Instituto Superior Técnico

Machine Learning
Lab 3 - Neural Networks

André Pereira 90016

Anton Erikson 98037

Group 16

2 Pm, Friday

Teacher: Catarina Barata

Overfitting

When a model is too specialized on the training data it will perform worse on new data. This is called overfitting. To verify if this happens during training we split the dataset in two parts. The training data and the test data. We only train the model on the train data while we evaluate it on both, obtaining a training loss and a validation loss. In this way we get a picture of how the model is performing on unseen data.

MLP

We started by training the model with an Early stopping monitor that stops the training when the validation loss stops decreasing. In this case both the training loss and the validation loss decreased in a similar way and the training stopped after 39 epochs. We then continued by training the model from scratch without the monitor for 200 epochs. In this case the training loss and the validation loss started to vary after approximately epoch 39 (see Figure 1). The training loss kept decreasing while the valdiation loss started to increase. Therefore we can assume that after roughly 39 epochs the model starts to overfit.

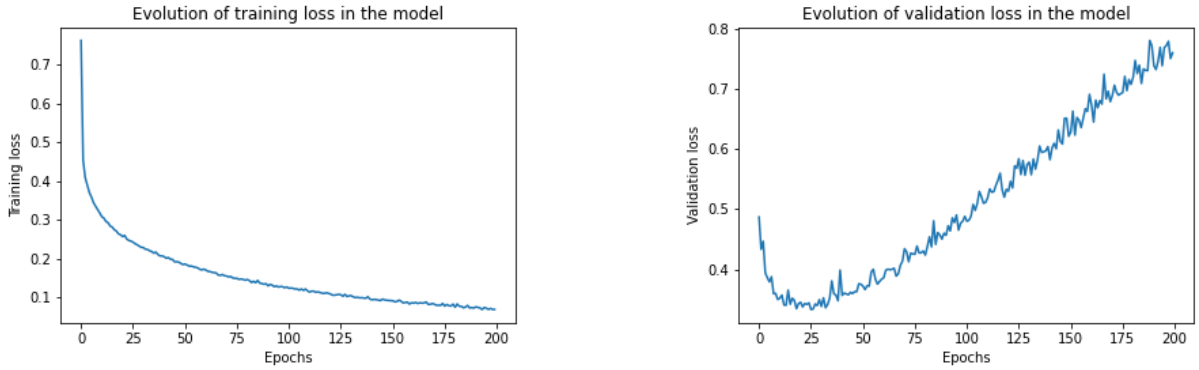


Figure 1: MLP training loss to the left. MLP validation loss to the right. Both were trained for 200 epochs. Note that the validation loss starts to increase while the training loss decreases throught he whole training.

CNN

For the CNN we used an Early stopping monitor. Therefore we did not observe any overfitting.

MLP compared to CNN

The MLP reached a minimum validation loss of about 0.3372 (see Fig.2) while the CNN reached about 0.2962 (see Fig.2). Thus the CNN had a substantially better performance. This happened even though the MLP had about twice as many paramaters (27882) compared to the CNN (15642). This is due to the design of convolutional layers. Since they convolve the earlier layer with a set of relatively small kernels they are spatially invariant and are only looking at local features. This lowers the amount of needed parameters and prevents overfitting. It has proven to be a very successful strategy in the problem of

image recognition. It would be possible to get the same behaviour from a dense layer. However it would require dramatically more weights and it is probably practically impossible to achieve through gradient descent. The CNN has one drawback though. It takes more time to train. This is also a consequence of the design. Convoluting requires more operations than the normal matrix multiplication in a dense layer.

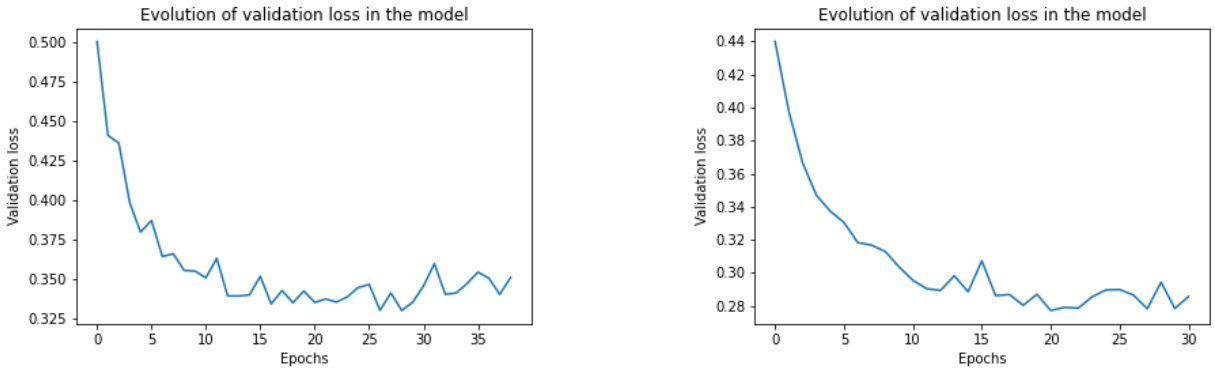


Figure 2: MLP validation loss to the left. CNN validation loss to the right. Both were trained using an Early stopping monitor. Note that the CNN validation loss reaches a lower value.

Confusion matrix

The confusion matrix gives you an idea of what classes the model confuses with each other. Its element with indexes i, j is the probability of classifying an image of class i with class j.

MLP

For the MLP we see that the highest values lay in the diagonal and resembles the identity matrix. This tells us that the model often makes the correct prediction. The lowest value on the diagonal is $i, j = 6$ (starting the count at 0). This is the shirt class. It is often confused with indexes 0, 2, 3 and 4 corresponding to T-shirt/top, Pullover, Dress and Coat. This is expected since these garments look quite similar.

$$P_{MLP} = \begin{bmatrix} 851 & 0 & 16 & 22 & 3 & 1 & 98 & 0 & 9 & 0 \\ 4 & 968 & 1 & 20 & 3 & 0 & 2 & 0 & 2 & 0 \\ 19 & 4 & 797 & 14 & 104 & 0 & 61 & 0 & 0 & 1 \\ 30 & 7 & 12 & 888 & 30 & 1 & 28 & 0 & 4 & 0 \\ 0 & 1 & 88 & 30 & 832 & 0 & 46 & 0 & 3 & 0 \\ 1 & 0 & 0 & 0 & 0 & 951 & 0 & 26 & 1 & 21 \\ 139 & 2 & 105 & 28 & 84 & 0 & 636 & 0 & 6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 21 & 0 & 937 & 0 & 42 \\ 6 & 1 & 6 & 7 & 4 & 5 & 5 & 5 & 961 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 1 & 27 & 0 & 968 \end{bmatrix}$$

CNN

The confusion matrix of the CNN resembles the one for the MLP. The difference is that most of the values on the diagonal are higher than the ones for the MLP. This tells us that the CNN model makes more accurate predictions. It is interesting though to note that the distribution of the confusion values varies a bit.

$$P_{CNN} = \begin{bmatrix} 852 & 0 & 29 & 16 & 3 & 1 & 91 & 0 & 8 & 0 \\ 2 & 979 & 2 & 11 & 2 & 0 & 2 & 0 & 2 & 0 \\ 16 & 0 & 869 & 9 & 49 & 0 & 55 & 0 & 2 & 0 \\ 15 & 5 & 16 & 884 & 36 & 0 & 30 & 0 & 13 & 1 \\ 3 & 0 & 70 & 28 & 832 & 0 & 64 & 0 & 3 & 0 \\ 0 & 0 & 1 & 1 & 0 & 967 & 0 & 18 & 2 & 11 \\ 149 & 1 & 82 & 23 & 76 & 0 & 652 & 1 & 16 & 0 \\ 0 & 0 & 0 & 0 & 0 & 20 & 0 & 949 & 1 & 30 \\ 4 & 0 & 1 & 3 & 6 & 2 & 7 & 5 & 971 & 1 \\ 0 & 0 & 0 & 0 & 0 & 8 & 1 & 33 & 0 & 958 \end{bmatrix}$$

Activation images

4. Comment on the activation images that are obtained.

Activation images provide a grasp of what the convolutional layers are achieving in the CNN model. Each layer is trained to extract different caracteristics of each data. Earlier layers are trained to extract really basic properties, like color gradients, sharp color transitions and basic shapes, while the deepest layers are specialized to extract advanced features (for example, a face). And this is exactly what we observe when we plot what each convolution layer is doing. In this case, we plotted the following layers:

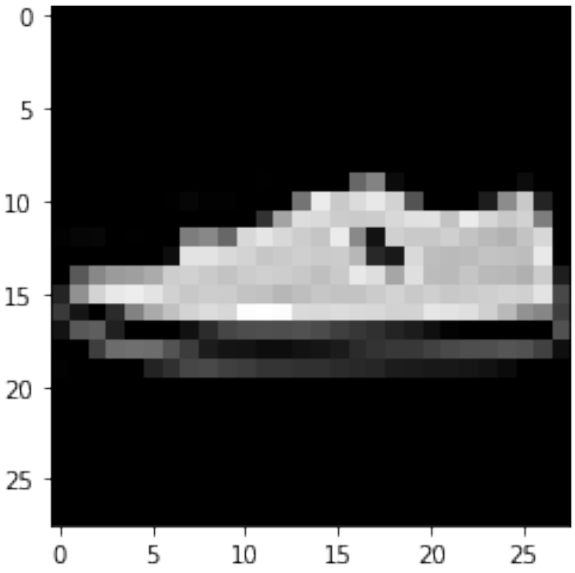
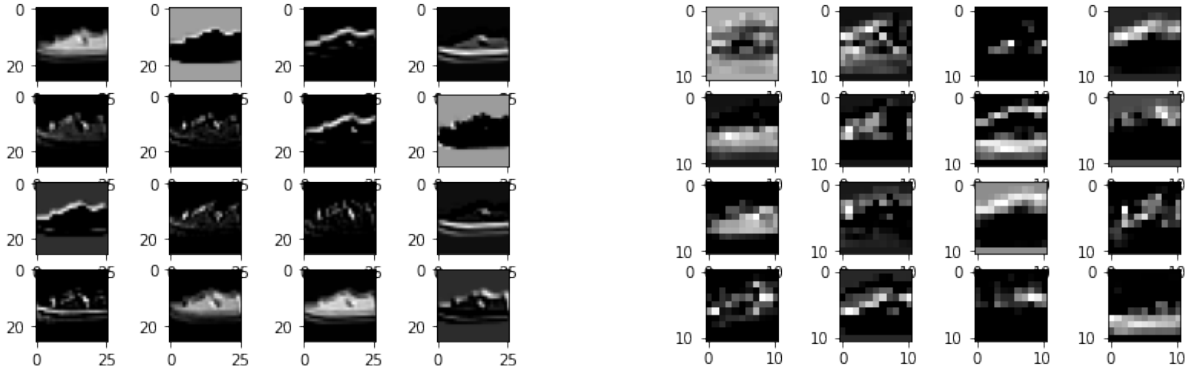


Figure 3: Image Analyzed



(a) Convolutional Layer 0

(b) Convolutional Layer 2

Figure 4: Convolutional Layers visualized

In the first layer, we observe that the convolutional layer is simply looking at sharp color transitions in the image, while the next convolutional layer that appears in the model is more focused in extracting something more advanced of the image, to the point where it started ”moving” the raw data and pixels of the image to extract it.