Supervised Learning and Neural Networks Report

Introduction

In this report, the process involved in designing and train Artificial Neural Networks (ANNs) to perform image recognition of fashion items will be documented and the results will be discussed. The Fashion-MNIST dataset which is a dataset of Zalando articles' images will be used. It is composed of 70000 grayscale images of different fashion products, and their correspondent labels.

Question 1: Learning rates and their corresponding effect on the loss over the training and validation data.

1a: Analysing the current loss of the network

On the right is the plot of the accuracy and loss over time that the initial code generates.

It can be seen from the plot of the accuracy and loss over time that the accuracy for both the training set and validation set converges quickly.

The validation and training loss appear to be converging initially, however after the first few cycles, the validation loss begins to spike as the training loss continues to decrease. Overall, as the training loss continues to decrease, the validation loss increases. As the error on the new data beings to increase, this indicates that the model is starting to overfit the training data.

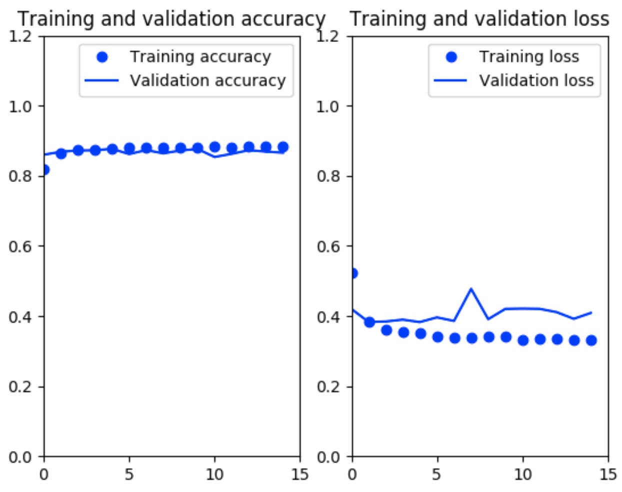


Figure 1: Original plot of accuracy and loss

1b: Varying learning rates to see the effect on the loss over the training and validation data.

The learning rate is a hyper-parameter that controls how much the weights of the network are adjusted in response to the loss gradient.

1. **High learning rate**

A high learning rate generally means a network will learn fast. If the learning rate is too high, the network becomes unstable. This can be seen by the plot in figure 2 where the learning rate was set to **0.095.** Both the training loss and validation loss increase and it can be observed how erratic the validation loss is compared to the original plot above.

As the network doesn't appear to be able to model the training data well nor generalize to new data, it can be said the model is underfitting.

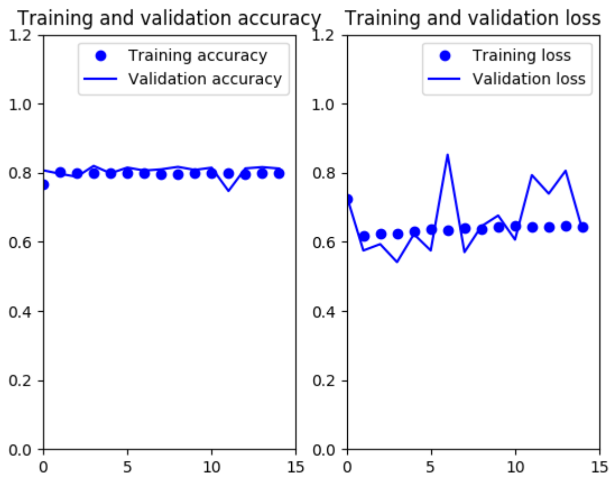


Figure 2: Plot with learning rate = 0.095

2. Low learning rate

Conversely, if the learning rate is too low, the network can be too slow to reach the optimal result. This can be seen by the plot in figure 3 where the learning rate was set to **0.0001.** It can be seen that the validation loss and training loss do converge but this is quite slow and a lower training and validation loss can be reached much quicker.

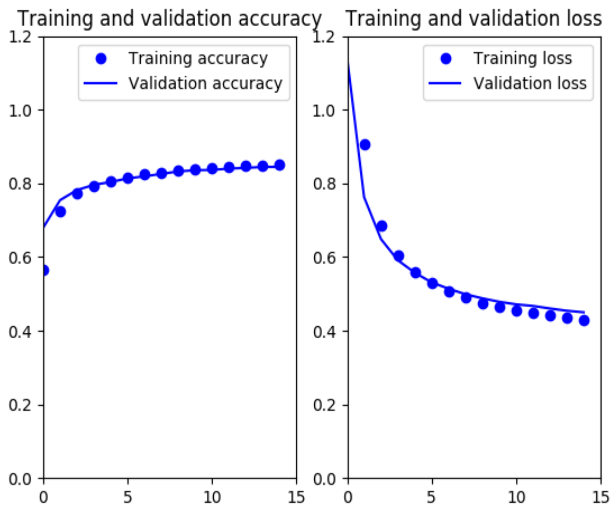


Figure 3: Plot with learning rate = 0.0001

3. Appropriate learning rate

Using a smaller learning rate helps to unsure any local minima are not missed. The optimal learning rate for this network is 0.001 which shows the training and validation loss converging but importantly, the time taken to convergence is less than shown in figure 3 where the learning rate was smaller. This can be seen below in figure 4.

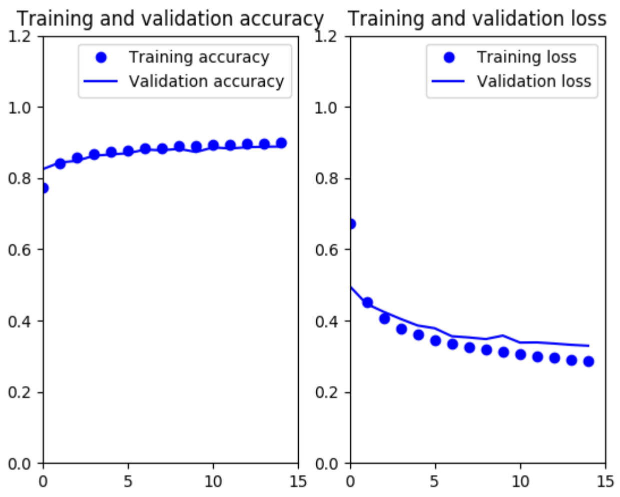


Figure 4: Plot with learning rate = 0.001

Question 2: Modifying the architecture

2a: Architecture to minimise the training loss

For the first part of this question, an architecture to achieve the minimum loss possible on the training data will be designed. The architecture specified below achieved a loss on the training data of **0.0123.** The rationale behind the choices will also be discussed.

Hyper-parameters

The architecture has 3 hyper-parameters; batch size, epochs and learning rate.

The batch size was set to **300**. This detailed the number of images with their corresponding categories to use per weights update step . Having a large batch size reduces the training error because the model is being fed more information about the training data helping it to fit the training data better each time the weights are updated.

The number of epochs used was **30**. Epochs represent the number of times the model looped over the entire training set. Having a large number of epochs reduces the training error each time because the fit to the training data improves each time as the model also begins to learn the 'noise' of the training data resulting in eventually the training data being "memorised" by the model. This eventually results in overfitting. Also the larger the number of epochs, the more computationally expensive the model becomes. Therefore the optimal number chosen was 30.

The learning rate was set to **0.01**. Using a small value helps to ensure local minimums are not missed and that the training loss will keep decreasing until it finally converges.

Convolutional layers

The following adjustments were made to the design of the convolutional layers.

The kernel size was chosen as **3 x 3.** Using a smaller filter size captures highly local features and as a result captures smaller complex features in the image and more information overall. All of which will help it learn the training data better resulting in a reduction of training loss.

An odd number is used for the kernel size to improve implementation simplicity as for odd-sized filters the previous layer pixels are symmetrical around the output pixel. This is not the case for even-sized filters.

The number of filters chosen was **37.** Filters are there to capture patterns. Giving the layer a larger number of filters captures more features and thus helps to get a matrix of more dimensions to export to the next layer.

Max Pooling

Max pooling effectively 'down-samples' the image to focus on higher-level patterns. It reduces the resolution of the pattern that the network learns, essentially just summarising what has been learnt. The dimension of the max pooling layer was adjusted to **1x1**. This is because a smaller down-sampling step would retain more information enabling the network to learn the training data better and reduce the training loss.

Dropout

Dropout is a technique designed to reduce overfitting by randomly disregarding a certain percentage of the weights of the network at learning time. Therefore a dropout layer has not been introduced given the aim of this architecture is to minimise the training loss only.

Dense layers

The architecture has **2** dense layers.

The first dense layer is placed after the model has been flattened. This layer uses activation Rectified Linear Units (ReLU) which makes sure that the output of the convolutional layer is still positive, thereby normalising the output. The number of neurons chosen for this layer was **55** using trial and error.

The second dense layer is used as the output layer with the activation Softmax. The number of neurons matched the number of classes to ensure that it gives us the probabilities to define the class of each input.

Result

The results on the training and validation loss for this architecture can be seen in figure 5 below. It can be seen that whilst the training loss is almost at 0, the validation loss increases drastically showing clearly that the model has overfit the training data. This indicates that trying to minimise training loss will result in a higher validation loss thereby impairing the model's ability to generalise to new data.

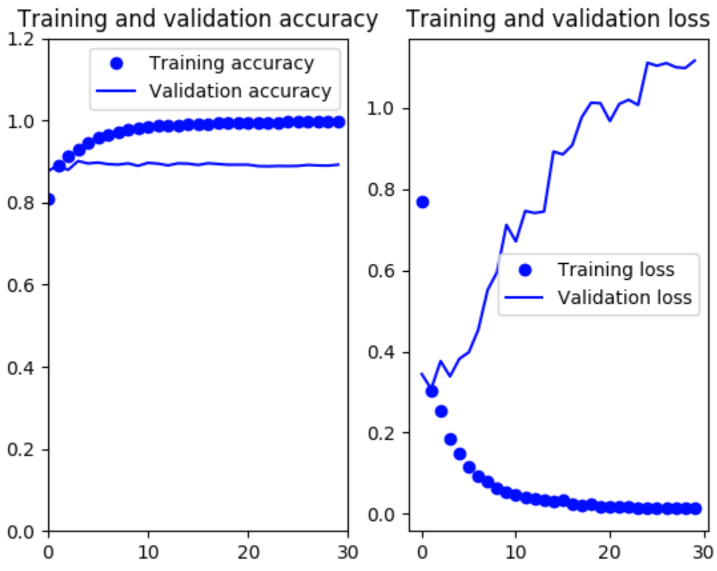


Figure 5: Plot for architecture trying to minimise training loss

2b: Architecture to minimise the validation loss

For the second part of this question, an architecture to achieve the minimum loss possible on the validation data will be designed. The architecture specified below achieved a loss on the validation data of **0.2148.** The rationale behind the choices will also be discussed.

Hyperparameters

The batch size was set to **64**. Having a large batch size reduces the training error because the model is being fed more information about the training data helping it to fit the training data better each time the weights are updated.

The number of epochs used was **4**. Having a large number of epochs increases the validation error because the fit to the training data improves each time as the model also begins to learn the 'noise' of the training data resulting in eventually the training data being "memorised" by the model. Therefore a lower number of epochs was chosen.

The learning rate was set to **0.001**. Using a small value helps to ensure local minimums are not missed and that the validation loss will keep decreasing until it finally converges.

Convolutional Layer

3 convolutional layers were used. The dimension of the filter for all the layers was chosen as **3 x 3.** Using a smaller filter size is more efficient and with more layers it learns more complex features.

The number of filters **(32, 64, 64)** increased after the initial layer to capture as many pattern combinations as possible as the patterns become more complex.

Dropout

2 dropout layers were used with rates **0.05** and **0.1** respectively. This is to ensure that the network overfitting to the training data and thus reduces validation loss.

Max Pooling

A max pooling layer with dimension **2x2** was used to down-sample the outputs. It reduces the resolution of the pattern that the network learns, essentially just summarising what has been learnt.

Dense layers

The architecture has **5** dense layers. The first three dense layers have **32** neurons each, use the ReLU activation and are placed after each convolutional layer. The fourth dense layer has **256** neurons and is placed after the model has been flattened.

The final dense layer is used as the output layer with the activation Softmax. The number of neurons matched the number of classes to ensure that it gives us the probabilities to define the class of each input.

Final result

The results on the training and validation loss for this architecture can be seen in figure 6 below. It can be seen from the plot that the validation loss performs better than the test loss.

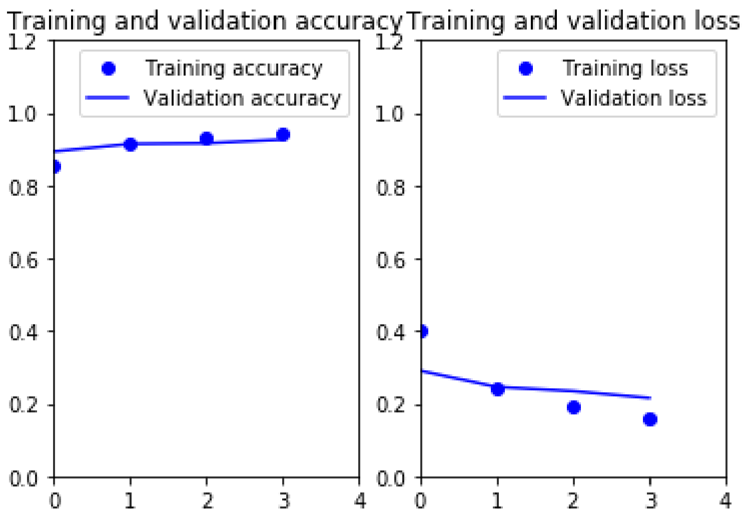


Figure 6: Plot for architecture trying to minimise validation loss

2c: Comparing the architectures to see what performs best on the test data

For the architecture designed to minimise the training loss, the results on the test set are as follows:

- Test loss = 0.54
- Test accuracy = 0.90

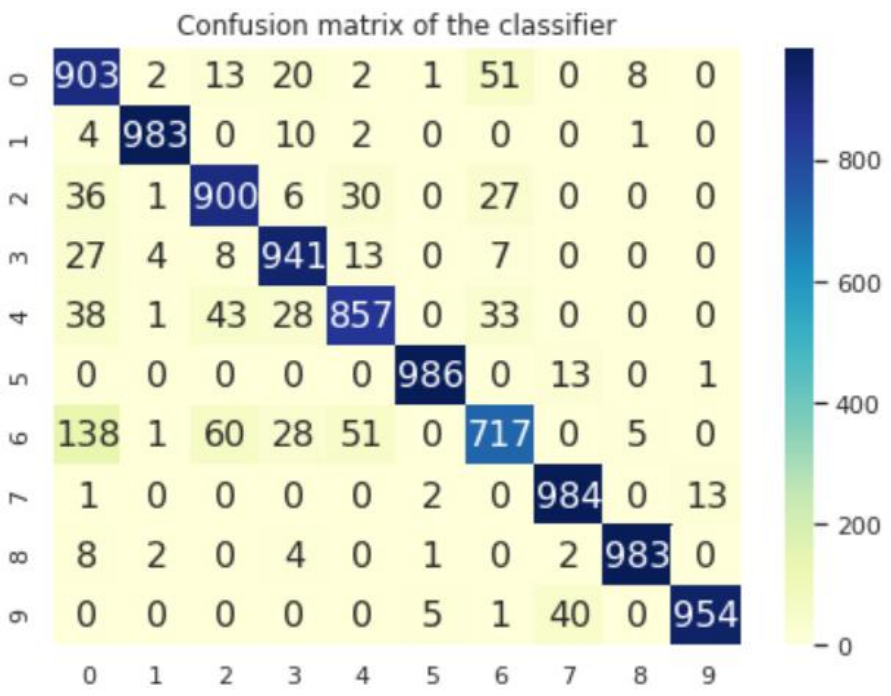
For the architecture designed to minimise the validation loss, the results on the test set are as follows:

- Test loss = 0.22
- Test accuracy = 0.92

Clearly, the architecture designed to minimise the validation loss performs best on the test data. A lower validation loss is a good indicator that the model can generalise to new data better whilst a low training loss suggests the data has been overfitted to the training data. These results serve to highlight the importance of training your network to reduce validation loss instead of focusing exclusively on training loss.

2d: Building a confusion matrix for the best architecture over the test set

To represent the accuracy of our model, a confusion matrix has been built for our second architecture designed to minimise the validation loss. This is a performance measure which measures the accuracy of our classifier. It can be seen below. The diagonal gives the number of correct classifications for each class



	0	1	2	3	4	5	6	7	8	9
0	903	2	13	20	2	1	51	0	8	0
1	4	983	0	10	2	0	0	0	1	0
2	36	1	900	6	30	0	27	0	0	0
3	27	4	8	941	13	0	7	0	0	0
4	38	1	43	28	857	0	33	0	0	0
5	0	0	0	0	0	986	0	13	0	1
6	138	1	60	28	51	0	717	0	5	0
7	1	0	0	0	0	2	0	984	0	13
8	8	2	0	4	0	1	0	2	983	0
9	0	0	0	0	0	5	1	40	0	954

Class	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

The class that is easiest to classify is class 5 with 986/1000 correct values which is the class of sandals. It is quite different from the other classes so it makes sense that the network was mostly accurate for these images. Almost all the false positives were for class 7, the class of sneakers which makes sense.

The class that is the most difficult to classify is class 6 with 717 correct values, which is the class of shirts. Most of the false positives from class 6 were for class 0 which is the class of t-shirt/tops. This makes sense intuitively, given how similar both classes are and therefore it would have been much harder for the network to correctly distinguish between the classes relative to other classes. The other class with a significant number of false positives for class 6 was class 4 which is the class of coats. This again would share many similarities with shirts making it more difficult for the network to correctly classify.

From this we can reasonably conclude that to improve the accuracy, the network needs to be trained to recognise more complex features and would benefit from more training examples to help it better distinguish between similar categories.