Assignment 4 group 1

By: J.N.A. Bergmans, M. Boone, J.N.A. van Mook, J.F. Peeters

Exercise 1

Transfer learning can be used when you want to train a neural network for a certain task B but do not have a lot of data available for this task. However, you do have another task A for which you have more data and from which the low level features can be useful for learning task B. It is important that task A and task B have the same input type. For example: images and images are the same input type, but images and sounds are not. It does not make sense to pre-train a neural network on a database that is significantly smaller than the database for the task you want the model to perform. You can train a model for task B by taking a 'pre-trained' model that has been trained for task A, replace the output layer by randomly initialized weights and then finetune the weights of the layers. (DeepLearning.AI [DeepLearningAI], 2017) ImageNet has 14.2 million images (Imagenet, 2021), whereas Patch-CAMELYON has 327.680 (Veeling, 2020). ImageNet has a significantly larger image database. Therefore it makes sense to do transfer learning from ImageNet to the Patch-CAMELYON dataset. However, due to the large differences between the ImageNet and the Patch-CAMELYON datasets it is important that not only the output layer is replaced, but that the weights of a number of hidden layers are finetuned as well.

Exercise 2

In this exercise, 3 models are compared. The first one is a transfer model (model 1) where the model weights are initialized using the ImageNet weights. In other words, pre-training of the model on ImageNet is performed. The second model (model 2) is a model with random initialization. The third model (model 3) is a convolutional neural network model. This model consists of two convolutional layers that are each followed by a max pooling layer, and finally by a fully connected layer with 64 neurons. A kernel size of 3x3 is used for the convolutional layer, and a pool size of 4x4 for the max pooling layer. It was trained using a lower number of epochs due to the higher amount of time per epoch. As a result, the accuracy curve for this model ends after less epochs compared to the other models.

The layers in all models use ReLu activation, with exception of the output layers. For the output layers, sigmoid activation is used. For the evaluation of the model performance, the validation accuracy was used. This was done since the used dataset did not contain a labelled test set. The validation dataset provided the best alternative for evaluating model performance.

When looking at the training set, the accuracy was highest for model 1 (0.9139), followed by model 3 (0.8477) and model 2 (0.8393). When looking at the validation set, the accuracy was highest for model 3 (0.8600), followed by model 1 (0.8575) and model 2 (0.5163).

Since the validation dataset was not used for training of the model but only for model selection, the validation accuracy can be seen as the best metric to evaluate model performance in the absence of test accuracy. From the validation accuracy, we can conclude that the convolutional neural network model (model 3) performs a bit better than the transfer model and much better than the randomly initialized model. Moreover, the transfer model has much higher performance than the model with randomly initialized weights.

Looking at the accuracy plot (Figure 2), it seems to be the case that the performance of model 3 does not increase with training. It's final value (0.5163) is lower than it's starting value (0.535). For model 1, the validation accuracy seems to have stopped increasing after 10 epochs. The validation

accuracy is not yet decreasing significantly, so overtraining has not yet occurred. The validation and training accuracy of model 3 are still increasing after 3 epochs. More training seems to be needed to improve the model performance.

Exercise 3

The Dropout layer sets input units to 0 randomly. The inputs that are not equal to 0, are scaled up by 1/(1-rate) so that the sum over all inputs is unchanged . Where 'rate' is the number of input units to be dropped (*Dropout Layer*, n.d.). Adding a Dropout layer to the neural network helps prevent overfitting because it prevents units from co-adapting too much (**Error! Reference source not found.** (Srivastava et al., 2014). However, applying dropout to a neural network typically increases the training time. This is caused by noisy parameter updates coming from when each training step tries to train a different architecture (Ko et al., 2017).

When running transfer.py without the dropout layer (model 4), the computational time was 29 minutes and 12 seconds. While the computational time with dropout layer was 19 minutes and 16 seconds (Figure 2: The accuracy curves of the training and validation set of model 1, 2, 3 and 4. and Table 1: An overview of the models used in this assignment including a description of the model, the training and validation accuracy and the computation time.). This contradiction with the literature stated earlier can be a result of running additional other programs while running Spyder during the training of the model without the dropout layer than during the training of the model with dropout layer.

The accuracies of the training set with dropout layer and without dropout layer are 0.9139 and 0.9206 respectively. The accuracies of the validation set with dropout layer and without dropout layer are 0.8575 and 0.7936 (Table 1: An overview of the models used in this assignment including a description of the model, the training and validation accuracy and the computation time.). To conclude, the dropout layer in these models does not affect the accuracy of the training set significantly. However, it does affect the accuracy of the validation set, since this is significantly higher for the model with dropout layer than for the model without dropout layer. This corresponds to the found literature that the dropout layer helps preventing overfitting.

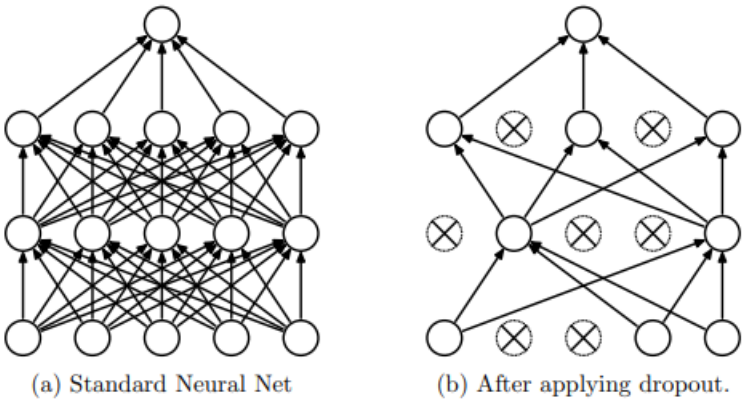

Figure 1:Dropout Neural Net Model. Left: A standard neural net with 2 hidden layers. Right: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped (Srivastava et al., 2014) .

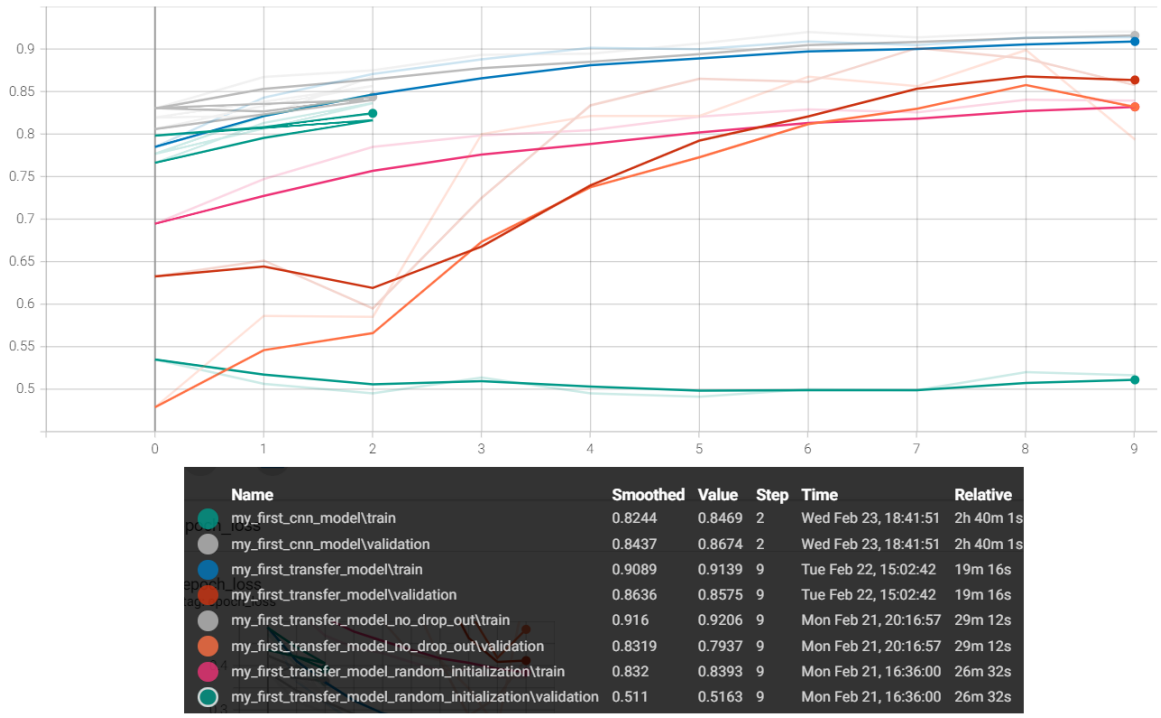

Figure 2: The accuracy curves of the training and validation set of model 1, 2, 3 and 4.

Model	**Description**	**Training accuracy**	**Validation accuracy**	**Computation time**
Model 1	Transfer model with initialized weights using the ImageNet weights.	0.9139	0.8575	19 min 16s
Model 2	Model with random initialization	0.8393	0.5163	26 min 32 s
Model 3	Convolutional neural network model	0.8477	0.8600	16 min 6 s
Model 4	Transfer model with initialized weights using the ImageNet weights with a dropout layer	0.9206	0.7936	29 min 12 s

Table 1: An overview of the models used in this assignment including a description of the model, the training and validation accuracy and the computation time.

References

DeepLearning.AI [DeepLearningAI]. (2017, 25 augustus). Transfer Learning (C3W2L07) [Video]. YouTube. https://www.youtube.com/watch?v=yofjFQddwHE

Dropout layer. (n.d.). https://keras.io/api/layers/regularization_layers/dropout/

Ko, B., Kim, H.-G., & Choi, H.-J. (2017). Controlled dropout: A different dropout for improving training speed on deep neural network. *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 972–977. https://doi.org/10.1109/SMC.2017.8122736

Srivastava, N., Hinton, G., Alex Krizhevsky, & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, *15*(56), 1929–1958.

Veeling, B. (2020, 28 april). GitHub - basveeling/pcam: The PatchCamelyon (PCam) deep learning classification benchmark. GitHub. Geraadpleegd op 9 maart 2022, van https://github.com/basveeling/pcam