TBMI26 – Computer Assignment Report

Deep Learning

Authors:

Ludvig Knast ludkn080, Erik Jareman erija971

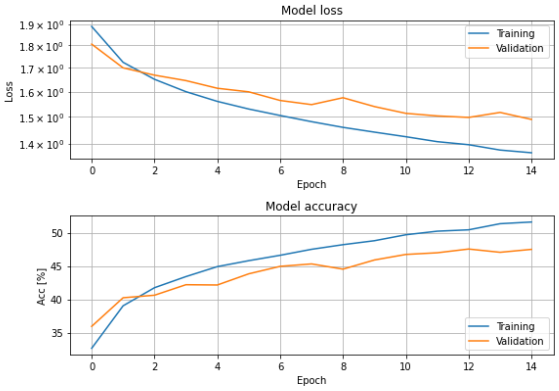
In order to pass the assignment you will need to answer the following questions and upload the document to LISAM. Please upload the document in PDF format. **You will also need to upload the Jupyter notebook as an HTML-file (using the notebook menu: File -> Export Notebook As…)**. We will correct the 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.

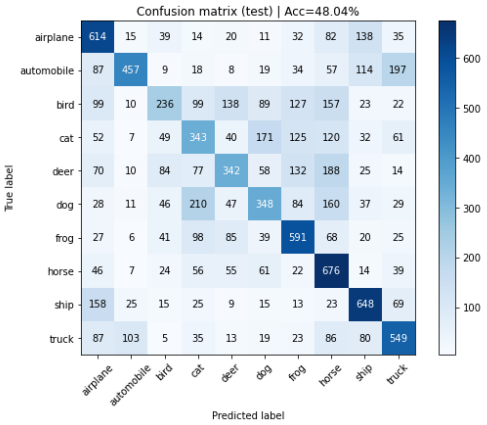
1. The shape of X_train and X_test has 4 values. What do each of these represent?

(sample, height, width, color channel)

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.

One hidden layer with 64 neurons, the hidden layer used the sigmoid activation function and the output layer used softmax. The outlayer had 10 nodes because each training image was labeled one of 10 different classes, so one output node for each class. Test acc = 48%.





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?

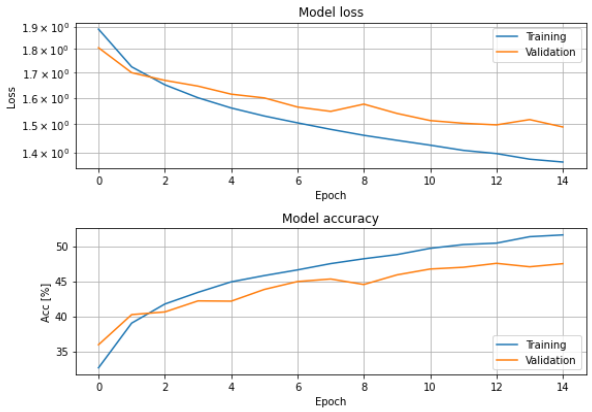
As stated the test accuracy in Q2 is worse than in the first assignment, which is a result of the increased complexity in the CIFAR-10 dataset. Each image in the MNIST consists of 8x8 grey scaled pixels i.e 64 input variables while the CIFAR10 images is the 32x32x3 i.e 3072 variables, so there is some clear increase of complexity. Also, the images in the CIFAR10 dataset consist of more complex objects (and in some cases very similar to each other despite belonging to different classes) which probably makes them harder to classify.

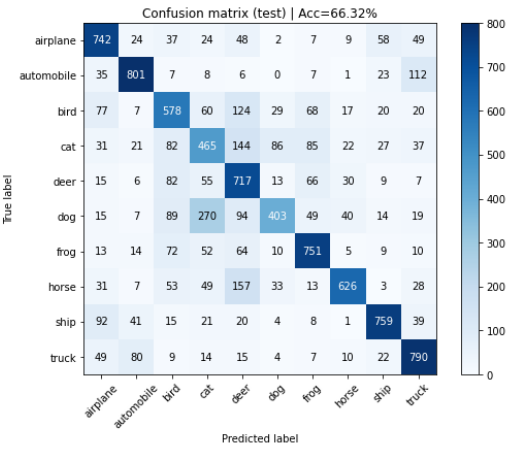
4. Train a CNN model that achieves at least 62% test accuracy. Provide a short description of your model and show the evaluation image.

Below is the overview of the model, from input to output. Number of filters, activation functions, etc can be seen in the code.

```
x_in = Input(shape=X_train.shape[1:])

# === Add your code here ===
x = Conv2D(20, 3, activation='relu')(x_in)
x = MaxPooling2D(pool_size=(3, 3))(x)

x = Conv2D(40, 3, activation='relu')(x)
x = MaxPooling2D(pool_size=(2, 2))(x)

x = Flatten()(x)
x = Dense(64, activation='relu')(x)
x = Dense(32, activation='relu')(x)
x = Dense(10, activation='softmax')(x)
```

Acc = 66%.





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.

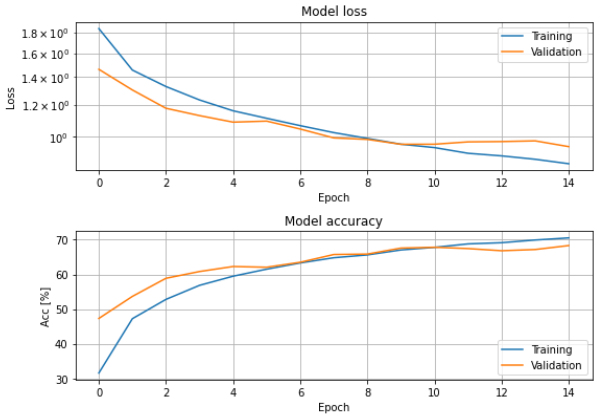
The CNN is learning **features** of the images that then can be used to classify the image while the fully connected model is only blindly looking at each pixel at a time. The kernel that is used for convolution is spatially local and shift invariant which improves the model and its efficiency. The maxpooling lowers the amount of parameters while still possessing important information about each section of the image, this also reduces over-fitting because it provides an abstracted form of the representation.

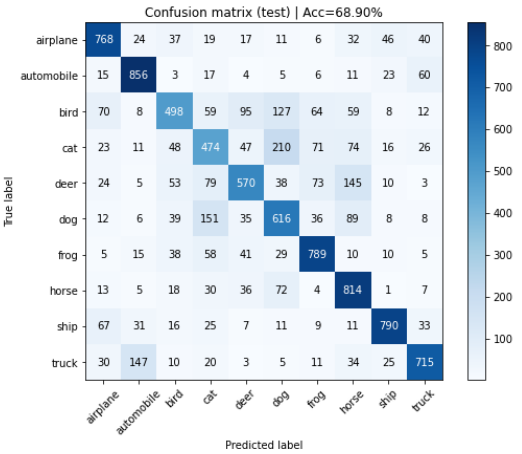
6. Train the CNN-model with added Dropout layers. Describe your changes and show the evaluation image.

Below is the overview of the model, from input to output. Number of filters, activation functions, dropouts, etc can be seen in the code.

```
x_in = Input(shape=X_train.shape[1:])

# === Add your code here ===

x = Conv2D(20, 3, activation='relu')(x_in)
x = MaxPooling2D(pool_size=(3, 3))(x)

x = Conv2D(40, 3, activation='relu')(x)
x = MaxPooling2D(pool_size=(2, 2))(x)

x = Flatten()(x)
x = Dense(64, activation='relu')(x)
x = Dropout(0.2)(x)
x = Dense(32, activation='relu')(x)
x = Dropout(0.2)(x)
x = Dense(10, activation='softmax')(x)
```

Acc = 69%



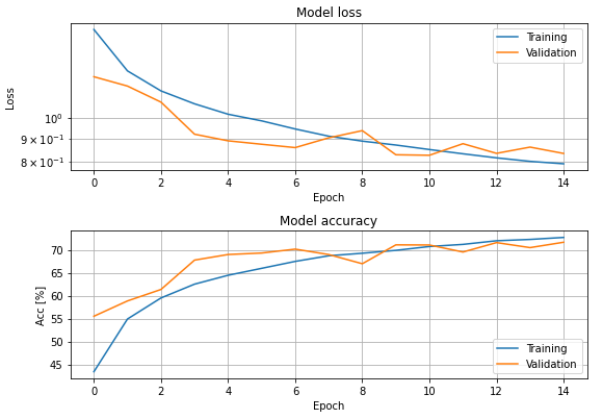


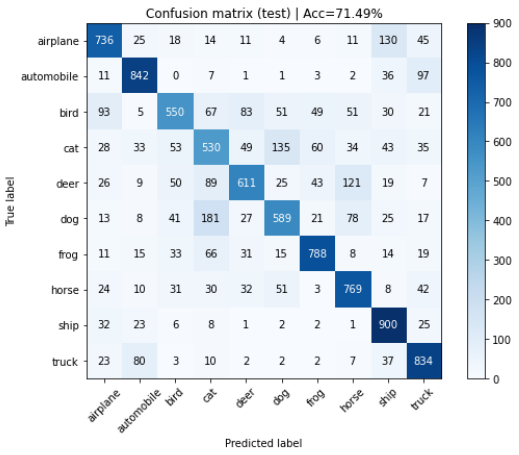
7. Compare the models from Q4 and Q6 in terms of the training accuracy, validation accuracy, and test accuracy. Explain the similarities and differences (remember that the only difference between the models should be the addition of Dropout layers).

Hint: what does the dropout layer do at test time?

The Q4 model has a lot better training accuracy, but this is because the dropout makes the model in Q6 worse during training which makes it more difficult to classify training data. But then when it's time for testing, the Q6 model does not drop out any neurons which in this case then results in a better test acc than Q4, to summarize the dropout makes training a lot harder for the model but in the end results in a model that is performing much better on new test data. The models have approx the same validation acc.

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!

Too small batch size might lead to bad representation of the real sample population which then makes the normalization “equally bad” i.e oscillatory behavior and obstructed convergence . However, too big batch sizes are more computationally heavy and not as robust as smaller ones, i.e risk of overfitting and poor generalization.

https://reader.elsevier.com/reader/sd/pii/S2405959519303455?token=D1576D460F8AD4B08F326764C322EC37C59F680FD820DFFF7435C823F1581AD9E2B18B21DE4985E91C71DAC6EB38D75F&originRegion=eu-west-1&originCreation=20220307134750

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.**

The mode is based on the previously made model from this notebook, with some additional layers and tweaks. Most noticeably we took some inspiration from Alexnet and wanted to try having some convolutional layers without max pooling with some padding, therefore we added one of those layers in the middle of the conv part of the model. We also chose momentum 0.8 for all batch norm layers as it seemed to give a bit better results than 0.75. Another big change is that we removed one dense layer and instead increased the number of neurons in the one we kept, which made it a bit less complex. All in all, it managed to pass the 75% acc mark in 15 epochs without changing that much from the previous model.