TUBAF
The University of Resources.
Since 1765.

Documentation

Scientific Computing Project

Winter Semester 2024/2025

Lars Wunderlich, Parsa Besharat, Toni Sand

April 6, 2025

TU Bergakademie Freiberg

Contents

1 Task

The goal of this project was to code the model of a convolutional neural network and train it for the task of classifying images which consists of assigning them to one out of 10 possible classes. The used training images are self-generated by the students of this course and collected into a database. A detailed description of the used training data can be found in chapter 2, whereas the architecture of the network is explained in chapter 3. Especially, the network should also be able to classify new images correctly. In chapter 4 we talk about the training process and the tuning of hyperparameters like learning rate, batch size or optimizer by using k-fold cross-validation. At the end in chapter 5, we evaluate the result of our work by looking at several performance measurements like the confusion matrix and the loss and accuracy curves. Finally, we discuss approaches for improvement.

2 Description of the Dataset

The data set that is used for the project consists of 10 classes of images. These are

1. bottles	2. mugs/cups	3. spoons	4. knifes	5. forks
6. shoes	7. t-shirts	8. plants	9. chairs	10. bikes

Every student had to contribute to this database by generating 15-20 pictures of pairwise disjoint objects for each class. In total there are approximately 400 images per class which yield a raw database of almost 4000 images. In the next step the images of

(a) 10. bikes

(b) 2. mugs/cups

(c) 3. spoons

Figure 1: Example images of the raw database

the database were normalized to a resolution of 56x56 pixels. That made the training possible, despite the different resolutions in the beginning. To enlarge the database and improve the training process, we used the following augmentation techniques:

- horizontal flip
- vertical flip
- crop
- rotation
- blur
- change of brightness
- change of contrast
- change of color

Here the intensity of the effects and attributes like the rotation angle are selected randomly based on a uniform distribution. It is also possible that more than one effect is applied simultaneously to one image. The number of these applied effects on this image is chosen randomly in the same manner but with a maximum of 4. As a result of

(a) 10. bike (b) 1. bottles (c) 3. spoons

Figure 2: Example images of the augmented database with different transformations and a resolution of 56x56 pixels

this preprocessing procedure, we got 1500 images for every class and therefore a total augmented database of around 15000 images with a resolution of 56x56 pixels for the training and testing of our neural network.

3 Architecture of the Network

For our convolutional neural network we were given the architecture of a simplified VGG-16 that we should implement. A VGG-16 is characterized by a very deep but uniform architecture, i.e. many convolutional layers in a row with a kernel size of 3, followed by a pooling layer. The simplified version of the VGG-16 uses fewer repetitions of these layers. In our case we used 4 repetitions after the initial convolutional layers with max-pooling. The first convolutional layer gets the 56x56 image with the 3 color channels as input and convolutes it to 64 channels while preserving the resolution because of the kernel size of 3 and the padding of 1. After that another convolutional layer is applied again preserving both resolution and number of channels. The following max-pooling halfes the image resolution to 28x28 due to the kernel size and stride of 2 (see Figure 3). This procedure is repeated with again 2 convolutional layers and a max-pooling that doubles the number of channels to 128 and halfes the image resolution to 14x14 in the end (see Figure 4). After this, there are 3 more repetitions acting in a similar manner but with 3 convolutional layers instead of 2 (see Figures 5, 6, 7). Subsequently 4 fully connected layers are applied (see Figure 8). The reason for choosing such a high number of channels is to enable the network to learn the high variety of features in the complex input images and focus less on the spatial resolution. In the whole network ReLU is used

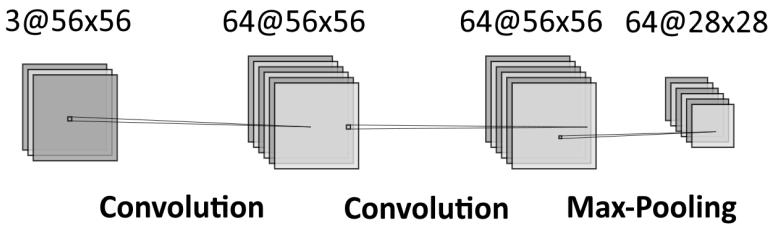

Figure 3: Inital sequence of 2 convolutional layers with max-pooling

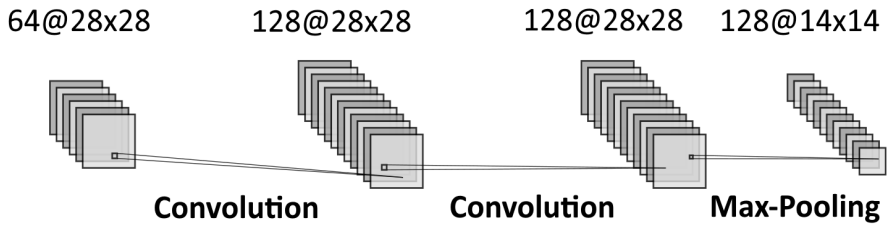

Figure 4: First repetition with 2 convolutional layers and max-pooling

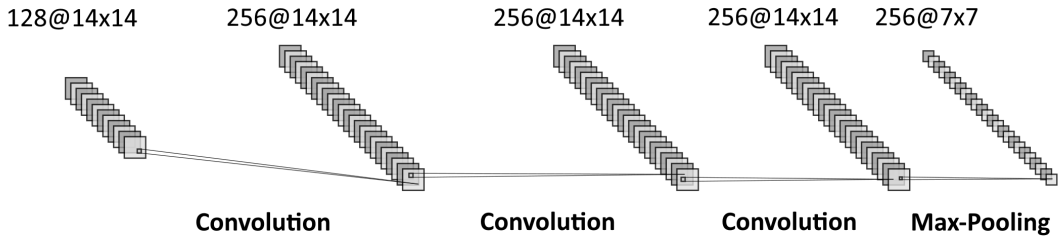

Figure 5: Second repetition with 3 convolutional layers and max-pooling

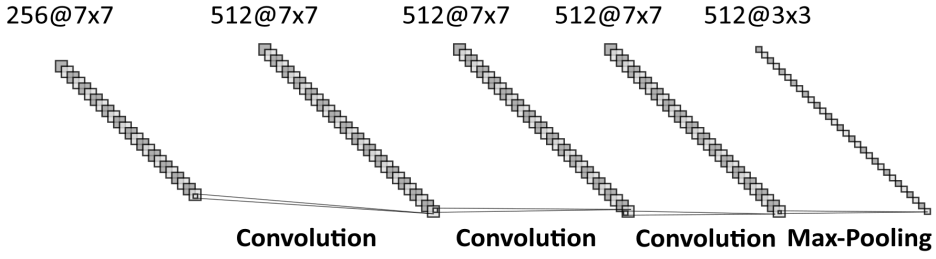

Figure 6: Third repetition with 3 convolutional layers and max-pooling

as activation function. Furthermore we applied a batch normalization to the output of each convolutional layer so that the training process behaves more stable by fixing the means and variances of each layers inputs. This helps speeding up the training and improves the generalization properties, i.e. reducing overfitting.

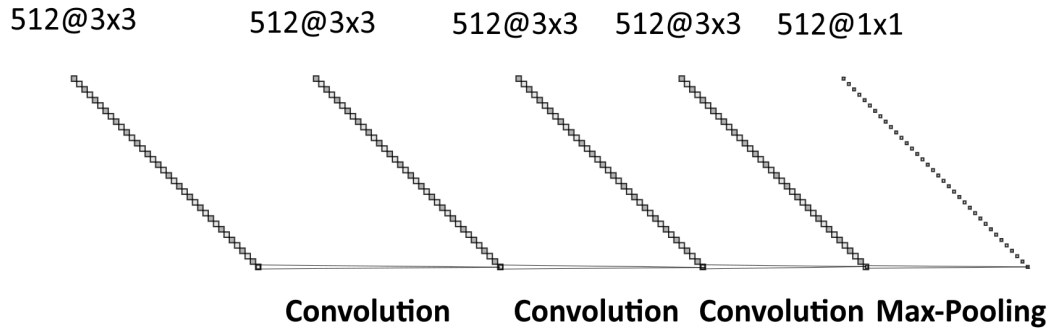

Figure 7: Fourth repetition with 3 convolutional layers and max-pooling

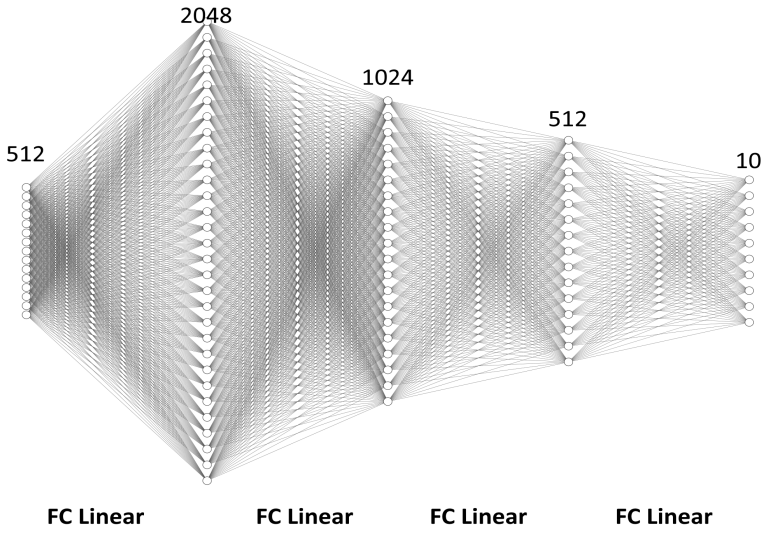

Figure 8: Fully connected linear layers at the end

4 Training

For training the network we started by finding the best hyperparameters using 3-fold cross-validation. Here the data is divided into 3 equally sized folds, where 2 of the folds is used for training and 1 fold is for testing. The reason for using $k = 3$ is because of the relatively small database and the limited computational power that is available. Because for $k = 3$ the model gets trained on a big partition of the data (two-thirds)

and for fewer times, as for higher values of k. We used this procedure for finding the best optimizer, learning rate and batch size. As possible optimizers we tried ADAM and SGD and several combinations of different learning rates from 0.0005 to 0.1 with batch sizes between 32 and 128. As a result we got as best hyperparameters for the ADAM optimizer a learning rate of 0.0005 and batch size of 64. For SGD we got a learning rate of 0.1 with a batch size of 32. For the actual training we used these hyperparameters in combination with a number of epochs equal to 50. In Addition to have a better comparison between the two optimizers, we trained for both ADAM and SGD with the aforementioned hyperparameters.
In the original task it was required to split the dataset into 80% training data and 20% test data. When we first tried this configuration we encountered the problem that the connection to the cluster got disconnected after 24 hours and the training process remained unfinished. After reaching out to the University we got informed that using the cluster for longer then 24 hours is not possible due to security reasons. Therefore we could only train in this 80/20 configuration for less training epochs which lead to low accuracy and unsatisfying results.
To solve this problem we tried training the model again with a 3-fold cross-validation while using the best hyperparameters we determined before. By that we could reduce the training time to a little less then 24 hours while also achieving a relatively high accuracy.

5 Results

As the result of our training process for both ADAM and SGD we got the following curves for the average accuracy and loss over all folds with respect to the training epochs (see Figure 9, 10).

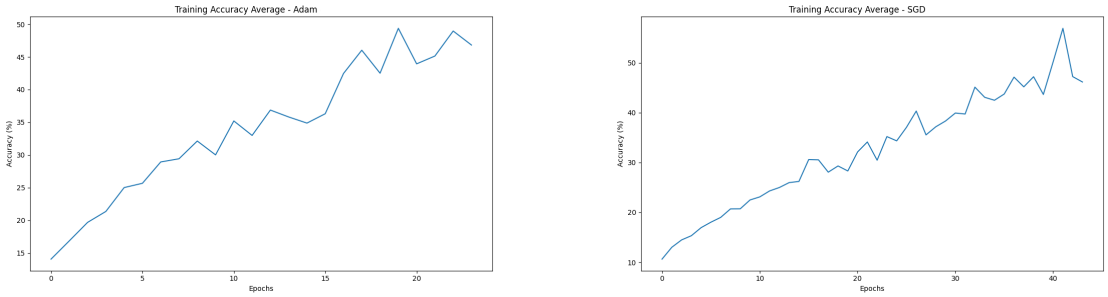

Figure 9: Average accuracy of ADAM and SGD with respect to training epochs

Here we can see that the accuracy of both optimizers increases over the training process but does not exceed 60%. Also the loss is decreasing which fits to the observations of the accuracy.

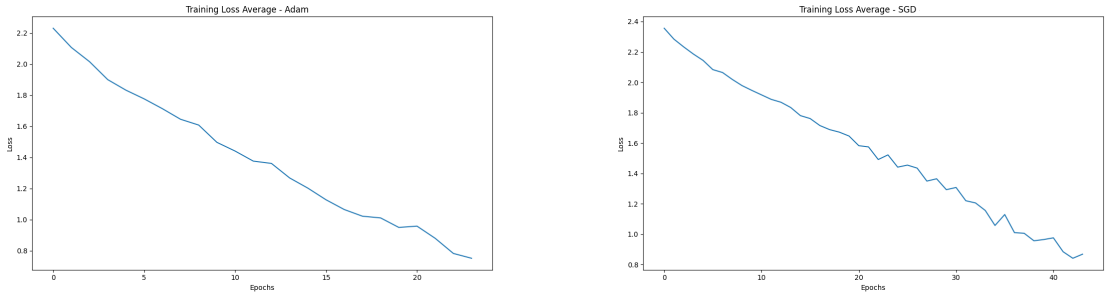

Figure 10: Average loss of ADAM and SGD with respect to training epochs

When testing the trained networks on new unknown data we got the following accuracies:

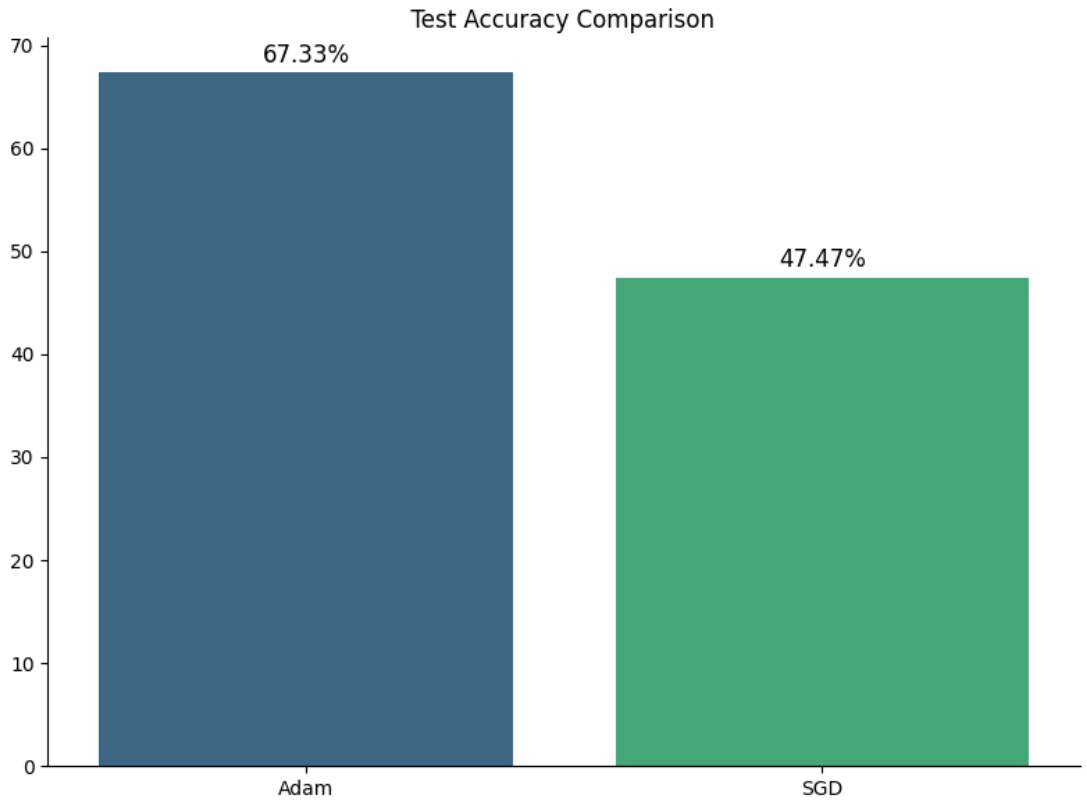

Figure 11: Comparison of the accuracy of both optimizers over test data

Here we can see that ADAM performs better on the unknown data with an accuracy of 67.33%, whereas SGD has less accuracy of just 47.47%. When looking at the corresponding confusion matrices, one can verify this result (see Figure 12).

To reduce the training time we also implemented a counter that counts the number of training epochs where no improvement happened. If this counter exceeds 7, the training for the current fold is skipped and the training with the new fold starts. Because in the training with ADAM there were no improvements after a certain epoch, the training wasn't executed until the 50th epoch but canceled earlier (see Figure 9, 10 for ADAM).

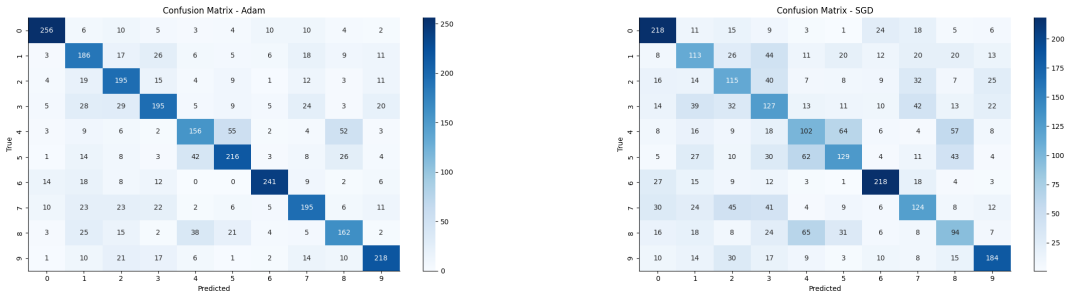

Figure 12: Confusion matrices for ADAM and SGD for the test data

In the confusion matrix of ADAM one can see that most of the predictions were correct, which is shown by the dark blue main diagonal. In comparison to that, the main diagonal in the confusion matrix of SGD is brighter, representing less correct predictions.
To sum up, the model which was trained with the ADAM optimizer (with the best corresponding hyperparameters) performed better on the new data then SGD. But overall the accuracies are not very high at all. We also tried to use a 64x64 resolution for the data with the hope for improvement, but due to the restricted time for computation we couldn't finish a training process with a higher accuracy. This result could have been improved by having more using time of the cluster and higher computational power to enable a more complex training process. Also the architecture of the network could be improved. The original idea of VGG-16 is using lots of repetitions for the convolutional layers with pooling. By using more repetitions then just 4 like in our case, the networks performance could probably be improved. Because of the complex structure of our used images (e.g. high resolution, color) there is a high chance that a more deeper network would perform better by recognizing the complex features in the data. But that would also yield a higher computational effort which is again the problem from before.