TBMI26 – Computer Assignment Reports Deep Learning

Deadline – March 14 2021

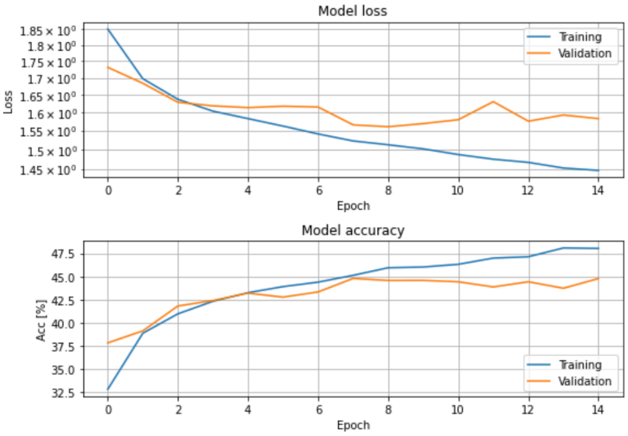
Author/-s:
Nicolas Taba (nicta839)
Tim Yuki Washio (timwa902)

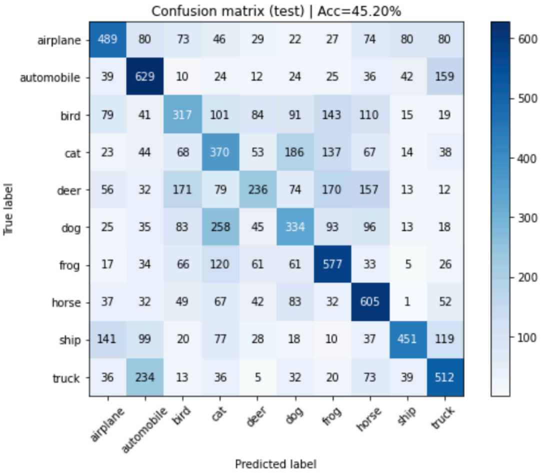
In order to pass the assignment you will need to answer the following questions and upload the document to LISAM. Please upload the document in PDF format. **You will also need to upload the Jupyter notebook as an HTML-file (using the notebook menu: File -> Export Notebook As...)**. We will correct the reports continuously so feel free to send them as soon as possible. If you meet the deadline you will have the lab part of the course reported in LADOK together with the exam. If not, you'll get the lab part reported during the re-exam period.

1. **The shape of X_train and X_test has 4 values. What do each of these represent?**

1. Value represents the number of images
2. Value represents the width
3. Value represents the height
4. Value represents the number of color channels (red, green, blue in our case)

2. **Train a Fully Connected model that achieves above 45% accuracy on the test data. Provide a short description of your model and show the evaluation image.**



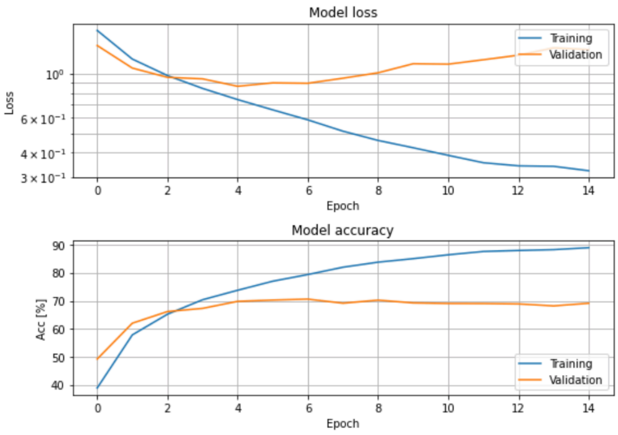


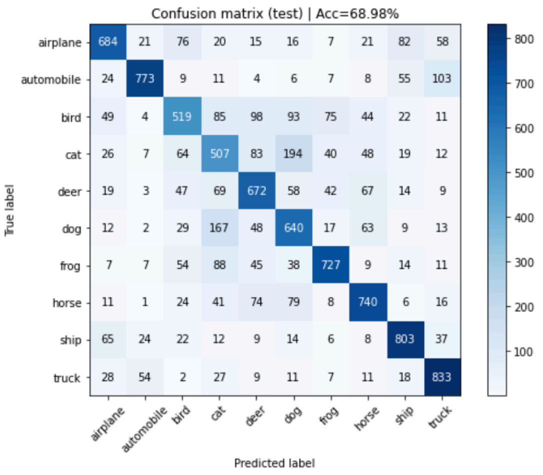
After flattening the data, we add 3 Dense layers to our model. The first layer uses relu as activation function and has an output size of 64. The second layer uses relu as activation function and has an output size of 128. The third layer is our final layer which uses the softmax as activation function and has an output size of 10 since we are classifying on 10 different classes. We achieve an accuracy of 45,2% on the test data.

3. **Compare the model from Q2 to the one you used for the MNIST dataset in the first assignment, in terms of size and test accuracy. Why do you think this dataset is much harder to classify than the MNIST handwritten digits?**

Unlike the MNIST dataset the CIFAR-10 dataset contains three color values per pixel (RGB). Therefore, we are working on a bigger dataset. So compared to the MNIST dataset which contains gray-scale photographs represented by 28x28 (flattened to 728) values the CIFAR-10 dataset containing photographs represented by 32x32x3 values is far more complex. Another reason why the CIFAR-10 dataset is harder to classify is the orientation of the photos. The numbers in MNIST dataset are always written in the same orientation and at most slightly different angles. The photos in the CIFAR-10 dataset though can contain objects that are mirrored or upside down and still represent the same class. Also, the objects in the CIFAR-10 dataset can look very different. For example, inside the dogs class we can find all kinds of different breeds and thus totally different looking dogs, which makes classifying a dog way harder than a number.

4. **Train a CNN model that achieves at least 62% test accuracy. Provide a short description of your model and show the evaluation image.**





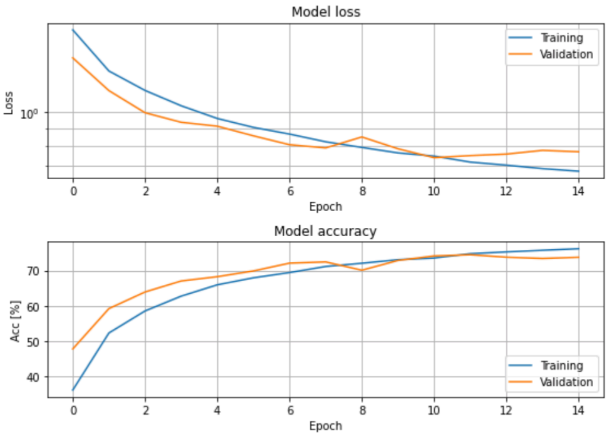
	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	684	21	76	20	15	16	7	21	82	58
automobile	24	773	9	11	4	6	7	8	55	103
bird	49	4	519	85	98	93	75	44	22	11
cat	26	7	64	507	83	194	40	48	19	12
deer	19	3	47	69	672	58	42	67	14	9
dog	12	2	29	167	48	640	17	63	9	13
frog	7	7	54	88	45	38	727	9	14	11
horse	11	1	24	41	74	79	8	740	6	16
ship	65	24	22	12	9	14	6	8	803	37
truck	28	54	2	27	9	11	7	11	18	833

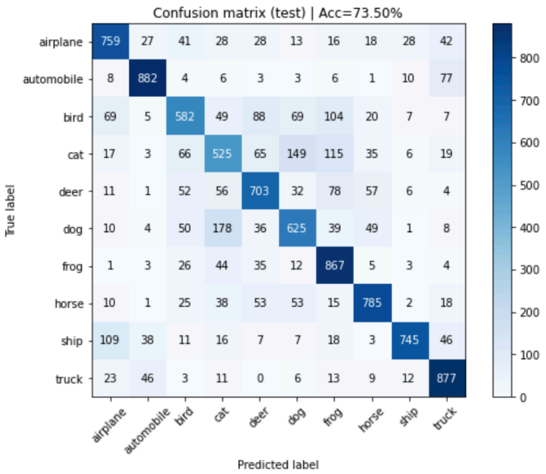
We are adding two VGG blocks to our current architecture. One VGG block is defined by two convolutional layers and one max pooling layer. This is a common architecture pattern first described by Karen Simonyan and Andrew Zisserman in 2014 (https://arxiv.org/abs/1409.1556). We also change our model build up by using a different approach. We import **layers** and **models** from the **tensorflow.keras** package, define our model first and add layers afterwards using the **add()** function. We achieve an accuracy of 68,98% on the test data.

5. **Compare the CNN model with the previous Fully Connected model. You should find that the CNN is much more efficient, i.e. achieves higher accuracy with fewer parameters. Explain in your own words how this is possible.**

Using a fully connected model for classifying images is less efficient and less successful since we need a huge amount of weights for the first hidden layer (in our case 32x32x3 = 3072) which could easily result in overfitting on the training data. By introducing a convolutional kernel (3x3 in our case) we can reduce the matrix representing an image to a lower dimension matrix. We can then further reduce that dimension introducing a pooling layer that finds the maximum matrix (2x2 in our case) inside the convolutional output. While fully connected models can't be used for feature extraction, a CNN is able to do this by using adjacent pixels to do a step by step downsampling of the initial image and extract the best image features. In other words, in a CNN each neuron is only connected to some neurons of the previous layer multiplying a common set of weights, which makes sense since images contain local features consisting of only pixels near to each other and not spread out to the whole picture. In contrast, in a fully connected model each neuron is connected to every neuron in the previous layer by a specific weight and bias which makes no assumptions about the features in the data. Thus, CNNs are the optimal fit for image classification and result in better performance and accuracy then traditional fully connected models.

6. **Train the CNN-model with added Dropout layers. Describe your changes and show the evaluation image.**



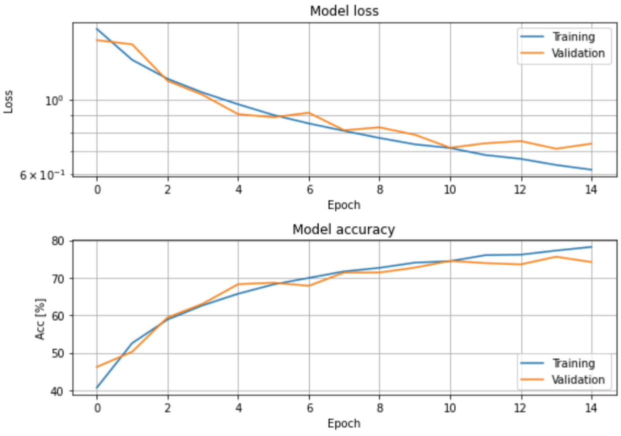

Confusion matrix (test) | Acc=73.50%

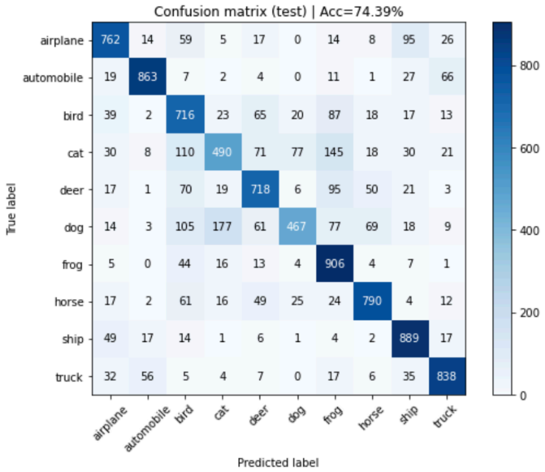
True label \ Predicted label	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	759	27	41	28	28	13	16	18	28	42
automobile	8	882	4	6	3	3	6	1	10	77
bird	69	5	582	49	88	69	104	20	7	7
cat	17	3	66	525	65	149	115	35	6	19
deer	11	1	52	56	703	32	78	57	6	4
dog	10	4	50	178	36	625	39	49	1	8
frog	1	3	26	44	35	12	867	5	3	4
horse	10	1	25	38	53	53	15	785	2	18
ship	109	38	11	16	7	7	18	3	745	46
truck	23	46	3	11	0	6	13	9	12	877

We add an additional dropout layer with dropout probability of 50% to prevent the observed overfitting from the previous model. We can clearly observe a better model generalization. We achieve an accuracy of 73,5% on the test data.

7. **Compare the models from Q4 and Q6 in terms of the training accuracy, validation accuracy, and test accuracy. Explain the similarities and differences (remember that the only difference between the models should be the addition of Dropout layers).**
   Hint: what does the dropout layer do at test time?

We can observe a big difference in training accuracy from the previous model. While the previous model reached an accuracy of almost 90% on the training data, the model introducing the dropout layer shows an accuracy of around 75% on the training data. Though, the validation as well as the test accuracy is higher in the model using the dropout layer. Thus, we the new model has a better generalization and is not overfitting the training data anymore. The reason for that is obviously the dropout layer that randomly ignores nodes (50% in our case) during training phase. Thus, our model doesn't rely on each parameter and turns out to be more robust (better generalization).



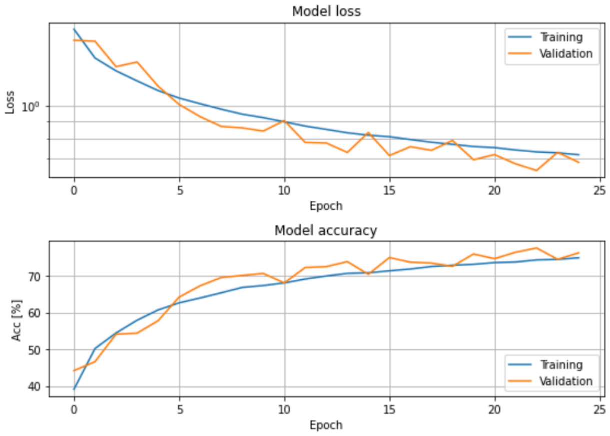

Confusion matrix (test) | Acc=74.39%

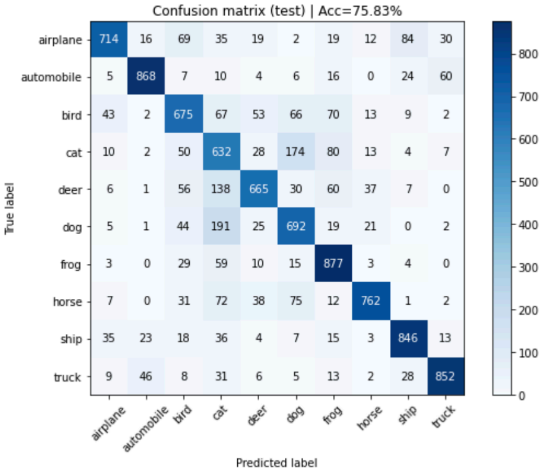
True label \ Predicted label	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	762	14	59	5	17	0	14	8	95	26
automobile	19	863	7	2	4	0	11	1	27	66
bird	39	2	716	23	65	20	87	18	17	13
cat	30	8	110	490	71	77	145	18	30	21
deer	17	1	70	19	718	6	95	50	21	3
dog	14	3	105	177	61	467	77	69	18	9
frog	5	0	44	16	13	4	906	4	7	1
horse	17	2	61	16	49	25	24	790	4	12
ship	49	17	14	1	6	1	4	2	889	17
truck	32	56	5	4	7	0	17	6	35	838

8. **Train the CNN model with added BatchNorm layers and show the evaluation image.**

9. **When using BatchNorm one must take care to select a good minibatch size. Describe what problems might arise if the wrong minibatch size is used.**
   You can reason about this given the description of BatchNorm in the Notebook, or you can search for the information in other sources. Do not forget to provide links to the sources if you do!
   Given the information in the notebook the wrong choice for the size of a minibatch might either end up in slow convergence (large momentum) or in non robust behavior (low momentum). By looking for further information on the web we found this thread (https://datascience.stackexchange.com/questions/18414/are-there-any-rules-for-choosing-the-size-of-a-mini-batch) that was referring to https://arxiv.org/abs/1609.04836 saying that a larger batch size results in lower generalization ability and thus less model quality.

10. **Design and train a model that achieves at least 75% test accuracy in at most 25 epochs. Explain your model and motivate the design choices you have made and show the evaluation image.**




Confusion matrix (test) | Acc=75.83%

True label \ Predicted label	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	714	16	69	35	19	2	19	12	84	30
automobile	5	868	7	10	4	6	16	0	24	60
bird	43	2	675	67	53	66	70	13	9	2
cat	10	2	50	632	28	174	80	13	4	7
deer	6	1	56	138	665	30	60	37	7	0
dog	5	1	44	191	25	692	19	21	0	2
frog	3	0	29	59	10	15	877	3	4	0
horse	7	0	31	72	38	75	12	762	1	2
ship	35	23	18	36	4	7	15	3	846	13
truck	9	46	8	31	6	5	13	2	28	852

Since our previous model already managed to achieve an accuracy of almost 75% we decided to only add dropout layers in our VGG blocks with increasing dropout probability. For the first block we added one dropout layer with 30% dropout probability and for the second block we added one dropout layer with 40% dropout probability. As already mentioned before we are using this VGG block model design since it has shown good results in other experiments.