CT diagnosis of COVID-19

A. Pascuet, D. Cantabella, M. Ballart, R. Cambray

Universitat Pompeu Fabra

Abstract

Context: Coronavirus Disease 2019 (COVID-19) is an infectious disease caused by a coronavirus discovered in December 2019. Besides a Polymerase Chain Reaction (PCR) test, a Computed Tomography (CT) scan can also be used to diagnose the disease, based on clinical findings.

Objective: The aim of this study is to classify 2D CT coronal scans of the lungs into COVID-19 positive or COVID-19 negative using different deep learning models.

Methodology: Various architectures were constructed and evaluated. The built networks were a simple Convolutional Neural Network (CNN), a Residual Network of 50 layers (ResNet50), a second version ResNet-50 (ResNet50 V2) and a Residual Network of 34 layers (ResNet34). Nevertheless, before training the models, images were preprocessed and two groups of images were created to compare the original set of images to one with more restricted conditions for the COVID-19 and non-COVID-19 classification.

Results: Best performance was achieved by the ResNet34 with original set of images, which obtained values of 0.854, 0.831, 0.885 for accuracy, F1-score, and Area Under the Curve metrics, respectively. Furthermore, the subjective selection of the most significant images to characterize the disease proved not to be a good option, since this worsened the obtained results. The results of the ResNet-50 were indicators of a possible erroneous implementation of the algorithm while the simple CNN showed quite good results scored from 0.695 to 0.675.

Conclusions: The implementation of the ResNet34 proved to be much more effective than the rest of the networks, but also the preprocessing of the input images showed its importance on the correct classification. This study suggests that a lack of information regarding the differences between normal and COVID-19 pneumonia's, negatively affects the design of an efficient architecture.

Keywords:
COVID-19, Grand Challenge, CT scan, CNN, ResNet

1. Introduction

Coronavirus Disease 2019 (COVID-19) is an infectious disease caused by a recently discovered coronavirus. Since it was first discovered in Wuhan, China, on December 2019, this virus has expanded worldwide affecting many countries and causing a pandemic. Flu-like symptoms are the most identifiable when detecting COVID-19, such as fever, cough or tiredness, but other not-so-common symptoms can be body aches, nasal congestion, loss of taste or smell and diarrhea. Shortness of breath or chest pain is also a clear sign of a more acute affection of the disease, which can end up causing a pneumonia (1). The commonalty of these symptoms with respect to other respiratory diseases asks for specific detection methods that can confirm the diagnosis.

To detect COVID-19, a PCR test is usually used, which can obtain a result in less than 24 hours. But there are limited resources for PCR tests, which has led hospitals to search for other fast methods to evaluate positive or negative subjects. As pneumonia can be a cause of COVID-19 infection, lung damage is assumed to be caused by the virus, and Computer Tomography (CT) scans have been suggested for its detection. CT scan findings can depend on the stage of affection of the disease. On the early stage, CT findings will include small patchy shadowing areas and interstitial abnormalities, both around the peripheral area of the bilateral lungs. On a progressive period of the disease, the lesions previously seen will increase in range and in number, and multiple ground glass opacity (GGO) with further infiltration into the bilateral lungs will also appear. On some severe cases, pulmonary diffuse consolidation will be observed, and pleural effusion can occur, but it is rare (2).

In the recent months, there has been a lot of studies trying to use any algorithm in order to classify lung CT images into positive or negative COVID-19, to find a faster and more effective detection method, as mentioned above. The common deep learning technique to use is Convolutional Neural Networks (CNN), after some preprocessing of the input images. In order to avoid manual feature extraction and selection methods, Narin et al. (3) compared the results obtained using three different architectures: ResNet50, InceptionV3 and Inception-ResNet V2. With the first method they achieve a 98% of accuracy, being the best among the other results. In (4), Singh et al. discusses the proposed multi-objective differential evolution (MODE)–based convolutional neural networks (CNN) for classification of COVID-19-infected patients from chest CT images, obtaining an accuracy of 93.3%.

The CT diagnosis of COVID-19 challenge from the Grand

Challenge website[1] has as main task to classify CT scan images in either COVID-19 positive or COVID-19 negative. Currently, on the leader board of the up-to-date most notable participants, only the leading group of the challenge, Merkh and his team (5), have published their methdodology. They used a transfer learning technique, as they had a pretrained deep convolutional network called COVID-NET and fine-tuned its parameters. The pretrained model was used for feature extraction, implementing several convolutions and pooling operators to evaluate possible and potential features. A Multi-objective fitness function was calculated at the end, as CNN usually suffers from hyperparameter tuning issues such as kernel size and type, activation function, learning rate etc. With all this process they obtained an accuracy value of 93.3%.

As mentioned, this project wants to find a competent classification for chest CT images into COVID-19 or non COVID-19. Considering the huge variety of options to classify images, our take on this challenge is to, using different network architectures, a simple CNN, a ResNet50, a ResNet50 V2 and a ResNet34, compare each network results and determine the one obtaining the best classification.

2. Materials and Methods

This project's pipeline follows the structure seen in Figure 1, subdividing it into three main parts: the preprocessing, the network training and the testing and classification of images.

The algorithms were written using Python 3.7 language in the Spyder 4[2] environment. Tensorflow[3] and Keras[4] libraries were used to implement the deep-learning, and Scikit-image[5] for the preprocessing of the images.

The database was composed by COVID-19 positive and COVID-19 negative images, separated into different folders. The images were 2D CT scans of different sizes. Also, an excel file was presented, with information on the patients' medical picture, containing its age, if they had fever or not or if they had had a PCR test and its outcome.

For the networks constructed, the parameters for compilation were 30 epochs, a batch size of 32, and an initial size of the images of 64x64x3.

2.1. Preprocessing

For a better predictions outcome, the images had to undergo some preprocessing, as it was seen on literature too. Our preprocessing consisted on different steps, and resulted in the obtention of two different data subsets.

2.1.1. Data separation

First, an initial separation of the images was done, in order to discard the marked images, like Figure 2. These marked images were standing out COVID-19 findings, but were of no use for our algorithm. This resulted in our first data subset.

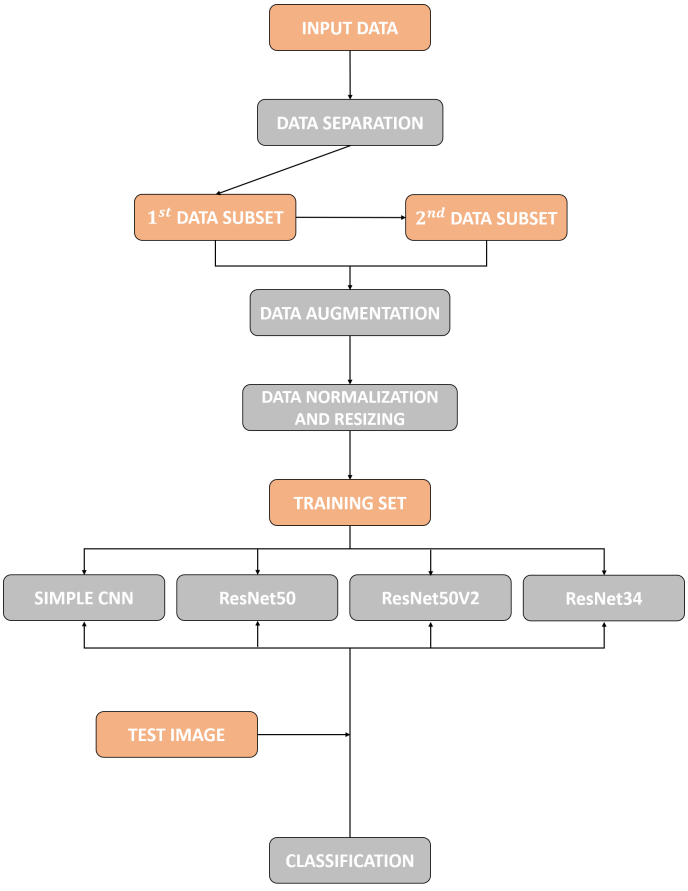


Figure 1: Workflow of the proposed methodology. Input data corresponds to the data-set proportioned by the challenge, which was divided into two subsets, one keeping all the images except for the marked ones and a reduced one which kept only images which seemed to give best characteristics for the classification. From here, both groups follow the scheme in parallel: data augmentation, normalization and resizing is applied to both of them and two different models are trained for each architecture, one for the original data-set and another for the reduced one.

Later, a second separation was done in order to discard poor quality images, like Figure 3. These images were discarded for confusion between COVID-19 or non COVID-19, not showing a proper slice of the CT, for lungs were barely appreciable, or low quality. This separation resulted in our second data subset.

2.1.2. Data augmentation

The next step was to do some data augmentation, because after discarding the mentioned images, the data-sets were poor in quantity. Augmentation of the data was done using the Augmentor library [6]in Python, and the changes applied to the images were flipping, left to right, turning them black and white, rotating, skewing and zooming in.

This way, a bigger subset of images could be obtained. In our case, we decided to get images subsets of a 1000 images in total, 500 for COVID-19 positive and 500 for COVID-19 negative. As mentioned above, two subsets of augmented images were obtained, the first one with all the original images minus

[1] Grand Challenge, https://COVID-ct.grand-challenge.org/
[2] https://www.spyder-ide.org/
[3] https://www.tensorflow.org/
[4] https://keras.io/
[5] https://scikit-image.org/
[6] Augmentor library, https://pypi.org/project/Augmentor/

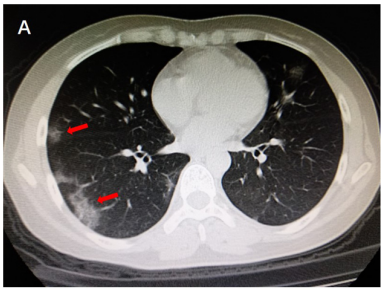


Figure 2: Marked discarded image.

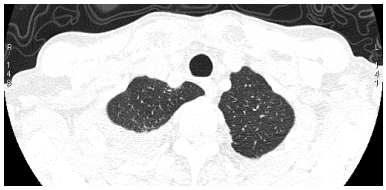


Figure 3: Bad quality discarded image.

the marked ones, and the second one, without both the marked and the low quality images, resulting in our original and separated data-sets, respectively.

2.1.3. Data normalization and resizing

Normalization of the images was the next step to take in the preprocessing, for all the images to have the same range of values.

After normalization, a resizing of the images was required in order to give them all the same size, as the inputs for the networks had to be equal and, in the data-set, the images had different sizes. It was decided for our images to be 64x64x3, and, although it is known that the bigger the size of the input images, the better the predictions, the computational power in the laptops used was not enough to yield greater images.

2.2. Deep learning algorithm

As the aim of our project was to compare different architectures to see their effect on COVID-19 classification, we first needed to establish global settings for all the cases, so comparisons could be correctly assessed.

If we look at the basic structure of a CNN algorithm, we have three main parts: the introduction of the data-set, the compiling of the model and the training and testing of the model. For the introduction of the data-set, images were all of the same size as explained at the preprocessing. Then, it only left to separate them in training and testing sets (80% and 20% of the images, respectively) and input its labels (COVID-19 or non-COVID-19) with a vector with a characteristic number for each label (0s and 1s). To compile the model, we need to define an optimizer and a loss function, which we decided to determine as the Adam optimizer (because seems to be computationally efficient and most of the tutorials that we looked for used it) the categorical crossentropy loss function (which is the general one for classification of images in CNN). Finally, the training and testing of the model was done with a batch size of 32 (since we have a reduced number of images) and a maximum of 30 epochs which were prematurely stopped if the model stabilized before achieving them.

2.2.1. Architectures

As stated before, several architectures to classify our images were used. In any case, all of them shared some layers that will be presented here.

The main step needed for all the models is the convolution one, which is in charge of extracting the image features, acting as a filter over the image. The number of extracted features and the size of the filter variate between each architecture and layer of the algorithm. Moreover, it needs of an activation that will determine how the features are extracted. For that, we usually used the Rectified Linear Unit (ReLU), which considers the positive part of its argument, and, at the last layer, the softmax function, which translates the results in a probabilistic way.

Other common steps are the batch normalization (an element-wise operation that normalizes the input image by rescaling and recentering it), maximum or average pooling (another convolutional layer that subsamples the image), dropout (a regularization technique to reduce overfitting by omitting some units during the training process) and flattening (converts the pooled feature maps to an input layer of 1D).

Simple CNN

As commented, various network architectures were tried in this project. The first one used was a simple Convolutional Neural Network (CNN). This method was used to get introduced in the deep learning field and also to evaluate the classification with the simplest possible architecture. A simple CNN basically passes the set of images through different sequential convolutional layers until they are connected with an output layer that classifies them.

To program this architecture, the code was based on the tutorial presented by the user *randerson112358* (7). The first step was to stablish the architecture as sequential, the first convolutional layer could be added to extract the features from the input image with 64 5x5 ReLU convoluted features. Next, features were subsampled to the half by applying a maximum pooling layer filter of 2 by 2. Afterwards, both last steps were repeated, but with 128 5x5 ReLU convoluted features now.

With that, the features extraction from the images is completed and they could be passed to the neural network, but before, they needed to be linearized in an array by a flattening layer. Next, three neural networks were created, reducing the number of neurons for each one (1000, 500 and 250). The three of them were ReLU activated and had a drop out layer between them of the 50%. Finally, a last neural network of 2 neurons (one for each label) was created and activated, but this time with the soft-max function.

ResNet50

After building the simple CNN, and basing our believes on current state-of-the-art, we decided to build a Residual Network

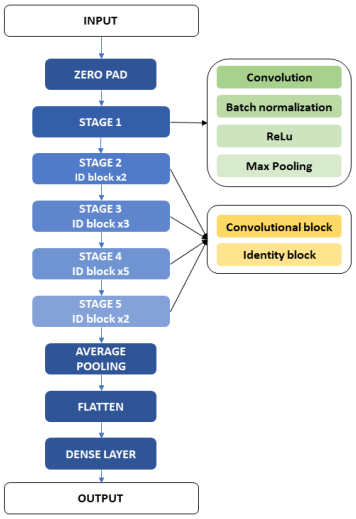


Figure 4: ResNet50 and ResNet50V2 architecture.

with 50 layers. A module graph of the network can be seen in Figure 4. The ResNet50 consists on different steps containing a mix of identity blocks and convolution blocks, whose composition can be seen in Figure Appendix B.2a and Appendix B.2b, respectively (8). These blocks are formed by 2D convolutions, Batch Normalizations and ReLU activations, in the order that can be seen in the mentioned figure.

Residual networks stand out because of a different feature they have compared to normal CNNs. Besides regular linear connections, ResNets also have connected hidden layers, so they can use skip connections. These connections allow the possibility not to use the features from the previous layer if they are meaningless, if the algorithm is not optimized.

Taking a deeper look on the ResNet50 architecture on Figure 4, it can be appreciated that the first stage consists on a Convolutional layer, a Batch Normalization, a ReLU Activation and a Max Pooling operation. The 2D convolution has 64 filters, uses a kernel of size 7, and a stride of size 2. Thus, the size of the input after the convolution has changed. The next layer is a Batch Normalization layer, followed by a ReLU activation, and finally a (3x3) Max Pooling operation with a (2,2) stride is done. After the first stage, a series of repetition blocks enters. These are the residual blocks, which stacked on top of each other make the Residual Network.

ResNet50 V2

In the *Introduction* section, the current leader of the challenge is mentioned (5). Merkh used an already built architecture, the COVID-NET, using transfer learning so this network could be used on the challenge database (6). We contacted one of the authors of the COVID-NET network, Linda Wang, to learn a bit more on it and find if she had an recommendations for us, and she kindly suggested for us to try the ResNet50 V2. Thus, the next step was to build a Residual Network with 50 layers, but the version 2. Version 2 differentiates from version 1 in the order the identity and convolution blocks are build, and how batch normalization is used before every weight layer (**?**). Its new internal architecture can be seen in Figure Appendix B.3a and Appendix B.3b, respectively.

The input parameters for this network, again, were the size of the input image, in our case, 64x64x3, and the number of classes, in our case, 2.

ResNet34

Lastly, another architecture was tried, the ResNet34. ResNet34 is another Residual Network, but this time with 34 layers. Figure 5 shows the model's decomposed architecture (9).

This network differentiates from the ResNet50 by the number of convolutional layers it has in it. Recalling the ResNet50 explanation, the beginning of both networks can be equally described. It starts by a 2D convolutional layer with 64 filters, a kernel of size 7 and s stride of 2, which gives an modified size output. Next, a 3x3 Max Pooling operation is done, so that the output value has the desired dimensions to be the input for the first convolutional block of the Residual Network. As explained before, the ResNet layers are composed by several blocks, which at the same time, consist on a convolution, a batch normalization and a ReLU activation to an input.

In this case, the network is also built by 5 stages, but the total number of block is, as said, 34, resulting in a more shallow model.

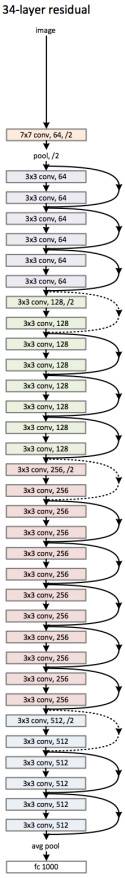


Figure 5: ResNet34 architecture.

3. Experiments and results

A quantitative analysis of the proposed networks was performed to gain a better understanding of how good they were in classifying the data. The chosen metrics were those required to evaluate the performance of the algorithms by the challenge itself. Thus, the accuracy of the algorithm, the F1-score and the Area Under the receiver operating characteristic Curve (AUC) were taken from the different networks. Accuracy measured the percentage of diagnostic predictions that matched exactly with the ground-truth. F1-score was computed by the harmonic mean of precision and recall, where precision was the fraction of true positives among the predicted positives and recall was the fraction of the total number of true positives that were predicted as positive. On the other hand, AUC showed how false positive rate increased as true positive rate increased. The obtained results are shown below in the form of bar plots for a better visualization.

Figure 6 shows the different metrics that were achieved when using the different networks discussed above. The best results were obtained by classifying the original data with a simple CNN and ResNet34 network. As the data obtained by selecting and separating the data worsened the metric results (indicated as "separated"), it was decided to directly use the original data for ResNet34, which is the one shown in black. This network achieved the highest scores with values of 0.854, 0.831 and 0.885 for accuracy, F1-score and AUC respectively. On the other hand, it can be observed that the use of the different ResNet-50s, did not improve the results at all, but on the contrary, results were obtained without consistency, with null-values in the F1-score and very similar values around 0.5 of score among the other metrics. For these reasons, it was considered that an error in the implementation of the ResNet50 algorithm caused these results.

In Figure 7, the two best networks are presented as COVID and new COVID which actually are the Simple CNN and the ResNet34, respectively. Both network metrics are compared with the results published in the challenge. The results obtained were not the best, but the new COVID fitted much better than the COVID when comparing the results with those of other groups. It can be appreciated that the difference between the results obtained with the COVID and the new COVID was considerable. The new COVID obtained much more similar results to the rest of the groups, while the COVID obtained quite bad results when compared to the rest. For the COVID network the achieved metrics had a range from 0.635 for the AUC and accuracy to 0.644 for the F1-score. On the other hand, for the new COVID, these values increased in a range from 0.831 for the F1-score to 0.885 for the AUC, which actually performed better than other published results.

4. Discussion

The low results obtained by most of the networks were probably due to different factors. The results of the simple CNN may be due to its simplicity, so the most relevant features are not considered by the network, so the classification was not accurate. It was seen that a subjective selection of the most representative images of the disease, worsened the results in all the metrics. This could have been due to the low number of images that could cause overfitting in our network, so the ranking worsened.

In the case of the different ResNet50, the results are very similar among the their two versions, but very different from the rest of networks. This makes us to suppose that some error occurred during the computational implementation of the architecture of the network, which prevented a good analysis of its effectiveness. It was seen that the results between them hardly varied few decimals in the case of the accuracy and the AUC, or were considered zero in the case of the F1-score. This fact, actually did not coincide with any logical reasoning, so it was assumed that it was due to an error in the application of the algorithm.

Using ResNet34, the highest results were obtained showing that using bottleneck blocks as a lightweight design patterns worked correctly. It enabled a strong representational capacity showing ResNet34 was better tailored for the classification task.

If we look at the challenge leader board, probably most of them used a wider database than the one that was used here, as it states the leader group that has published the results, and their network designs may be more personalized to the data. These factors suggest that they are obvious indicators of better efficiency when classifying the data.

A solution to improve the obtained results could have been the use of a human-machine collaborative strategy to create a better and more personalized design of the network for our data. It would act as a guide to a design exploration strategy to learn and identify the optimal macroarchitecture and microarchitecture designs with which to construct the final tailor-made deep neural network architecture, and it will enable much greater granularity and much greater flexibility than it is possible through manual human-driven architecture design, while still ensuring that the resulting deep neural network architecture satisfies domain-specific operational requirements.

Moreover, a higher preprocessing of the images in which segmentation and region growing were applied, could help the CNN to focus on the important features (basically, the lighter marks of the lungs). For that, some manual segmentation could be applied to afterwards train a model to segment images, so that this images could be used as the input to our classification models.

On the other hand, during the whole process we have only been looking to the images itself. However, COVID-19 CT characteristics are really similar to pneumonias caused by other factors. This could lead to a wrong diagnose of the disease if the decision it is only based on the image. Letting the clinician add extra information on the status of the patient, as common symptoms (fever, dry cough or dizziness) or other clinical tests (as PCR detection), could lead to a more robust diagnose combined with the information of the classified CT image.

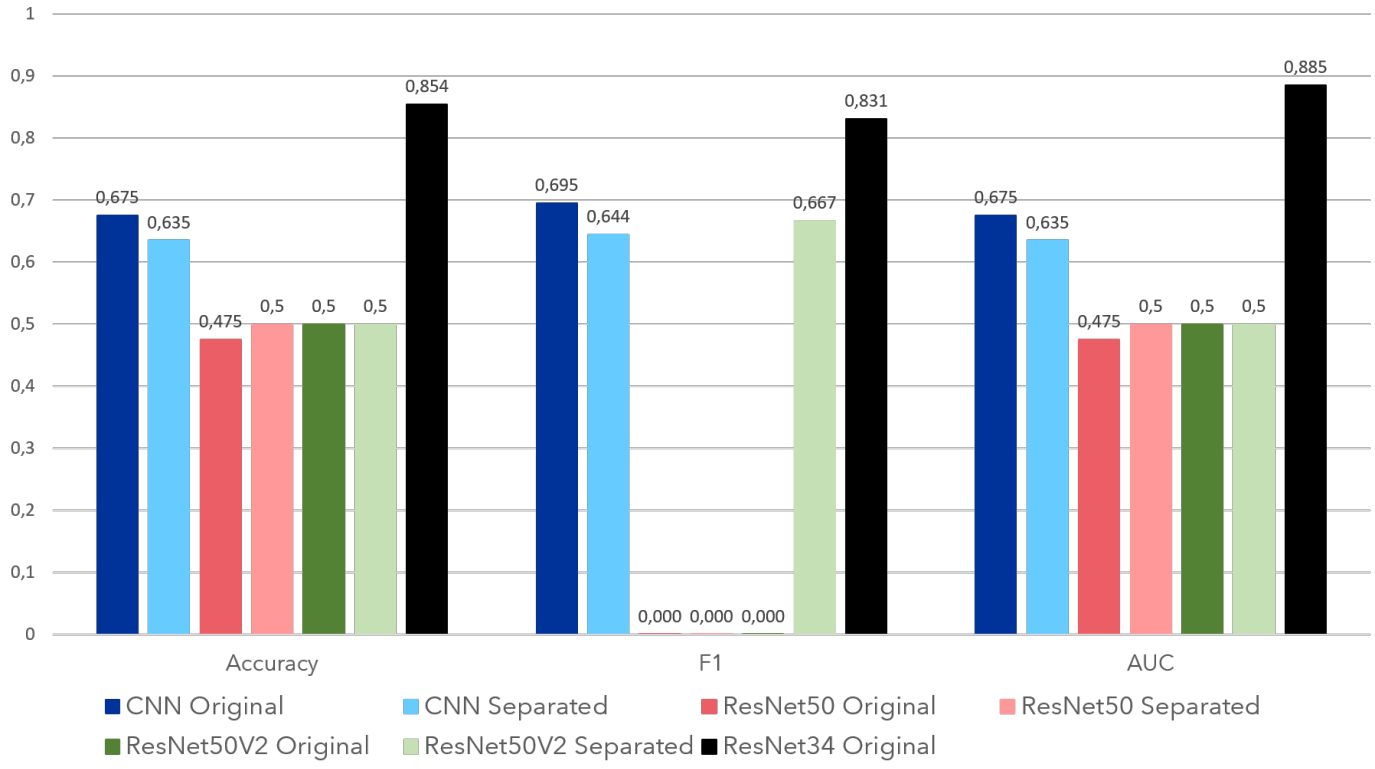


Figure 6: Final metrics of the proposed networks.

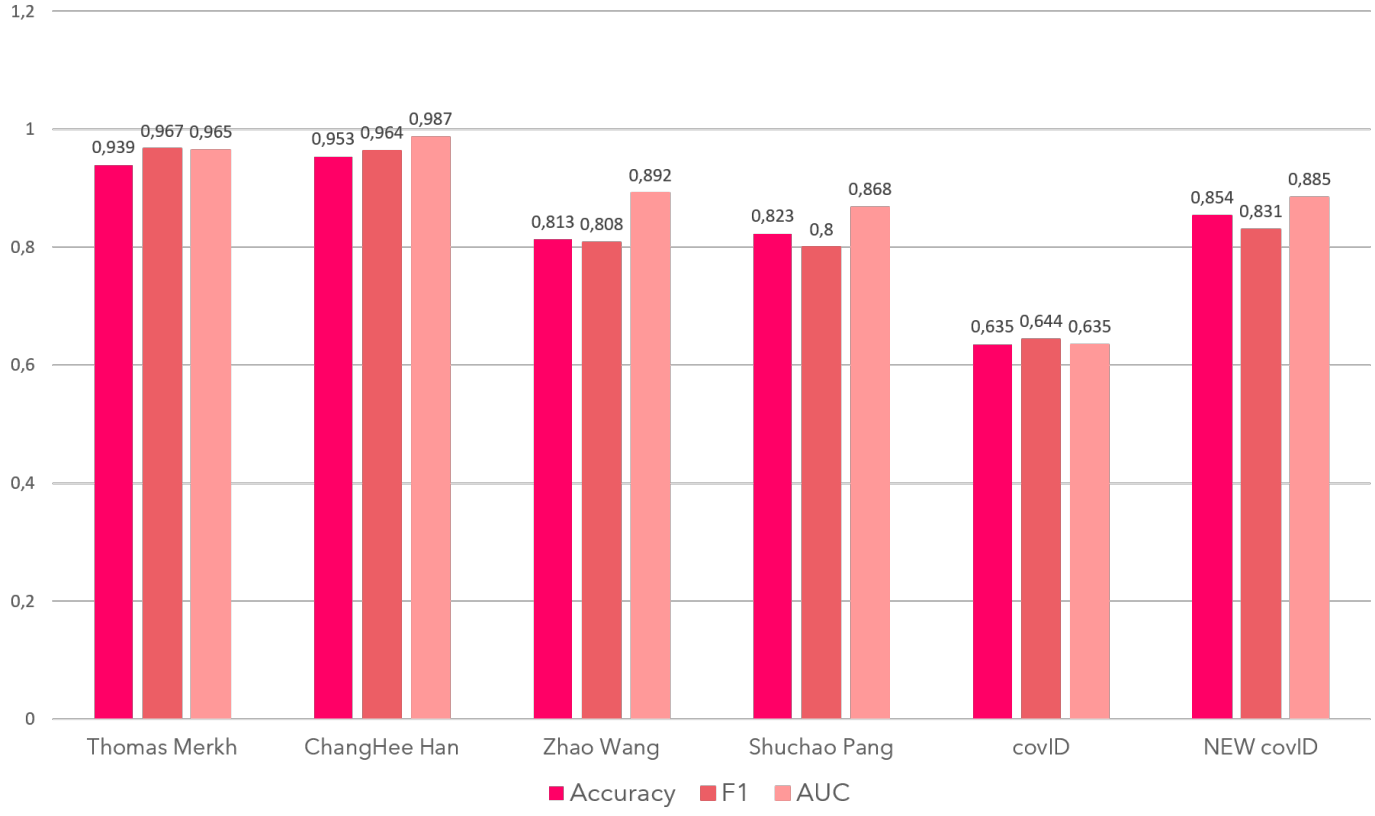


Figure 7: Challenge metrics comparison.

5. Conclusions

CNNs have proven to correctly classify COVID-19 and non-COVID-19 2D CT images, being residual networks a much better option than a simple CNN because of its higher efficiency. It is noticeable that one of the most important steps is not the architecture of the model, but the preprocessing of the input images so that the network gets the correct features of the images for classifying. Moreover, as CT images of COVID-19 infected people show characteristics really similar to other kinds of pneoumonia, more parameters should be taken into account in the classification process. All in all, an automatic classifier trained by deep learning to classify COVID-19 CT images could be helpful in the diagnosis of the disease, but further information should be taken into account to give a final verdict.

References

[1] **QA on coronaviruses (COVID-19)** (2020). [online] Available in: https://www.who.int/emergencies/diseases/novel-coronavirus-2019/question-and-answers-hub/q-a-detail/q-a-coronaviruses

[2] Yang, R., Li, X., Liu, H., Zhen, Y., Zhang, X., Xiong, Q., Luo, Y., Gao, C., Zeng, W. (2020). **Chest CT Severity Score: An Imaging Tool for Assessing Severe COVID 19**. *Radiology: Cardiothoracic Imaging* https://doi.org/10.1148/ryct.2020200047

[3] Narin, A., Kaya, C., Pamuk, Z. (2020). **Automatic Detection of Coronavirus Disease (COVID-19) Using X-ray Images and Deep Convolutional Neural Networks** https://arxiv.org/abs/2003.10849

[4] Singh, D., Kumar, V., Vaishali, Kaur, M. (2020). **Classification of COVID-19 patients from chest CT images using multi-objective differential evolution-based convolutional neural networks**. *European journal of clinical microbiology infectious diseases*: official publication of the European Society of Clinical Microbiology, 39(7), 1379–1389, 2020. https://doi.org/10.1007/s10096-020-03901-z

[5] Thomas Merkh (2020). Math.ucla.edu. Thomas Merkh - COVID19. [online] Available at: https://www.math.ucla.edu/~tmerkh/COVID19.html

[6] Wang, L. and Wong, A. (2020). **COVID-Net: A Tailored Deep Convolutional Neural Network Design for Detection of COVID-19 Cases from Chest Radiography Images** https://arxiv.org/abs/2003.09871v4

[7] User randerson112358 (2019). **Classify Images Using Convolutional Neural Networks Python**. [online] Availabe at: https://medium.com/@randerson112358/classify-images-using-convolutional-neural-networks-python-a89ce

[8] Rokas Balsys (2019). **Convolutional Neural Networks (CNN) explained**. [online] Available at: https://pylessons.com/Keras-ResNet-tutorial/
bibitemresnet50v2 Ankit Sachan. **Detailed Guide to Understand and Implement ResNets**. [online] Available at: https://cv-tricks.com/keras/understand-implement-resnets/

[9] Jeff Heaton (2020). **Applications of Deep Neural Networks** [online] Available at: https://github.com/jeffheaton/t81_558_deep_learning/blob/master/t81_558_class_06_3_resnet.ipynb

Appendix A. Code sources

All the used codes to create the different models and a user interface to use the simple CNN as a classificator of images for a clinical environment can be found at `https://github.com/apascuet/covID`

Appendix B. Supplementary figures

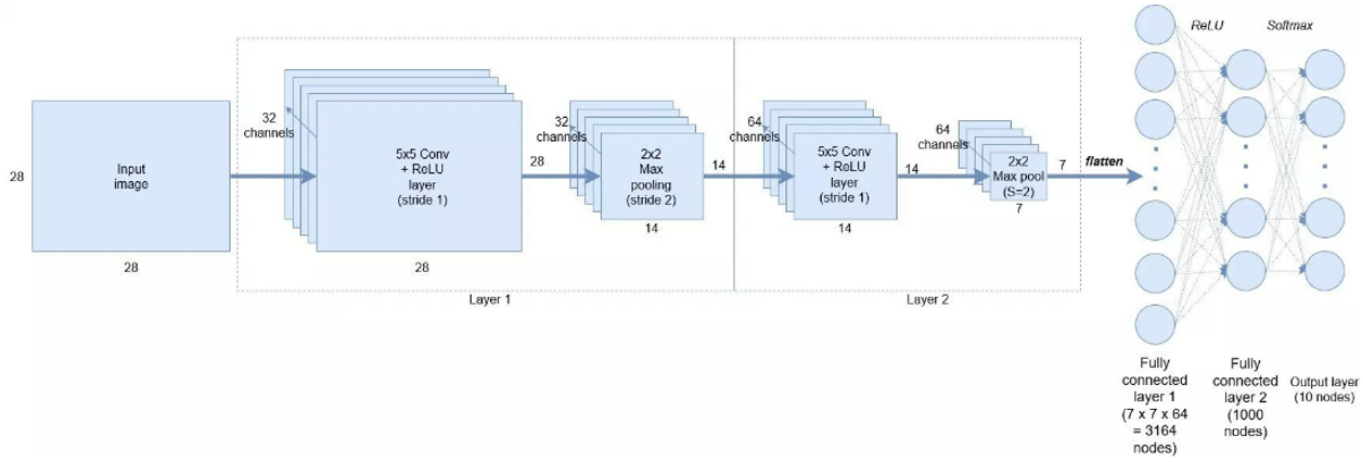


Figure Appendix B.1: Simple CNN architecture from (7). Our model is modified by having 64 channels at the first convolutional layer and 128 at the second one. Also, our output layer it only consists of two nodes because we only have two labels for classification, COVID-19 positive and COVID-19 negative.

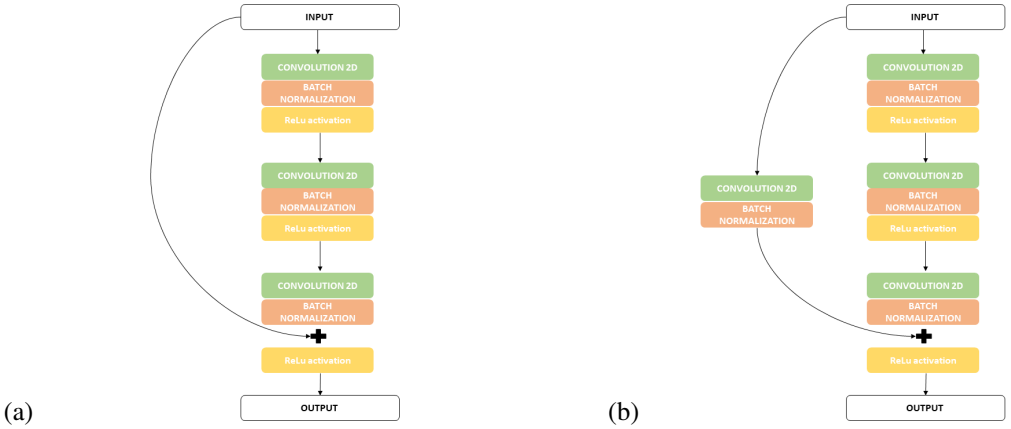
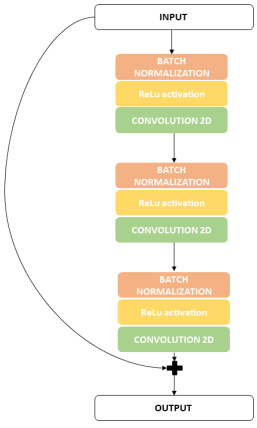


Figure Appendix B.2: (a) Identity block of the ResNet50. (b) Convolutional block of the ResNet50.

(a)



(b)

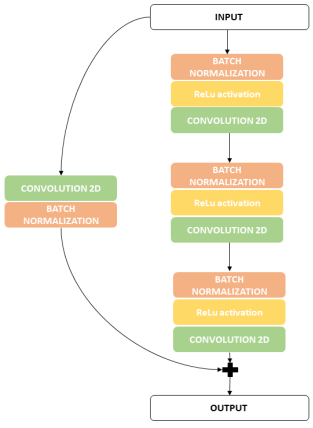


Figure Appendix B.3: (a) Identity block of the ResNet50V2. (b) Convolutional block of the ResNet50V2.