8P361 Project Imaging - BIA Group 4 | Assignment 4
Myrthe van den Berg, Rick van Bergen, Nils Bodenstaff, Cris Claessens and Jeanine van Ingen

Exercise 1
When does transfer learning make sense? Hint: watch the video. Does it make sense to do transfer learning from ImageNet to the Patch-CAMELYON dataset?

As mentioned in the video, transfer learning (using a model from task A for a model for task B) makes sense when task A and B have the same input x and when task B has a relatively smaller dataset than task A to train the parameters. When a model is trained to perform, for example, image recognition on a large dataset, only a few adaptations need to be made to retrain the model for your own purpose with a small dataset. Lastly, transfer learning makes sense when low-level features from task A could be helpful for learning task B [1].
So, when comparing the data from ImageNet to the Patch-CAMELYON dataset, it does make sense to do transfer learning from ImageNet to the Patch-CAMELYON dataset, since both inputs are images. Thereby is the model from ImageNet trained on much more data than available for the Patch-CAMELYON challenge and both tasks are essentially based on object recognition, so this low-level feature can be helpful.

Exercise 2
Run the example in transfer.py. Then, modify the code so that the MobileNetV2 model is not initialized from the ImageNet weights, but randomly (you can do that by setting the weights parameter to None). Analyze the results from both runs and compare them to the CNN example in assignment 3.

After running the original script of transfer.py and the script was modified by setting the weight parameter to None (as mentioned in the question). The results of these two models and the CNN model from assignment 3, are shown in table 1.

Table 1 - Final loss and accuracy values of the original model, the model with random initialization and the CNN model of assignment 3

	training loss	training accuracy	validation loss	validation accuracy
Original	0.2230	0.9139	0.2466	0.8675
Modified	0.3761	0.8289	0.6937	0.5250
CNN	0.2569	0.8942	0.2586	0.8831

Without initializing with the ImageNet weights, the modified model shows a higher loss and a lower accuracy compared to the model which uses the original ImageNet weights. Especially in the validation loss, the difference between the original model and the modified model is large, the differences are around 0.4. Another remarkable note is that the accuracy of the validation data is lower than the loss of the validation data.
When comparing the ImageNet models to the CNN model of assignment 3, it can be seen that the values of the CNN model lie in between the values of the original ImageNet model and the modified ImageNet model.
In figures 1 and 2 the course of the accuracy and loss for both training and validation data are shown for all models.

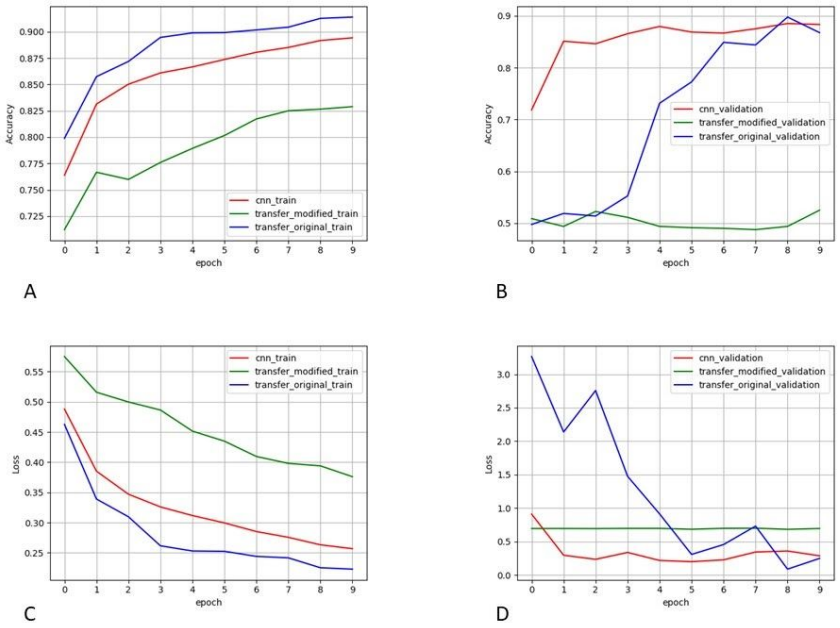

Figure 1, The accuracy of the train data (A), and of the validation data (B). The loss of the train data (C), and of the validation data (D). The graphs are shown for the original, the modified and the CNN model.

Figure 1 shows that in comparison with the original and CNN model, the modified model with the random initialised weights, has the weakest performance. This model overfits the training data, since the validation accuracy and loss stay practically constant during the iteration process. The training accuracy and loss of the original transfer model are better than the CNN model, because of the higher accuracy and the lower loss. However, for the validation data, the CNN model has a higher accuracy and lower loss at the beginning of the iteration process which indicates that for the validation data the CNN model is better in classifying the right category. The CNN model also shows a more smooth curve than the original transfer model.

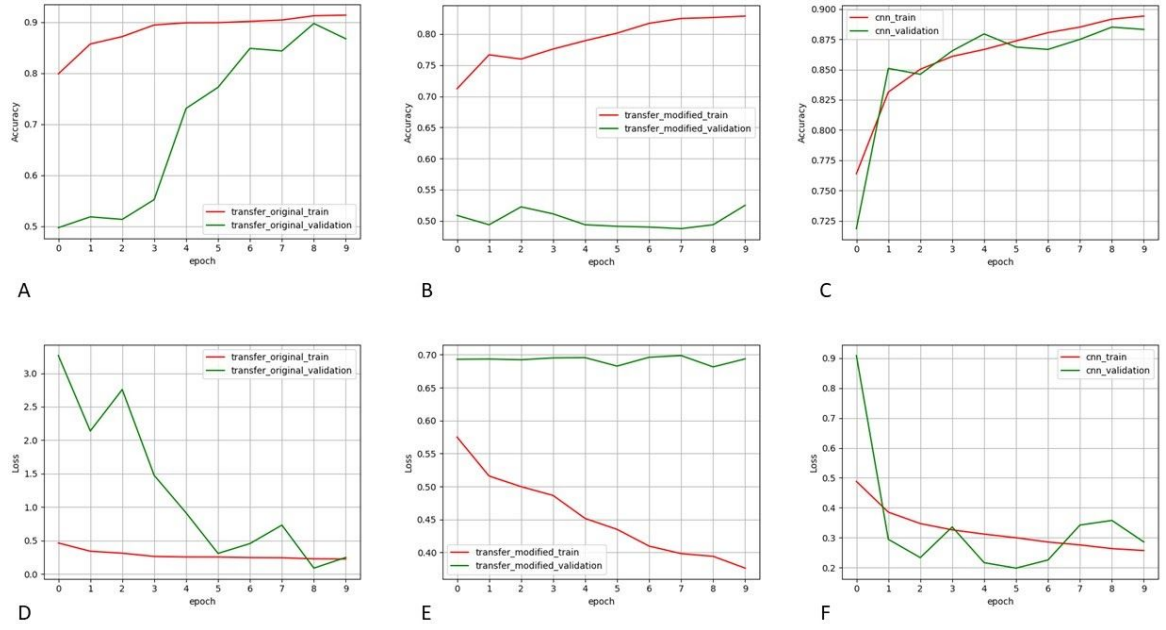

Figure 2, The accuracy and loss curves of the train and validation data of the original model (A,D), the modified model (B,E) and the CNN model (C,F).

Figure 2 also confirms that for higher iterations the modified model overfits because the differences between the accuracy and loss values of the training and validation data in 2B and 2E is increasing. For the CNN model the accuracy (2C) of both training and validation data are pretty similar from the beginning of the process. The loss curve (2F) shows that the validation loss oscillates around the training loss values. At the end of the process, for the original transfer model, the accuracy (2A) and loss (2D) of both training and validation data end up around the same value. However, the validation accuracy is decreasing and the loss increasing from epoch 8 to 9, so it is possible that overfitting can occur after 10 epochs or more.

Exercise 3

The model in transfer.py uses a dropout layer. How does dropout work and what is the effect of adding dropout layers to the network architecture? What is the observed effect when removing the dropout layer from this model? Hint: check out the Keras documentation for this layer.

Problems that occur within Deep Neural Networks are overfitting and that large networks are slow which makes it difficult to deal with overfitting by combining the predictions during the test time. A dropout layer addresses this problem. It is based on randomly dropping units during training. This prevents units from co-adapting excessively. During training, the Keras Dropout function samples from an exponential number of networks that consist of the remaining units. At test time, the effect of averaging the predictions of all these thinned networks can be approximated by using a single unthinned network that has smaller weights [2].

Table 2 - Final loss and accuracy values of the original model and the model without the dropout layer

	loss	accuracy	validation loss	validation accuracy
Original	0.2230	0.9139	0.2466	0.8675
Without Dropout	0.2082	0.9192	0.3675	0.8087

When comparing the final loss and accuracy values of a model with dropout and a model without dropout (Table 2), only the validation loss shows a rather big difference around 0.12. This is in accordance to the theory since the dropout layer is supposed to prevent overfitting. In figures 3 and 4 the curves of the accuracy and loss for both training and validation data are shown for all models.

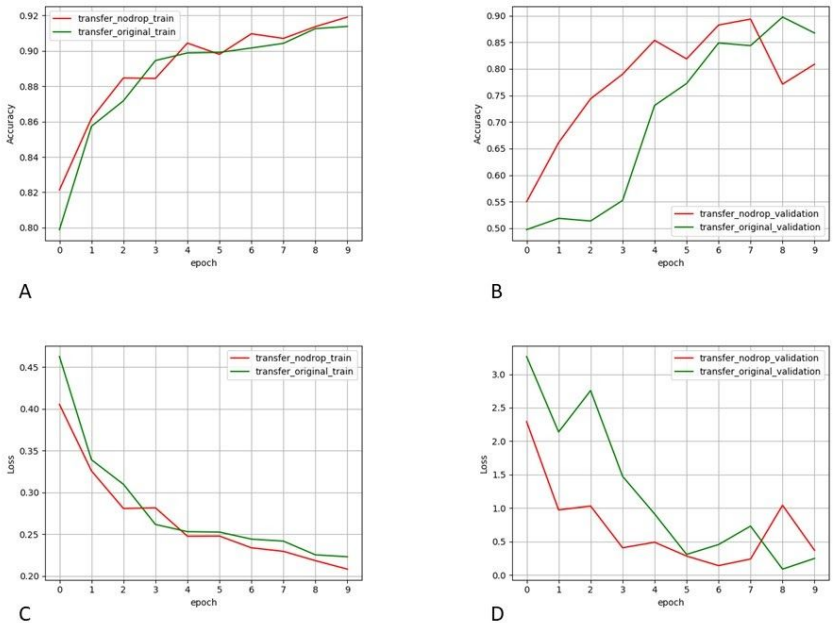

Figure 3, The accuracy and loss curves of the train data (A,C) and validation data (B,D) for the original model and the model without dropout

When looking at figure 3, the expectation that the layer with dropout constructs the best model is validated, because after 7 epochs the validation accuracy from the original model drops while the accuracy resulting from the drop layer model keeps rising. Similar behaviour is seen in the validation loss of the model since the loss increases after 7 epochs. This all while the training loss and accuracy keep improving. These signs hint at overfitting. Therefore a closer look at the validation and training curves is needed as can be seen in figure 3.

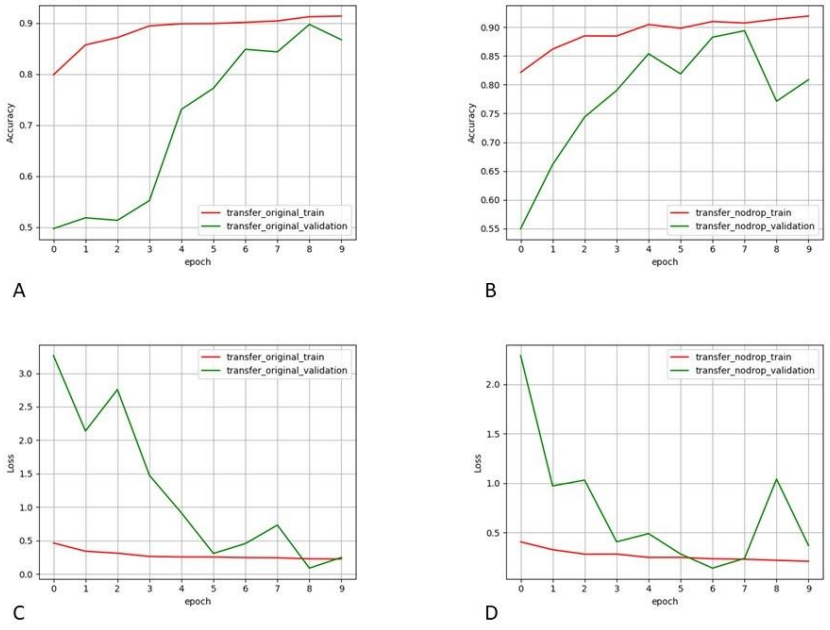

Figure 4, The accuracy and loss curves of the train and validation data of the original model (A,B) and the model without the dropout layer (C,D)

When looking at figure 4, the suspicion that overfitting occurs when the dropout layer is removed is confirmed. Namely when looking at the 3C it is visible that the difference

between the train and validation accuracy increases significantly. This is also visible for 3A yet not as drastic as in 3C. Thus it can be concluded that the dropping layer significantly reduced the possibility of overfitting.

References

[1] Deeplearning.ai. (2017, August 25) Transfer Learning (C3W2L07) [Video file] Retrieved from https://www.youtube.com/watch?v=yofjFQddwHE

[2] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15, 1929–1958.