TBMI26 – Computer Assignment Reports Deep Learning

Deadline – March 14, 2022

Author/-s: Theodor Emanuelsson (theem089)
Chayan Shrang Raj (chash345)

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?**

The shape of train and test data is given as (50000,32,32,3) and (10000, 32,32,3) respectively. They are images of 10 different objects (classes).

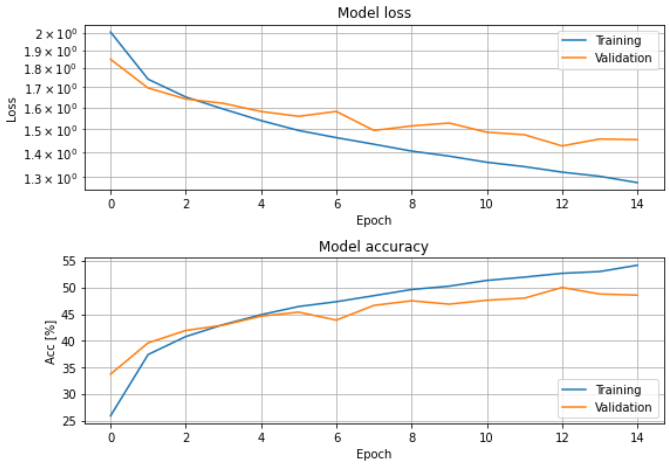
• The first value represents the number of images in each dataset, so training data consists of 50 000 images and test data contains 10 000 images.
• The second value represents each pixel the image is made of in the height of the image. So, the image has 32 rows that cover the vertical structure of the images.
• The third value represents each pixel on the width of the image. So, the image has 32 columns or features that cover the horizontal structure of the image. Combining this dimension and the above dimension makes a 32x32 pixel image.
• The fourth value represents the color channel of the image. Since these pictures are colored and hence, they include Red, Green and Blue (RGB) channels to show true colors in an image. On the contrary, a grayscale image only contains white and black channels.

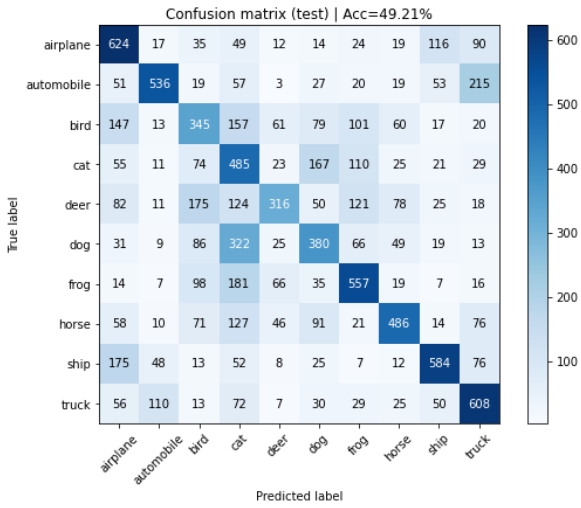
2. **Train a Fully Connected model that achieves above 45% accuracy on the test data. Provide a short description of your model and show the evaluation image.**

We train a Fully Connected model by first flattening the data and proceed with 6 connected dense layers with 64 hidden units each. These layers are all activated using the ReLU activation function. The output of the last dense layers is given to a dense layer with 10 hidden units which utilize the softmax function in order to be able to interpret the results as a probability. Thus, each of the hidden units in the final layer represents the predicted probability for each class label.

On the test data we achieved an accuracy of 49.21%.

Evaluation image model 1:





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?**

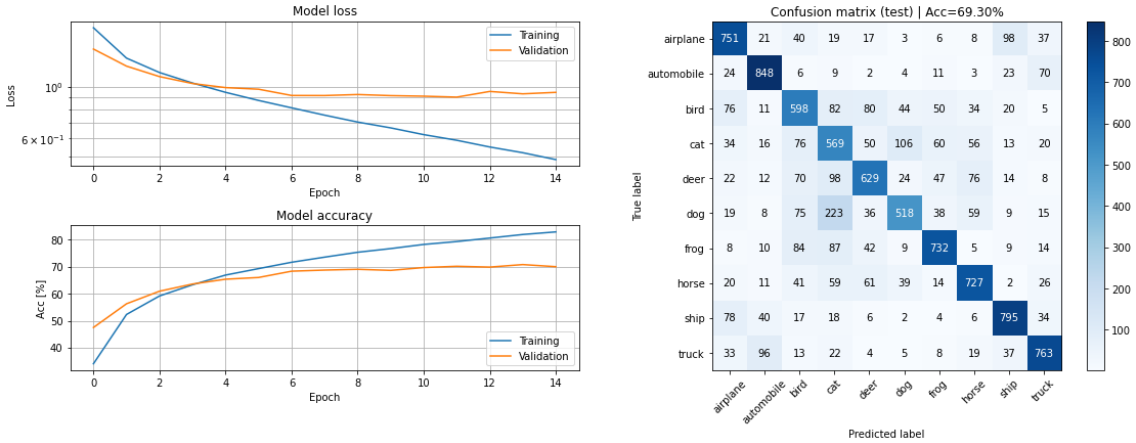
The models in Q2 and in the MNIST dataset are comparable. However, in the CIFAR10 model above the decision boundaries are a bit more complex since we have five more fully connected dense layers. For the MNIST dataset we achieved 96 % accuracy on the test data with only one dense hidden layer, compared to 49.21% for the CIFAR10 model.

This is mostly due to the MNIST dataset being a much simpler dataset and one where flattening is more suited. For one, the MNIST dataset has images that are 8x8x1 and for CIFAR we have images that are 32x32x3. The MNIST dataset has pixel drawn images of numbers and flattening the data essentially means that every pixel is represented as a feature. In addition, the images in this dataset are also centered, which is useful for this type of model, and the image is only represented on a grayscale (black to white). In the CIFAR10 dataset on the other hand, we have three different color channels and classes are much less consistent in terms of pixels. For example, a picture of a cat can be seen from a distance, up close, from an angle and so forth. This creates problems for having each pixel (and even each pixel's different values on the color channels) as an input to the model.

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

Using the idea of convolution, activation and pooling we create a model that achieves 69.3% accuracy on the test data. It has three convolution-activation-pooling blocks, where we start by doing a (3,3) convolution over the image (using 32 filter in the first layer and 64 in the rest) and applying the ReLU activation function. Then we perform a (2,2) maxpooling before going on to the next block. Finally, we flatten the results, apply a 64 hidden unit dense layer which then goes through to a 10 hidden units dense layer with softmax activation.

Evaluation image model 2:



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.**

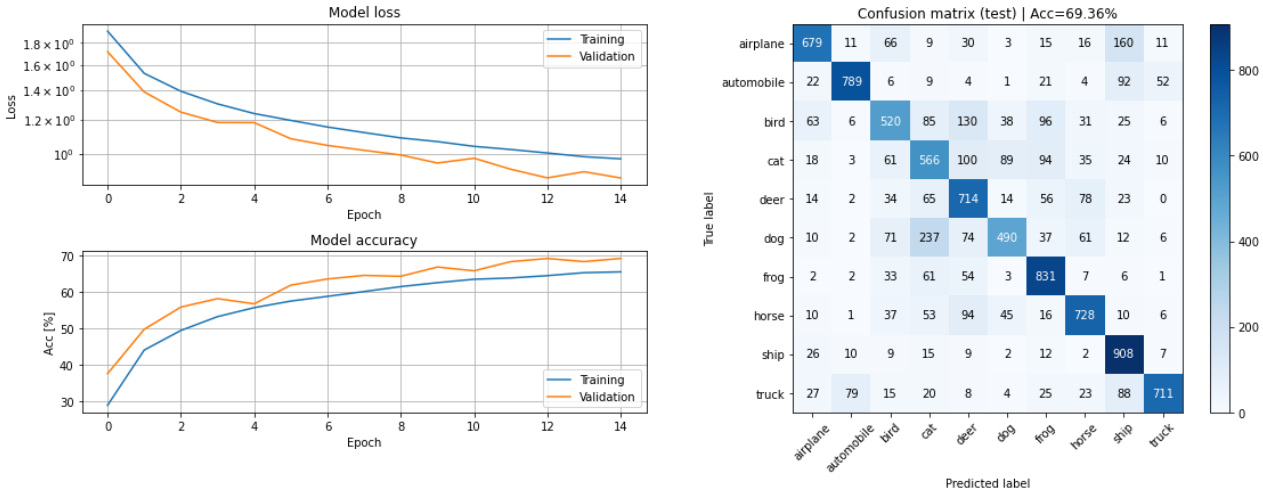
In the Fully Connected model we had 213 962 parameters and achieved 48.6% accuracy on the test data. For the CNN model we only have 73 418 parameters and easily achieve 68.67% accuracy. The fact that this work is due to the idea of convolutions. Instead of having every single pixel as a feature, we can properly utilize the spatial aspects and the color aspects of the data. In our case we use a set number of (3,3) kernels (essentially 3x3x3 arrays) that goes over all the pixels of our 32x32x3 image. A single kernel only has 27 parameters that need to be learned by the model, one for each cell in the 3x3x3 array. We then take the 3x3x3 array and multiply each cell with the corresponding cell in a 3x3x3 cube that goes over the image from top left to bottom right. The results of the multiplication between the 3x3x3 kernel and a 3x3x3 cube going over the image are summed up and stored in a 30x30 matrix. We are, however, using more kernels (or filters as it called in Keras) so our final output will be 30x30x32 if we use 32 filter for a single layer. Using many filters is advantageous since each kernel, when going over the image, can be used to identify a specific feature such as vertical lines or corners that are present in the image. The initial 2D convolutions extract the higher-level features of the image. In the convolutions that are deeper in the network, work on the previous information from the higher-level features and identifies lower-level features within them. After each convolutional layer, we use MaxPooling2D which is also another summarize information from the output of CN layer. We use a pool size of (2,2) meaning that for the 30x30x32 output of a convolutional layer we will take the max value in a 2x2 square going from top left to bottom right with non-overlapping of cells. This condenses the information from the convolutional layer. The resulting array will be 15x15x32 if we have 32 filters.

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

We added a dropout layer with drop rate set to 0.25 after each pooling layer in the model above. This means that on average 25% of the hidden nodes the layer affected by the dropout will be turned off (their output put to zero). We achieved less overfitting to the data which can be seen in the model loss plot below. The loss of the training data and validation data is much more similar to each

other compared to the previous model. Therefore, we see the regularizing effect of adding dropout layers. We even achieved higher a higher accuracy on the test data at 69.4%.

Evaluation image model 3:



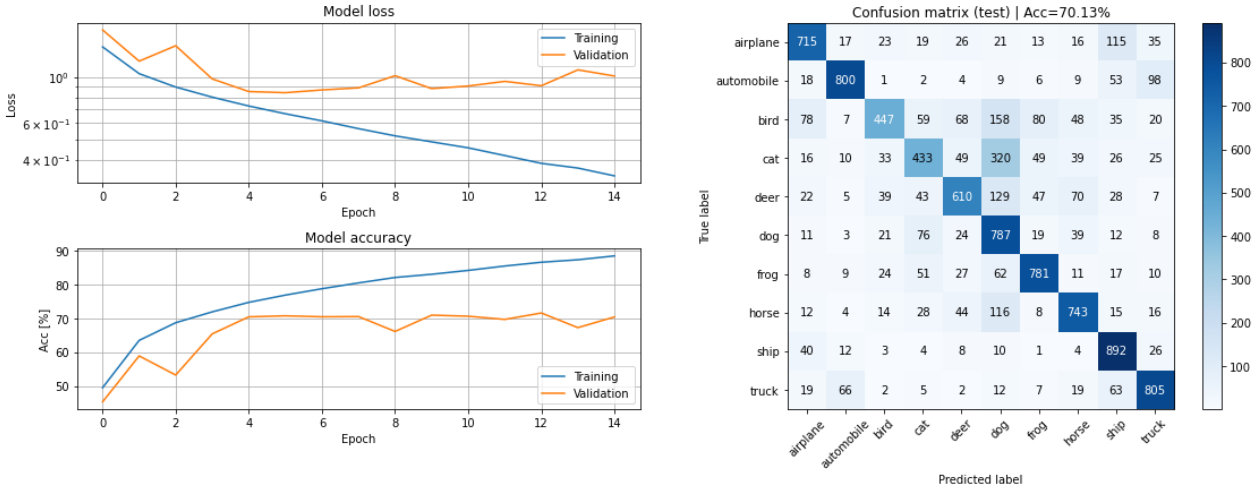
7. Compare the models from Q4 and Q6 in terms of the training accuracy, validation accuracy, and test accuracy. Explain the similarities and differences (remember that the only difference between the models should be the addition of Dropout layers). Hint: what does the dropout layer do at test time?

In the Q4 model training accuracy moves towards 90%, while the validation accuracy seems to converge to somewhere in the 65-70% range. The test accuracy is 69.3%. The training accuracy is significantly higher than the validation and test accuracy, indicating that the model is overfitting to the training data. In the Q6 model on the other hand, training accuracy moves towards about 70%, with validation accuracy moving very closely to the training accuracy over the epochs. It is even typically higher than the training accuracy. There are some, seemingly random, spikes in the validation accuracy that can most likely be due to the random turning off for some hidden units. The test accuracy for this model is 69.36%. In this comparison we can see the effect that the dropout layers had on the training. Instead of overfitting to the training data, adding dropout regularizes the parameters to not rely too heavily on specific parameter values, much like L2 regularization forces parameter values to become lower.

The models are very similar, with the exception that the Q6 model turns off on average 25% of the hidden nodes in a maxpooling layer for every epoch. This only tweaks the optimization of parameters and dropout is not implemented when predicting new unseen data. The Q6 model has other parameter values than the Q4 model due to some hidden units being turned off during backpropagation. As mentioned above, this generally makes parameter values be more spread out in terms of magnitude, as for different epochs, the network will have different hidden units available.

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

Evaluation image model 4:



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!

When performing BatchNorm we are essentially scaling the output for a hidden unit by the minibatch's mean and standard deviation to make it centered around zero and to have a standard deviation of one. The model also learns parameters transform this back to its original values.

The issue with BatchNorm is that we require a large enough minibatch size so that the sample statistics of that minibatch is similar to that of the overall data. If for example minibatch size is set to 2, we cannot assume that the mean and standard deviation of these two observations is similar to that of the whole dataset and we would get unstable results. If we instead choose a larger minibatch, then it is much more likely that these sample statistics are similar to the population statistics. Specifically must be able to rely on the sample in the last epoch to be representative of the population as these are the parameters that will be used in prediction. If the minibatch size is small, then the effect of adding BatchNorm actually has a slight regularizing effect on the parameters.

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.**

For this model we use 2 different types of blocks (Henceforth referred to as B1 and B2). B1 starts with a convolution, then a batch normalization and finally an activation. B2 starts with a convolution, then a batch normalization and activation and finally a max pooling. Our structure is: B1 -> B2 -> Dropout with rate 0.25 -> B1 -> B1 -> B2 -> Dropout with rate 0.45 and then flatten and two dense layers, and a final softmax dense layer. We also start off in the first B1 and B2 with only 32 filters applied in the convolution and increase the filters to 64 and 128 the deeper the model does. We do this so that we can capture basic patterns such as lines, corners, and edges in the initial part of the model further in we increase the filter so that we can capture more complex patterns. We achieved 76.8% accuracy on the test data using this strategy.

Evaluation image model5: