Machine Learning

Università della Svizzera italiana

Assignment 2

Hatas Tomas

June 7, 2022

Because of my broken laptop I had to do this assignment in colab google. But, I had an opportunity to check the code in Visual Studio Code in my laptop from the company, where I work. But, I did not want to run the code even in the virtual environment.

Therefore, I did 2 versions of the assignment. One version I downloaded from colab with the py.format and then I modify it in Visual Studio Code. However, I am not sure whether you can run the code without any problem because I did not run it in VS code.

Secondly, I downloaded the second version from colab with ipynb format, which I did not modify.

In other words, I upload as2_Hatas_Tomas_ipynb and as2_Hatas_Tomas_py.

However, I now figure out that I can upload only 1 file. Hence, I only upload as2_Hatas_Tomas_py. If it is not working, I can send you via email the as2_Hatas_Tomas_ipynb.

I hope everything will be working. If it does not work or you have any kind of questions, please contact me: hatast@usi.ch.

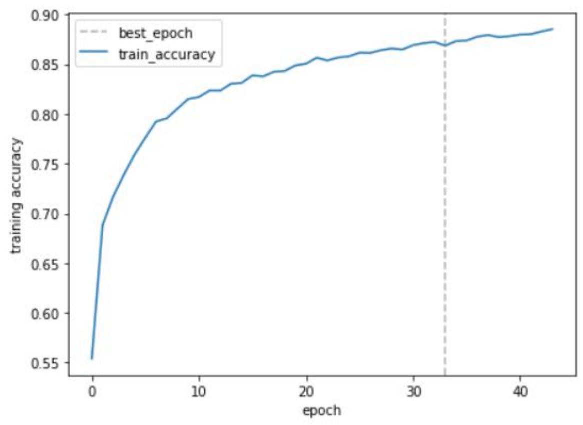
Task 1

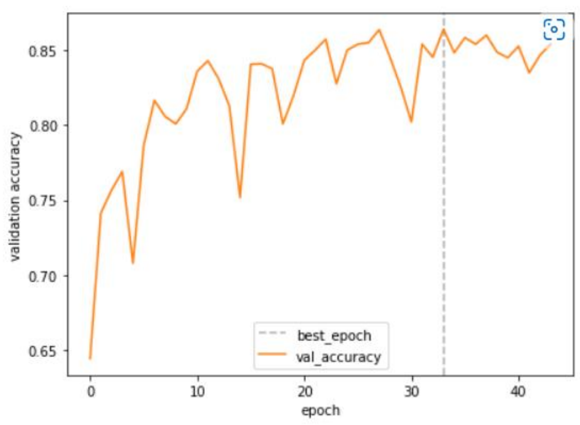
model1.py

In the main function in the first stage I split the dataset into training, validation and test set. Then I normalized the x. Then I used one-hot encoding of the label y for training, validation and test set. After that I was ready to build the convolutional neural network.

I created an object from EarlyStopping with necessary parameters. After that I fit the model with the training data and also validation data using the early stopping on validation accuracy. Then I evaluated model on the test set (scores = model.evaluate(x_test, y_test). I got 0,361 as loss on the training set and accuracy with 0,857.

Below you can see a plot with epochs on the x-axis and with training accuracy and the validation accuracy.





We can see the training terminated at the epoch 44. Because of patience 10, we achieved max validation accuracy at the epoch 34 with 0,8637. It means that there were 10 epochs after the 34th epoch with no improvement on validation accuracy. Under these circumstances/parameters, the epoch 34 achieved the highest validation accuracy.

After that I saved the model.

For more detailed description see model1.

run_task1.py

After loading model and predicting on testing set, I get accuracy 0.9044. This result is great.

However, I needed to modify the y_pred_task1. The point is that the y_pred_task1 provides probabilities of each classes for the image. Therefore, I needed to find a max value in each row which is my best guess for the image. For example, I got the result for the one image: [0.8773714 0.08719977 0.03542893]. It means that the image is airplane with the probability of 0.8773714, or automobile with the probability of 0.08719977 or bird with the probability of 0.03542893. Based on the function max value I found out that the image is airplane with the highest probability and hence I convert the value into 1. All other probabilities will be converted into 0. In this way I create new_y in the same shape like y_pred_task1 and I fill it with the 0s or 1s.

After that I just compared the y_test dataset with the created new_y binary numbers. The result is just great.

Task 2

model2.py

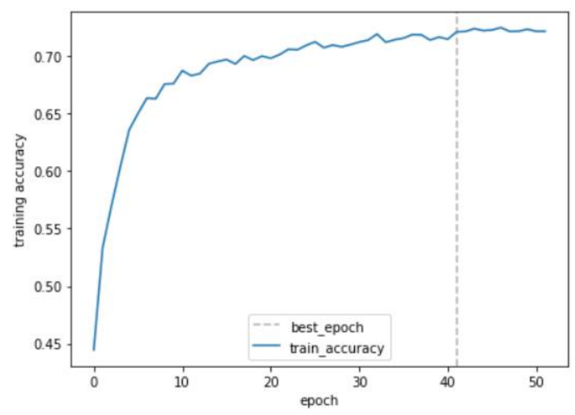
In the main function in the first stage I split the dataset into training, validation and test set. Then I normalized the x. Then I used one-hot encoding of the label y for training, validation and test set. This is the same procedure like in the task 1.

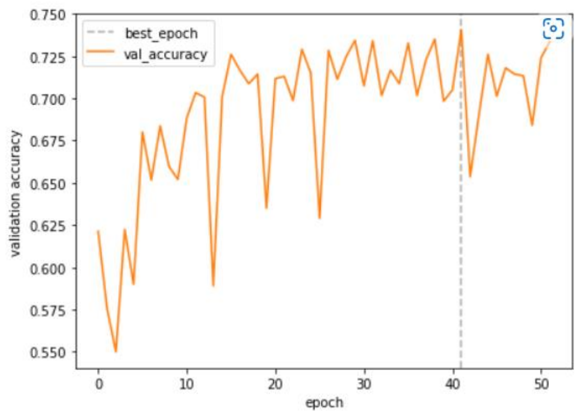
In the 3rd section I had to flatten the images into 1D vectors. I achieved this by using tf.reshape.

Then I build the Feed Forward Neural Network according to the instructions.

Moreover, I compile the network with the given parameters and I also created an object from EarlyStopping with necessary parameters. After that I fit the model with the training data and also validation data using the early stopping on validation accuracy. Then I evaluated model on the test set (scores = model.evaluate(x_test, y_test). I got 0,658 as loss on the training set and accuracy with 0,7317.

Below you can see a plot with epochs on the x-axis and with training accuracy and the validation accuracy.





We can see the training terminated at the epoch 52. Because of patience 10, we achieved max validation accuracy at the epoch 42 with 0,7407. It means that there were 10 epochs after the 42nd epoch with no improvement on validation accuracy. Under these circumstances/parameters, the epoch 42 achieved the highest validation accuracy.

After that I saved the model.

For more detailed description see model2.

run_task2.py

After loading model and predicting on testing set, I get accuracy 0.8211. This result is pretty good.

However, I needed to modify the y_pred_task1. First, I needed to flatten the images of the x_test_set into 1D vectors. I achieved it by using tf.reshape. This reshaped x test set was a parameter into predict function. After that, I followed the same procedure like in the task 1. The y_pred_task1 provides probabilities of each classes for the image. Therefore, I needed to find a max value in each row which is my best guess for the image. For example, I got the result for the one image: [0.8773714 0.08719977 0.03542893]. It means that the image is airplane with the probability of 0.8773714, or automobile with the probability of 0.08719977 or bird with the probability of 0.03542893. Based on the function max value I found out that the image is airplane with the highest probability and hence I convert the value into 1. All other probabilities will be converted into 0. In this way I create new_y in the same shape like y_pred_task1 and I fill it with the 0s or 1s.

After that I just compared the y_test dataset with the created new_y binary numbers. The result is pretty good.

Comparing results obtained in T1 and T2

The accuracy model task 1 is: 0,9044 and the accuracy model in task 2 is 0,8211. The procedure how to build these models were same except of flatten the images into 1D vectors. This is the reasons why the accuracy model in task 2 was lower than in task 1. As a matter of fact, by flattening the x of training, validation and test sets, we lost 2 dimensions. These 2 dimensions were valuable information. This loss of information leads to the lower accuracy of model in task 2.