November 25, 2021

COVID-19 CT SCANS CLASIFFICATION

OBJECTIVES

The following document briefly describes the experimental sessions over a public dataset, including workflow, activities and model evaluation for Covid-19 screening in CT scans. The challenges, methodology and results are discussed.

DATA AND CHALLENGES

A systematic literature review of applied deep learning approaches in the field is reported in the previous stage. Based on the findings, an appropriate dataset is identified. The SARS-CoV-2 CT consists of 2482 CT scans in PNG format collected from hospitals in Sao Paulo, Brazil (Public Hospital of the Government Employees of Sao Paulo (HSPM) and the Metropolitan Hospital of Lap). The balanced data distribution is shown in the figure below.

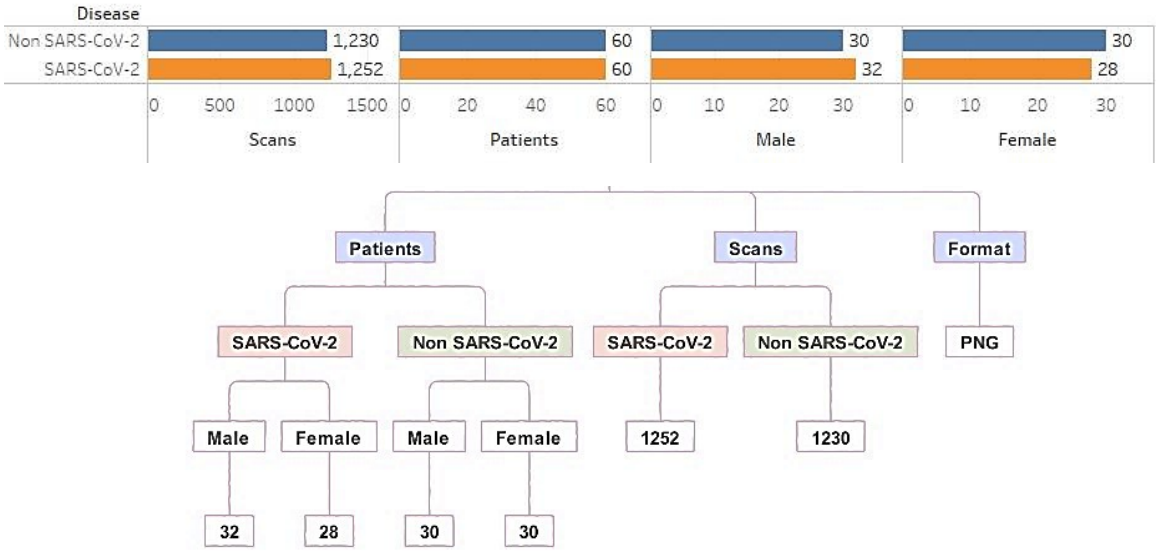


Figure 1. Data distribution

The primary dataset challenge is connected with the following issue:

The same patient may appear in the training and test sets simultaneously as one patient is presented by several scan slices, and the patients and their images are not grouped. That may produce misleading outcomes, especially in splitting training and test sets at random. It can lead

to an overestimated result due to data leakage from the same patient/individual into training and test sets.

A KMeans-based patient clustering method is applied to address the problem. Then the group shuffle split is applicable to ensure no images from the same group appear in both training and validation sets.

METHODS AND ARCHITECTURES

Transfer learning is usually done for tasks with too little data to train a full-scale model from scratch. It consists of taking features learned on one problem and leveraging them on a new problem. The most common incarnation of transfer learning is the following workflow:

- instantiate a base model with pre-trained weights
- freeze the base model
- create a new model on top of the output of one (or several) layers from the base model
- train the model on new data

Different transfer learning approaches depend on the data and the problem. Therefore, modifications are possible.

The popular lightweight workflow could be used as an alternative, including the steps: instantiate a base model and load pre-trained weights into it, run the new dataset through it and record the output of one or several layers from the base model (feature extraction), finally, use that output as input for a new, smaller model.

The second method is a lot faster and resources "cheaper." But it is not appropriate for the current issue as it doesn't allow dynamically modifying the new model's input data during training, which is required when data augmentation is needed. The last one is essential in scenarios with too little data. It is a technique to increase the diversity of training sets by applying transformations, such as image rotation, thus enlarging the data volume.

The main advantage of non-trainable layers is the considerable decrease in computational time. But such methods are appropriate enough to deal with data similar to a subset of the Imagenet (data) on which the pre-trained model is learned. If the data is very different from the Imagenet, then freezing is decreasing accuracy. If there is enough computation resource and time, unfreezing layers allow optimizing the whole feature space, so finding better optima.

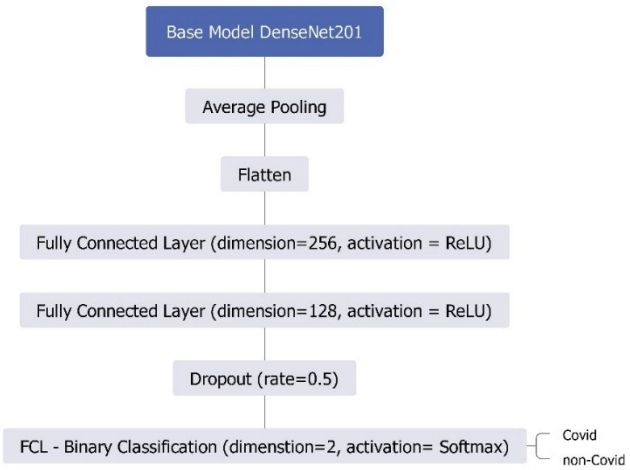
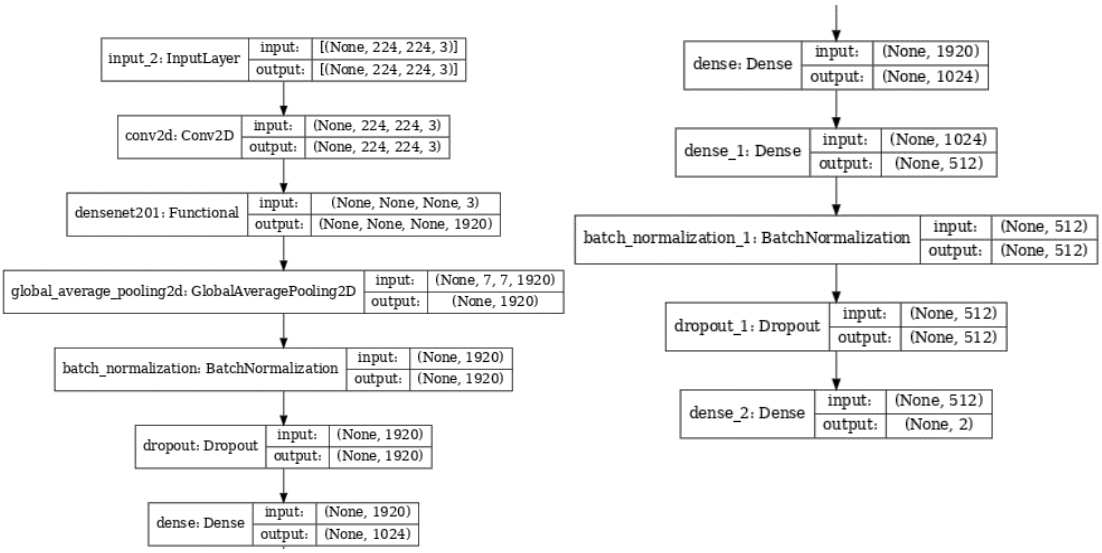
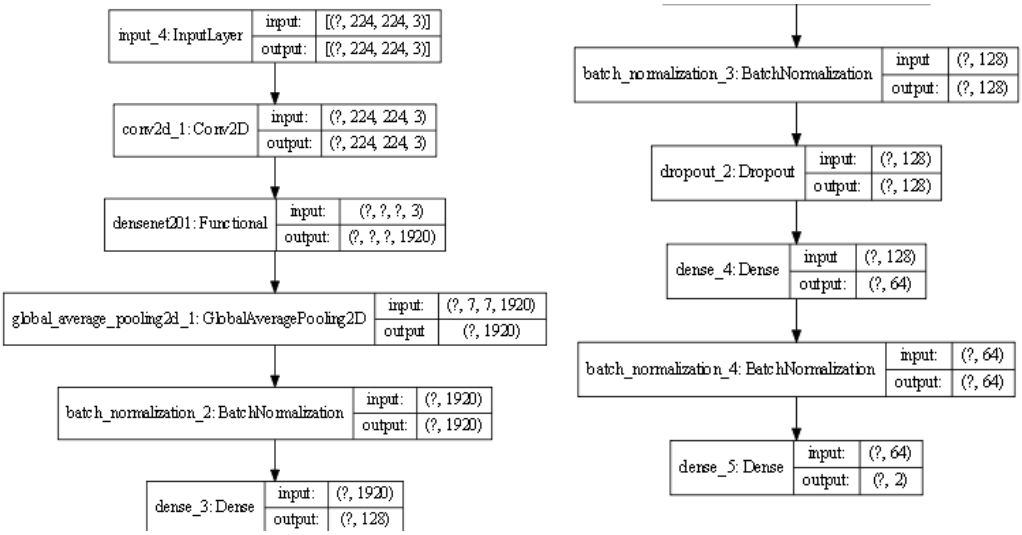


Figure 2. On-top proposed model architecture



A



B

Figure 3. Pre-trained model architectures and fully connected layers

DenseNet 201 is the chosen pre-trained model. It is one of the novel discoveries (2019) in neural networks for visual object recognition. DenseNet is similar to ResNet, but while the last uses an additive method that merges the previous layer with the future layer, DenseNet concatenates the previous layer's output with the future layer - composite function operation. It consists of the convolution layer, pooling layer, batch normalization, and non-linear activation layer. These connections mean that the network has L(L+1)/2 direct connections. L is the number of layers in the architecture.

EXPERIMENTAL WORKFLOW AND PARAMETERS

Before actual implementation (Python), it is essential to import all the relevant libraries: TensorFlow, Keras, Sklearn, Open CV, Matplotlib, Pandas and NumPy. The main drivers are tensorflow.keras.applications to import DenseNet201 and tensorflow.keras.layers to import layers involved in building the network. If the model sees no change in validation loss, the ReduceLROnPlateau function reduces the learning rate, which often benefits the model. The ImageDataGenerator function performs real-time data augmentation over generated tensor image data batches in a loop.

The sequence of the following steps is:

- Load the data
- Exploratory data analysis: data distribution and visualization (plot the images)
- Image pre-processing: cleaning, standardizing to fixed image size (224x224, tensorflow.keras.layers.Resizing), normalizing (scaling, temsorflow.keras.layers.Rescaling): the data are good quality, with no missing items.
- K-means clustering (k=60, sklearn.cluster (K-means)): results visualised in figure 5.
- Group shuffle split (split rate is 0.2 or 80% for training and 20% for the testing set, sklearn.model_selection (GroupShuffleSplit, GroupKFold))
- Build the models (basis on first and second architecture for pre-trained model & real-time data augmentation over generated tensor image data batches in a loop). The setting parameters are shown below.
- Evaluation (Accuracy, Precision, Recall, AUC, Confusion Matrix)

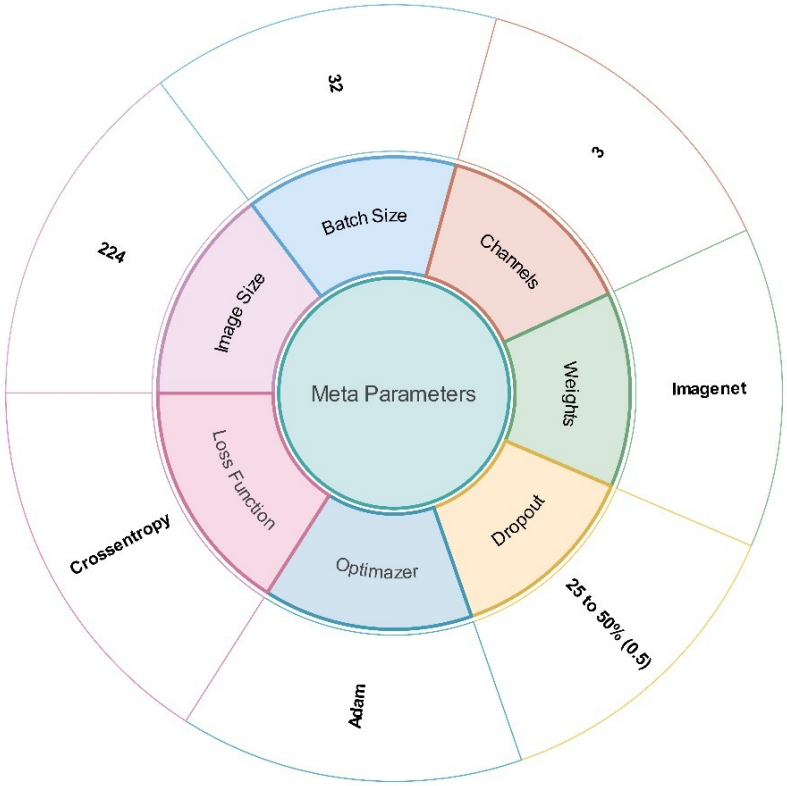


Figure 4: Models meta-parameters

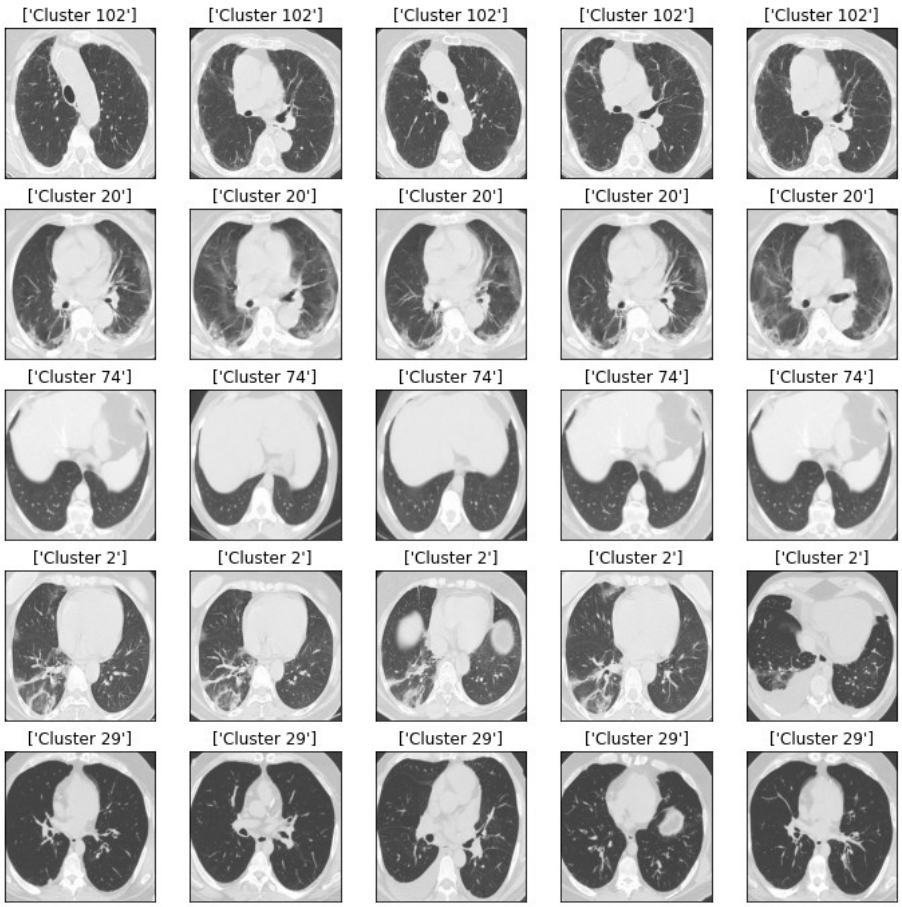


Figure 5. The results visualization of K-means clustering

RESULTS

Several performance evaluation metrics for binary classification tasks are proposed, such as accuracy, recall, precision, F1-score, and specificity, as shown in Table 1. The true-positive and true-negative refer to the numbers of non-COVID and COVID-19 images that are correctly classified. The false-positive and false-negative present the numbers of non-COVID and COVID-19 images that have been wrongly classified.

Table 1: Evaluation performance metrics of proposed models

EVALUATION METRIC	MEANING & CALCULATION
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
SENSITIVITY (RECALL, TRUE	TP/(TP+FN)
Specifity (True Negative Rate)	TN/(TN+FP)
PRECISION (POSITIVE PREDICTION	TP/(TP+FP)
Negative Predictive Value	TN/(TN+FN)
False Positive Rate	FP/(FP+TN)
False Negative Rate	FN/(TP+FN)
False Discovery Rate	FP/(TP+FP)
Accuracy	(TP+TN)/(TP+TN+FP+FN)
F1-score	2*(Precision*Recall)/(Precision+Recall)

A series of experiments are done with tunning the meta-parameters to avoid overfitting. A Dropout layer before the classification layer for regularisation is applied.

Here is the visualization of the best so-far results using typical graphs and metrics for model evaluation.

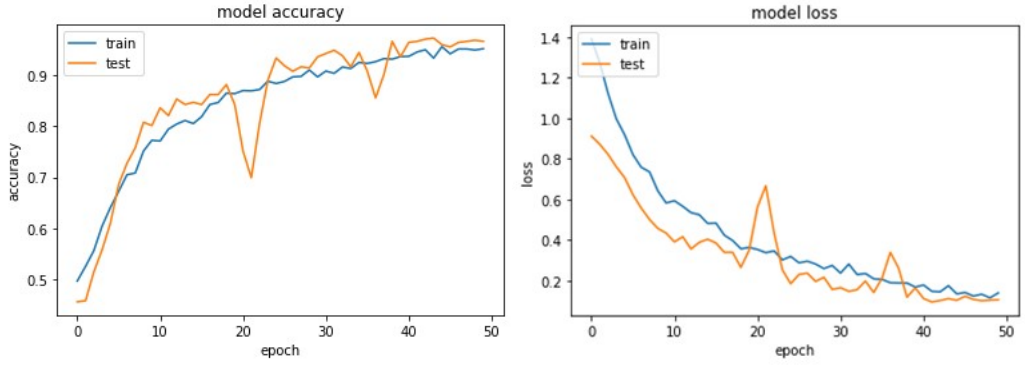


Figure 6. Model accuracy and loss for architecture A

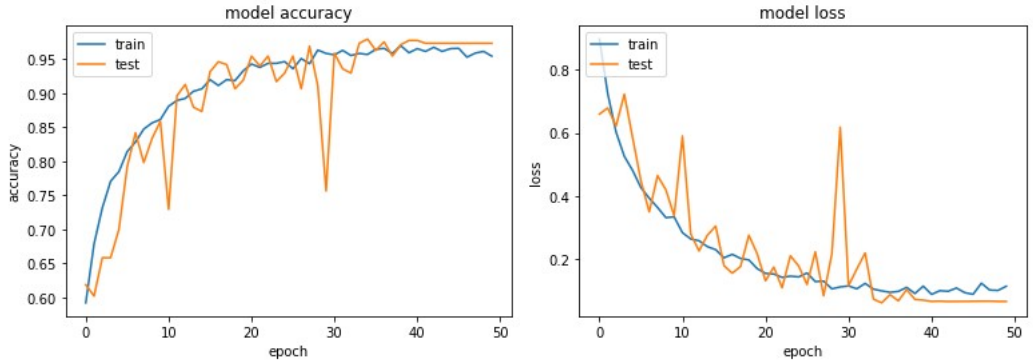


Figure 7. Model accuracy and loss for architecture B

In both diagrams (figure 6 &7), the test curve stays slightly above train one, explained by the variance in train data added through real-time data augmentation and its absence in the test set. Dropout layers also could infer as they are “on” for training but “off” (skipped) when doing testing. In other worlds in training, due to disabling neurons, some of the information about each sample is lost, and the subsequent layers attempt to construct predictions based on incomplete representations. However, all of the units are available during validation, so the network has its full computational power - and thus, it might perform better than in training. The split rate is 0.2. The learning rate is not too high, so it is ignored as a reason. Nevertheless, the slight difference decreases with the increasing number of epochs.

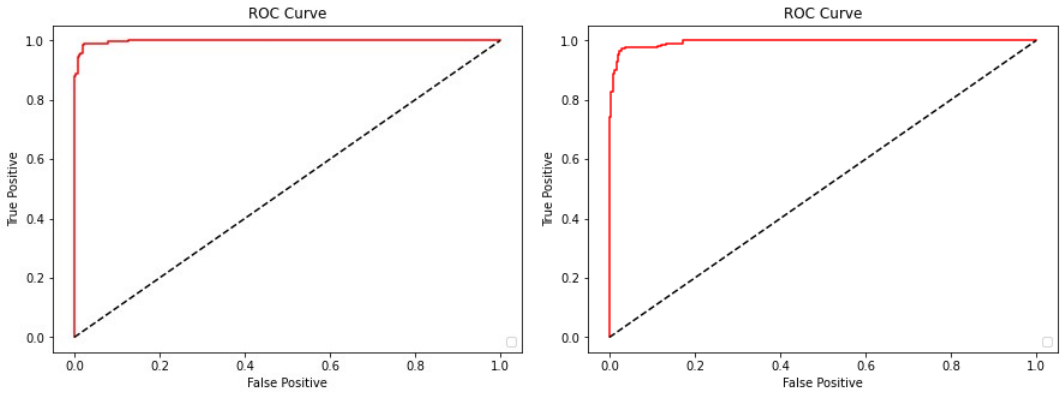


Figure 8. ROC Curve A & B

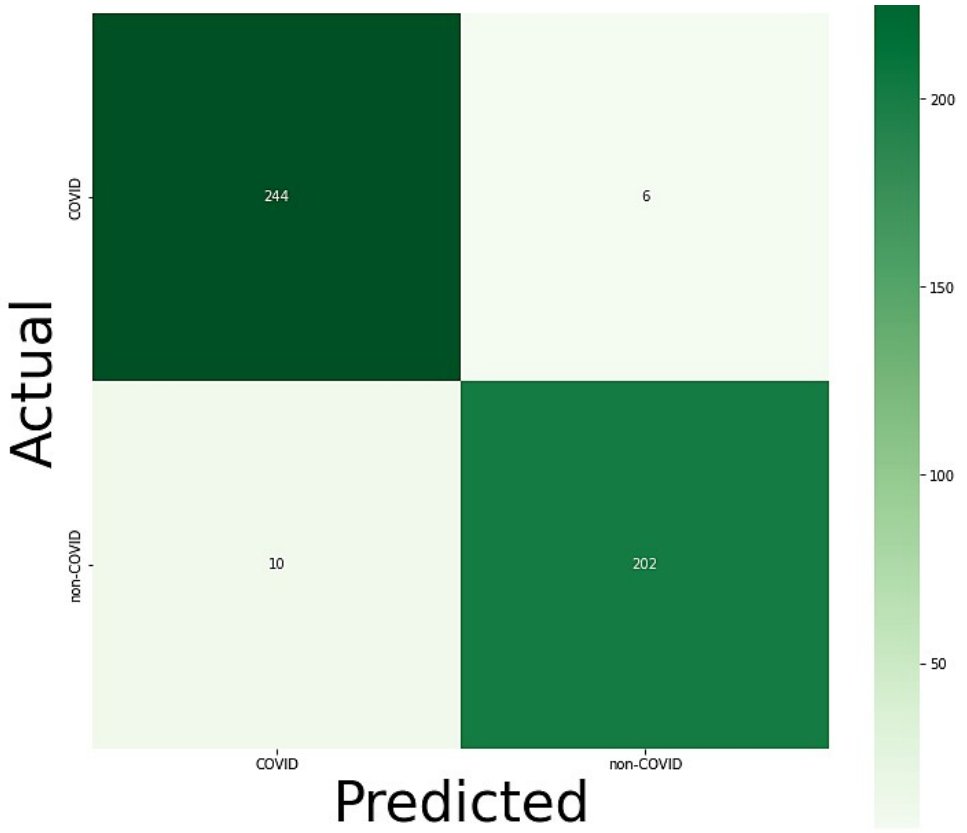


Figure 9. Confusion matrix A

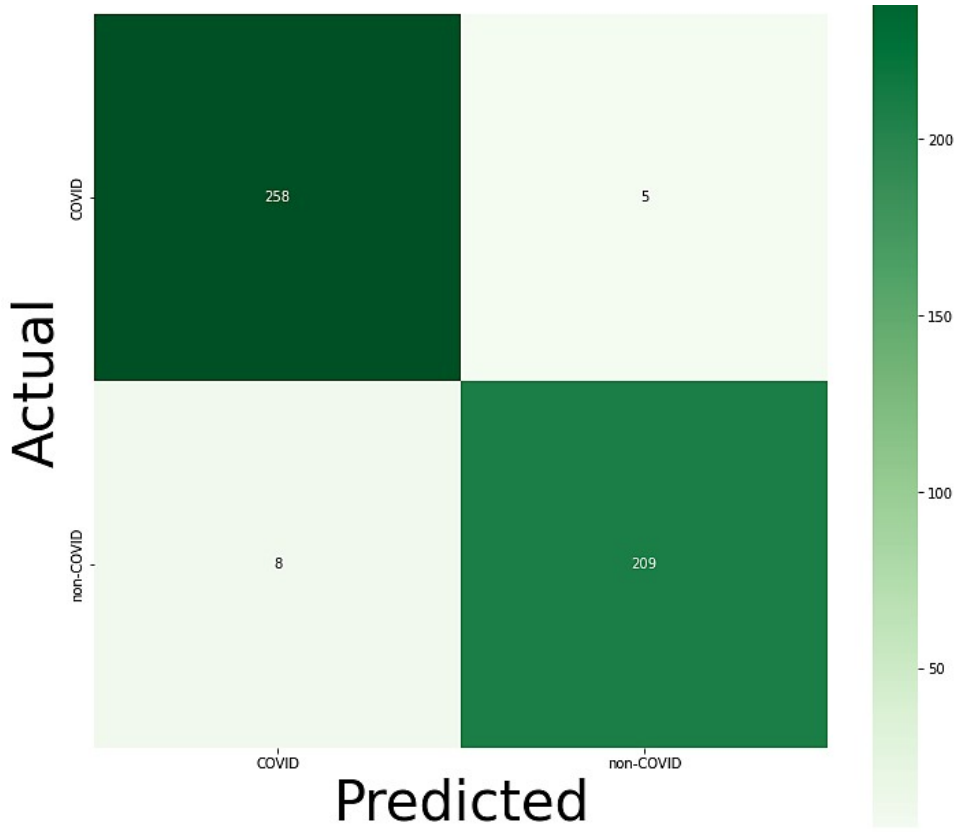


Figure 10. Confusion matrix B

Table 2. Evaluation metrics of model based of architecture A

EVALUATION METRIC	MEANING & CALCULATION	VALUE
TP	True Positive	202 / 202
TN	True Negative	244 / 196
FP	False Positive	6 / 4
FN	False Negative	10 / 7
SENSITIVITY (RECALL, TRUE POSITIVE RATE)	TP/(TP+FN)	95.3%
Specifity (True Negative Rate)	TN/(TN+FP)	97.6%
PRECISION (POSITIVE PREDICTION VALUE)	TP/(TP+FP)	97.12%
Negative Predictive Value	TN/(TN+FN)	96.06%
False Positive Rate	FP/(FP+TN)	2.4%
False Negative Rate	FN/(TP+FN)	4.7%
False Discovery Rate	FP/(TP+FP)	2.9%
Accuracy	(TP+TN)/(TP+TN+FP+FN)	96.53%
F1-score	2*(Precision*Recall)/(Precision+Recall)	96.52%
Best Threshold		0.53

Table 3. Evaluation metrics of model based of architecture B

EVALUATION METRIC	MEANING & CALCULATION	VALUE
TP	True Positive	209
TN	True Negative	258
FP	False Positive	5
FN	False Negative	8
SENSITIVITY (RECALL, TRUE POSITIVE RATE)	TP/(TP+FN)	96.3%
Specifity (True Negative Rate)	TN/(TN+FP)	98.1%
PRECISION (POSITIVE PREDICTION VALUE)	TP/(TP+FP)	97.66%
Negative Predictive Value	TN/(TN+FN)	96.99%
False Positive Rate	FP/(FP+TN)	1.9%
False Negative Rate	FN/(TP+FN)	3.7%
False Discovery Rate	FP/(TP+FP)	2.3%
Accuracy	(TP+TN)/(TP+TN+FP+FN)	97.29%
F1-score	2*(Precision*Recall)/(Precision+Recall)	96.97%
Best Threshold		0.55

CONCLUSION

A K-means clustering is done to avoid the issue with the data leakage. Also, data augmentation is applied, adding some variance and volume. The model evaluation is high and relevant over the current open dataset or similar. But the question of generalization is still existing. Cross-data analysis has to be completed with the data from the original source. Till then, models are not so adequate for applying in the real-world scenario. The reason lies in the centering of the images, their quality, dispersion, and other features/characteristics that depend on the types and settings of the CT machines. However, the primary model architectures are developed, implemented, and tested, ready to be fine-tuned and fitted over the original data.