SARS-COV-2 Ct-Scan Dataset

Gitali Naim & Avital Abergel

1 Introduction

Severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) is the virus that causes COVID-19 (coronavirus disease 2019), the respiratory illness responsible for the COVID-19 pandemic. Also colloquially known simply as the coronavirus, it was previously referred to by its provisional name, 2019 novel coronavirus (2019-nCoV), and has also been called human coronavirus 2019 (HCoV-19 or hCoV-19).

CT scans are used for patients with serious symptoms of Covid-19, to evaluate the severity of infection. CT scans gives an estimate about the blockage in the lungs and doctors can treat the patient accordingly.

Recently, deep learning-based algorithms have bestowed enormous progress in medical data interpretation and automation due to their capability to elicit valuable features from the available medical datasets. The proposed work considers CNN, a specific class of neural networks having a grid-like structure for processing the data. Data could be a one-dimensional time series or a two-dimensional array of pixels, as found in the images. This type of neural net employs the convolution operation, which is a kind of linear operation. CNN uses convolution instead of standard matrix multiplication for calculation in at least one layer of the network. CNN forms the basic building block of most of the deep and complex image-based deep learning algorithms.

Success in the project will allow savings in time, money and resources, and most importantly will allow for a quick diagnosis resulting in earlier treatment which may lead to saving a human life.

In this project, our main goal is to identify, using deep learning, if a person is infected by SARS-CoV-2 through the analysis of his/her CT scans.

2 Related work

In recent years, as a result of the relevance and importance of SARS Covid, a number of articles and studies have been published on modeling methods in deep learning on CT images of a patient in order to predict the severity of a corona patients.

One article[1] that has been published in February 2021 proposes an automated diagnosis of COVID-19 infection from CT scans of the patients using deep learning technique. The proposed model, ReCOV-101, uses full chest CT scans to detect varying degrees of COVID-19 infection. To

[1] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7903153/

improve the detection accuracy, the CT-scans were preprocessed by employing segmentation and interpolation. The proposed scheme is based on the residual network that takes advantage of skip connection, allowing the model to go deeper.

Another relevant article[2] that has been published in Nature proposes a deep learning model that is based on CT scans to predict severity. They then construct the multimodal AI-severity score that includes 5 clinical and biological variables in addition to the deep learning model. They show that neural network analysis of CT-scans brings unique prognosis information, although it is correlated with other markers of severity explaining the measurable but limited 0.03 increase of AUC obtained when adding CT-scan information to clinical variables. They show that when comparing AI-severity with 11 existing severity scores, they find significantly improved prognosis performance.

The two articles we described above deal with different methods of deep learning in order to study differences between CT scans of corona patients and healthy people. Some of these methods were also learned in the course, so we saw a point in using them in our work.

3 Dataset and Features

The data that we used in this project have been collected from real patients in hospitals from Sao Paulo, Brazil.[3] Our data contains two classifications and is divided almost in a one-to-one ratio: 1252 positive CT scans of sick people and 1229 negative CT scans of healthy people.

We used a number of different manipulations on the data so that we could then apply different learning methods. First of all, we resized the images to have a uniform size of 128 * 128 pixels. Then We divided our data into 80% training and 20% validation. We then decided to do augmentation in order to increase our data range, which may improve how the model is learned. We decided not to use common augmentation methods (such as flips, rotation, shifts etc.) as it is less suitable for case of lung scans which are quite uniform and symmetrical so these methods will not only not benefit the model but will confuse it.

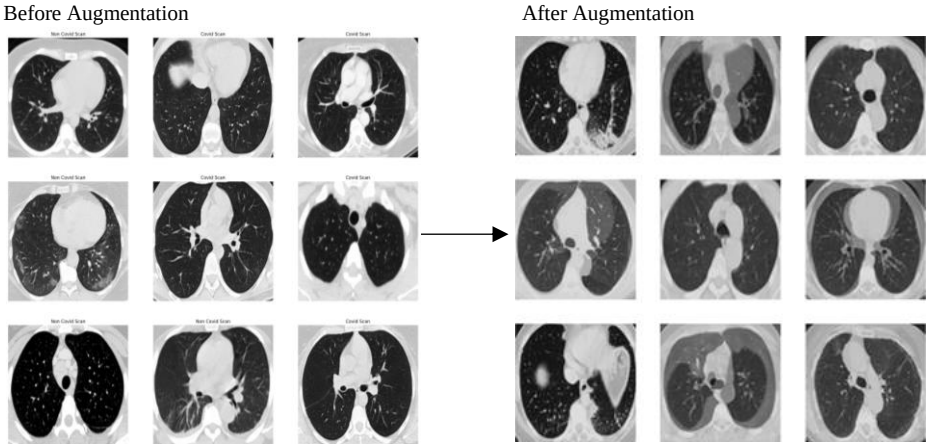
Therefore, we finally decided to use a relatively new augmentation method called MixUp that was first introduced in 2018. The implementation of MixUp is really simple, but still it can bring a huge benefit to our model performance. The technique is quite systematically named - we are literally mixing up the features and their corresponding labels. Neural networks are prone to memorizing corrupt labels. MixUp relaxes this by combining different features with one another (same happens for the labels too) so that a network does not get overconfident about the relationship between the features and their labels.

[2] https://www.nature.com/articles/s41467-020-20657-4
[3] https://www.kaggle.com/plameneduardo/sarscov2-ctscan-dataset

MixUp is specifically useful when we are not sure about selecting a set of augmentation transforms for a given dataset, medical imaging datasets, for example.

$$\tilde{x} = \lambda x_i + (1-\lambda)x_j, \qquad \text{where } x_i, x_j \text{ are raw input vectors}$$
$$\tilde{y} = \lambda y_i + (1-\lambda)y_j, \qquad \text{where } y_i, y_j \text{ are one-hot label encodings}$$

The lambda parameter in the formula is the percentage selected from each image. Its value ranges from 0 to 1, but it is important that there is a preference for one of the images, i.e. not near or equal percentages, but rather at the edges (in our code, the selection of the lambda is randomly in a uniform distribution between 0.1 and 0.3). X_i and X_j represent vectors with numbers that represent the images and after we have activated the formula we get an $\tilde{x}$ which is basically the combination vector of the two images. Similarly for $\tilde{y}$ that represents the label of the image. When the model learns he will categorize the combined image according to the label with the highest percentage.



4 Methods

4.1 Perceptron – non deep learning method

At first we tried to implement a non-deep learning method, so we chose to implement the "Perceptron" learning algorithm which is intended to supervised learning of binary classifiers and was therefore appropriate to our case.

Since our images are 128 * 128 pixels, and each pixel consists of 3 numbers representing the RGB values, we inserted a vector input with size of 41,952 (3*128*128). The output of the perceptron is a vector with 2 cells, each one of them is the probability of being each of the possible labels (sick or healthy). The changes from input to output in perceptron occurs through linear action.

When we ran the perceptron algorithm we got results of about 50% accuracy. Also, the loss does not appear to be declining significantly. Of

course this is a pretty naive result that we would probably get in any random experiment without using learning.

4.2 CNN – deep learning method

Because of the naive results we got when we did a non deep learning method, we decided to try a CNN model**.**

Our network is made up of 5 layers, with each superlayer containing in particular 5 layers, so in total our network contains 25 layers. The difference between each overlay is the size of the data, where the output of a particular layer should be the same size as the input of the next layer. We started with a 3*128*128 input and finished with a 2 size output, as explained in the previous model.

The first four layers consist of the following layers:

1. "Conv2d" - This layer increases the depth.
2. "BatchNorm2d" - normalizes the values according to the parameters studied.
3. "ReLU" - Causes non-linear activation and allows the depth of the network.
4. "MaxPool2d" - In the case of our parameters, causes the length and width to be divided by 2.
5. "Dropout" - randomly deletes 50 percent of the entries, in order to make the training phase more difficult in order for there to be better success in validation.

Between the fourth superlayer and the fifth superlayer there is a transition from a three-dimensional state to a one-dimensional vector state by performing a reshape. After activating all the layers and getting the output at the intended size, we activated the "softmax" function which turns the numbers we got into probabilities. This is a relatively common architecture in building a CNN network.

The sizes of the convolution layers we chose are 16,32,64,128 and then in the transition between the fourth and fifth layer we move to one dimension vector. Additional parameters we set to define the length and width of the image:

- Kernel size = 5
- Stride = 1
- Padding = 2

These values ensure that the length and width of the image do not change as long as the maxpool2d function explained above is not enabled.

The learning rate we chose for the model is 0.1, and our optimizer is SGD since these parameters gave us the best results. SGD (Stochastic gradient descent) is an iterative method for optimizing an objective function with suitable smoothness properties, this method has become an important optimization method in machine learning.

The loss function we chose is MSE. In statistics, MSE (the mean squared error) measures the average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value. The reason we chose this method is that it is relatively popular, suitable for our case, and also because we have used it in the past in this course and other courses.

We chose 64 epochs because it allowed a good learning. We adjusted the parameters while performing a number of runs in different combinations and different values until optimal results were obtained.

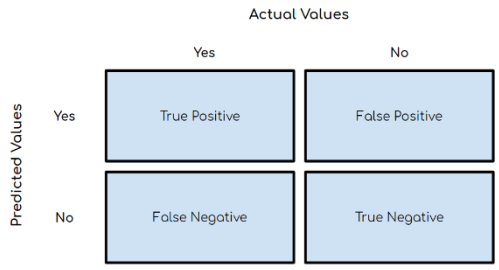
$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

5 Results and analysis

In this project we compared two models of machine learning. The first model is the Perceptron algorithm, this algorithm does not use deep learning, and since our data is not linearly separable the results we obtained when running the algorithm were not good, and even naive. In other words we would get such results randomly without performing learning.

In contrast, when we used deep learning through CNN we got great results. Our accuracy percentages reached over 90%, and the loss value was very low as expected from a machine that learned. For example, in one of the runs we received that out of 497 images the model detected 446 images at the validation stage, i.e. the percentage of accuracy in this run is 90%. This is a high percentage of accuracy as we would expect in a binary classification problem.

Confusion matrix is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one. Each row of the matrix represents the instances in an actual class while each column represents the instances in a predicted class, or vice versa – both variants are found in the literature.



The prediction matrix in our case is:

Predicte \ Actual	Covid	Non covid
Covid	224	24
Non covid	27	222

$$Accuracy = \frac{\#\ of\ correct\ predictions}{total\ \#\ of\ predictions} = \frac{TP + TN}{TP + TN + FP + FN}$$

As we explained above, the percentage of accuracy came out 90%. Using the confusion matrix it can be understood that TP is 0.45, FP is 0.048, FN is 0.054 and TN is 0.446. From here the precision and recall can be calculated. Precision is the fraction of relevant instances among the retrieved instances.

$$Precision = \frac{TP}{TP + FP}$$

In our case, the precision is 0.903.
Recall is the fraction of relevant instances that were retrieved.

$$Recall = \frac{TP}{TP + FN}$$

In our case, the recall is 0.892.

6 Discussion

Due to the shortness of time, we implemented non-deep learning of one type as well as deep learning of one type. If we had more time, we would run more models and compare them and we might even get better accuracy results. Also, if we had a more powerful processor so that the models would run at a faster time, we could better adjust the parameters and pinpoint them more optimally. In addition, we would try to increase the variety of existing CT images by using additional databases, and we might even slightly change the research question so that it is not binary. For example, we would compare corona patients, flu patients, and healthy people.

7 Conclusion

In this project we used medical data, which includes over a thousand CT scans of corona patients and healthy people. The aim of the project was to apply methods that were learned in the course and succeed in determining a classification for the selected data. It can be deduced from the findings of the project, as well as from the literature known that when performing analyzes on medical data it is advisable to use methods of deep learning, since they have better performance.

8 References

- Our data: https://www.kaggle.com/plameneduardo/sarscov2-ctscan-dataset
- Link to the code via google colab: https://colab.research.google.com/drive/1wgBXHFMoQktjOP3mpAYBBiNkZFql1sPv?usp=sharing
- https://www.nature.com/articles/s41467-020-20657-4
- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7903153/
- https://keras.io/examples/vision/mixup/
- https://towardsdatascience.com/understanding-the-confusion-matrix-and-how-to-implement-it-in-python-319202e0fe4d
- https://towardsdatascience.com/what-the-hell-is-perceptron-626217814f53