Exercise 4: Image classification using machine learning

Max Tamussino, 01611815

December 3, 2020

Contents

1 Different neural networks **2**

2 Optimisers **3**

2.1 Stochastic Gradient Descent 3

2.2 Adam 4

3 Prediction results for deer images **5**

3.1 CNN and MLP models 5

3.2 Linear classifier model 5

4 Perceptive field of CNNs **6**

1 Different neural networks

Various different types of neural networks were examined for their learning curves when classifying images. Specifically, the dataset *CIFAR10* was split into a train and a test sample and mapped to ten different categories. The prediction accuracy and loss function of train and test samples were plotted for 30 epochs. The performance of linear classifiers can be obtained from Figure 1. It shows increasing accuracy over epochs for the training set, however does not show a clear trend of increased accuracy for the test sample. A final validation accuracy of 34.09 % is reached. The multilayer perceptron in Figure 2 uses two additional hidden layers. It shows significantly higher values for the accuracy, however strongly overfits to the sample, as the loss function of the test sample is increasing after epoch five. This technique yields a final validation accuracy of 51.75 %.

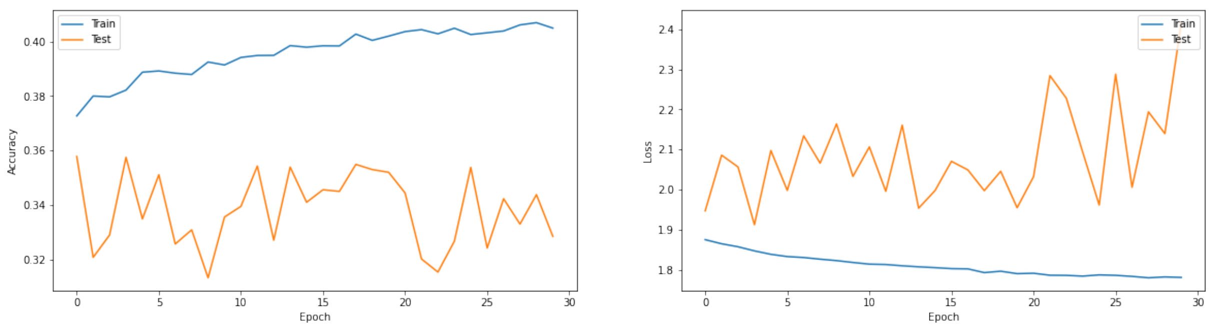


Figure 1: Linear classifier model

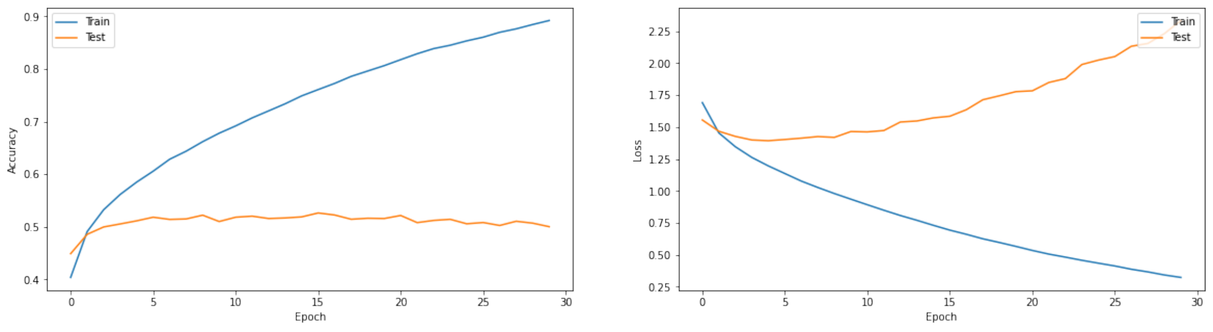


Figure 2: Multilayer perceptron model

The overfitting behaviour of the multilayer perceptron can be tackled by introducing regularisation techniques. The results for this are shown in Figure 3. The accuracy of train and test shows higher values and better convergence, while additionally the loss function for the test sample does not increase for later epochs. This model leads to 53.41 % validation accuracy in total. A further performance increase can be reached by using convolutional neural networks: Convolutional layers are introduced while still using normalisation techniques to avoid overfitting. CNNs were found to be very effective, as a final validation accuracy of 77.65 % was reached.

Finally, the pre-trained model *ResNet50* was used and fine-tuned by retraining only the last layer to be able to match to the given categories. This lead so significantly higher

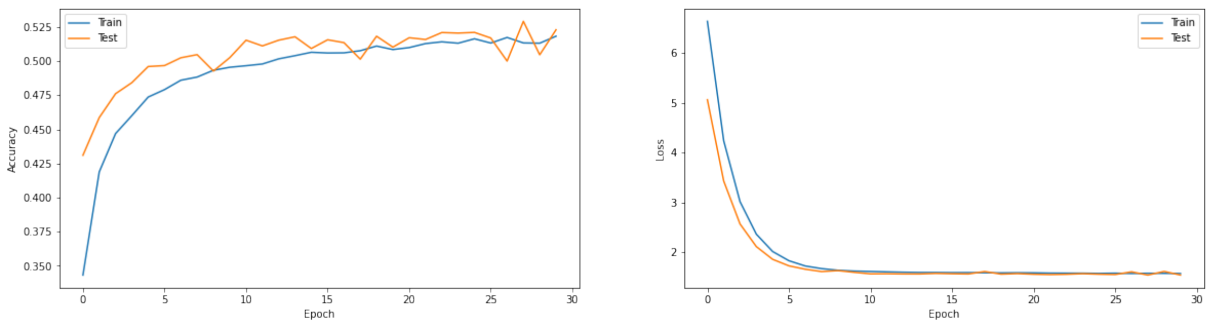


Figure 3: Multilayer perceptron model with regularisation

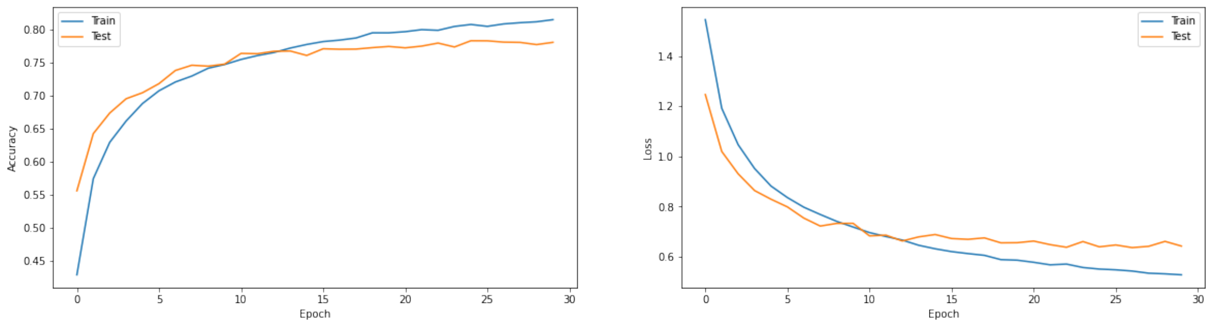


Figure 4: Convolutional neural network model

computational effort. The final validation accuracy after five epochs was 78.47 %, which is a higher result than the normalised multilayer perceptrons after 30 epochs.

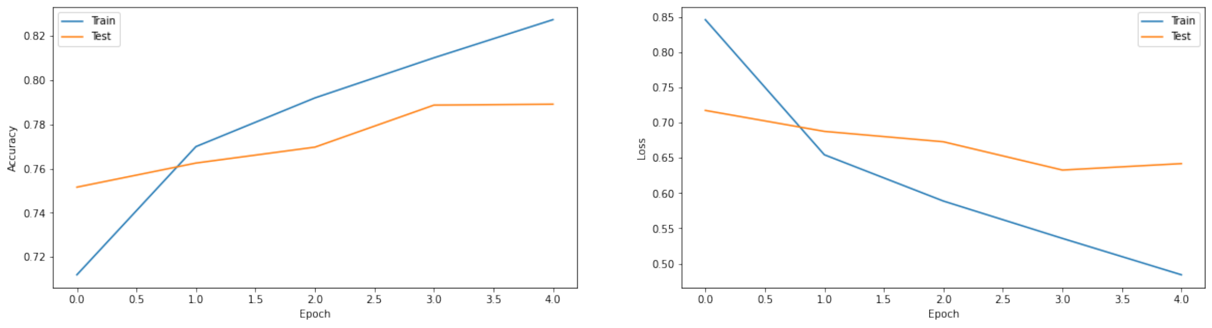


Figure 5: Resnet model

2 Optimisers

2.1 Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is parametrised by the value of the learning rate r. Figures 6 and 7 depict the learning outcome using regularised multilayer perceptrons. Figure 6 using $r = 0.1$ shows a rapid drop of the cost function in the first epoch due to its high learning rate. The subsequent epochs do not show any increase in accuracy for the test set. If the learning rate is decreased to $r = 0.001$, which is done in Figure 7, the learning time is increased significantly - however the outcome shows a more directed and smooth increase of accuracy towards later epochs. Also, this rate yields much higher test set accuracy for epoch 30.

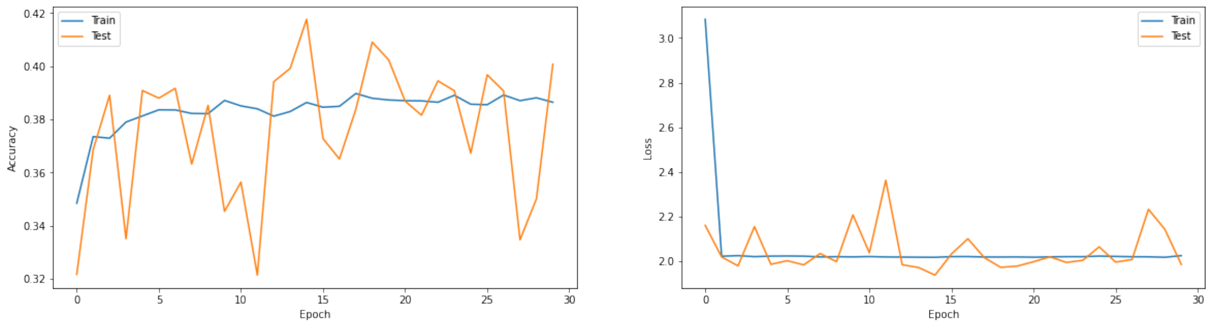


Figure 6: Multilayer perceptron with regularisation using SGD optimiser with learning rate $r = 0.1$

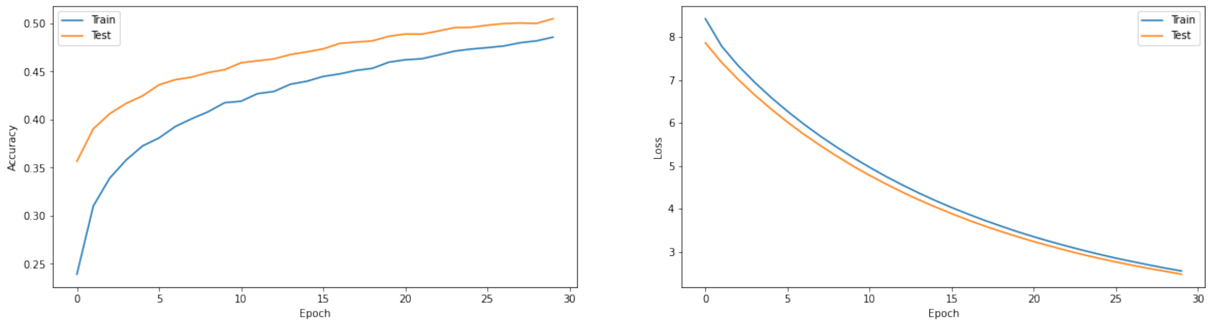


Figure 7: Multilayer perceptron with regularisation using SGD optimiser with learning rate $r = 0.001$

2.2 Adam

Adam is a method which gradually decreases the learning rate of SGD optimisation. The goal of this procedure is to reduce learning time by initially using high learning rates but maintaining smooth convergence towards the last epochs. This method shows inferior performance when compared to the basic low learning rate SGD in Figure 7, both train and test accuracy are lower than the ones for basic SGD. The outcome of the Adam learning procedure is shown in Figure 8.

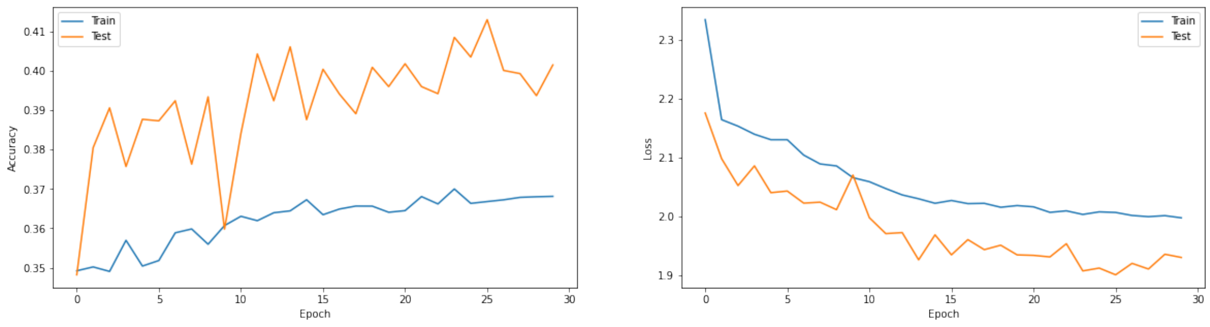
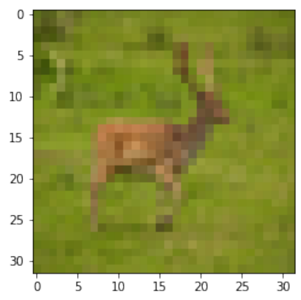


Figure 8: Multilayer perceptron with regularisation using Adam optimiser

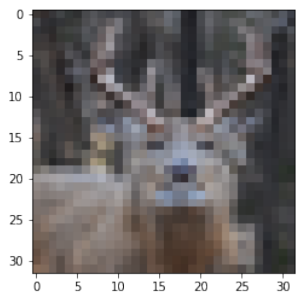
3 Prediction results for deer images

3.1 CNN and MLP models

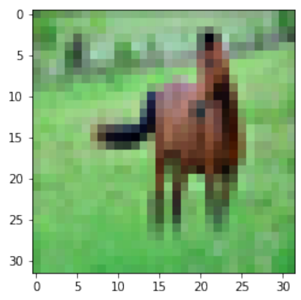
The performance of the convolutional neural network and the multilayer perceptron can be examined by looking at the predictions for the images in Section 3.1. The latter yields inferior performance: Figure 9a is correctly classified as a deer with 66 %, while Figure 9b is wrongly classified as a cat with 31 % and Figure 9c is also wrongly classified as a deer with 50 %. It can be observed, that the classification process is significantly dependant on image colour. Green background color leads to classification as a deer, while Figure 9b is not recognised. The CNN predicts all images more accurately. Figure 9a is even more strongly correctly classified as a deer with 98 %. Figure 9b is surprisingly classified correctly with 99 %, and even Figure 9c is detected correctly as a horse with 87 %.



(a) Deer in front of green background



(b) Deer in front of brown background



(c) Horse in front of green background

Figure 9: Images to be classified by CNN and MLP

3.2 Linear classifier model

In Figure 10, the image class predictions of the linear classifier model are shown for the image class *deer*. As assumed from the outcomes of the previous experiment, correct predictions are predominantly showing green or dark backgrounds, as do false positives. False negatives show problems with other animals which are often depicted in front of similar backgrounds. The strong color dependency of MLPs is also the reason for the superior performance of CNNs, which have convolutional layers preserving edge information.

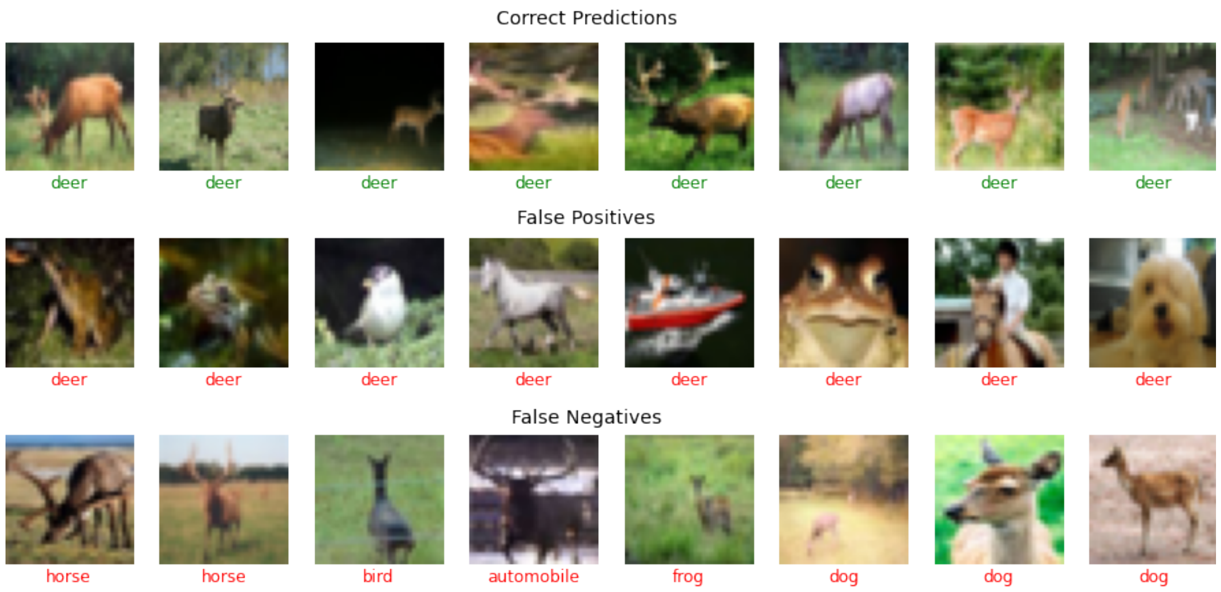


Figure 10: Predictions of the linear classifier model for image class *deer*

4 Perceptive field of CNNs

As CNNs use convolutions, each layer of convolutions has an increased perceptive field - which is the number of pixels influencing one particular node. Additionally, pooling between the layers increases this number. The perceptive field can be calculated using Equation (1), with l_k being the perceptive field of the current layer, l_{k-1} the perceptive field of the last layer, f_k the filter size $(k \times k)$ and s_i the strides $(s \times s)$ of past pooling layers. For the implemented three convolutional layers with filter size 3, which are interleaved by two pooling layers with $s = 2$, this yields $l_1 = 3$, $l_2 = 7$ and $l_3 = 15$.

$$l_k = l_{k-1} + (f_k - 1) \cdot \prod_{i=1}^{k-1} s_i) \tag{1}$$