TBMI26 – Computer Assignment Reports Deep Learning

Deadline – March 14 2021

Author/-s:

Martynas Lukosevicius

Alejo Perez Gomez

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?**

Dimensions: (# data instances, # pixels, #pixels, #number of channels RGB)

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.**

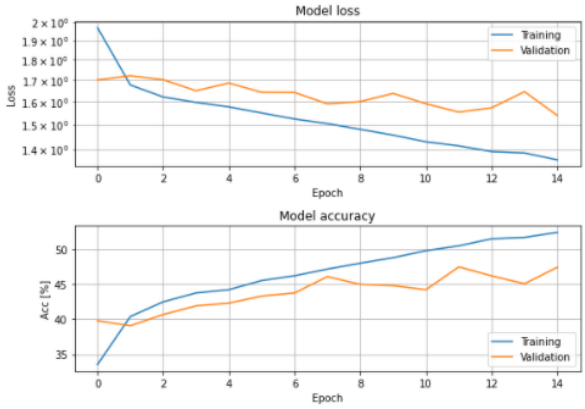
The model (which is outlined below) used 2 hidden layers and 12,523,266 total and training parameters. In the last layer a softmax activation function was used whereas in the hidden layers a tanh function was used. The optimization algorithm was Stochastic Gradient Descent, using a momentum coefficient of 0.9, a weight decay of 1e-6, a learning rate of 0.01 and Nesterov accelerating gradient. The loss function was Categorical Cross Entropy and Accuracy was used to display the performance.

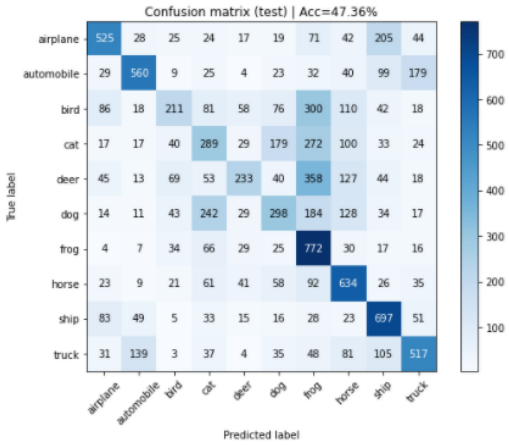
Model: "model_1"

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 32, 32, 3)]	0
flatten_1 (Flatten)	(None, 3072)	0
dense_3 (Dense)	(None, 3072)	9440256
dense_4 (Dense)	(None, 1000)	3073000
dense_5 (Dense)	(None, 10)	10010

Total params: 12,523,266
Trainable params: 12,523,266
Non-trainable params: 0

Below it is shown the learning curves of both loss and performance metrics along with the confusion matrix. The latter showed a bias towards the misclassification of "Frog" when the real categories where "Cat", "Dog" and "Deer".





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?**

   In the first assignment model there was just one hidden layer with 200 Neurons while the current one has 2 hidden layers with 3072, 1000. In Lab 1 we reached 96% of accuracy whereas now we are reaching 47%. The classification problem now is much more difficult because of the dimensionality of date. We are using 3 channels of 32x32 pixels, which will end up like 3072 features when flattening images.

   The dimensionality of data has scaled up as the complexity of the classification in terms of local features and categories to classify. That is to say, it is not the same classifying hand-written digits than animals (with all the complexity that their local structures entail, eg: orientation, size, animal treats).

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

   The model (which is outlined below) used 3 hidden layers and 81,834 total and trainable parameters. The 2 first hidden layers consisted of a Convolutional Layer- Max Pooling Layer sequence. The last hidden layer is a fully connected layer that receives the flattened output of the previous layer. The final output of the output layer goes through a softmax activation whereas the activation function for the hidden ones was RElu.

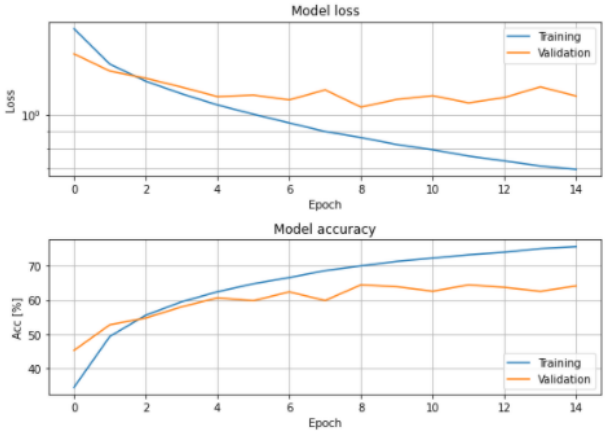
   The optimization algorithm was Stochastic Gradient Descent, using a momentum coefficient of 0.9, a weight decay of 1e-6, a learning rate of 0.01 and Nesterov accelerating gradient. The loss function was Categorical Cross Entropy and Accuracy was used to display the performance.

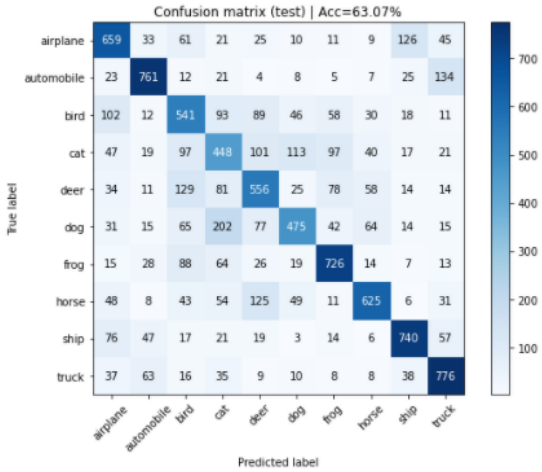
Model: "model_2"

Layer (type)	Output Shape	Param #
input_3 (InputLayer)	[(None, 32, 32, 3)]	0
conv2d (Conv2D)	(None, 28, 28, 32)	2432
max_pooling2d (MaxPooling2D)	(None, 14, 14, 32)	0

conv2d_1 (Conv2D)	(None, 11, 11, 64)	32832
max_pooling2d_1 (MaxPooling2D)	(None, 3, 3, 64)	0
conv2d_2 (Conv2D)	(None, 2, 2, 120)	30840
max_pooling2d_2 (MaxPooling2D)	(None, 1, 1, 120)	0
flatten_2 (Flatten)	(None, 120)	0
dense_6 (Dense)	(None, 120)	14520
dense_7 (Dense)	(None, 10)	1210

Total params: 81,834
Trainable params: 81,834
Non-trainable params: 0





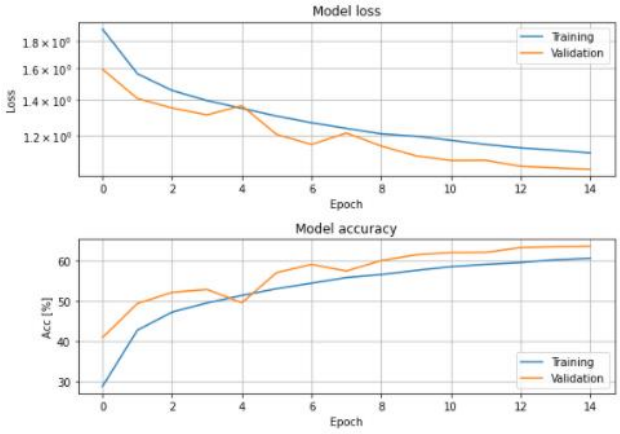
As it can be seen in the confusion matrix, we got better accuracy

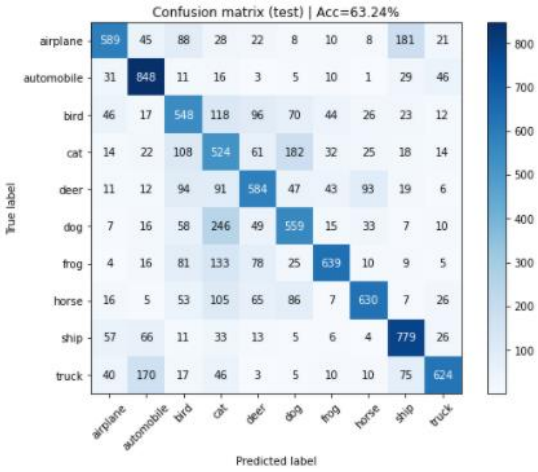
Compare the CNN model with the previous Fully Connected model. You should find that the CNN is much more efficient, i.e. achieves higher accuracy with fewer parameters. Explain in your own words how this is possible.

As it can be seen in the confusion matrix of the last question, we got better accuracy using convolutional layers. Using convolutional operations, we are accounting for local features in images and underlying structures that exists within each category meant to classify.

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

As observable in the confusion matrix, the accuracy increased a tiny little bit. We added dropout layers that turned off neurons with a probability of 0.2. The dropout layers were added after each maxpooling layer.





6. **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 training accuracy in the case of dropout-model decreased aiming not to rely on the best features and involve different sets of kernels to the training. When testing, all neurons are used so the dropout layer doesn't turn neurons off.

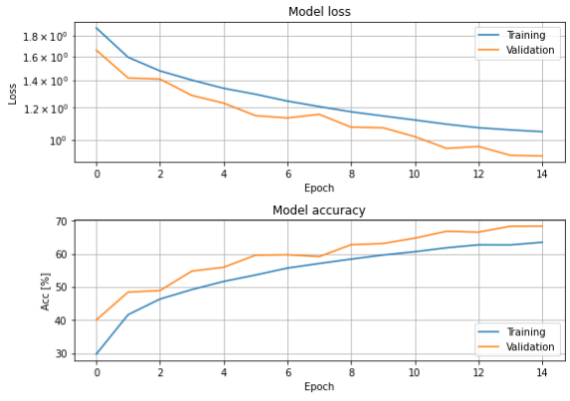
   The training accuracy in the non-dropout-model reaches 0.7559 while the dropout-model gets to 0.6061. That leads us to think the second model increases from training to testing less drastically and prevents overfitting, giving a more generalizing response. In terms of validation accuracy in the training, both models behave alike.

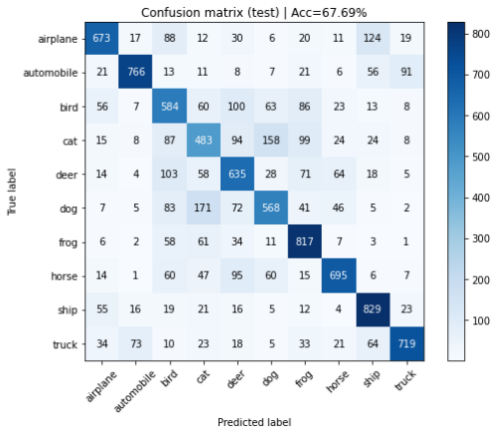
   The test accuracy increased slightly from 0.6307 to 0.6324.

   The validation accuracy in dropout-model resulted in 0.6361 while in the first model was 0.6413.

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

   Output evaluation image after 15 epochs.





8. **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!

   If the latest minibatches used are not representative enough of the dataset in terms of size, it will lead to instabilities when using a low momentum value of the BatchNormalization layer. That is because latest minibatches are taking more into account in the updates. However, if a proper behavior is attained, it will adapt to the data quicker.

9. **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 scheme of the model built is displayed below. As it can be seen, there are multiple stacks of convolution layer + batch normalization + Activation function (RElu) + Max Pooling + Dropout. The dimensions will be downsampled until the data pass through a fully connected layer (Dense function). There result 654,454 total parameters out of which 654,068 were trainable.

   Moreover, for dropouts after convolutions we used from 0.05 to 0.15, and for dropout after dense layer 0.5 was use as the dropout coefficient. The momentum in batch normalization used (0.75) ensured stability in training although it leads to a slower convergence.

   The test accuracy obtained was 78.2%.