the final accuracy of each model on the test set. There is then some plots representing the activation functions and the weights within the networks.

2 Regression

The data set for this task comprised of only one hundred data points. This is a very small number. Additionally, when visualising the training points, we see that there are none between the ranges of $x = \{-3, -1\}, \{2, 3\}$. This means the network will have to predict on areas where it has no training data.

2.1 Methods

The feed-forward network architecture is as follows:

1 **Input layer:** one unit.

2 **First Linear layer:** input=1, output=Nh1. ReLU activation. Probability of dropout=dropout1.

3 **Second Linear layer:** input=Nh1, output=Nh2. ReLU activation. Probability of dropout=dropout2.

4 **Output layer:** one unit.

The tuneable parameters are therefore the number of neurons in the two hidden layers, along with the dropout probabilities within those two layers. Additionally, the optimisation function, the weight decay rate, the learning rate and the number of epochs of training will be varied.

The best hyperparameters are found by randomly sampling a new set at every iteration and then training and finding the average validation loss over the training epochs. The lowest average validation loss is used because that implies that is the model which is converging the fastest and has overall the best accuracy.

The set of parameters to be sampled for the random search is as follows:

- **dropout1:** sampled uniformally in the range {0, 0.3}
- **dropout2:** sampled uniformally in the range {0, 0.5}
- **Nh1:** [16,32,64]
- **Nh2:** [32,64,96]
- **Optimiser:** [Adam, SGD]
- **Learning rate:** [$1e^{-5}$,$5e^{-5}$,$1e^{-4}$,$5e^{-4}$, $1e^{-3}$,$5e^{-3}$,$1e^{-2}$]
- **Number of epochs:** [100,200,300]
- **Weight decay:** [$1e^{-3}$,$1e^{-2}$,$1e^{-1}$]

These parameters were sampled to make 300 sets, which were then initiated and trained then tested on the validation set as three hundred individual trained networks.

2.2 Results

The best set of parameters was found to be as follows (rounded to 4 decimal places):

- **dropout1:** 0.0127
- **dropout2:** 0.1310
- **Nh1:** 32
- **Nh2:** 96
- **Optimiser:** Adam
- **Learning rate:** 0.001
- **Number of epochs:** 300
- **Weight decay:** 0.001

The network is then trained using this optimal set of hyperparameters. The loss is assessed on the test set, which was separated at the beginning. The model prediction is

plotted alongside the test dataset in Figure 1.

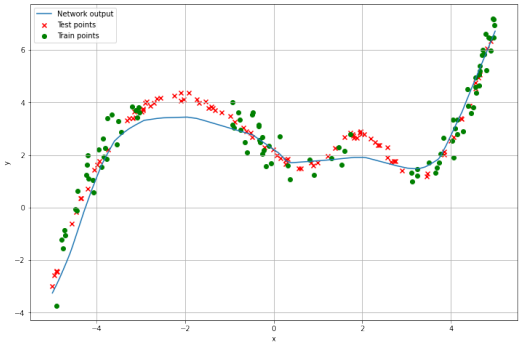

Figure 1: Best set performance

The model is unable to fit the data in the sections where there was no available training data. Instead it just connects the gap by a straight line, which is expected of the ReLU activator.

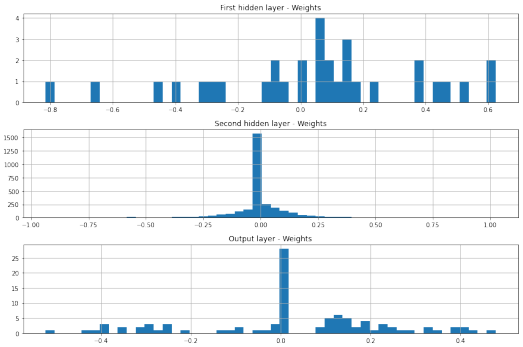

Figure 2: Histograms of the network weights

The visualisation of the weights shows that the regulariser is successfully preventing them from blowing up in any particular range.

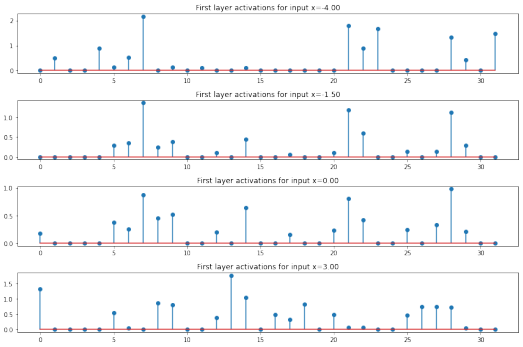

Figure 3: Activation functions of the first layer

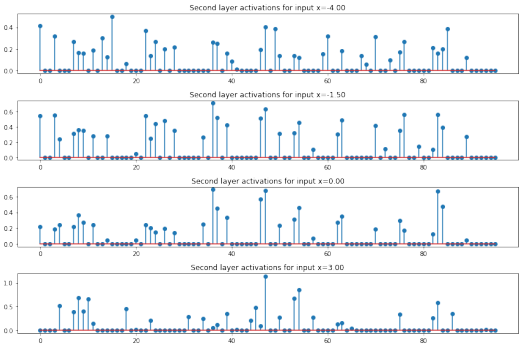

Figure 4: Activation functions of the second layer

The activation profiles shown in Figures 3 and 4 show that the first layer is minimal compared to the second. This shows that perhaps too many neurons were used for too simple a task and thus many of them were inactive.

3 Classification

The classification task comprised of training a convolutional neural network to correctly determine the labels of a set of clothing images. The inputs are the 28 x 28 pixel images, and there are ten possible output labels.

3.1 Methods

The training set this time has 60,000 entries. The test set contains an additional 10,000. The training set is then split 80:20 into a training set and a validation set. Thus, the validation set contains 12,000 images and the training set has 48,000.

First, the data was imported and then converted into PyTorch tensors. The number of samples with each label was assessed in order to check that the training data would be representative of the dataset. This can be seen in Figure 5. The number of images for each label is almost the same, thus this set is a good, representative set to use for training the model.

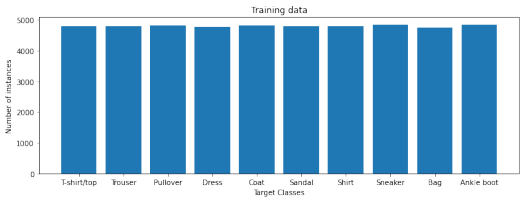

Figure 5: Training Set Label Counts

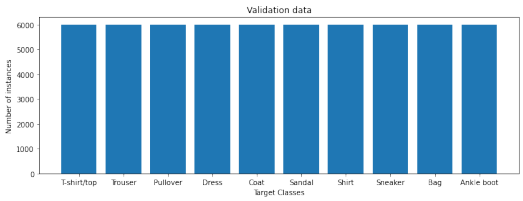

Figure 6: Validation Set Label Counts

The network architecture is as follows:

1 **Input layer:** one unit.

2 **First Convolutional layer:** input channels=1, output channels = n_channels. 5 x 5 kernel. ReLU activation. Image dimensions reduced to 24 x 24.

3 **Max Pooling layer:** Reduces dimensions of the image to 12 x 12.

4 **Second Convolutional layer:** input channels=n_channels, output channels=2*n_channels. 3 x 3 kernel. ReLU activation. Image dimensions reduced to 10 x 10.

5 **Max Pooling layer:** Reduces image dimensions to 5 x 5.

6 **Dropout layer:** Probability of dropout=dropout.

7 **First Linear Layer:** input channels=5*5*2*n_channels, output channels=100.

8 **Output Linear Layer:** input channels=100, output channels = 10.

The loss function is CrossEntropyLoss. The set of hyperparameters for sampling is as follows:

- **dropout:** sampled uniformally in the range {0, 0.6}
- **n_channels:** [3,4,5,6,7,8,9,10]
- **Optimiser:** [Adam, SGD]
- **Learning Rate:** $[1e^{-5},5e^{-5},1e^{-4},5e^{-4},1e^{-3},5e^{-3},1e^{-2}]$
- **Number Epochs:** [40]
- **Weight Decay:** $[1e^{-3},1e^{-2},1e^{-1},1e^{0}]$

In the same way as the regression task, the best model will be chosen as the one which has the lowest average validation loss over the training period. This network has a higher computational strain than in the first case, so the number of models tested in the random

search will be fewer. This is also the reason for having fewer number of epochs during this searching phase.

3.2 Results

For the random search, the validation set was further divided and used as a smaller training set and validation set. This is because the computation time on the whole training set for that many iterations was excessive. Additionally, Figure 6 shows that the validation set is representative of the dataset as a whole. After 50 iterations, the best set of hyper-parameters was as follows (rounded to 4 decimal places):

- **dropout:** 0.1335
- **n_channels:** 5
- **Optimiser:** Adam
- **Learning Rate:** 0.005
- **Weight Decay:** 0.01

The model was then trained on the entire training dataset with these hyperparameters with an increased number of epochs (100), which was significantly more computationally intensive than for the model selection process. This resulted in a test accuracy of 85.41%. Figure 7 shows the first nine images from the test dataset with the true labels and the labels predicted by the network.

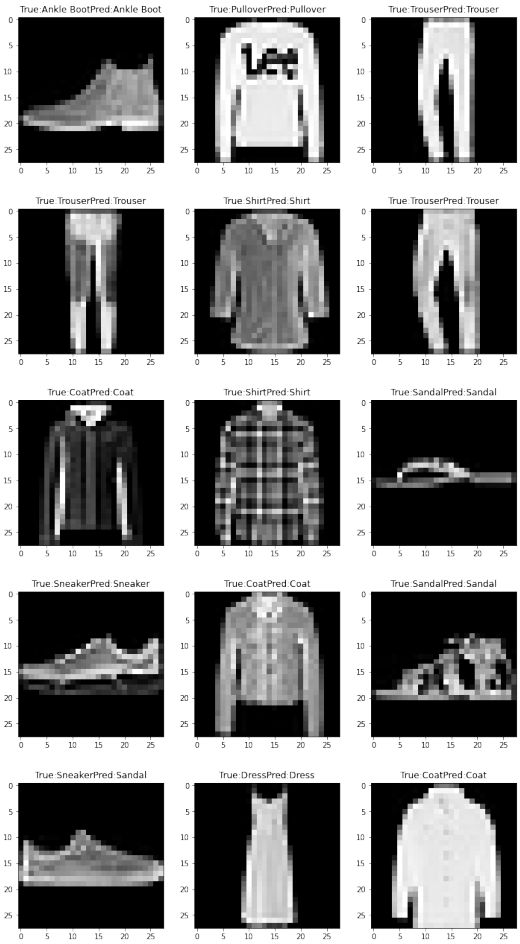

Figure 7: Output predictions with true labels and input images

The one which the model gets wrong is the bottom left image, where it predicts a sandal when the image is instead of a sneaker. This is an understandable error to make as the two items of footwear are visually quite similar.

Plots of the filters of the two convolutional layers can be seen in Figures 8 and 9. In the first layer the weights are concentrated largely in one area of the image. In the second layer this changes as the model learns more complexity.

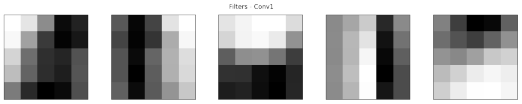

Figure 8: Filters of the first convolutional layer

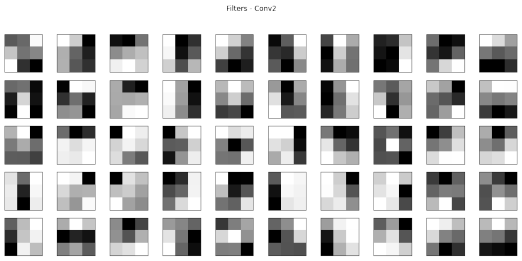

Figure 9: Filters of the second convolutional layer

Figures 10 and 11 show the activation profiles of the first and second convolutional layers. In the first layer you can still see what the image is clearly. However, once in the second layer this is no longer possible as the network learns more abstract features from the images.

Figure 10: Activations of the first convolutional layer

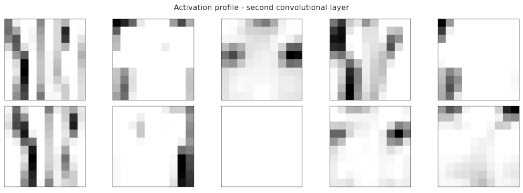

Figure 11: Activations of the second convolutional layer

4 Conclusion

In conclusion, both of the models were tuned and able to effectively complete the tasks for which they were trained. The regression network achieved an average loss of 0.28 on the test dataset, whilst a 85.41% accuracy was attained for the classification task.

The regression task had very limited data, and additionally, gaps in the training data. The predictive power on this part of the graph is hence lacking. The models could be improved by some data boosting procedures, such as adding noise. This could help to make the model more generalisable and therefore perform better on unseen data.

The classification task had significantly more data leading to less possibility of overfitting the training data at the expense of losing performance on unseen samples. The network was able to perform well, and achieved an accuracy on the test set of 85.41%. The data in this case could have been augmented or had noise added which would also have improved the generalisation of the model.