Deep Learning Project

MASTER DEGREE PROGRAM IN DATA SCIENCE AND ADVANCED ANALYTICS

Using Deep Learning to Classify Pneumonia Disease in Human Chest X-ray Images

Diogo Peixoto - 20210993
João Costa - 20211005
Gonçalo Gomes - 20211007
Gabriel Avezum - 20210663
Danilo Arfelli - 20211296

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

1. Introduction

Deep learning is a subset of machine learning, which is essentially based on artificial neural networks. These neural networks attempt to simulate the behavior of the human brain, allowing it to "learn" from large amounts of data [1].

In this project we used deep learning methods applied in the field of medical imaging, more specifically radiology. Chest x-rays are one of the most requested and used exams by healthcare professionals, to assess the presence of thoracic diseases, due to its low-cost and non-invasive nature [2]. Deep learning can, therefore, be an important tool in this area not only by providing a second opinion as a way of validating diagnostic interpretation by humans but also as an attractive way to increase the productivity and efficiency in the interpretation allowing clinicians to better manage their time and resources. Several studies comparing human and deep learning already point to equivalent performance of both in diagnostic performance [3].

The dataset used on this project was downloaded from Kaggle [4], with the images of chest x-rays being labeled as normal status, viral pneumonia, and bacterial pneumonia. In the original project available in Kaggle the author built a deep learning model to classify the images for a binary outcome: normal status or pneumonia (virus or bacteria). However, because the original data granulation allowed further distinction for the pneumonia conditions, we decided to go further and create a multi-label deep learning model with a single-label multiclass outcome: *normal status*, *viral pneumonia*, or *bacterial pneumonia*.

Figure 1 - Illustrative Examples of Chest X-Rays in Patients with Pneumonia. The normal chest X-ray (left panel) depicts clear lungs without any areas of abnormal opacification in the image. Bacterial pneumonia (middle) typically exhibits a focal lobar consolidation, in this case in the right upper lobe (white arrows), whereas viral pneumonia (right) manifests with a more diffuse "interstitial" pattern in both lungs [2].

2. Dataset

The data set contains 5859 chest x-rays images. These chest x-rays images, in anterior-posterior projection, were selected from retrospective cohorts of pediatric patients of one to five years old from Guangzhou Women and Children's Medical Center, Guangzhou [4]. The images were uploaded to *google drive,* and then *google colab* was used to develop the model itself.

From the original dataset folder, we split manually the images into three different folders: *train, test and validation.* The division is summarized on the table below:

	Train	Test	Val	Total
Normal	1000	300	283	1583
Virus	1001	200	294	1495
Bacteria	2000	300	480	2780
Total	4001	800	1057	5858

Table 1 – Number of images by training, test and validation folder

3. Model development

In the first step of the model building process, we have decided to build from scratch the convolutional neural network (CNN) model. During this step, we have maintained constant parameters such as: **stride**:1 ; **filter**: 3x3; **number of filters**: 32/64/128; **train/val batch size**: 16; **epochs**: 30; **step_per_epoch**:250; **validation_steps**:66; **optimizer**: 'rmsprop", **max pooling**:(2,2), **activation**: 'relu'. The remaining parameters and topologies considered tested during this process are summarized in table 2. All the graphs, including accuracy and loss training results, are in the appendix.

Model Num.	Parameters						Results				
	Architecture	Input_shape	Dropout	Data Aug	Geome. Transf	RGB	Time (HH:mm)	Best Acc Train	Best Acc Val	Final Acc Train	Final Acc Val
1	2 conv	(150,150)	no	no	no	no	0:35	0.998	0.745	0.998	0.686
2	2 Conv	(150,150)	yes-0.5	no	no	no	0:32	0.997	0.752	0.994	0.705
3	2 Conv	(150,150)	yes-0.5	yes	yes	no	0:35	0.752	0.763	0.746	0.714
4	2 Conv	(80,80)	yes-0.5	yes	yes	no	0:32	0.744	0.761	0.738	0.716
5	3 conv	(80,80)	yes-0.5	yes	yes	no	0:33	0.753	0.808	0.748	0.805
6	2 conv (2x)	(80,80)	yes-0.5	yes	yes	no	0:33	0.749	0.781	0.749	0.658
7	2 conv (2x)	(80,80)	yes-0.5	yes	yes	yes	0:37	0.740	0.792	0.735	0.761
8	2 conv (2x)	(80,80)	yes-0.5	yes	no	yes	0:38	0.772	0.807	0.771	0.789
9	2 conv (2x)	(80,80)	yes-0.2	yes	no	yes	0:38	0.781	0.809	0.781	0.788
10	VGG16	(80,80)	yes-0.2	yes	no	yes	0:31	0.818	0.828	0.808	0.787

(Geom. Transf - Geometric transformation; RGB – The use of 3 Channels and 'rgb' in the color_mode parameter; Model num. – Model Number)

Table 2 – Models, parameters and topologies considered during the build from scratch CNN model process

The first model, **model 1**, with only two convolution layers and trained with images of size 150x150 is depicted in the figure below. The graph curves of loss and accuracy depicted in graph 1, display *overfitting*. The training accuracy increases, until it reaches nearly 100% on epoch 12, whereas the validation accuracy stalls at around 70%. The validation loss increases linearly over time, whilst the training loss decreases until it reaches nearly 0.

```
model.add(layers.Conv2D(32,(3,3),activation = 'relu',input_shape=(80,80,1)))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(64,(3,3),activation = 'relu'))
model.add(layers.MaxPooling2D((2,2)))
```

Figure 2 – two convolution layer topology

Based on the above, the following model iteration, **model 2**, has included the dropout technique to help mitigate overfitting. The dropout *constant rate* has been set to 0.50. The plots displayed in graph 2, still show a similar behavior to the previous model, with the clear presence of overfitting. The only difference is the training time, with the training accuracy reaching nearly 100% on epoch 22 rather than on epoch 12.

Taking that into account and considering the small-medium training dataset size, the **model 3** has introduced the data augmentation technique, which is specific to computer vision problems. The idea behind is to augment the training set via a number of random image transformations, applied to the original ones, which include the following ones defined by ourselves:

- *rotation_range* = 40, meaning the new images rotate between 0 and 40
- *width-shift range* = 0.30 sets the new images width as a fraction of the original

- *zoom range* = 0.20 is for zooming inside pictures
- *horizontal_flip* = True, refers to flipping half the images horizontally

The **model 3** resulting plots display a different output. Now both the training and validation loss decrease linearly over time. On top of that, the training accuracy reaches 75% after the 30 epochs, with similar values upon the validation accuracy. We might conclude that the overfitting issue is sorted out.

Now, the goal was to change other different hyperparameters and test different topologies to improve the results. Our first approach was to understand the impact of the training image size reduction to (80,80), which defines the **model 4**. Comparing the results between model 3 and 4 from table 2 it is noticeable that the performance is quite similar. Together with theoretical less computational effort, we decided that there was good evidence to maintain *the target_size* as (80,80) in the further model's testing.

Afterwards, we have decided to understand the effect of variations into the network topology itself. The first change was by adding one convolution layer, with the new **model 5** adding up to a total of three convolutional layers, with max pooling layers between them as represented in the figure below.

```
model.add(layers.Conv2D(32,(3,3),activation = 'relu',input_shape=(80,80,1)))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(64,(3,3),activation = 'relu'))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(128,(3,3),activation = 'relu'))
model.add(layers.MaxPooling2D((2,2)))
```

Figure 3 – three convolution layer topology

The results are displayed in graph 5, the model presents a final accuracy with 0.805, and the previous one presents 0.716. It's also worth mentioning that from graphical analysis we can see that the validation outperforms the train in almost all epochs.

Another topology variation was also tested, **model 6**, by connecting two convolution layers followed by a max pooling layer, and repeating this structure one more time, as represented in the figure below.

```
model.add(layers.Conv2D(32,(3,3),activation = 'relu',input_shape=(80,80,1)))
model.add(layers.Conv2D(32,(3,3),activation = 'relu'))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(64,(3,3),activation = 'relu'))
model.add(layers.Conv2D(64,(3,3),activation = 'relu'))
model.add(layers.MaxPooling2D((2,2)))
```

Figure 4– two times two convolution layers topology

The results from this new topology are presented in graph 6 and show both loss and accuracy with slightly more volatility for validation than the results obtained from the previous topology (model 5). The previous model had a best accuracy of 0.808 during validation, whilst the model 6 presented a best accuracy of 0.781. We acknowledge the slight decline in performance. However, in light of the solution implemented in Kaggle[5] for the same dataset, with a binary classification output instead, the model's 6 architecture showed the better performance results. With this in mind, we have decided to keep the topology described in the model 6 and further focus on manual parameter tuning combinations.

We have decided to look deeper into the *color_mode* parameter, which is responsible for whether the images presented to the network will be converted to 1 or 3 color channels [6].

Initially, since we were dealing with x-ray medical images, we decided to use only 1 channel and set the *color_mode* as "*grayscale*" based on the empirical belief, the most relevant x-ray image' features were not captured on coloration. However, we have still decided

to test the previous topology, now with *"rgb",* in **model 7***,* since there was a possibility that some visual details could have been lost when using "grayscale". The results presented in graph 7. The previous model had a best accuracy of 0.781 during validation, whilst the model 7 presented a best accuracy of 0.792 and a 5-minute increase in network running time. Based on this information, we have made the decision to proceed with the modelling with 3 channels *"rgb"*.

As a further parameter testing step, we have decided to reassess the data augmentation technique that we have been applying from model 3 onwards, since it proved to be fundamental in overfitting prevention. A study using deep learning in chest x-ray of patients with COVID-19 [7] concluded "*that geometrical data augmentation in X-ray images may not be effective in detecting COVID-19*". Taking that into account, we have decided to reformulate the data augmentation process by removing the geometrical parameters such as *rotation_range* and setting *horizontal_flip* as false. We have implemented these changes, on top of model 7, resulting in **model 8**. Although there were more spikes in the validation curve of both loss and accuracy (graph 8), the model showed improvement in max accuracy value and showed a lower loss end-value. Therefore, we have decided to continue with further model testing without geometrical transformations in the data augmentation step.

The parameter adjustment testing process continued with the focus on the dropout step. Dropout is a regularization technique that tries to prevent neural networks from overfitting [8]. Throughout the topologies tested so far, with the exception of the first one, we have always applied a *constant rate* of 0.50, which means that half of the layer's outputs are randomly *dropped out* [8]. However, in several models, namely model number 5,7 and 8, the accuracy of the training set was slightly lower than the validation, contrary to what was expected. This could be an indication that our models were not fully capturing the complexity of the images provided. As a result, in the next test step, we have decided to reduce the *dropout*'s *constant rate* to 0.2, leading to the **model 9**.

The results, displayed in graph 9, show small improvements, when compared with previously described models, in both the training and validation set. Therefore, the group decided to maintain the last topology test (model number 8), with a *constant rate* for *dropout* of 0.2.

All the previous architectures used were based on a *built from scratch* approach, until the best possible model was reached (model 9). In our last step of model development, the group has decided to test a different approach by using Kera's transfer learning pre-trained CNN vgg16. After internal discussion, the group decided to train this network using a *fine-tuning* approach, rather than *feature extraction*. It's important to mention that structurally, this CNN is far more complex that the networks that we have been building from scratch, and it's recognized by its efficiency in image classification. The *dropout*'s *constant rate* choice of 0.2 together with the decision to not include geometrical transformation in the data augmentation step, was based on the empirical evidence from our previous *built from scratch* models. The results, expressed in the graph 10, show an overall good performance with the highest validation accuracy score of 0.828, which is higher than the best one *built from scratch* model 9 with a value of 0.809.

4. Prediction

On the prediction step, we decided to include model 9 and model 10, which were the models with best overall performance. Model 9 was the model built from scratch and model 10 was the last model training that was based on a *fine-tuning* approach of the vgg16 pre-trained CNN. Both results are displayed in table 2 and 3 respectively.

Model 9

True label	Predicted label			All
	Bacterial	Normal	Viral	
Bacterial	**285**	8	7	300
Normal	0	**300**	0	300
Viral	105	18	**77**	200
All	390	326	84	800

Table 3– Model 9 predictions on test set

Model 10

True label	Predicted label			All
	Bacterial	Normal	Viral	
Bacterial	**277**	16	7	300
Normal	0	**300**	0	300
Viral	96	22	**82**	200
All	373	338	89	800

Table 4 – Model 10 predictions on test set

The results on the **model 9** (*built from scratch*) showed an overall accuracy of 82,7% in our test data. However, a closer look shows that *our built from scratch* CNN model performs very differently according to the intended predicted label. Bacterial pneumonia and normal images had a classification accuracy of 95,0% and 100%, respectively. However, for viral pneumonia, the model showed poor results, only being able to accurately classify 38,5% of the images. The model showed satisfactory performance in distinguishing if a patient has pneumonia or not, but struggles to identify if the image is suggestive of bacteria or viral pneumonia. The model might be interesting for a binary classification scenario (pneumonia vs non-pneumonia) but is unsatisfactory for a single-label multiclass classification scenario involving images with viral and bacterial pneumonias.

Regarding **model 10** (*fine-tuning* model), it outperforms model 9, with an overall accuracy of 82.3%. However, a closer look reveals the same difficulties registered before, in classifying viral pneumonia images, where it displayed an accuracy of 41%.

5. Conclusion

As we saw earlier, our project follows two distinct approaches. First, we have decided to build from scratch a CNN model based on the knowledge acquired during our course. Second, we have tried to implement a transfer learning approach via *fine-tuning* of the pre-trained model vgg16 from Keras. While building from scratch, more than 30 models with different hyperparameters were tested within the four main topologies presented. To ease the presentation results and based on its relevance, we have ultimately decided to present only 10 models. During this first phase, we have implemented different techniques and methodologies with the objective of developing a competent model in solving the task at hand. Here, our main goal was to try to establish a sequential logic of the parameters and topologies that contributed to an improvement in the performance of the model in question, taking in consideration both accuracy and processing/computational time. To do this, we have applied a trial-and-error methodology so that, along model 1 through 9, we were restricting the final structural characteristics of our end model. In the model 10, the goal was to implement the vgg16, enhanced with the parameters we selected as the most relevant in the previous pipeline and compare them.

In both approaches, one of the major points of discussion was the data augmentation step. Although this is crucial to tackle the overfit present in our models, we are aware that human chest X-rays obey specific projection rules in order to be valid for diagnosis, which empirically might limit the good results coming from this process.

On the other hand, after some research, we found a scientific paper [9] that could be applied to this project. The paper sought to optimize the performance of CNN in predicting the probability of a patient suffering from COVID-19, using x-ray images together with pre-processing algorithms. These were used to detect and consequently remove the region corresponding to the diaphragmatic area of the initial images. The conclusion of this study was

clear: The implementation of the pre-processing algorithms contributed greatly to the improvement of CNN performance (from 88% to 94.5%). This could be an interesting implementation in future work development.

Regarding the performance results on the test data, despite the initial overall results being satisfactory with the best performance seen on the fine-tuning CNN model (**model 10**), the truth is that none of them show satisfactory ability to correctly classify viral pneumonias. One possible reason could have been the fact that our dataset was initially unbalanced. However, both viral pneumonia and normal X-rays had the same number of images, but with very different results. Also, the nature of the image's clinical characteristics itself might play a role, since the radiological patterns associated with viral pneumonias are empirically harder to diagnose by the lack of pathognomonic features [10]. The best accuracy for viral pneumonias was 41% versus 100% of accuracy for normal X-rays. In order to counteract this underperformance for viral pneumonia, we could add more data to this sub-category or, even, develop a model dedicated exclusively to identify differences between images of bacterial and viral pneumonia. However, we considered that the last one was out of the scope of this project, and we were unable to find more viral pneumonia image data available.

To conclude, the built CNN models are not ready to be deployed in the single-label multiclass classification scenario proposed in this project, especially in medical context.

References

1 - Deep Learning - IBM Cloud Education - May 2020. Available: https://www.ibm.com/cloud/learn/deep-learning.

2 - Elgendi M, Nasir MU, Tang Q, Smith D, Grenier J-P, Batte C, Spieler B, Leslie WD, Menon C, Fletcher RR, Howard N, Ward R, Parker W and Nicolaou S (2021) The Effectiveness of Image Augmentation in Deep Learning Networks for Detecting COVID-19: A Geometric Transformation Perspective. *Front. Med.* 8:629134. doi: 10.3389/fmed.2021.629134

3 – Liu, X., Faes, L., Kale, A. U., Wagner, S. K., Fu, D. J., Bruynseels, A., Mahendiran, T., Moraes, G., Shamdas, M., Kern, C., Ledsam, J. R., Schmid, M. K., Balaskas, K., Topol, E. J., Bachmann, L. M., Keane, P. A., & Denniston, A. K. (2019). A comparison of deep learning performance against health-care professionals in detecting diseases from Medical Imaging: A systematic review and meta-analysis. *The Lancet Digital Health*, *1*(6). https://doi.org/10.1016/s2589-7500(19)30123-2

4 – Chest X-Ray Images (Pneumonia) – Kaggle – Accessed: February 2022. Available at: https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia

5 - Beating everything with Depthwise Convolution - Kaggle - Accessed: April 2022. Available at:https://www.kaggle.com/code/aakashnain/beating-everything-with-depthwise-convolution

6 – Keras – Documentation – available at: https://faroit.com/keras-docs/1.0.6/preprocessing/image/

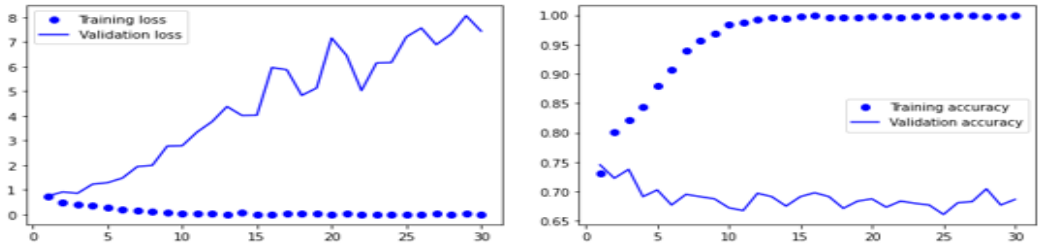
7 - Rocha, J., Mendonça, A. M., & Campilho, A. (2021). Review on deep learning methods for chest X-ray based abnormality detection and thoracic pathology classification. *U.Porto Journal of Engineering*, *7*(4), 16–32. https://doi.org/10.24840/2183-6493_007.004_0002

8 - Chollet François. (2021). *Deep learning with python*. Manning Publications.

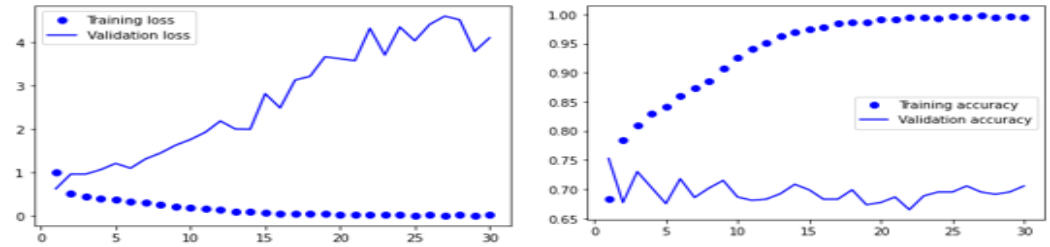
9 - Heidari, M., Mirniaharikandehei, S., Khuzani, A. Z., Danala, G., Qiu, Y., & Zheng, B. (2020). Improving the performance of CNN to predict the likelihood of COVID-19 using chest X-ray images with preprocessing algorithms. *International Journal of Medical Informatics*, *144*, 104284. https://doi.org/10.1016/j.ijmedinf.2020.104284

10 - Koo, H. J., Lim, S., Choe, J., Choi, S.-H., Sung, H., & Do, K.-H. (2018). Radiographic and CT features of viral pneumonia. *RadioGraphics*, *38*(3), 719–739. https://doi.org/10.1148/rg.2018170048

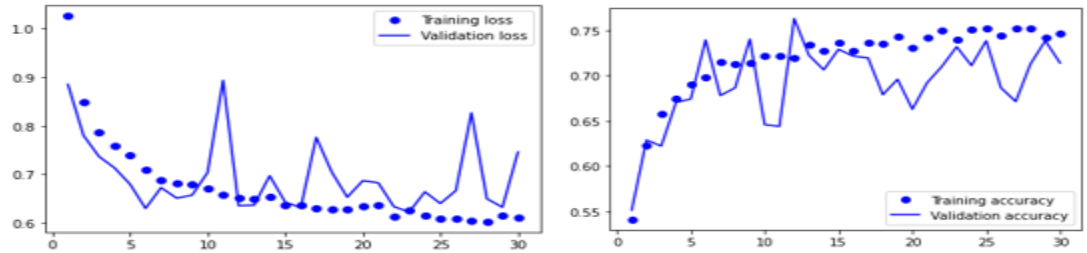
Appendix



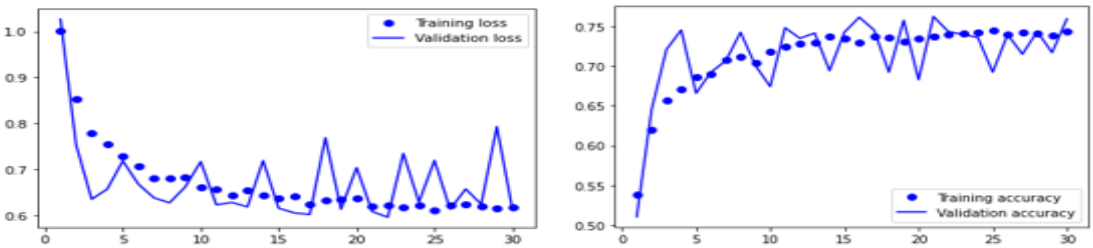
Graph 1: 2 convolution layers, (WITHOUT augmentation and WITHOUT dropout) and target size (150,150)



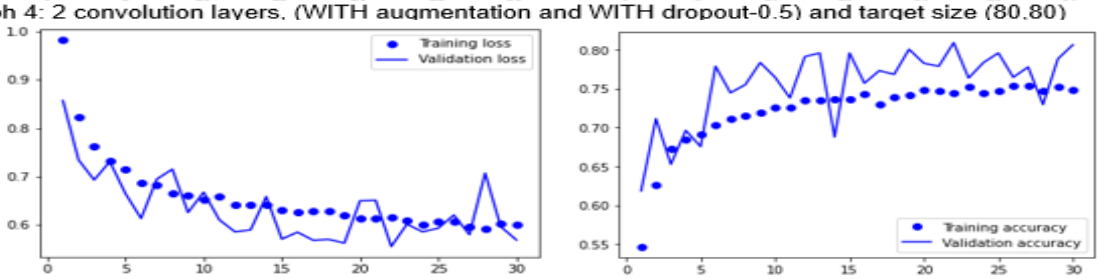
Graph 2- 2 convolution layers, (WITHOUT augmentation and WITH dropout-0.5) and target size (150,150)



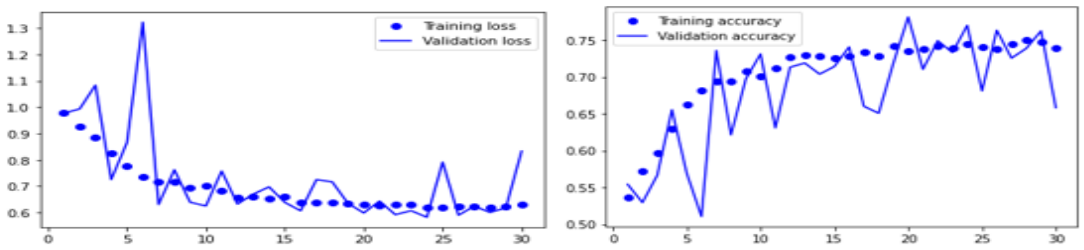
Graph 3:- 2 convolution layers, (WITH augmentation and WITH dropout-0.5) and target size (150,150)



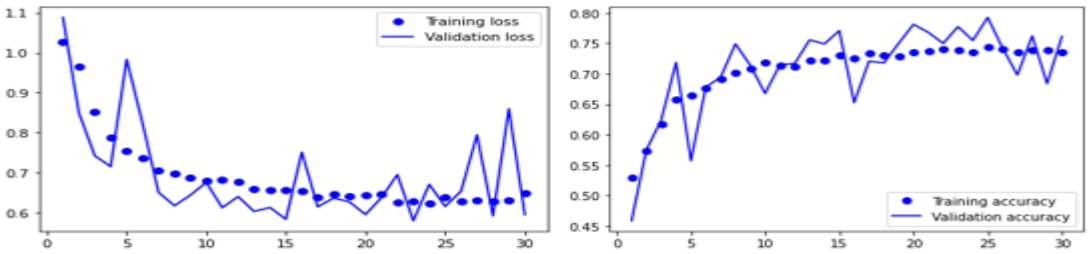
Graph 4: 2 convolution layers, (WITH augmentation and WITH dropout-0.5) and target size (80,80)



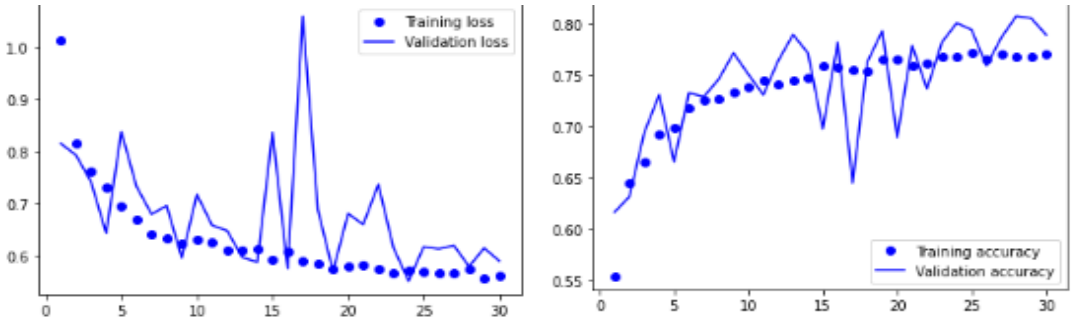
Graph 5 - 3 convolution layers, (WITH augmentation and WITH dropout-0.5) and target size (80,80)



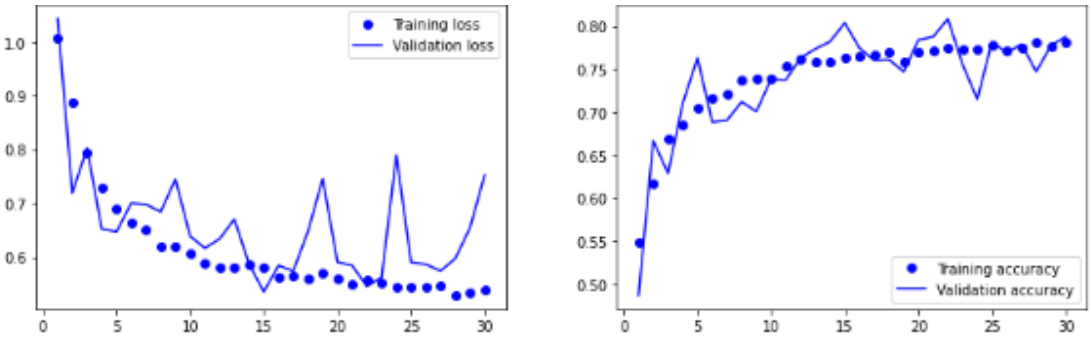
Graph 6 - Conv ->Conv ->max pooling->Conv ->Conv ->max pooling->dense layers(WITH augmentation and WITH drop out 0.50) and target size (80x80)



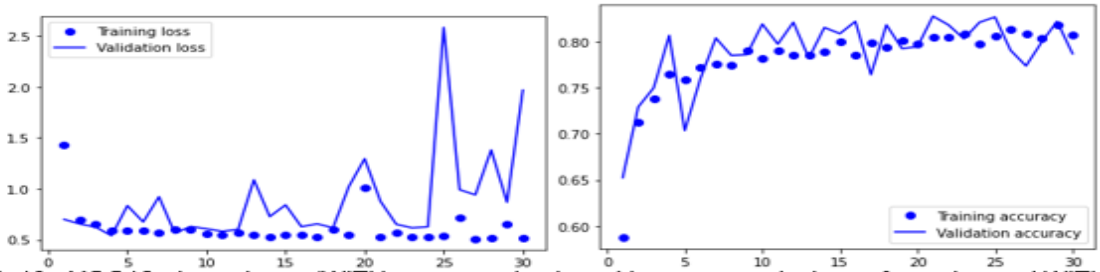
Graph 7 - Conv ->Conv ->max pooling->Conv ->Conv ->max pooling->dense layers(WITH augmentation and WITH drop out 0.50), target size (80x80) and with RGB



Graph 8 - Conv ->Conv ->max pooling->Conv ->Conv ->max pooling->dense layers(WITH augmentation but without geometrical transformation and WITH drop out 0.50), target size (80x80) and with RGB



Graph 9 - Conv ->Conv ->max pooling->Conv ->Conv ->max pooling->dense layers(WITH augmentation but without geometrical transformation and WITH drop out 0.20), target size (80x80) and with RGB



Graph 10 - VGG16>dense layers(WITH augmentation but without geometrical transformation and WITH drop out 0.50), target size (80x80) and with RGB