Neural Networks Project
“COVID-ResNet: A Deep Learning Framework for Screening of COVID19 from Radiographs”

Leandro Maglianella – 1792507

Contents

Abstract

The COVID-19 pandemic has completely monopolized our societies for over a year. An important phase in the fight against the virus is the creation of effective methods to identify its presence in patients; methods that are therefore able to distinguish whether the subject is healthy or sick and distinguish COVID-19 from other forms of diseases. Recent studies have shown that, among all of COVID-19's symptoms, the virus involves the presence of anomalies in the radiographies of people positive to it: the present work is based on this assumption and attempts accordingly to create an instrument that can perform the aforementioned task. This project builds on, re-implements and enhances the concepts described in the paper "COVID-ResNet: A Deep Learning Framework for Screening of COVID19 from Radiographs". The study is performed by creating and training a model based on one of the major state-of-the-art Neural Networks, ResNet50: this network is here modified and is able to gain even better results thanks to the application of transfer learning and, in particular, fine-tuning and progressive resizing. The overall work consists in a classification task of radiography images where the model will be able to achieve an F1-score of 0.946 after training for 21 epochs.

Introduction

Since the COVID-19 outbreak, huge variations have been introduced in our daily life and the scientific community is working hard day by day to understand and discover all its effects to finally eradicate it. This project implements an effective COVID-19 screening method by means of chest radiography examination: in fact, recent studies have shown that the virus involves the presence of anomalies in the radiographies of people positive to it. Based on this assumption, this research will build up a particular neural network architecture capable of carrying out a classification task among the "Normal", "Pneumonia" and "COVID-19" categories.

First of all, in the continuation of this report, in the "Related work" section the concept of Residual Neural Network and the state-of-the-art network employed as the foundation of the used model will be analysed: ResNet50, a pre-trained 50 layers residual neural network that provide a good combination of performance, number of parameters and have proved to be one of the fastest training architectures. It will be explained how ResNet solves the "vanishing gradient effect" and, subsequently, the principles applied in this project to the network to improve its accuracy will be described: in particular the theoretical notions of Progressive Resizing, Transfer Learning and Fine-Tuning will be treated.

Next, the core of this work will start to be described in the "Experimental Procedure" section. It will be pointed out how the dataset has been constructed, its data distribution and how it has been pre-processed to make it suitable for the model's training. After that, the variations to the structure of ResNet50 will be assessed in reference with the training and the adopted principles. Finally, this research will analyse the peculiar three stages training technique designed specifically for the used network to maximize accuracy and reduce training time.

The report will end with the "Results" and the "Model comparisons" sections where the outcome of the training will be considered, the validation metrics and confusion matrices achieved by the best trained model will be shown and it will be confronted with other models.

1. Related work

1.1. Residual Neural Networks

In this study, a variant of the residual neural network with a total of 50 layers called ResNet50 has been leveraged. Residual Networks (ResNets) came about in 2015 winning the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) where they were able to show to be reliable neural network architectures that are easier to train to high performance, using a lower number of training parameters and taking less training time. Moreover, all of these reasons enable deeper architectures (hence more robust and capable of representing more complex features) to be built successfully. The architecture of ResNet50 will be analysed in particular in the next section, in the "Network architecture" subsection.

1.1.1. Vanishing Gradient Effect

Let us now explain the functioning of Residual Learning. One of the major problems concerning standard feedforward neural networks was the impossibility of increasing their accuracy in proportion to the depth of the network: in fact, in deep neural networks the phenomenon of accuracy degradation was observed and adding more and more layers to the networks either made the accuracy value to saturate or it all of a sudden started to decrease. The cause of this event was traced back to the "vanishing gradient effect" which is present, precisely, only in deeper networks.

During training, in the error's backpropagation phase, each of the neural network's weights is given a new value that is proportional to the partial derivative of the error function and compared to the current weight in each iteration of training. The problem here is that sometimes it happens that the gradient will be extremely small, so much that the update significance will vanish and the weight will be prevented from changing its value (Figure 1). This is the case especially for the initial layers of the network where the parameters' improvements will occur very slowly. In the worst case, this degradation may completely stop the neural network from further training.

$$\delta^{(l)} = \mathbf{W}^{T(l+1)}\delta^{(l+1)} \odot \mathbf{\Phi}'\left(\mathbf{s}^{(l)}\right)$$

$$\mathbf{W}^{(l)}_{new} = \mathbf{W}^{(l)}_{old} + \mu\delta^{(l)}\mathbf{x}^{T(l)} \qquad \textit{generalized delta rule for MLP}$$

$$\delta^{(1)} = \underbrace{\mathbf{W}^{T(2)}\mathbf{W}^{T(3)}\cdots\mathbf{W}^{T(L)}}_{\textbf{exploding gradient term}}\mathbf{e}\odot\underbrace{\mathbf{\Phi}'\left(\mathbf{s}^{(L)}\right)\odot\mathbf{\Phi}'\left(\mathbf{s}^{(L-1)}\right)\cdots\odot\mathbf{\Phi}'\left(\mathbf{s}^{(2)}\right)\odot\mathbf{\Phi}'\left(\mathbf{s}^{(1)}\right)}_{\textbf{vanishing gradient term}}$$

Figure 1: the local error gradient term $\delta^{(l)}$ decreases exponentially with number of hidden layers.

1.1.2. Residual Connections

Among the various possible solutions that can be adopted to solve this problem, the application of a ResNet as a model is one of the most recent and effective. In general, Residual Networks refer to network architectures that include "residual connections" or "skip connections": these are the critical component of what allows a successful fast training of deeper neural networks allowing gradient information to propagate through the layers, by creating a channel where the output of a previous layer is added directly to the output of a deeper layer without passing through the non-linear activation functions, which generally is a ReLu function (Figure 2).

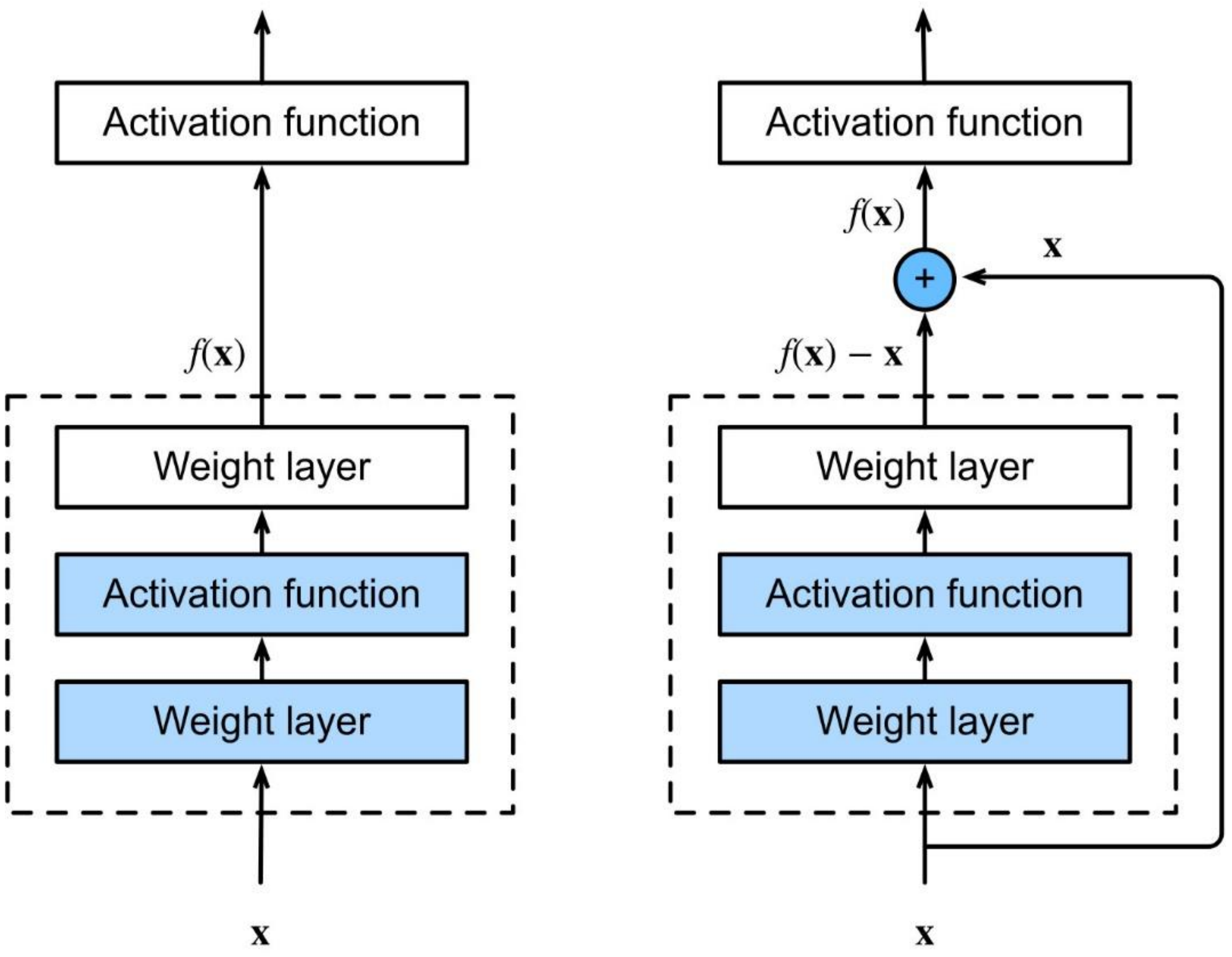


Figure 2: A regular block (left) and a residual block (right).

In Figure 2, a regular block (right) and a residual block (left) are represented as the dotted-line boxes. The regular block must directly learn the mapping *f(x)* while the residual block first learns the residual mapping *f(x)* − *x* and then the input *x* is added, carried by the solid arrow line which represents a skip connection.

Because of this, information from the earlier parts of the network can be passed to the deeper parts of the network, helping maintain signal propagation even in deeper networks and compensating for the vanishing effect. However, it should be remarked that ResNets do not resolve the vanishing gradient effect by preventing the gradient to flow throughout the entire depth of the network. In fact, creating a Residual Network in the proposed way, the outcome network is rather actually constituted of an ensemble of relatively shallow networks which are not affected by the gradient problem.

1.2. Transfer Learning

As mentioned in the Introduction, a consistent part of this work is a transfer learning application. This type of deep learning is based on the assumption that a model pretrained on a large and general enough dataset will qualify it as a generic model of visual world, readopting its weights without the need of training once again a large model on an even larger dataset.

Usually, transfer learning is divided in two categories: Feature Extraction and Fine-Tuning. In Feature Extraction the representations learned by a previous network is used to extract meaningful features from new samples: therefore, these weights are “freezed” and will be prevented from changing during the learning procedure. Only a new small classifier will be trained from scratch, on top of the pretrained model so that it is possible to repurpose the feature maps learned previously for the dataset. Regardless, however, this project is an example of Fine-Tuning.

1.2.1. Fine-Tuning

This method consists on unfreezing a few of the top layers of a frozen model base and jointly train both some newly-added classifier layers and the last layers of the base model. This allows us to "fine-tune" the higher-order feature representations in the base model in order to make them more relevant for the specific task and the new dataset: in this way it is feasible to transfer the knowledge already acquired by the pre-trained model into the new one. This technique is usually recommended when the training dataset is large and very similar to the original dataset that the pre-trained model was trained on.

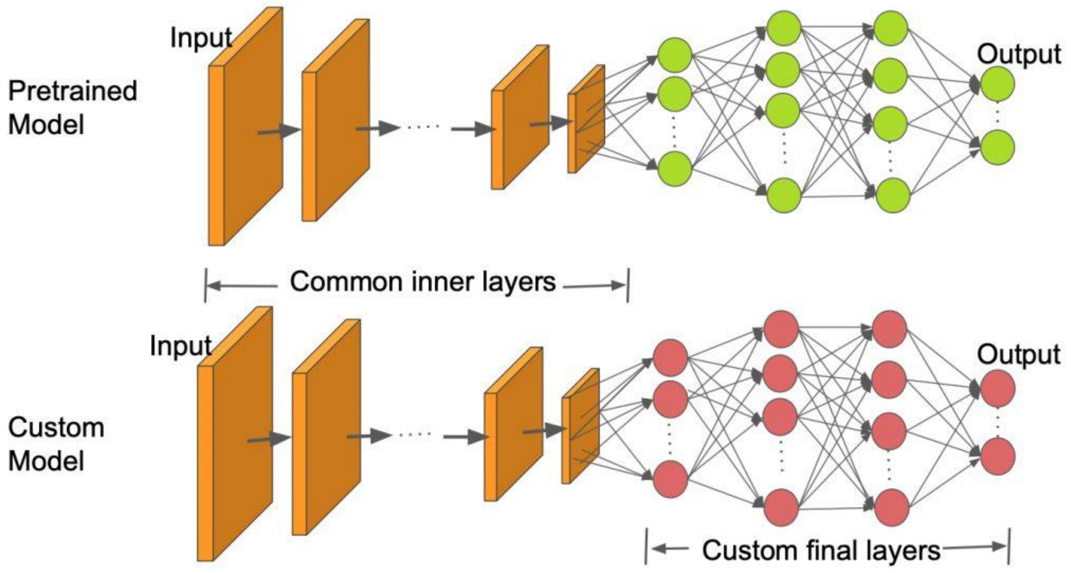


Figure 3: Transfer Learning example.

More specifically for this project, the weights that have been loaded and put as foundation in the model’s architecture were previously obtained by training ResNet50 on ImageNet, a huge dataset with more than fourteen million of images belonging to more than twenty thousand of classes. Following the “COVID-ResNet: A Deep Learning Framework for Screening of COVID19 from Radiographs” paper, in the fine-tuning training’s phases not only the last layers of ResNet50 has been trained, but the whole network.

1.2.2. Progressive Resizing

Progressive resizing is an approach used where the network is initially trained with smaller images and the size of the input images is gradually increased as the training progresses. This can be done as the features learned by the different layers of the CNN are independent of the input image size. Furthermore, the global features are still present in the same resized image with different pixel resolutions. Training the model in multiple stages with different input image sizes using progressive resizing tends to achieve a better generalization. In this project three different input image's sizes have been used consecutively: *128×128×3*, *224×224×3*, and *229×229×3* pixels.

Both applications of Fine-Tuning and Progressive Resizing will be further resumed in the next section, in the "Training" subsection.

2. Experimental Procedure

2.1. Dataset Generation

Author's Note: the COVIDx dataset is constantly evolving and each new update expands it by adding new samples. The COVIDx V7A version of the dataset, released on the 28th January 2021, was used in this project.

In this work it was used COVIDx (version V7A), an open access benchmark dataset, usable for free by anyone for research purposes. It is constructed by samples coming from five different main open-source chest radiography datasets hosted online containing X-ray chest images of anonymous patients from all over the world:

- Covid Chest X-ray Dataset: COVID-19 public dataset collection hosted on GitHub by Joseph Paul Cohen, Postdoctoral Fellow at Mila and University of Montreal, Canada. He is also the Director of the Institute for Reproducible Research, AcademicTorrents.com and ShortScience.org.

- Figure 1 COVID-19 Chest X-ray Dataset: COVID-19 image data hosted on GitHub by DarwinAI Corp. (Canada) and Vision and Image Processing Research Group (University of Waterloo, Canada). These medical data were compiled by Figure 1.

- Actualmed COVID-19 Chest X-ray Dataset: COVID-19 image data hosted on GitHub by DarwinAI Corp. (Canada) and Vision and Image Processing Research Group (University of Waterloo, Canada). These medical data were compiled by Actualmed.

- COVID-19 Radiography Database: Winner of the COVID-19 Dataset Award by Kaggle Community. This database of chest X-ray images is hosted on Kaggle by a team of researchers from Qatar University, Doha, Qatar, and the University of Dhaka, Bangladesh along with their collaborators from Pakistan and Malaysia in collaboration with medical doctors. Italian Society of Medical and Interventional Radiology (SIRM) is one of the major contributors to this dataset.

- RSNA Pneumonia Detection Challenge Dataset: This database of chest X-ray images is hosted on Kaggle by the Radiological Society of North America in collaboration with the US National Institutes of Health, The Society of Thoracic Radiology, and MD.ai. It is used in a challenge with prizes where the participants have to build an algorithm to detect a visual signal for pneumonia in medical images.

These datasets were downloaded (6.60 GB) and their radiography images were extracted and combined together to generate a new general dataset (5.91 GB). Simultaneously with the generation of the general dataset, the data were divided into two folders (training images and test images) using a train/test split equal to 0.1 and in particular two .txt files where produced: "train_split.txt" and "test_split.txt". Each line of these text files represents a sample, each sample has four columns (respectively divided by a space key) which stand for the patient id, image filename, classification label for the subsequent Supervised Learning and Validation ("Pneumonia", "COVID-19" or "Normal") and dataset of origin of the sample.

The following tables show the Images' and the Unique patients' distributions of the Dataset resulted from this first experimental phase:

- Images' distribution:

Type	**Training images**	**Test images**	**Total images**
Normal	7966	885	8851
Pneumonia	5475	594	6069
COVID-19	1645	67	1712
Total	15086	1546	16632

- Unique patients' distribution:

Type	**Training patients**	**Test patients**	**Total patients**
Normal	7966	885	8851
Pneumonia	5451	591	6042
COVID-19	1483	46	1529
Total	14900	1522	16422

2.2. Pre-processing

The previously produced .txt files and the two images' folders were pre-processed and their information combined together to produce .npy array files: these are particular files which contain NumPy array (created in Python using the NumPy library). NPY files store all the information required to reconstruct the arrays, including their dtype and shape information. This procedure was repeated three times to get all the data needed for the progressive resizing method: each array, one for each of the three proposed input dimensions, contains the normalized pixel values (on a scale of 0 to 1) for each image. The labels' arrays correspondingly contain integer values which associate to the classes through this mapping: {'normal': 0, 'pneumonia': 1, 'COVID-19': 2}.

In detail, the following files were generated:

File	**Description**	**Dimensions**
x_train_128.npy	*15086×128×128×3* training array	2.76 GB
x_train_224.npy	*15086×224×224×3* training array	8.45 GB
x_train_229.npy	*15086×229×229×3* training array	8.84 GB
y_train.npy	*15086×1* training labels' array	60 KB
x_test_128.npy	*1546×128×128×3* test array	289 MB
x_test_224.npy	*1546×224×224×3* test array	887 MB
x_test_229.npy	*1546×229×229×3* test array	927 MB
y_test.npy	*1546×1* test labels' array	7 KB
Total		22.1 GB

In addition, the "COVID-ResNet: A Deep Learning Framework for Screening of COVID19 from Radiographs" paper suggested to use Data Augmentation (which helps in creating newer examples by applying different transformation randomly to the available training images) transforming using vertical flips of the images and random rotation of the images (with a maximum rotation angle of 15 degree) and changing the lighting conditions. I decided not to apply this step here in this project because medical data are very sensitive data and the COVIDx V7A dataset is already more than large enough.

2.3. Network Architecture

Let us now break down the adopted ResNet50's architecture. As it was mentioned, three input sizes were used however, to the benefit of this explanation, only the *224×224×3* input size will be considered. This network is divided in two macro-parts: the body and the head.

2.3.1. Network's body

The architecture of ResNet50's body is composed by 4 stages (follow Figure 4 and Figure 5 at the following page to clearly understand):

0. Initially, a convolution and max-pooling are performed using *7×7* and *3×3* kernel sizes respectively and stride = 2.

1. Afterward, Stage 1 of the network starts. It has a Residual block, reiterated 3 times, containing 3 layers: the size of kernels used to perform the convolution operation in these 3 layers of the block of stage 1 are 64, 64 and 256 respectively. The curved arrows refer to the residual connections. The curved dashed arrows represent that the convolution operation in that particular Residual Block is performed with stride = 2 (therefore the size of input will be reduced to half in terms of height and width but the channel width will be doubled). As the network progress from one stage to another, the channel width is doubled and the size of the input is reduced to half.

2. Stage 2 has a Residual block, reiterated 4 times, containing 3 layers: the size of kernels used to perform the convolution operation in these 3 layers of the block of stage 2 are 128, 128 and 512 respectively.

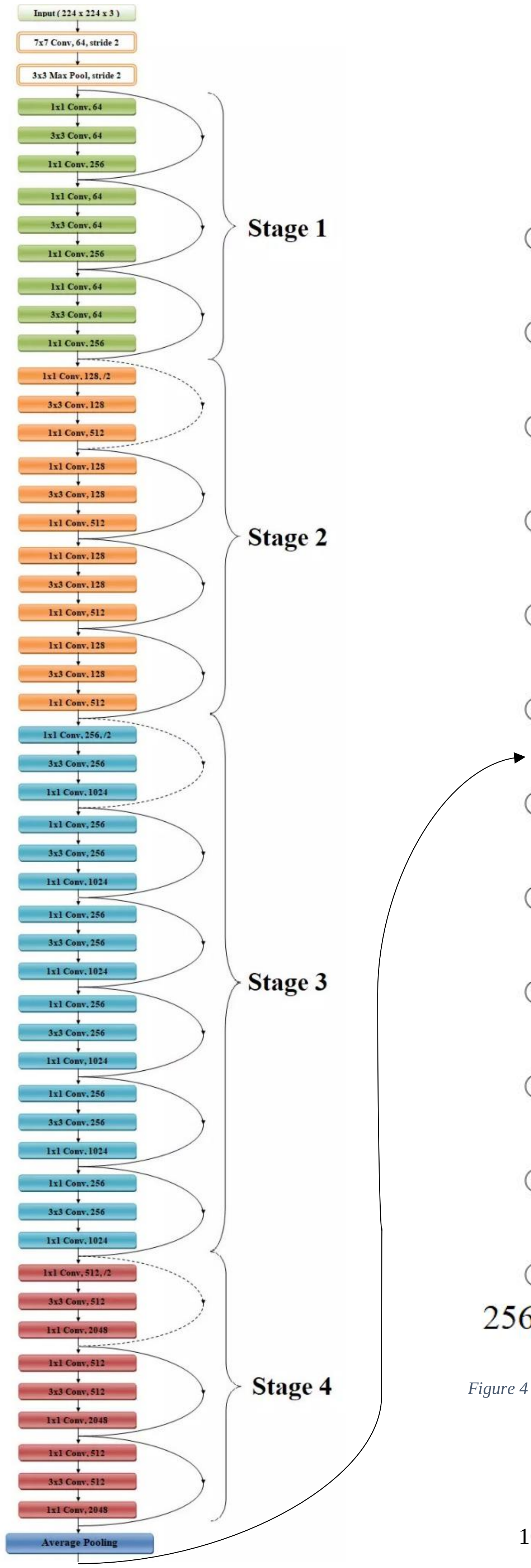
3. Stage 3 has a Residual block, reiterated 6 times, containing 3 layers: the size of kernels used to perform the convolution operation in these 3 layers of the block of stage 3 are 256, 256 and 1024 respectively.

4. Stage 4 has a Residual block, reiterated 3 times, containing 3 layers: the size of kernels used to perform the convolution operation in these 3 layers of the block of stage 4 are 512, 512 and 2048 respectively. For the whole network a bottleneck design is used: for each residual function (each block) 3 layers are stacked one over the other. The three layers are *1×1, 3×3, 1×1* convolutions. The *1×1* convolution layers are responsible for reducing and then restoring the dimensions. The *3×3* layer is left as a bottleneck with smaller input/output dimensions.

5. Finally, the network has an Average Pooling layer which connects the body to the head.

2.3.2. Network's head

For transfer learning, the head of the pre-configured ResNet50 model is replaced by another head containing a sequence of drop out, batch normalization and fully connected linear layers having respectively 256, 128, 64, 32 and, finally, 3 neurons (as many as the output categories "Normal", "Pneumonia" and "COVID-19" for this classification problem). Every layer uses a ReLu activation function, while the last output layer uses a Softmax activation function.



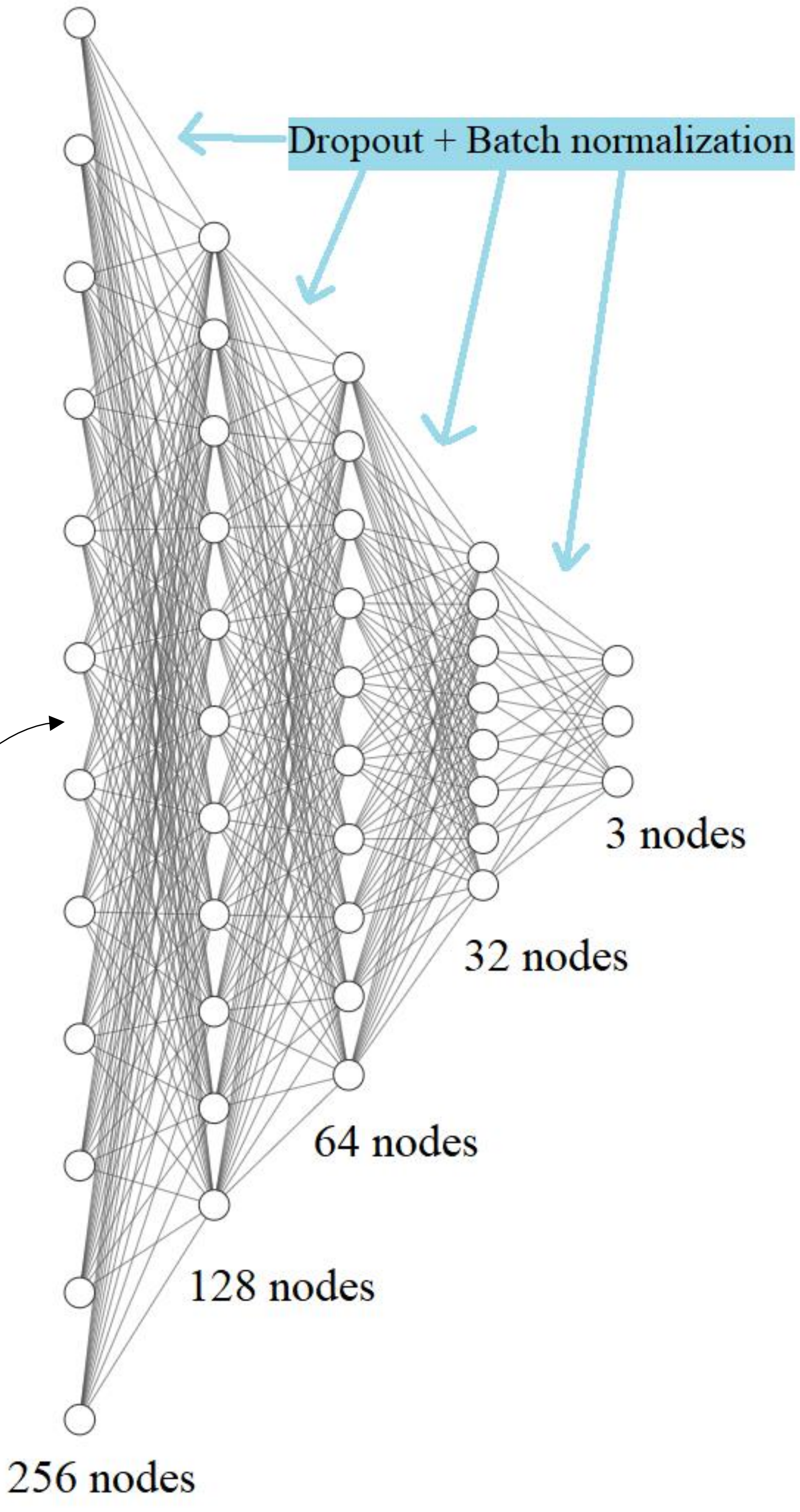


Figure 4 and Figure 5: Boddy (left) and Head (right) of the Network.

2.4. Training

Training is performed in three stages where each stage corresponds to images of different input dimensions. The entire training process is performed using the TensorFlow framework, adopting the Adam optimizer with a batch size of 32 and a categorical cross-entropy loss function. Moreover, a Model Checkpoint callback was used to save only the best model (the one with the maximum accuracy) at every step. One last thing to notice is that as we proceed to the later stages of the training, we ensure to reduce the learning rates. This guarantees that the weights are not significantly modified from one stage to successive stage.

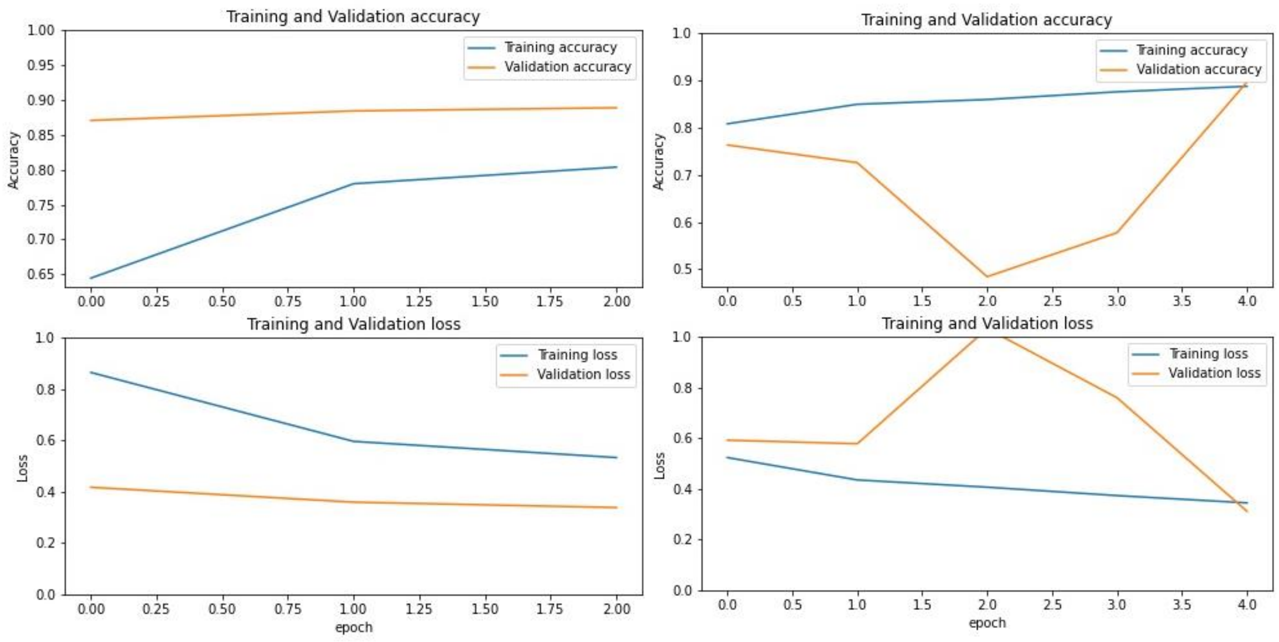
Note: this phase is particularly hardware-dependent, I have personally used the following:

- GPU: NVIDIA® GeForce® RTX 2070 SUPER™ 8GB GDDR6
- CPU: Intel® Core™ i7-10875H Processor 2.3 GHz (16M Cache, up to 5.1 GHz, 8 cores).

If you get a “ResourceExhaustedError”, I would suggest executing the stages separately one at a time.

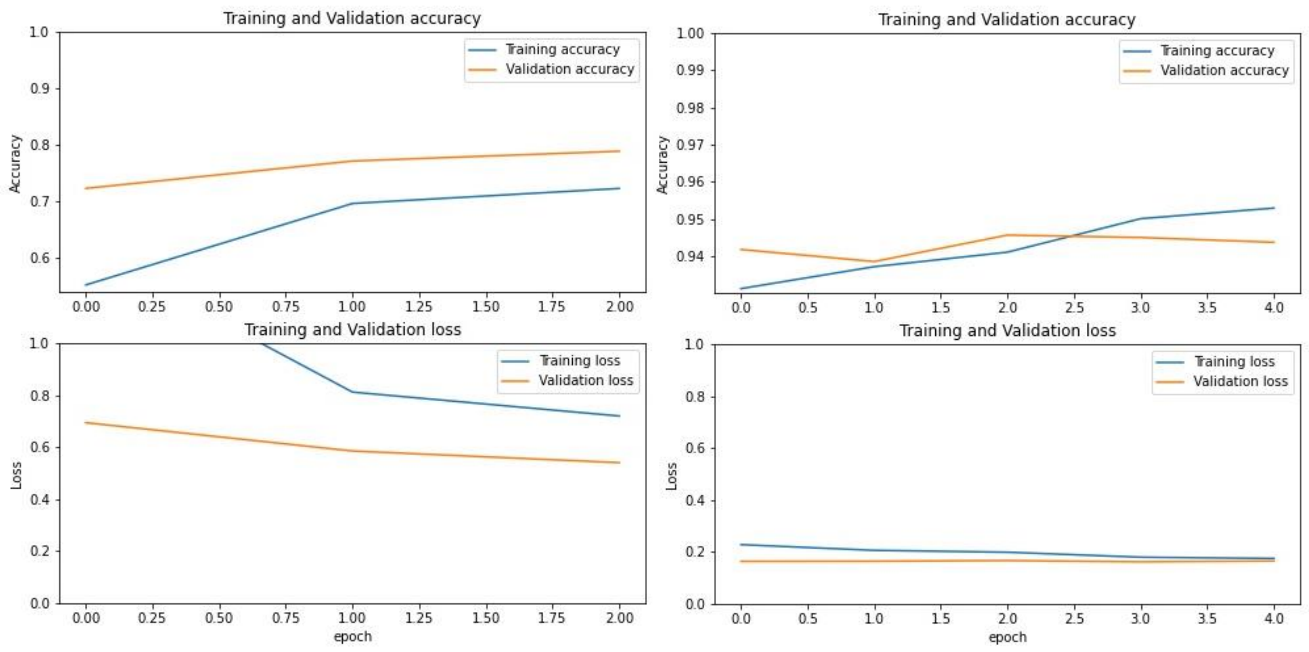
2.4.1. Stage 1

Model input’s size of *128×128×3*. This stage is formed by 2 steps: firstly, only the newly added head of the network is trained with a learning rate of 1e-3 for 3 epochs, while preserving the ImageNet weights for the rest of the body which is freezed. Secondly, the body is unfreezed and the whole network is fine-tuned (both the body and the head of the model) with a learning rate of 1e-3 for 5 epochs. The best model produced in this stage is saved as model1.h5. Below are the accuracy (top) and loss (bottom) rates during this stage (first step on the left and second step on the right), the blue line indicates the progress of the metrics in training while the orange line in testing:



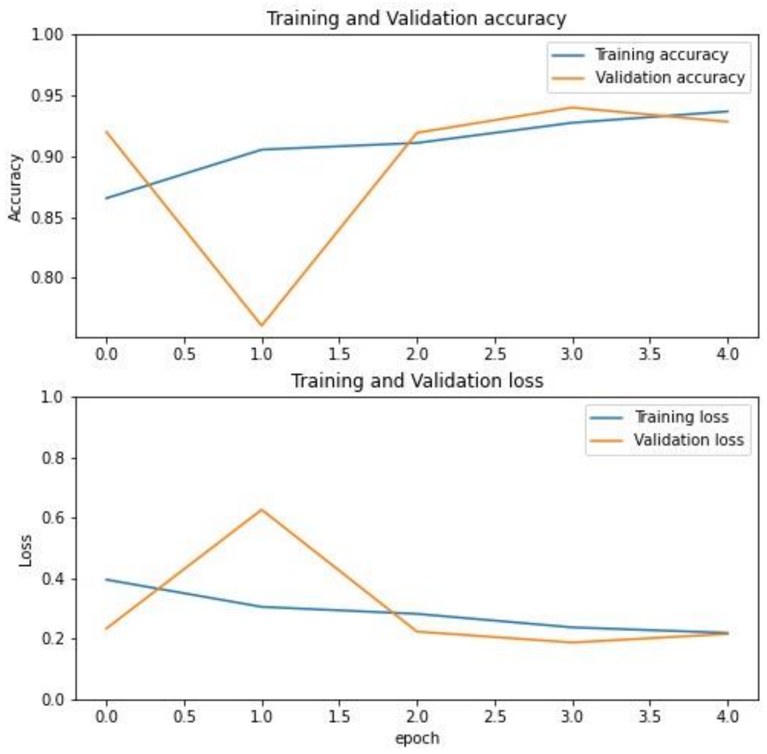
2.4.2. Stage 2

Model input’s size is progressively resized to *224×224×3*. The training goes on from the weights achieved in the last stage. This stage is formed by 2 steps: firstly, only the head of the network is trained with a learning rate of 1e-4 for 3 epochs, while preserving the weights for the rest of the body which is freezed. Secondly, the body is unfreezed and the whole network is fine-tuned (both the body and the head of the model) with a learning rate of 1e-4 for 5 epochs. The best model produced in this stage is saved as model2.h5. Below are the accuracy (top) and loss (bottom) rates during this stage (first step on the left and second step on the right), the blue line indicates the progress of the metrics in training while the orange line in testing:



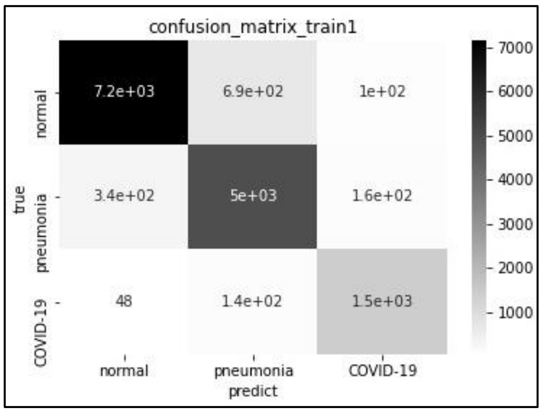
2.4.3. Stage 3

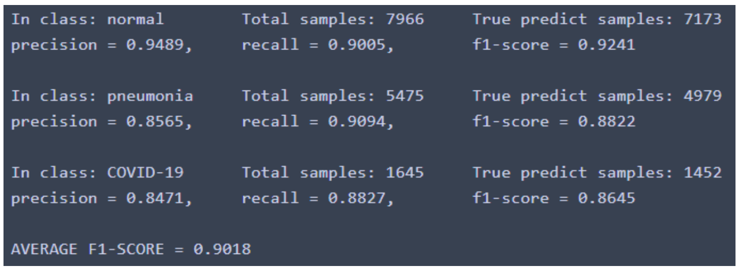
Model input’s size is progressively resized to *229×229×3*. The training goes on from the weights achieved in the last stage. This stage is formed by only one step: the whole network is fine-tuned (both the body and the head of the model) with a learning rate of 1e-5 for 5 epochs. Note that the original paper claimed to use 25 epochs for this last stage, however I observed that this involved a strong overfitting in my network where training accuracy tended to 100% while test accuracy began to converge or decrease: because of this I decided to reduce the number of epochs. The best model produced in this stage is saved as model3.h5. Below are the accuracy (top) and loss (bottom) rates during this stage, the blue line indicates the progress of the metrics in training while the orange line in testing:



3. Results

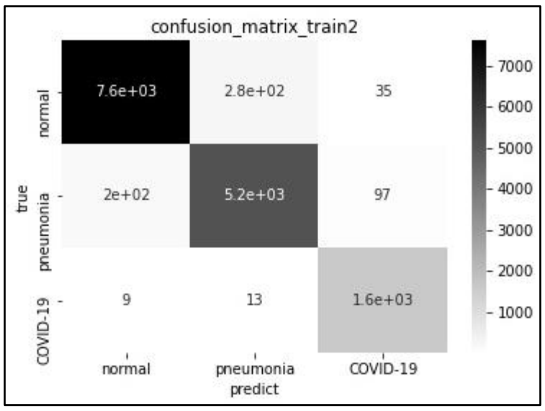
Below are the confusion matrices and validation metrics obtained by applying the three models to the training and testing samples:

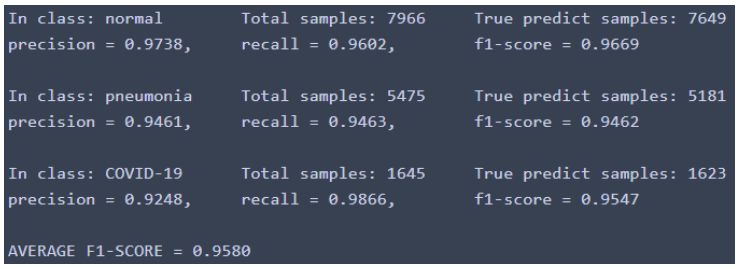




```
In class: normal          Total samples: 7966       True predict samples: 7173
precision = 0.9489,       recall = 0.9005,          f1-score = 0.9241

In class: pneumonia       Total samples: 5475       True predict samples: 4979
precision = 0.8565,       recall = 0.9094,          f1-score = 0.8822

In class: COVID-19        Total samples: 1645       True predict samples: 1452
precision = 0.8471,       recall = 0.8827,          f1-score = 0.8645

AVERAGE F1-SCORE = 0.9018
```

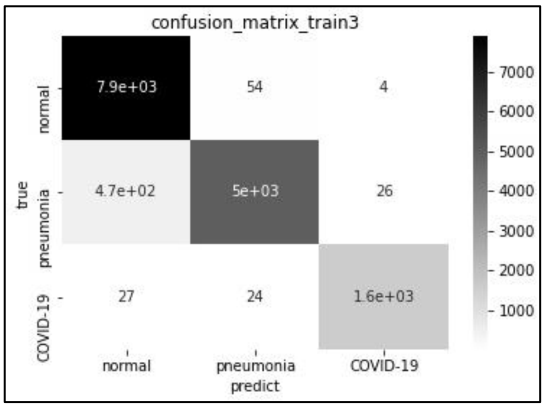
Results of model 1 applied to training data.

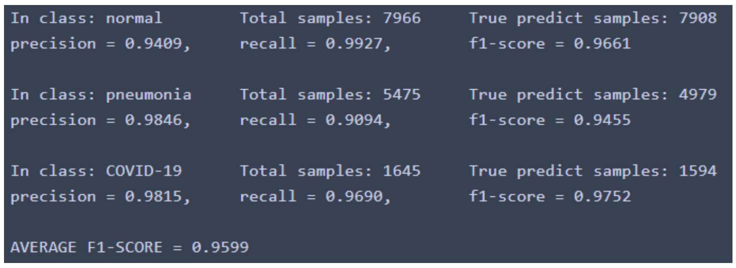




```
In class: normal          Total samples: 7966       True predict samples: 7649
precision = 0.9738,       recall = 0.9602,          f1-score = 0.9669

In class: pneumonia       Total samples: 5475       True predict samples: 5181
precision = 0.9461,       recall = 0.9463,          f1-score = 0.9462

In class: COVID-19        Total samples: 1645       True predict samples: 1623
precision = 0.9248,       recall = 0.9866,          f1-score = 0.9547

AVERAGE F1-SCORE = 0.9580
```

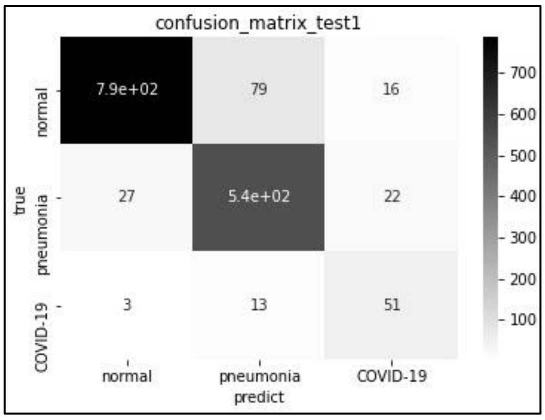
Results of model 2 applied to training data.

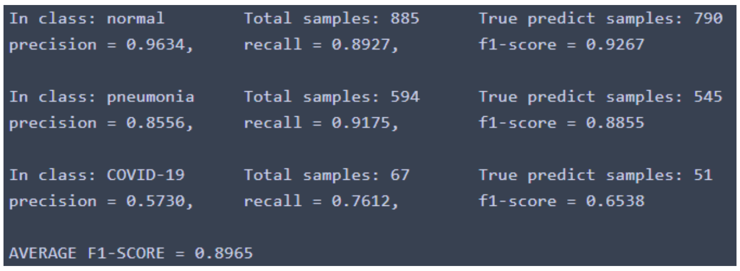




```
In class: normal          Total samples: 7966       True predict samples: 7908
precision = 0.9409,       recall = 0.9927,          f1-score = 0.9661

In class: pneumonia       Total samples: 5475       True predict samples: 4979
precision = 0.9846,       recall = 0.9094,          f1-score = 0.9455

In class: COVID-19        Total samples: 1645       True predict samples: 1594
precision = 0.9815,       recall = 0.9690,          f1-score = 0.9752

AVERAGE F1-SCORE = 0.9599
```

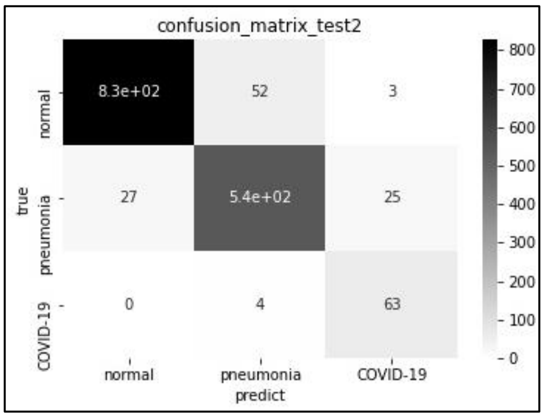
Results of model 3 applied to training data.

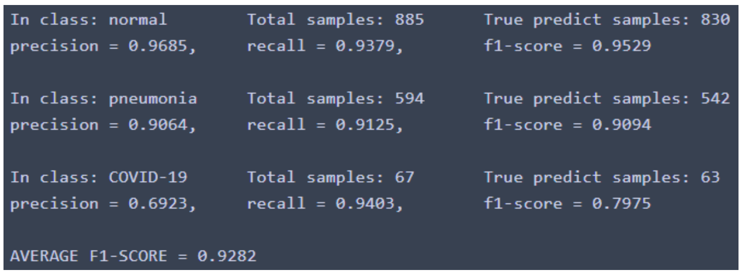




```
In class: normal        Total samples: 885      True predict samples: 790
precision = 0.9634,     recall = 0.8927,        f1-score = 0.9267

In class: pneumonia     Total samples: 594      True predict samples: 545
precision = 0.8556,     recall = 0.9175,        f1-score = 0.8855

In class: COVID-19      Total samples: 67       True predict samples: 51
precision = 0.5730,     recall = 0.7612,        f1-score = 0.6538

AVERAGE F1-SCORE = 0.8965
```

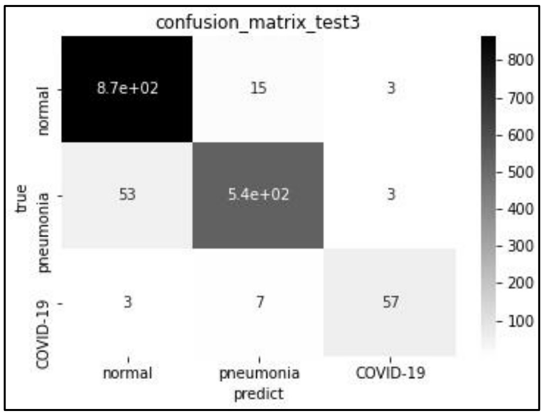
Results of model 1 applied to validation data.

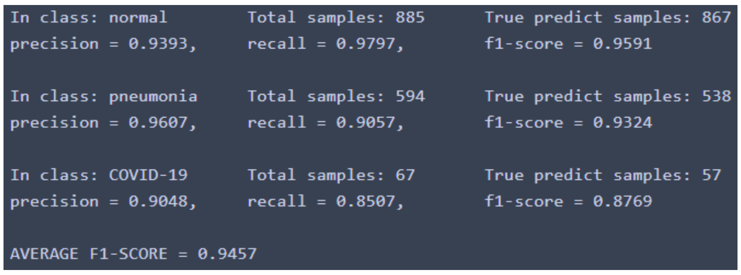




```
In class: normal        Total samples: 885      True predict samples: 830
precision = 0.9685,     recall = 0.9379,        f1-score = 0.9529

In class: pneumonia     Total samples: 594      True predict samples: 542
precision = 0.9064,     recall = 0.9125,        f1-score = 0.9094

In class: COVID-19      Total samples: 67       True predict samples: 63
precision = 0.6923,     recall = 0.9403,        f1-score = 0.7975

AVERAGE F1-SCORE = 0.9282
```

Results of model 2 applied to validation data.





```
In class: normal        Total samples: 885      True predict samples: 867
precision = 0.9393,     recall = 0.9797,        f1-score = 0.9591

In class: pneumonia     Total samples: 594      True predict samples: 538
precision = 0.9607,     recall = 0.9057,        f1-score = 0.9324

In class: COVID-19      Total samples: 67       True predict samples: 57
precision = 0.9048,     recall = 0.8507,        f1-score = 0.8769

AVERAGE F1-SCORE = 0.9457
```

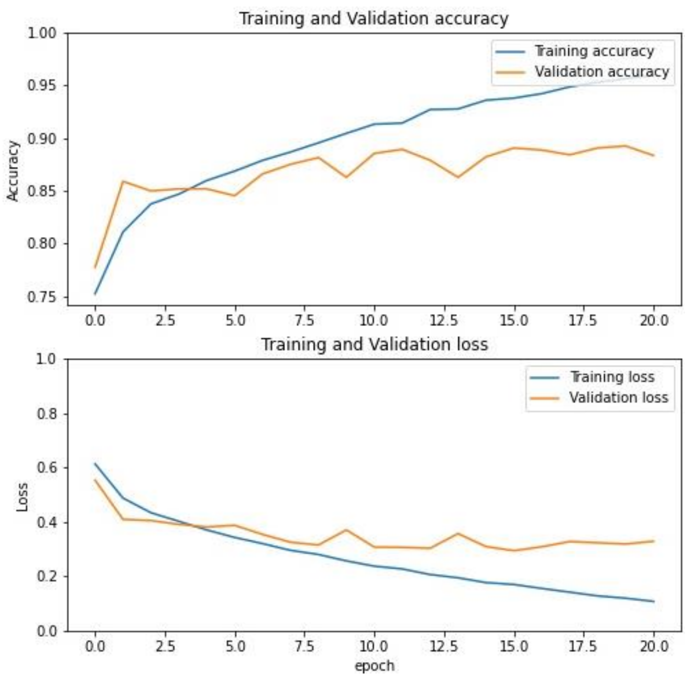
Results of model 3 applied to validation data.

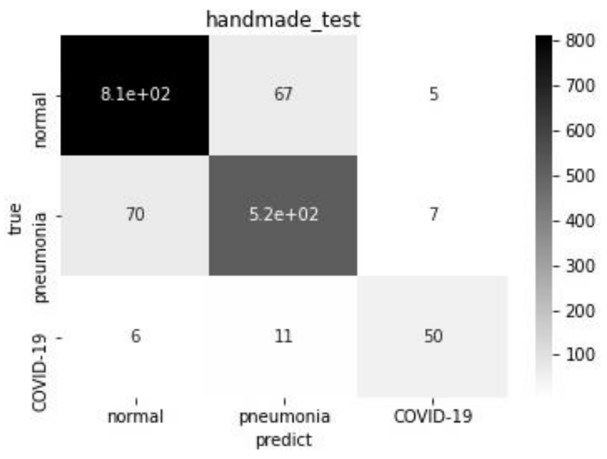
Since the dataset is strongly unbalanced (the samples are not uniformly distributed: the radiographies of the COVID-19 class are much less than the other two classes), evaluating merely the accuracy of the models is not sufficient: for this reason, I chose the F1-score as the main testing metric. The training phase was empirically repeated several times, varying in each instance the hyperparameters, and the combination of hyperparameters previously described in the course of the report are the ones that led to the results listed here, which were the best achievable. As can be observed, the results improve from stage to stage until reaching a final F1-score of 0.9457, which is comparable to the result obtained by the original paper.

4. Model Comparisons

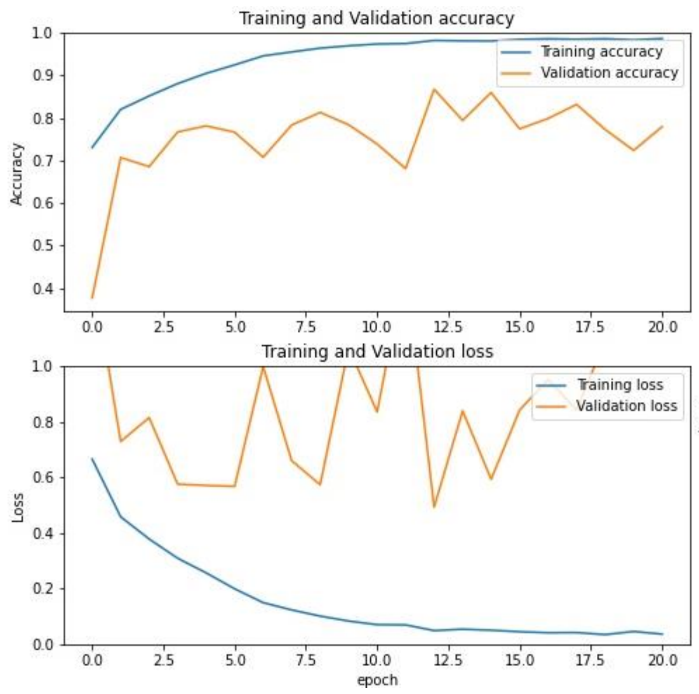
The best model generated by the experimental procedure has been compared with three networks to prove its validity (the hyperparameters are the same as before and they were trained with the *229×229×3* input images):

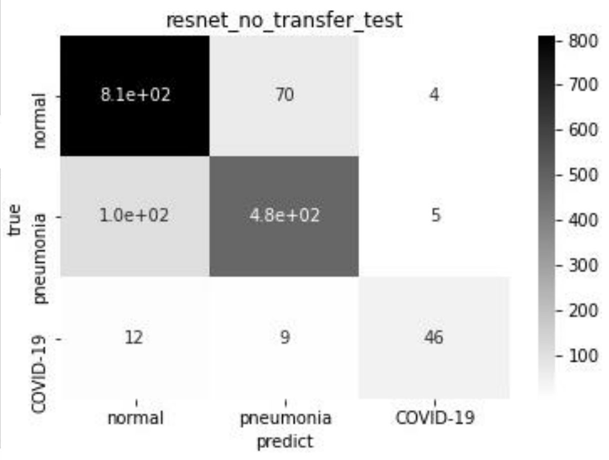
- **A simple convolutional neural network.** It is formed by three convolution layers (respectively with 32, 32 and 64 output filters and *3×3* kernels), a max-pooling, a drop out, a dense layer of 128 neurons and an output layer of 3 neurons. Softmax is the loss function for the output layer while ReLu is used in the rest of the network, it was trained with a learning rate of 1e-4. Results:



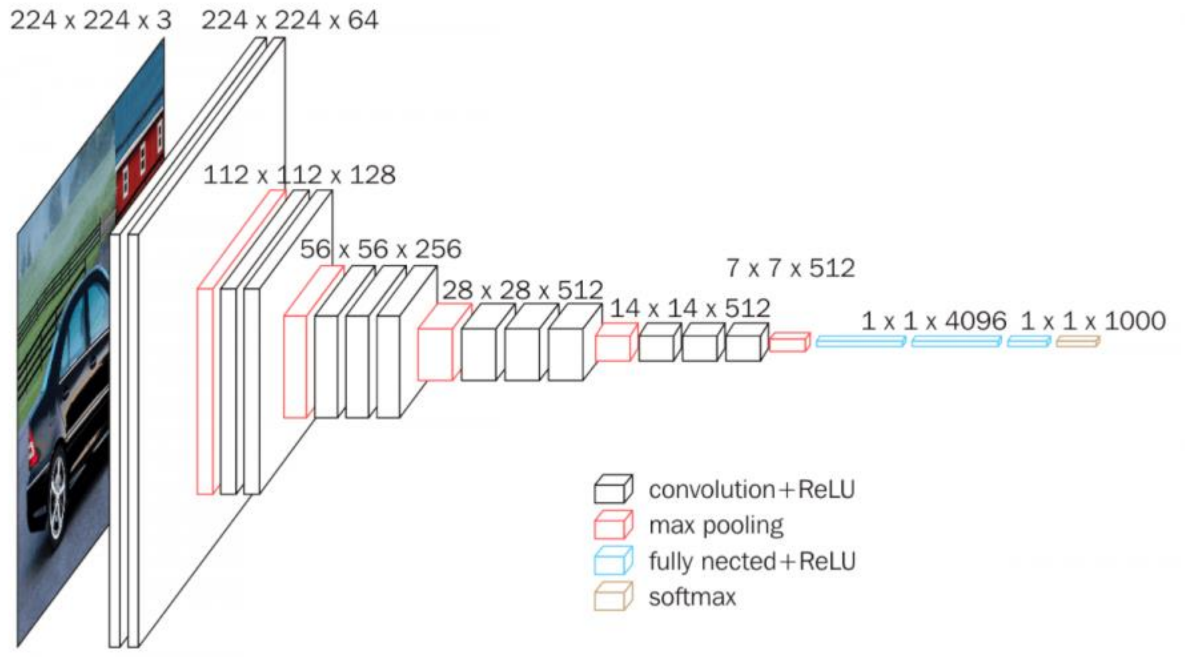


- **The same ResNet50 architecture, but without applying all the transfer learning methods.** This has been done with the intent of proving the effectiveness of transfer learning: ResNet50's weights were randomly assigned and trained with a learning rate of 1e-5 (I also tried using 1e-4 but the network was not able to learn at all). Results:

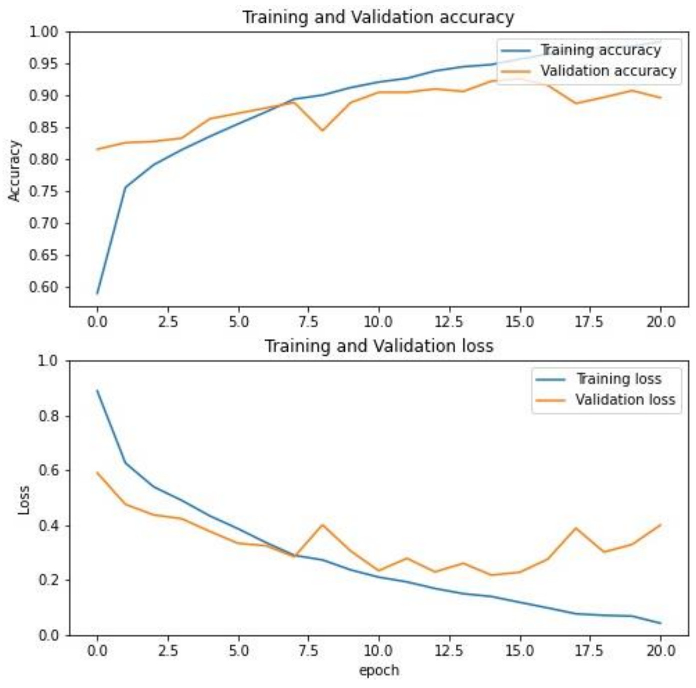


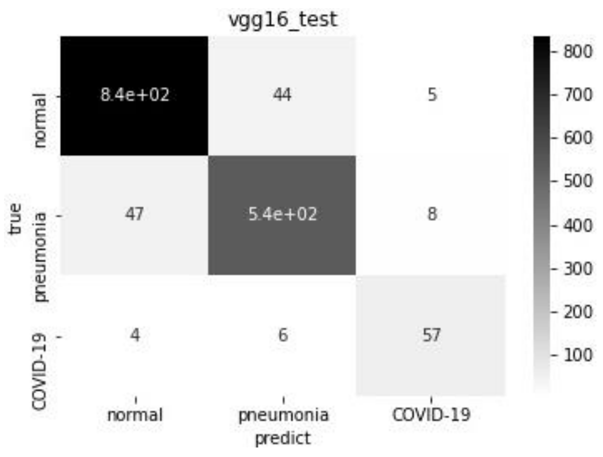


- **VGG16 network (without applying all the transfer learning methods).** As ResNets, also this convolutional network came about in a ILSVRC (2014 edition); I am not going to explain its functionality in depth as for Residual Networks, this image sums up its architecture:



  Its weights were randomly assigned and it was trained with a learning rate of 1e-4 and these results were obtained:





Finally, the following table summarizes the best F1-score and the training time (on my machine) for all of the models:

	Main model	**Simple CNN**	**Resnet50 w/o TL**	**Vgg16 w/o TL**
F1-score	0.9457	0.8926	0.8674	0.9263
Time to train (seconds)	1453	304	2201	2832

As it was easy to imagine, the "main model" of this report remains the one with the best performances, this underlining the importance of transfer learning and the other techniques described by this project.

Although VGG16 almost manages to reach the main model, however its training time is particularly slow (about the double). A similar reasoning can also be applied to ResNet50 without transfer learning, where F1-score gets worse and especially training time increases by approximately 66%. The simple CNN on the other hand, against all odds, manages to get pretty good results in a very short time (only one seventh of the time compared to the other models!): this demonstrates how important it is to pursue simplicity in the implementation of a neural network and that an overly extended model is rarely the right design choice.

5. Conclusions

In this report it was presented an effective way to solve the classification problem of radiographies for the COVID-19, Pneumonia and Normal categories; problem which emerged only recently due to the pandemic nature of the aforementioned virus, which has shaken the modern societies' habits right from their roots.

All the theoretical relevant aspects of Residual Networks, Transfer Learning and Progressive Resizing used in the project were addressed and explained. The publicly available dataset COVIDx V7A was subsequently described, generated and pre-processed for the training phase. One particular Residual Network, ResNet50, was adopted, studied and its architecture modified accordingly to maximize its performance adopting the changes proposed by the "COVID-ResNet: A Deep Learning Framework for Screening of COVID19 from Radiographs" paper and further expanded by researching and experimenting. Likewise, the specific training process was reimplemented and varied. Finally, the excellent classification results that the network was able to achieve across all the classes of the independent test dataset were disclosed and compared with other famous networks.

In conclusion, I would like to emphasize that even though the proposed model is very promising and accurate, it is not meant to be directly employed for clinical diagnosis: in order to make it clinically useful world-wide it would require to train with a lot larger dataset from the real-world cases. The goal of this work was to show that using different training techniques enable us to train models that are computationally efficient and accurate, this proves that it can be confidently stated that by continuing our scientific researches it would be very likely to build increasingly effective tools and significantly contribute to the fight against the COVID-19 virus.

6. References

- Cortés Aguas, K. (2020) A guide to transfer learning with Keras using ResNet50. Available at: https://medium.com/@kenneth.ca95/a-guide-to-transfer-learning-with-keras-using-resnet50-a81a4a28084b (Last Accessed: 26 March 2021).

- COVID-Net original implementation. (2021) Available at: https://github.com/lindawangg/COVID-Net (Last Accessed: 26 March 2021).

- Farooq, M. & Hafeez, A. (2020) COVID-ResNet: A Deep Learning Framework for Screening of COVID19 from Radiographs. Available at: https://arxiv.org/ftp/arxiv/papers/2003/2003.14395.pdf (Last Accessed: 26 March 2021).

- He, K. & Zhang, X. & Ren, S. & Sun, J. (2015) Deep Residual Learning for Image Recognition. Available at: http://arxiv.org/abs/1512.03385 (Last Accessed: 26 March 2021).

- Sachan, A. (2019) Detailed Guide to Understand and Implement ResNets. Available at: https://cv-tricks.com/keras/understand-implement-resnets/ (Last Accessed: 26 March 2021).

- Tensorflow Core. (2021) Transfer learning and fine-tuning. Available at: https://www.tensorflow.org/tutorials/images/transfer_learning (Last Accessed: 26 March 2021).

- Uncini, A. (2020) Neural Networks Course Material: Convolutional Neural Networks. Available at: http://www.uncini.com/dida/nn/nn_2020/ (Last Accessed: 26 March 2021).

- Uncini, A. (2020) Neural Networks Course Material: Convolutional Neural Networks lessons. Available at: https://youtu.be/4mZ0a9XbldA and https://youtu.be/rNHWq3dwhE0 (Last Accessed: 26 March 2021).

- Wang, L. & Zhong Qiu, L. & Wong, A. (2020) COVID-Net: A Tailored Deep Convolutional Neural Network Design for Detection of COVID19 Cases from Chest Radiography. Available at: https://arxiv.org/pdf/2003.09871.pdf (Last Accessed: 26 March 2021).