Medical Image Classification

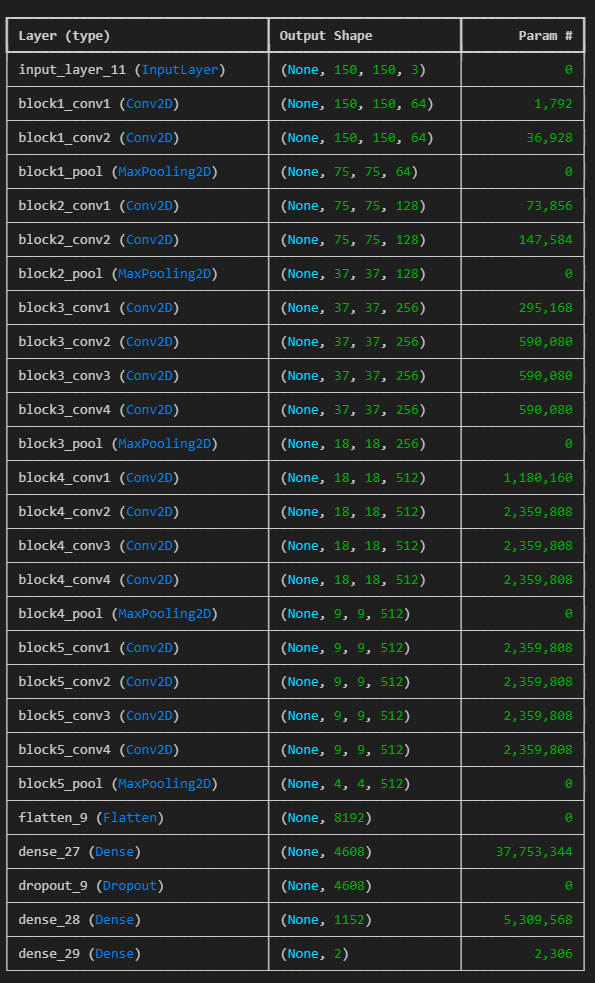
The 2 neural networks for the task will be described one after the other in order to improve the readability of the report instead of constantly comparing them in the same text.

Both networks use the same data set for pneumonia found on Kraggle **(ADD CITATION HERE)**

The naming of the dataset subfolders has been altered slightly but they retain their original structure of 3 folders – training, testing and validation. There are a total of 5863 files in the dataset stored in the form of JPEGs which eliminates the need to sanitize the data.

# Network №1

For the first network I decided to use the Visual Geometry Group (VGG-19) architecture. It consists of 19 layers – 16 convolutional and 3 fully connected. One of the reasons for my choice was the simplicity and modularity of the baseline model.

My approach consisted of creating a first-iteration model using the pretrained weights available from the base model and training it on the training data in my dataset. After that I would save the evaluate the model against the testing and validation data and save the weights. Furthermore, this process is repeated 2 more times by utilizing the saved weights from the previous iteration. For initial runs I utilized parameters with the following values for all 3 iterations:

* 1 epoch with 50 steps
* Plateau learning rate reduction with a factor of “0.5” and patience of “3”
* Stochatic Gadient Descent with a leaning rate of “0.0001” and momentum of “0.01”
* Image size of 150x150

The images were loaded without any changes for the validation and testing data. The training data was loading with the following parameters:

* Horizontal\_flip = 0.4
* Vertical\_flip = 0.4
* rotation\_range=40,
* shear\_range=0.1
* width\_shift\_range=0.4
* height\_shift\_range=0.3

Image 1. Model summary during initial testing

During the first testing period the network achieved an accuracy of “0.5000” against the testing data and “0.6308” against the validation data. Which only slightly improved to “0.6420” against the validation data due to using the same parameters everywhere.

For my next iteration I decided it increase the image size to 224x224 to match the designed size for VGG-19. This change did not change the evaluation accuracy which was expected. On the other hand, it increased the average accuracy when fitting the training data to the model from “0.6201” to “0.8021”. However, this increased the execution time by a factor of 3, which highlights one of the drawbacks of the VGG-19 architecture – very taxing on hardware in case of bigger datasets.

For the next iteration I decided to increase the batch size of the training data generator to 64 (from 32) and the epochs to the following:

* Model #1 is increased to 15 epochs with 250 steps per epoch
* Model #2 is increased to 10 epochs with 200 steps
* Model #3 is increased to 5 epochs with 200 steps

All models have had their patience increased to 8 for early stopping and 5 for rate reduction on plateau. These changes resulted in the following results:

* Model #1 – 0.8012 accuracy against testing data and 0.75 accuracy against validation data
* Model #2 – 0.7804 against testing data and 0.625 against validation data
* Model #3 – 0.7916 against testing data and 0.625 against validation data

Such results most likely indicate overfitting. In order to combat this I made changes to the training image generation in the following ways:

* Reduce the rotation range from 40 degrees to 20 degrees
* Reduce the width and height shift changes to 0.1
* Disable vertical flipping
* Add brightness range between 80% and 120% of the original

The changes resulted in 0.75 validation and 0.8493 testing accuracy.  
The next iteration consisted of changing the momentum of the final model to be “0.8”. This change lead to no increase in validation accuracy but increased testing accuracy to 0.889 which means that the model is still overfitting. In order to combat this I decided to lower the steps per epoch in the final model down to 100 which brought the validation accuracy to “0.8750” and the testing accuracy to “0.9208”