COVID-19 DETECTION USING RESNET18 WITH PYTORCH AND FLUTTER IMPLEMENTATION

Christian Ricardo Conchari Cabrera
Mechatronics Engineering Student
Universidad Católica Boliviana "San Pablo"
La Paz, Bolivia
chrisconchari@gmail.com

Hamed Emmerson Quenta Alvarez
Mechatronics Engineering Student
Universidad Católica Boliviana "San Pablo"
La Paz, Bolivia
hamedquenta@gmail.com

Fabricio Alejandro Jallaza Maldonado
Mechatronics Engineering Student
Universidad Católica Boliviana "San Pablo"
La Paz, Bolivia
fabricio.jallaza@ucb.edu.bo

Eliot Rubén Santos Orellana
Mechatronics Engineering Student
Universidad Católica Boliviana "San Pablo"
La Paz, Bolivia
eliot.santos@ucb.edu.bot

***Abstract*—This report seeks to show the methodologies, results and problems encountered when implementing a CNN capable of classifying chest x-ray images in three classes of lung conditions and lungs in a normal state. For this, it was thought to approach this issue through Transfer Learning methods, taking advantage of powerful networks such as ResNet18. Our results show that the trained model can classify input images in the target classes with considerably good accuracy percentages, finally, the model was implemented in a mobile application to facilitate the use for the end-user.**

***Index Terms*—Transfer Learning, COVID-19, Deep Learning, Flutter, Pytorch.**

I. Introduction

Nowadays, we are living in the COVID-19 pandemic, a typical COVID-19 infection case will damage the walls and linings of the air sacs in human lungs. As the human body tries to fight it, the lungs become more inflamed and fill with fluid. There are multiple methods to detect if a person has been infected with COVID-19, one of them is PCR testing, usually, the results of these tests are sufficient to determine whether or not a patient has been infected with COVID-19, but in many cases, it is also possible to perform imaging tests to support the diagnosis, determine the severity of the disease and assess therapeutic response. In this way, chest X-ray is usually the first imaging test in patients with suspected or confirmed COVID-19 because of its usefulness, availability and low cost. The optimal study includes posteroanterior (PA) and lateral projections in the standing position.

It is very common in today´s artificial intelligence industry to come across implementations of convolutional neural network models for object detection or image classification into different classes. In simple words, it is a matter of giving a computer an input image and having it answer you with whether this image corresponds to a certain class, which if we deal with numbers could be anyone from 0 to 9 or if we deal with transport vehicles, it could be a car, a motorbike, or a bicycle, etc. Because of this, together with the context we are living with the pandemic caused by COVID-19, we thought of generating a tool to support health professionals in the diagnosis of COVID-19 employing chest X-ray imaging tests. In this way, we will use the recently mentioned COVID-19 imaging tests (chest X-radiographs) together with artificial vision, but one of the problems we might encounter in doing this would be to design a CNN model robust and accurate enough to classify chest X-ray images and thus detect COVID-19 infection, for this project designing our model was not seen as an alternative, as it would involve an undesirable amount of time and work, as we have other options on the table that we can take advantage of. One of these options is the use of models that have already been pre-trained with gigantic amounts of images under certain accuracy margins, known as the Transfer Learning technique. For this implementation we would be using a modified model based on Resnet18 CNN, to classify X-ray images in four classes, where three of them corresponds to respiratory complications (viral pneumonia, pulmonary opacity and COVID-19) and the last one corresponds to normal lungs. In this way, we will consequently be able to detect COVID-19 and rule out other options.

Finally, although we were able to implement an artificial intelligence model capable of detecting COVID-19 in chest x-ray images, for this model to be used it is necessary to facilitate its use, so that the end-user, in this case, the health professional, can simply upload a test image and obtain diagnostic support by classifying it as either COVID-19, other respiratory complications or, if applicable, lungs in a normal state. To do this, we thought of implementing a mobile application using the cross-platform framework Flutter based on the Dart programming language, which in terms of feasibility and

implementation time was considered the best option.

II. Background

A. Deep Residual Networks (Resnet18)

Deep Residual Network was undoubtedly the most significant achievement in the computer vision/deep learning community in the previous few years, following AlexNet's acclaimed triumph at the LSVRC2012 classification challenge. ResNet allows you to train hundreds or even thousands of layers while still achieving excellent results [2]. ResNet's accomplishments can be summarized as follows:

- Won 1st place in the ILSVRC 2015 classification competition with top-5 error rate of 3.57% (An ensemble model).
- Won the 1st place in ILSVRC and COCO 2015 competition in ImageNet Detection, ImageNet localization, Coco detection and Coco segmentation.
- Replacing VGG-16 layers in Faster R-CNN with ResNet-101. They observed a relative improvement of 28%.
- Efficiently trained networks with 100 layers and 1000 layers also [3].

ResNet was invented by Kaiming He and his team of Xiangyu Zhang, Shaoqing Ren, and Jian Sun in 2015. Since its debut, various researchers have probed into the secrets of its effectiveness, and the architecture has undergone countless upgrades. Kiming is driven to develop a better CNN after demonstrating that the existing CNN model has a maximum depth threshold [4].

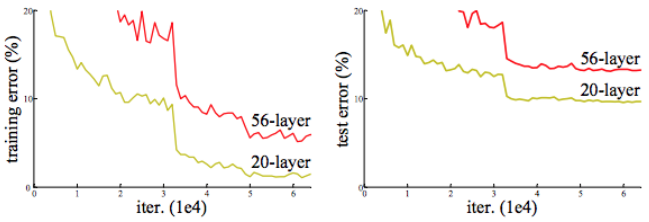


Fig. 1: Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error.[2]

This plot defies our expectations that adding more layers will result in a more complex function, with the failure due to overfitting. Additional regularization parameters and procedures, like as dropout or L2-norms, might be effective in correcting these networks if this was the case.The training error of the 56-layer network, on the other hand, is larger than that of the 20-layer network, indicating a separate phenomenon causing its failure. The best ImageNet models with convolutional and fully linked layers have between 16 and 30 layers, according to evidence[2].

1) Residual learning: The core idea of ResNet is introducing a so-called "identity shortcut connection" that skips one or more layers, as shown in the following figure [1]:

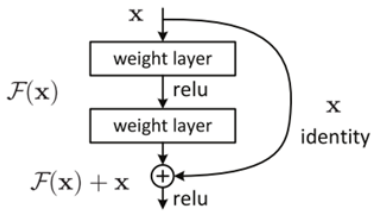


Fig. 2: A residual block architecture example.[3]

So, by processing x with two weight layers, the residual unit yields F(x). After that, it adds x to F(x) to get H. (x). Assume H(x) is your ideal anticipated output, which is identical to your ground truth. Because H(x)=F(x)+x, obtaining the desired H(x) is contingent on obtaining the ideal F. (x). That is, the two weight layers in the residual unit must be able to provide the desired F(x) in order to obtain the ideal H(x)[1].
F(x) is obtained from x as follows.

$$x \rightarrow \text{weight1} \rightarrow \text{ReLU} \rightarrow \text{weight2}$$

The following is how to get H(x) from F(x).

$$F(x)+x \rightarrow \text{ReLU}$$

The authors believe that residual mapping (i.e., F(x)) is more straightforward to optimize than H. (x). Assume that the ideal H(x)=x for the sake of illustration. Then, for a direct mapping, learning an identity mapping would be challenging due to the following stack of non-linear layers[1].

$$x \rightarrow \text{weight1} \rightarrow \text{ReLU} \rightarrow \text{weight2} \rightarrow \text{ReLU...} \rightarrow x$$

It would be tough to approximate the identity mapping with all of these weights and ReLUs in the middle[1].
Now, if we define H(x)=F(x)+x, all we have to do is get F(x)=0, as shown below.

$$x \rightarrow \text{weight1} \rightarrow \text{ReLU} \rightarrow \text{weight2} \rightarrow \text{ReLU...} \rightarrow 0$$

Achieving the above is easy. Just set any weight to zero and you will get a zero output. Add back x and you get your desired mapping.
ResNet's skip connections alleviate the problem of disappearing gradient in deep neural networks by allowing the gradient to flow along an additional shortcut channel. These connections also aid the model by allowing it to learn the identity functions, ensuring that the higher layer performs at least as well as the lower layer, if not better. Allow me to expand on this[4].
The authors suggest that stacking layers should not affect network performance since identity mappings (a non-functional layer) could be stacked on top of the current network and the resulting design would perform similarly.As a result, the deeper model should not have a bigger training error than its shallower equivalents. They believe that allowing the stacked layers to fit a residual mapping rather than the desired underlaying mapping is easier. And the leftover block above expressly authorizes it to do so[2].

2) Versions : We have ResNet-18, ResNet-34, ResNet-50, ResNet-101, ResNet-110, ResNet-152, ResNet-164, ResNet-1202 etc. The name ResNet followed by a two or more digit number simply implies the ResNet architecture with a certain number of neural network layers[4].

3) Aplications : Taking advantage of its powerful representational ability, the performance of many computer vision applications other than image classification have been boosted, such as object detection and face recognition[3].

4) Diagram : ResNet network uses a 34-layer plain network architecture inspired by VGG-19 in which then the shortcut connection is added. These shortcut connections then convert the architecture into the residual network as shown in the figure below[5]:

III. Methodology

A. Data Acquisition and Pre-processing

For the implementation of this project, the following 4-class dataset was used.[6][7] A test set with 30 images of each class was set aside, in addition to a train set with 21045 images distributed as follows:

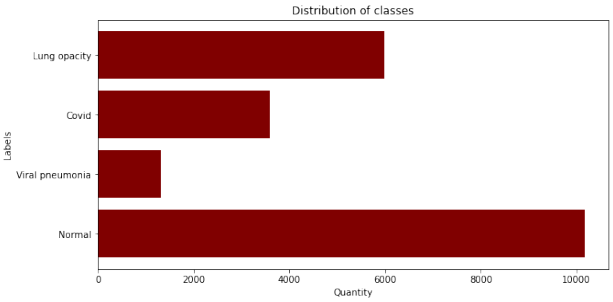
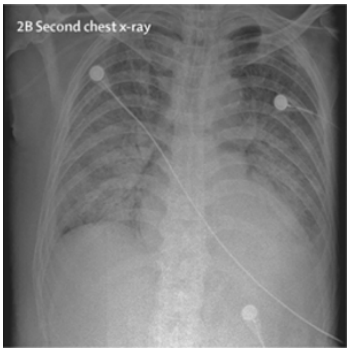
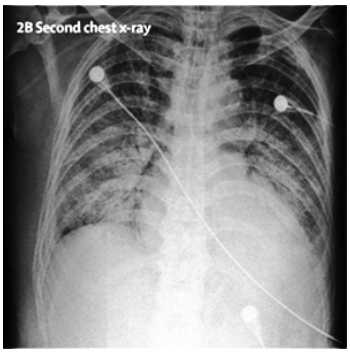


Fig. 3: Distribution of classes in the train set. Only the labels of the training set were considered.

Subsequently, a contrast stretch was performed for all images before training the model, this in order to obtain a better visualization of features in images with low contrast.



(a) Image from dataset before performing contrast stretching



(b) Image from dataset after performing contrast stretching

Fig. 4: Contrast stretching image comparison.

B. Transfer Learning and Model Training

Once the data acquisition and preprocessing stage was carried out, several transformations were applied to the dataset, such as a Resize, a RandomHorizontalFlip and a normalization. This was done using Pytorch dependencies.

Subsequently, based on a transfer learning model, a ResNet18 neural network was implemented starting from a pre-trained model within Pytorch dependencies. By evaluating the predictions made by the model during the training, this process was continued until an accuracy of 0.95 was obtained.

C. Model Compiling for Mobile Implementation

Once we have trained the model, our next step will be to obtain a TorchScript capable of being loaded and used to make inferences, either in a desktop Python environment or in a mobile environment with Dart and the Flutter framework as is being done in the present project. Remember that a Torch Script is a technique to generate serializable and optimizable models from PyTorch code, as described in [8]. That way, instead of running your jit-able modules as an interpreter, you may compile them, allowing for different optimizations and improved efficiency during training and inference.

IV. Results

A. Evaluation Metrics

In order to evaluate the project some metrics must be obtained, these will be the most important feedback to the

project and may be applied to the training or data augmentation part. In order to obtain the required metrics a data extraction must be made, the model on .pt and .pth format will be loaded and the important characteristics will be extracted from it. The relevant metrics for a ResNet-18 neural network are:

1) Accuracy Percentage: Accuracy was obtained by adding the correct trained labels and dividing it into the total trained labels.

$$\sum_{i=1}^{n} x_{acc} = \frac{x_{CL_1} + x_{CL_2} + x_{CL_3} + x_{CL_n}}{x_{L_1} + x_{L_2} + x_{L_3} + x_{L_n}}$$

The test set was set to 120 images where 103 were labeled as correct images with a resulting accuracy of 0.86.

2) Confusion Matrix: The matrix was obtained for both train and test data-set in order to compare the results for both parts.

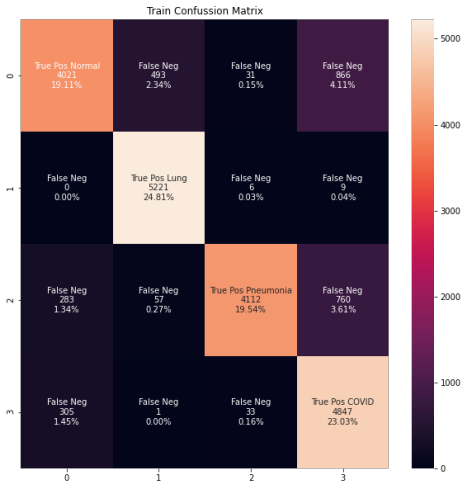


Fig. 5: The figure is for the Train confusion matrix with little deviations from the diagonal path.

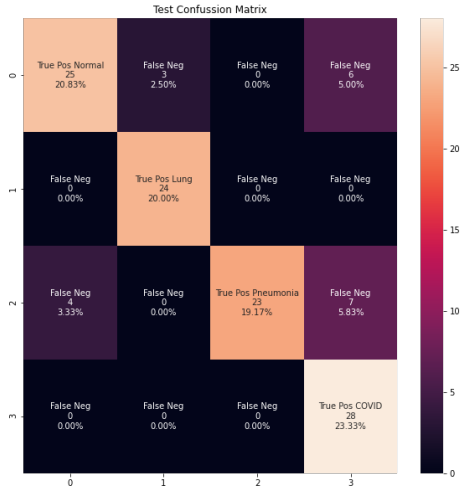


Fig. 6: The figure is for the Test confusion matrix with little deviations from the diagonal path.

3) Classification Report: The classification report contains the precision, recall, F1-Score and support values from each class of the data-set, it also contains the macro and weighted average from the previous mentioned metrics.

	Precision	Recall	F1-Score	Support
Normal	0.90	0.75	0.82	24
Lung opacity	0.90	1.00	0.95	36
Pneumonia	0.96	0.69	0.80	32
COVID-19	0.73	0.96	0.83	28
Accuracy			0.86	120
Macro-avg	0.87	0.85	0.85	120
Weighted-avg	0.88	0.86	0.85	120

TABLE I: Classification Report for test-set

	Precision	Recall	F1-Score	Support
Normal	0.87	0.74	0.80	5238
Lung opacity	0.91	1.00	0.95	5290
Pneumonia	0.98	0.79	0.87	5250
COVID-19	0.75	0.93	0.83	5267
Accuracy			0.86	21045
Macro-avg	0.88	0.86	0.86	21045
Weighted-avg	0.88	0.86	0.86	21045

TABLE II: Classification Report for train-set

4) GradCAM: A localization with Gradient-based Class Activation Maps was obtained

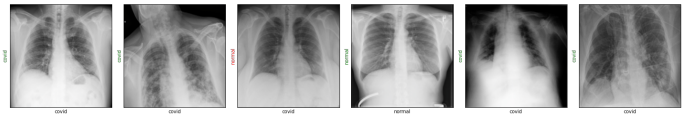


Fig. 7: This figure shows a black and white prediction of the model.

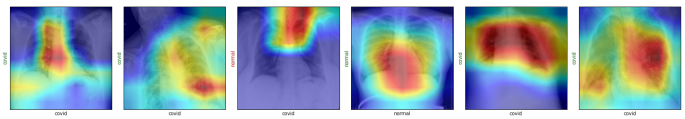


Fig. 8: This figure shows a gradient prediction of the model.

The mentioned metrics tell us that the learning process is optimal and the model is looking for the correct sections of the images. As we can see in the GradCAM section, the confusion matrix shows little deviations from the diagonal path and compare the accuracy to other implementations found online with values between 85% and 96% shows an optimal result. This feedback was applied into the training including also a cpu and gpu training and obtaining similar values in the accuracy percentage.

B. Flutter App Implementation

In order to make the usability of the model easier for the end user, a mobile interface has been created using the

Flutter framework, for this purpose we made use of the pytorch_mobile library, which allows us to work loading the Pytorch script generated with the information of the trained model, to make inferences taking as input images entered by the user and giving as output the corresponding classification label.

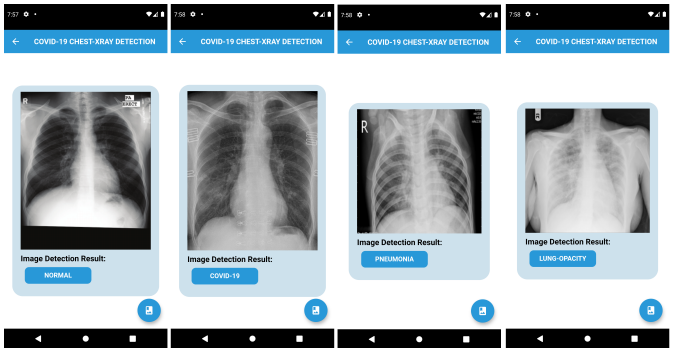


Fig. 9: This figure shows a gradient prediction of the model.

V. Discussion

As presented in the previous section, a model capable of classifying chest radiography images into the four classes corresponding to lungs infected with COVID-19, viral pneumonia, lung opacity and normal lungs has been trained. It was also shown that the model achieved an accuracy percentage of 95%, in addition to quite good percentages in the confusion matrix and classification report values, which demonstrate that the training of the model has been sufficiently good and this is reflected in the validation carried out with the test set. It is worth mentioning that there are points that could be improved in the future in this project. We could consider balancing the number of images for the four existing classes to a sufficiently high number, such as 10,000 images per class, even if we were thinking of making an implementation that could be used to support diagnosis in the field, we recommend increasing the dataset to 50,000 images per class, that way we could cover many more cases and we could avoid the existence of FP, NP, TP & TN, which in such a sensitive subject as this can generate many problems. Another problem that is worth mentioning is the size with which the trained model was exported by compiling the file covid_model. pt file, which in this case was generated with a memory size of 40MB, which is quite a lot if we take into consideration that it is an implementation in a mobile device, to reduce the size of the exported model it would be necessary to quantize it, in this way we would change it from Float to Uint, which would reduce its size in memory, but on the other hand it would lose a little accuracy, although it is worth mentioning that since a high percentage of accuracy was achieved in the case of the trained model for this project it would not be a significant loss and good results could still be obtained.

VI. Conclusions

In this project we have presented an implementation of a Convolutional Neural Network (CNN) model generated with Transfer Learning techniques, modifying the pre-trained ResNet18 network to classify chest X-ray images into four classes: normal lungs, lungs infected with COVID- 19, with viral pneumonia and pulmonary opacity. The current scenery in which we find ourselves requires us to contribute through new solutions, in our case applying artificial intelligence to support the medical diagnosis of respiratory diseases such as COVID-19 and the others mentioned in this report, and in this way provide our support for this fight that health personnel are carrying out. To show the classification efficiency of our trained model we have presented evaluation metrics such as accuracy percentages, confusion matrices and GradCAM & Classification report values in which it has been possible to show that the predictions made with the images of the test set are sufficiently good to say that the model could work appropriately with images captured from the field, although as mentioned in the discussion section there are changes to be made before releasing the model to the real world. We believe that with this work we will have stepped into a study that can be expanded in the future. improving the current project with the points previously raised and thus being able to implement the model in a mobile application that can be reliable and can truly support the diagnosis of medical personnel.

References

[1] Shuzhan Fan (Nov 2,2018), M. Dietz, "Understand Deep Residual Networks — a simple, modular learning framework that has re-defined…", Medium, 2021. [Online]. Available: https://medium.com/@waya.ai/deep-residual-learning-9610bb62c355. [Accessed: 11- Jun- 2021]

[2] V. Feng, "An Overview of ResNet and its Variants", Medium, 2021. [Online]. Available: https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035. [Accessed: 11- Jun- 2021]

[3] J. Prakash, "Understanding and Implementing Architectures of ResNet and ResNeXt for state-of-the-art Image…", Medium, 2021. [Online]. Available: https://medium.com/@14prakash. [Accessed: 11- Jun- 2021]

[4] M. Hussain, "What is Resnet or Residual Network — How Resnet Helps?", GreatLearning Blog: Free Resources what Matters to shape your Career!, 2021. [Online]. Available: https://www.mygreatlearning.com/blog/resnet/#:~:text=ResNet%2C%20short%20for%20Residual%20Network,Residual%20Learning%20for%20Image%20Recognition%E2%80%9D. [Accessed: 11- Jun- 2021]

[5] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[6] M. E. H. Chowdhury et al., "Can AI Help in Screening Viral and COVID-19 Pneumonia?," in IEEE Access, vol. 8, pp. 132665-132676, 2020, doi: 10.1109/ACCESS.2020.3010287.

[7] Rahman, A. Khandakar, Y. Qiblawey, A. Tahir, S. Kiranyaz, S. B. Abul Kashem, M.T. Islam, S. Al Maadeed, S. M. Zughaier, M. S. Khan, and M. E. Chowdhury, "Exploring the effect of image enhancement techniques on covid-19 detection using chest X-ray images, "Computers in Biology and Medicine, vol. 132, p. 104319, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S001048252100113X

[8] "torch.jit.script — PyTorch 1.8.1 documentation", Pytorch.org, 2021. [Online]. Available: https://pytorch.org/docs/stable/generated/torch.jit.script.html. [Accessed: 11- Jun- 2021]

Supplementary Material

https://github.com/ChristianConchari/COVID-19-detection-with-Chest-X-Ray-using-PyTorch