Coursework Approach

All three tasks classify RGB images which consist of one digit in each color channel. Each image is associated to the sum of the three digits is subtracted by 1. In total numbers from 0 to 19, and therefore 20 classes, are possible results. Because this is a multi-classification, it is recommended to use categorical cross entropy as loss function and a softmax unit at the output (Ketkar, 2017). The softmax activation function is fitting in this context, as the results are between 0 and 1 and adding them up equals 1. The range of numbers is therefore limited just like the number of possible classes. Additionally, every model is rated with the categorical accuracy, which is the fraction of successful prediction to the total number of predictions, as well as the performance of the training, which is the time per epoch.

While in the first task the RGB images are split into three sub-images which are then classified, in the other two tasks the model is supposed to predict the classes of the given RGB images as whole. The second task uses a pre-built Deep Learning Network. Fully connected, convolutional and pooling layers are added based on testing the model's accuracy in order to increase this metric. Because of the high number of images and the long running time (in task 2 and 3) the k-fold validation was dispensed.

Summary Report

Task 1

The first tasks started with preprocessing the data which includes separating the color channels and normalizing the values of the given RGB-images and the MNIST training images in order to have value between [0, 1]. The total testing accuracy, loss, trainable parameter and time per epoch are shown in *Table 1* to compare the results between selected models. The second number in the column of accuracy is calculated through testing random examples of the given data and subtracting the sum of the digits with 1. The most accurate predictions are made by the last model but both others are good as well with 0.9895 and 0.9838 for a single image. The table shows what a difference the reduction of nodes in the first fully connected layer has (Model 1 and Model 3) which increases the accuracy about 0.006 and the small improvement of the additional pooling layer between Model 2 and Model 3.

	accuracy	**loss**	**Trainable parameter**	**Time per epoch**
Conv2D, Con2D, MaxPooling2D, Flatten, Dense (256), Dense (128), Dense (10) [Model 1]	0.9838/0.9539	0.0783	1,446,734	6s
Conv2D, Con2D, Flatten, Dense (128), Dense (10) [Model 2]	0.9895/0.9628	0.0535	2,818,638	6s
Conv2D, Con2D, MaxPooling2D, Flatten, Dense (128), Dense (10) [Model 3]	0.9898/0.9856	0.0502	711,246	7s

Table 1: Result of Task 1

Task 2

For the second task the values were normalized as well and according to the specifications of the pre-built models resized and reshaped. Just like *Table 1*, *Table 2* compares the results of the second task. While the evaluated accuracy is above 60% for all models, *Xception* has the largest with 0.9683%. It also uses the second most parameters and takes the longest time per epoch. Additionally, the pre-build models fully-connected layers were used to produce to right number of classes and improve the accuracy further.

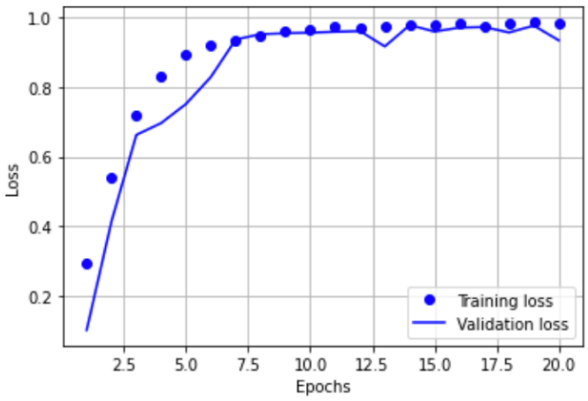

Figure 1 Accuracy of Xception in Task 2

	accuracy	loss	Trainable parameter	Time per epoch
Xception	0.9683	0.1665	21,866,300	238s
VGG16	0.0395	2.9986	14,987,604	53s
ResNet50	0.9097	0.4657	24,593,940	70s
ResNet50V2	0.9349	0.3463	24,578,708	99s
ResNet101	0.7846	0.8624	43,612,180	118s
ResNet101V2	0.6016	0.6016	43,588,244	120s
InceptionV3	0.9621	0.2055	22,827,700	124s
MobileNet	0.0395	2.9986	4,856,084	94s
DenseNet121	0.9462	0.2785	7,488,916	70s
DenseNet169	0.9440	0.2722	13,347,220	128s
EfficientB0	0.7991	0.6758	4,673,680	60s

Table 2 Results of task 2

Task 3

The Deep Learning Model of task 3 takes the normalized RGB image with the three digits as an input and classifies it. Through testing different sequences, epoch and batch sizes the model with the best accuracy was *Model 1* in *Table 3* with 0.8233. However, the figure showing the development of training and validation accuracy shows overfitting. In order to prevent this, the number of trainable parameters and layers was decreased and adjusted. An example for that is *Model 3* which is more likely to underfit the data as the accuracy is still below *Model 1*. *Model 2* and *Model 4* are other examples of the testing process in order to increase the accuracy. *Model 3* has a high number of trainable parameters and consists only of fully connected layers and Model 4 takes advantage of 3D-Convolution and needs therefore more time per epoch, but both are still below *Model 1*.

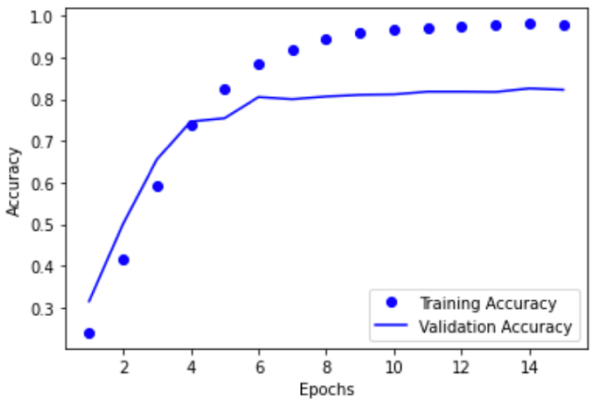

Figure 2 Accuracy of Model 1 in Task 3

	accuracy	loss	Trainable data	Time per epoch
Conv2D, Conv2D, Flatten, Dense (512), Dense (256), Dense (128), Dense (20) [Model 1]	0.8233	1.0657	1,254,491	6s
Flatten, Dense (512), Dense (256), Dense (128), Dense (20) [Model 2]	0.7615	1.3628	1,371,540	5s
Conv2D, Conv2D, Flatten, Dense (256), Dense (128), Dense (20) [Model 3]	0.7739	1.2983	585,307	6s
Conv3D, Flatten, Dense (256), Dense (128), Dense (20) [Model 4]	0.6787	2.0818	8,109,376	7s

Table 3 Results of task 3

In comparison between the different deep learning networks the ones of the first task make the highest number of right predictions followed by the pre-build networks and task 3. The process of maximizing the accuracy was similar in task 1 and 3 but with the simpler image classification the first one has higher accuracies.

References

Ketkar N. (2017) Introduction to Keras. In: Deep Learning with Python. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-2766-4_7