Deep Neural Network models for COVID-19 diagnosis from CT-Scan, Explainability and Analysis using trained models

Tahsin Islam, Shahriar Absar, S.M. Ali Ijtihad Nasif, Sadman Sakib Mridul, Rafeed Rahman, Md. Saiful Islam
Department of Computer Science and Engineering
BRAC University, Bangladesh
{tahsin.islam, shahriar.absar, s.m.ali.ijtihad.nasif, sadman.sakib.mridul}@g.bracu.ac.bd
{rafeedrahmansham2015}@gmail.com {md.saiful.islam}@bracu.ac.bd

***Abstract*—The world is going through a severe viral pandemic caused by the Novel Coronavirus, also known as "COVID-19". People infected with this virus, experience severe respiratory illness. The virus spreads through particles of saliva or droplets from an infected patient's nostril. There are ways of identifying COVID-19 based on the symptoms such as fever, dry cough, tiredness, but these symptoms are similar to other existing viral or respiratory infections. There is no quick approach to diagnosing if a patient is infected by the virus or not. To overcome the drawbacks mentioned above a faster diagnosis is needed which leads us to the objective of this study, which is to construct a diagnostic approach that uses pre-existing data mostly on COVID-19, as well as take datasets from other respiratory diseases. We will apply deep learning models to the acquired datasets enabling us to obtain more accurate and efficient results. We aim to use Deep Neural Network models namely convolutional neural network models (CNN) such as VGG19, Inception v3, MobileNetV2, and ResNet-50. These four models are pre-trained and they classify the CT-Scan images based on the trained learning approaches. The result of each model is compared among the models to get faster and more accurate results. This paper also proposes a "Hybrid" model which is composed of a convolutional neural network (CNN) and Support Vector Machine (SVM). The Hybrid Model is shallow and just as accurate as the pre-trained models. Based on the accuracy of the outcome and the least amount of time required for image classification we will be able to diagnose more accurately and effectively.**

***Index Terms*—COVID-19, Respiratory Diseases, X-ray, CT-Scan, Deep Neural Network, CNN, VGG19, Inception v3, MobileNetV2, ResNet-50, Rapid approach**

I. Introduction

In late 2019 the world began to see the effects of the novel coronavirus (COVID-19) which is officially known as "SARS-CoV-2". The spread of this virus was so rapid that within a month of its initial report, it became a global health emergency. Even with the most advanced medical treatment, it was not possible to contain the outbreak of this virus. IT generally spreads from the droplets from the sneeze and coughs of an infected person. It quickly damages the respiratory system of the person it infects, causing suffocation due to lack of oxygen and later death.

Though there are various ways the virus can be prevented from spreading, there are still not many ways to accurately diagnose it. Generally, it takes 2-14days to show the symptoms and sometimes the symptoms may not show at all. Now the way the virus is mutating and getting stronger day by day most people may not live 2-14 days after coming in contact with the virus. There are several ways of identifying based on the symptoms. Fever, dry cough, and tiredness are some of the signs. Breathing difficulties or shortness of breath, chest discomfort or pressure, loss of speech or movement are all symptoms of a heart attack, etc. These symptoms can also be for other diseases. So based on this, the diagnosis will not be accurate. Moreover, there are COVID-19 test centers that take samples of the infected people but the results take more time than the patient may have for which most of the cases the patients die due to the lack of proper diagnosis and timely treatment.

So in our research we try to introduce a diagnostic approach that detects COVID-19 and various respiratory diseases based on their pre-existing data and use deep learning models on them, so that we can obtain faster, accurate, and efficient results. We will use Deep Neural Network models namely convoluted neural network models (CNN) such as VGG19 which uses deep Convolutional neural layers to improve accuracy, Inception v3 which is computationally efficient, MobileNetV2, and ResNet-50. As they are pre-trained models we will use them to classify the X-ray and CT-Scan images of COVID-19 and other respiratory diseases. We also propose a "Hybrid" model which is composed of Convolutional Neural Network (CNN) and Support Vector Machine (SVM). The Hybrid Model is shallow and just as accurate as the pre-trained models. Based on the accuracy of the outcome and the least amount of time required for image classification we will be able to diagnose more accurately and effectively if the patient or person is infected or not or which respiratory disease the patient might be suffering from, which will, in turn, save lives.

II. Literature Review

Recently a number of research have been done regarding COVID-19 diagnosis and this segment of the article covered our topic's relevant research efforts. We also reviewed a few

research papers that compare several Machine Learning and Deep Learning models for detecting COVID-19 and other respiratory diseases.

Panwar, H. et al. [1], came up with a deep transfer learning algorithm, which uses various datasets of chest X-ray and CT-Scan to enhance the diagnosis of COVID-19. The approach depends mainly on binary image classification. Therefore, to categorize the numerous CXR and CT-scan images, which provides timely and efficient determination of COVID-19 positive cases with much less than two seconds. They claim that their process is speedier than RT-PCR testing.

Singh, R. K. et al. [2], took an innovative deep learning-based approach to boost up the treatment of COVID-19 infected patients which was based on their X-ray images. COVID-19, pneumonia, and normal were taken as classes that used the method of enhancing the images and also creating segmentation based on it. With this method, Naïve Bayes was used as a meta-learner to create a classification and an updated stacked model of four CNN-based learners was used. Their proposed framework introduces an efficient pruning technique that improves the model's performance and generalizes ability and reduces its complexity. Moreover, while dealing with small training samples; a range of cutting-edge GAN architectures were able to generate practical synthetic COVID-19 chest X-rays. Various numbers of datasets were used in their purpose of classification, segmentation, and weight initialization study. On standard datasets, their proposed approach outperforms current approaches by 98.67% precision, 0.98 Kappa, and F-1 ratings of 100, 98, and 98 the classes, respectively. They said that their approach can be used for the purpose of patient analysis and further medical research.

Alshazly, H. et al. [3], investigated the reliability of deep learning models by training them with CT-scan images of the thorax to properly exploit an automated process and also effectively identify patients infected by COVID-19. To obtain the prime results, they established refined deep network architectures to present a transfer learning technique that used precision input tuned for each deep architecture and trained them with the LAMB Optimizer. Based on two CT-scan image datasets, namely the SARS-CoV-2 and the COVID19-CT, they carried out sets of experiments thoroughly and even concluded that their models perform better as compared to the previous studies.

Li, L. et al. [4], in their research intended to build and analyze a completely automated system for detecting COVID-19 by using thoracic CT scan images in which they created a COVID-19 identification neural network (COV-Net). This deep learning algorithm was created to conduct the COVID-19 diagnosis by extracting visual attributes from "holographic thoracic CT- Scan images". CT-Scans of pneumonia that were obtained from outside the hospital, mostly from the general population also known as "CAP" and other non-pneumonia malformations were used to test the model's potential from August 2016 to February 2020, various data were obtained from six hospitals. The efficiency of the diagnostic was tested using the area under the receiver operating characteristic curve, as well as their sensitivity and specificity. In the separate test range, the pet-scan sensitivity and specificity for detecting COVID-19 were 90% and 96%, with an area under the receiver operating characteristic curve of 0.96 (P¡.001).

Yeşilkanat, C. M. et al. [5], used a different approach for predicting the number of instances of the occurrence of pandemics such as COVID-19 in their research paper. In their research, the efficiency of the Random Forest is also known as the "RF" machine learning algorithm. The Algorithm was evaluated in this study to determine near future case counts for 190 nations all over the world. The results were plotted against actual verified cases of COVID-19 outcomes, which were recorded between 23/01/2020 to 17/06/2020. The number of confirmed cases was divided into three sub-datasets for the random forest model which revealed that the random forest machine learning algorithm does an excellent job of predicting the number of cases in the upcoming future if such outbreak like COVID-19 ever emerges.

Shahid, F. et al. [6], discussed that it is feasible to build well-thought-out schemes in the public health system to prevent death by COVID-19 and manage patients. Their prediction models included ARIMA, SVR, LSTM, and Bi-LSTM. In this study, these models were analyzed for time series prediction of confirmed cases, fatalities, and cures in ten key COVID-19-affected countries. Mean Absolute error, root mean square error, and r2 score indices are utilized to compute model efficacy. The Bi-LSTM model surpasses recommended indices in the vast majority of instances. Bi-LSTM, LSTM, GRU, SVR, and ARIMA are the models that rank from best to worst in all cases. For deaths in China, Bi-LSTM produces the lowest MAE and RMSE values. According to the recovered cases in China. They deduced that Bi-LSTM can be used for pandemic forecasting of more conservation and restoration due to its demonstrated resilience and upgraded prediction precision.

Guhathakurta, S. et al. [7], came up with a method for determining whether or not a person is contaminated with COVID-19. They used a support vector machine to classify the patient's health into three categories namely mild infection, no infection, and serious infection, based on the critical effect of the symptoms. Only a few of the symptoms associated with COVID-19 patients have been selected. The major frequent symptoms like fever, breathing rate, and cough are present in 90% of confirmed cases. To categorize these features/symptoms into the given classifications, they used the support vector machine (SVM) classifier. They also used visual programming to compare and contrast common supervised learning models. They were able to forecast the instances with an accuracy of 87%.

III. Model Backgrounds

In our research, we have used Convolutional Neural Network models Such as VGG19, Inception v3, ResNet-50 and MobileNetV2 and apply and compare our dataset on them.

VGG19 is a VGG model type with 19 layers, 16 of which are convolution layers, 5 of which are MaxPool layers, 3

of which are fully-connected layers, and 1 of which is the SoftMax layer. It was pre-programmed to separate the images into 1000 segmented groups, each with its own set of components. As a result, the network has developed a sophisticated categorization model for a wide range of pictures.

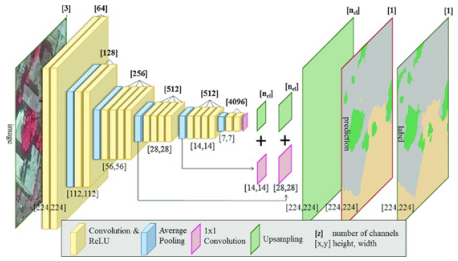

Fig. 1. Architecture of VGG19

Secondly, ResNet-50 or Residual Networks learn residual functions based on layer inputs. These stacked layers can use residual nets to fit a residual mapping. These relationships are easier to optimize and can benefit from more depth.

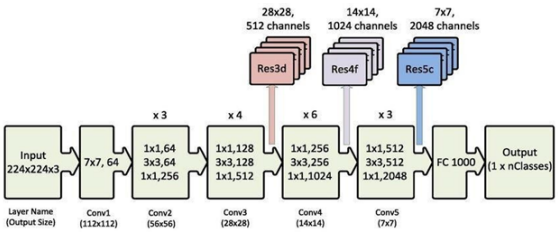

Fig. 2. Architecture of ResNet50

Likewise, Inception v3 is a convolutional neural network architecture that incorporates Smoothing of Labels that is factorized in 7x7 convolutions. It also employs an auxiliary classifier to generate an Information label as the network progresses, as well as batch normalization for layers in the side head. In terms of the number of parameters propagated by the network and the economic cost incurred for both memory and other resources, and has been proven to be more computationally well-planned and considerably more organized.

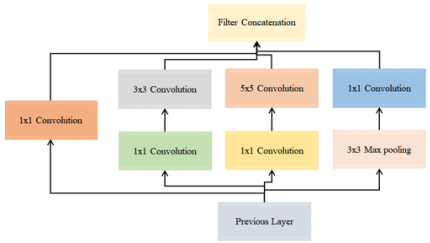

Fig. 3. Architecture of Inception V3

Finally, MobileNetV2 is a convolutional neural organization engineering that is improved for cell phones. It is the second form of engineering that powers numerous well-known versatile applications' picture handling usefulness. The essential guideline behind MobileNetV2 is that the bottlenecks decipher the model's moderate information sources and yields, while the internal layer exemplifies the model's ability to change from lower-level ideas.

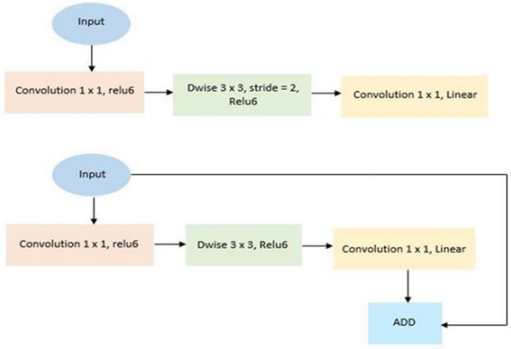

Fig. 4. Architecture of MobileNetV2

IV. Methodologies

In this part we are going to explain the dataset we have used and also explain the way we used, processed and trained our datasets and implemented them on the Convolutional Neural Network (CNN) models and discuss the outcomes we get.

A. Dataset description and Pre-processing

In our research, we collected the dataset from 2 sources, namely Kaggle and GitHub. We narrowed it down to the images of chest X-rays and CT-scans. Then we merged both the datasets into our final dataset that we used for training and testing by implementing different models in python programming. The final dataset is composed of 10 different classes labeled as 'Bacterial', 'Covid', 'Fungal', 'Lipoid', 'Normal', 'Other', 'Pneumonia', 'Tuberculosis', 'Unknown' and 'Viral'. The entire dataset contains 877 images and we categorized each of these images according to its respective category. The images in the dataset are in the format of either PNG or JPEG.

Datasets are imbalanced in a way that the training sets will not be divided evenly across the target classes. In this type of case, the model being applied will be more partial to the class having a huge number of training occurrences that will lead to a downfall in the model's prediction power. It also increases the number of false-positive outcomes in the case of a standard binary classification problem. Machine Learning is used to handle an imbalanced dataset, where the sampling process is applied to the training dataset only and no modifications are made to the testing dataset. In python programming, imblearn is the library needed to execute this method. For Data-Duplication, the K-Nearest-Neighbors algorithm is used. The artificially generated data points are correlated with the minority class. They are pushed into the dataset so that both

the labels are rough of equal size. This process prohibits the model from being biased towards the majority class and the interaction among the target classes remains unchanged. Because of the additional data brought up by this process, the system gets introduced with bias. The input records must not contain any null values in applying this method in particular.

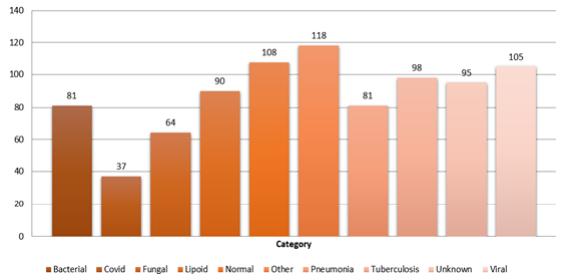

Fig. 5. Dataset Overview

B. Processing & model-train

We used Google Collaboratory for our python programming tasks. We had to implement the models mentioned above to perform transfer-learning and fine-tuning, we had to change the run-time type by choosing GPU as the hardware accelerator. Then we had to mount our notebook to Google Drive where our dataset was previously uploaded so that the program can execute accordingly by assessing the image files. Then we installed Tensor Flow which is an open-source platform for deep learning purposes. After that, we imported Keras which is a neural network library. Both Tensor Flow and Keras provide high-level APIs for building and training the models easily. We also have to specify the model that we will be implementing on our dataset itself. We set the size of the images of our dataset to 244x244. Then we used scikit-learn for the statistical purposes of each of the models we applied. For splitting the data set into training and testing sets, we had to apply it in each of the different categories of our data set. We assigned the test size to 0.2, meaning that 20% for testing and 80% for training; and we assigned the random-state to 40 which is just a measure of randomness (the higher the number, the higher the randomness) and the splitting of the categories into training and testing sets are done by choosing the images randomly. Then we set up our neural network by specifying each of the models as we pass the image size as input, assigning weights to ImageNet which is an image database for deep learning purposes, and by assigning include-top to false so that the neural network does not get activated right away. Then we freeze the training layers. dataset where each of them is contained in separate folders which are all located in the same Google Drive link. So, splitting the set of images into training and testing sets has to be done to every folder in our data set. Then we flatten the layers and activate the network with the Softmax function. After that, we compile the model by assigning the loss function as 'categorical cross-entropy, optimizer as 'Adam' and metrics as 'accuracy'. Categorical Cross-entropy deals with an output tensor and a target tensor. Adam is a deep learning direct training method that substitutes gradient mechanism. Accuracy is the number of correct predictions as per the model applied. For our test-data-generator, we re-scale the pixel of every image in the dataset, and for our train-data-generator, we do the same thing and also assign shear-range to 0.2, zoom-range to 0.2 And horizontal-flip to 'true'. Then for both our training and testing sets, we set up the test data-generator flow from the image directory; we also set the target size to 244x244, batch size (subset size of our training sample) to 16, and class mode to 'categorical'. Finally, we train our deep learning models by using the model-fit-generator function and assigning epochs to 20.

V. Proposed Hybrid Model

We propose to do image classification on a hybrid model which is an SVM on the CNN (Convolutional Neural Network) itself. SVM stands for Support Vector Machine and is a linear model used to address regression and classification problems. It is a conventional machine learning technique that helps in the segmentation of huge amounts of data and is particularly helpful for simulation and model-based applications in a massive data environment.

In our proposed model, at first, we imported an image data generator which was used for data-augmentation techniques like augmenting the data to create more additional data in the memory. After that, we re-scaled the images in both the train and test data and applied image-data-generator in the training data set. Then we set up the directories for both training and testing datasets by setting the batch size to 32, target size to 64 x 64, and class mode to categorical as we are doing multi-classification over here.

once again we imported two layers namely the dense layer (to add the nodes for the hidden layers) and the convolutional-2D layer (for the convolutional operation) because we just want to create a plain neural network. Then we imported L2 regularize. After that created our layer by adding our sequential layer; a convolutional-2D layer with a filter of 32, padding as same, kernel size as 3, activation function as relu, strides as 2, and input shape as 64 x 64; a max-pooling layer of pool size as 2 and strides as 3; another convolutional-2D layer (same as before) followed by another max-pooling layer (same as before) and then we flatten the layer. A dense layer with units of 128 and activation function as relu will be created and in our final layer, all the categories (10 according to our dataset), L2 regularize as 0.01, and activation function as softmax. Afterward, we will compile our neural network by setting the metrics like accuracy, loss as a squared hinge, and optimizer as adam. Finally, we run the entire algorithm by setting up epochs to 20 and training-set to test-set.

VI. Result Analysis & Comparison

A. Deep Learning Models

As we have mentioned before that we have trained each of our deep learning models by assigning epochs to 20, we have obtained the following as given below:

- **Inception v3:** It took 4335 seconds (1 hour, 12minutes and 15 seconds) in total. The first epoch took 504 seconds (8 minutes and 24 seconds) where each step took 9 seconds; and the rest of the 19 epochs took an average time of 201 seconds (3 minutes and 21 seconds) where each step took 4 seconds. At the very first epoch, the initial values of train-loss, train-accuracy, validation-loss and validation-accuracy are 6.4922, 0.6454, 0.8518 and 0.9019 respectively. Then at the very last epoch, the final values of train-loss, train-accuracy, and validation-loss and validation-accuracy are 0.3464, 0.9692, 0.2202 and 0.9806 respectively. Finally, we plotted two graphs, one showing how the values of both train-loss and validation-loss changed over 20 epochs; and the other one showing how the values of both train-accuracy and validation-accuracy changed over 20 epochs.

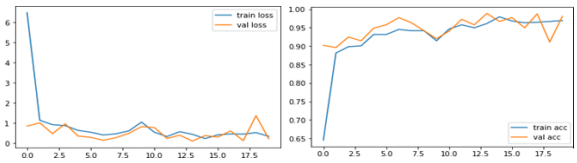

Fig. 6. Train-loss VS Validation-loss & Train-accuracy VS Validation-accuracy

- **ResNet-50:** It took 1235 seconds (20 minutes and 35 seconds) in total. The first epoch took 525 seconds (8 minutes and 45 seconds) where each step took 10 seconds; and the rest of the 19 epochs took an average time of 284 seconds (4 minutes and 44 seconds) where each step took 5 seconds. At the very first epoch, the initial values of train-loss, train-accuracy, validation-loss and validation-accuracy are 6.5914, 0.1893, 1.6879 and 0.4390 respectively. Then at the very last epoch, the final values of train-loss, train-accuracy, validation-loss and validation-accuracy are 1.2457, 0.6705, 0.9659 and 0.7206 respectively. Finally, we plotted two graphs, one showing how the values of both train-loss and validation-loss changed over 20 epochs; and the other one showing how the values of both train-accuracy and validation-accuracy changed over 20 epochs.

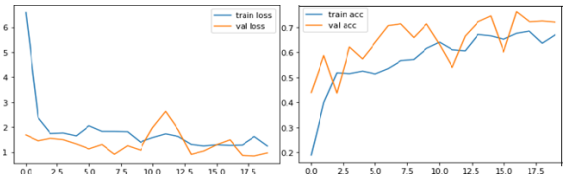

Fig. 7. Train-loss VS Validation-loss & Train-accuracy VS Validation-accuracy

- **VGG19:** It took 8614 seconds (2 hours, 23 minutes and 34 seconds) in total. The first epoch took 4677 seconds (1 hour, 17 minutes and 57 seconds) where each step took 16 seconds; and the rest of the 19 epochs took an average time of 196 seconds (3 minutes and 16 seconds) where each step took 8 seconds. At the very first epoch, the initial values of train-loss, train- accuracy, validation-loss and validation-accuracy are 0.4475, 0.8406, 0.2918 and 0.8890 respectively. Then at the very last epoch, the final values of train-loss, train-accuracy, validation-loss and validation-accuracy are 0.1274, 0.9685, 0.3869 and 0.9107 respectively. Finally, we plotted two graphs, one showing how the values of both train-loss and validation-loss changed over 20 epochs; and the other one showing how the values of both train-accuracy and validation-accuracy changed over 20 epochs.

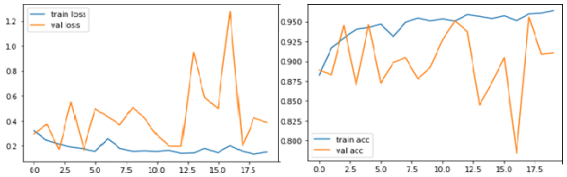

Fig. 8. Train-loss VS Validation-loss & Train-accuracy VS Validation-accuracy

- **MobileNetV2:** It took 2343 seconds (39 minutes and 3 seconds) in total. The first epoch took 715 seconds (11 minutes and 55 seconds) where each step took 12 seconds; and the rest of the 19 epochs took an average time of 1615 seconds (26 minutes and 55 seconds) where each step took 5 seconds. At the very first epoch, the initial values of train-loss, train-accuracy, validation-loss and validation-accuracy are 7.4631, 0.5730, 0.9946 and 0.9042 respectively. Then at the very last epoch, the final values of train-loss, train-accuracy, validation-loss and validation- accuracy are 0.3589, 0.9846, 0.3586 and 0.9806 respectively. Finally, we plotted two graphs, one showing how the values of both train-loss and validation-loss changed over 20 epochs; and the other one showing how the values of both train-accuracy and validation-accuracy changed over 20 epochs.

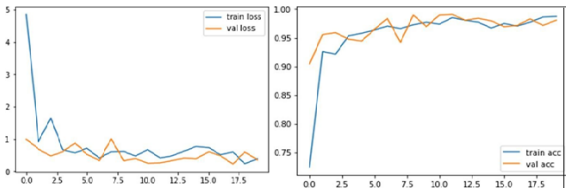

Fig. 9. Train-loss VS Validation-loss & Train-accuracy VS Validation-accuracy

B. Hybrid Model Analysis

We executed this algorithm three times and the results of each are as follows.

The first iteration took 2153 seconds (35 minutes and 53 seconds). The first epoch took 672 seconds (11 minutes and 12 seconds) and it took 23 seconds per step. The rest of the 19 epochs took an average time of 76 seconds (1 minute and 16 seconds) and it took 3 seconds per step. The initial values

of train-loss, train-accuracy, validation-loss and validation-accuracy are 1.3135, 0.1881, 1.2733 and 0.3170 respectively; and the final values of those are 0.9825, 0.8198, 0.9767 and 0.8301 respectively.
The second iteration took 1538 seconds (25 minutes and 38 seconds). The 20 epochs took an average time of 76 seconds (1 minute and 16 seconds) and it took 3 seconds per step. The initial values of train-loss, train-accuracy, validation-loss and validation-accuracy are 0.9804, 0.8233, 0.9749 and 0.8335 respectively; and the final values of those are 0.9452, 0.9065, 0.9416 and 0.9133 respectively.
The third iteration took 1567 seconds (26 minutes and 7 seconds). The 20 epochs took an average time of 78 seconds (1 minute and 18 seconds) and it took 3 seconds per step. The initial values of train-loss, train-accuracy, validation-loss and validation-accuracy are 0.9434, 0.9088, 0.9403 and 0.9202 respectively; and the final values of those are 0.9313, 0.9361, 0.9296 and 0.9384 respectively.

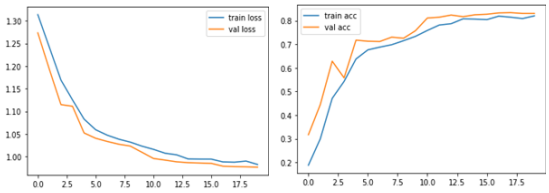

Fig. 10. Train loss VS Validation loss & Train accuracy VS Validation accuracy

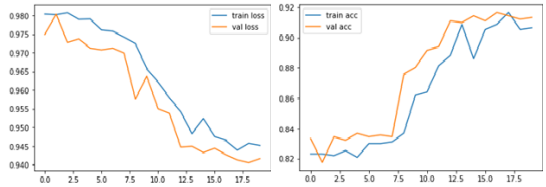

Fig. 11. Train loss VS Validation loss & Train accuracy VS Validation accuracy

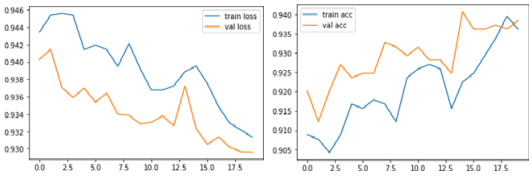

Fig. 12. Train loss VS Validation loss & Train accuracy VS Validation accuracy

C. Comparison with other trained models

In our hybrid model, the SVM (Support Vector Machine) is which is a very good algorithm for classification; be it binary or multi-class classification. We have implemented four pre-trained models and one hybrid model for transfer-learning and fine-tuning by using the same dataset for image classification. The four pre-trained models are Inception v3, VGG19, ResNet-50 and MobileNetV2.

Comparing our hybrid model with The Inception V3 we can see that it took 4335 seconds (1 hour, 12 minutes and 15 seconds), The ResNet-50 took total 1235 seconds (20 minutes and 35 seconds). The VGG19 took total 24279 seconds (6 hours, 44 minutes and 39 seconds) and finally The MobileNetV2; took 3083 seconds (51 minutes and 23 seconds) respectively to be successfully executed. Each of these four models were run only once. The Hybrid model we proposed was executed Three times.The first iteration took 2153 seconds (35 minutes and 53 seconds). The second iteration took 1538 seconds (25 minutes and 38 seconds) and finally the third iteration took 1568 seconds (26 minutes and 8 seconds).

Here we can see that 3 out of 4 pre-trained models took significantly more time than our Hybrid Model and only ResNet-50 was faster. The VGG19 took significantly more time than the other models. Here our Hybrid Model is a simple and shallow model which gives perfect accuracy with faster time.

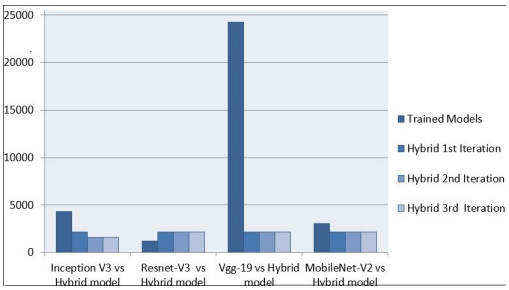

Fig. 13. Time Comparison between Hybrid model VS pre-trained models

To summarize, our hybrid model has 273706 trainable parameters with 0 non-trainable parameters in total it is the lowest number of parameters among all the models we have applied; and those models had non-trainable parameters which increased the total number of parameters. This model is composed of 2 max-pooling layers, 2 dense layers and 2 convolutional 2D layers which in total is the lowest number of layers a neural network has among the applied models; and those models have some other layers and even the same layers as those in the hybrid model more than twice. This model is shallower than the rest of the model applied. On average, the time taken to finish each epoch in our hybrid model was the least among all the models we applied. Our hybrid model performed well without using any boosting methods.

Table 1 shows a comparison of the train-loss, train-accuracy, validation-loss and validation-accuracy among the algorithms used at a glance for the final iteration.

VII. Conclusion & Future Works

Science has relieved us from countless withering diseases and if given time it will be able to treat COVID-19 patients,

	Train Loss	Train Accuracy	Validation Loss	Validation Accuracy
Inception V3	0.3464	0.9692	0.2202	0.9806
VGG-19	0.1274	0.9685	0.3869	0.9107
ResNet-50	1.2457	0.6705	0.9659	0.7206
MobileNetV2	0.3589	0.9846	0.3586	0.9806
Hybrid	0.9313	0.9361	0.9296	0.9384

TABLE I
PERFORMANCE EVALUATION OF ALGORITHMS USED (FINAL ITERATION)

comprehensively. For developing countries like Bangladesh, most of the hospitals are not well equipped. Hence, they are unable to treat COVID-19 infected patients properly and effectively. With the help of our model, the doctors will be able to quickly diagnose the infected people and any other respiratory diseases. By doing so, there will be less chance of getting false positives from the test results. Furthermore, with proper diagnosis, the growing number of COVID-19 infected people will decrease. The death rate will also come down if the virus is caught in its infancy stage. In the future, attempts will be taken to diversify the data-set with more relevant data and also increase the size. We would like to train the "Hybrid" model further for better accuracy and modify the models used for detecting other diseases. Also, we plan to use Grad-CAM for further explainability and visualization purposes.

VIII. ACKNOWLEDGEMENT

First of all, we would like to start by thanking the Almighty Allah, the most gracious and merciful who blessed us to complete this research. In this part, special gratitude to our supervisor Md. Saiful Islam for his valuable guideline, consultation, correction and advice. This thesis would never have been possible without his constructive suggestion, continual encouragement and assistance. Moreover, our profound thanks to our co-supervisor, Rafeed Rahman, who encouraged and invested his valuable time in clarifying our doubts during this thesis work. Lastly, we would like to thank our family and friends for their constant source of inspiration.

REFERENCES

[1] H. Panwar, P. K. Gupta, M. K. Siddiqui, R. Morales-Menendez, P. Bhardwaj, and V. Singh, "A deep learning and grad-CAM based color visualization approach for fast detection of COVID-19 cases using chest x-ray and CT-Scan images," *Chaos Solitons Fractals*, vol. 140, no. 110190, p. 110190, 2020.

[2] R. K. Singh, R. Pandey, and R. N. Babu, "COVIDScreen: explainable deep learning framework for differential diagnosis of COVID-19 using chest x-rays," *Neural Comput. Appl.*, vol. 33, no. 14, pp. 1–22, 2021.

[3] H. Alshazly, C. Linse, E. Barth, and T. Martinetz, "Explainable COVID-19 detection using chest CT scans and deep learning," *Sensors (Basel)*, vol. 21, no. 2, p. 455, 2021.

[4] L. Li, L. Qin, Z. Xu, Y. Yin, X. Wang, B. Kong, J. Bai, Y. Lu, Z. Fang, Q. Song, K. Cao, D. Liu, G. Wang, Q. Xu, X. Fang, S. Zhang, J. Xia, and J. Xia, "Using artificial intelligence to detect COVID-19 and community-acquired pneumonia based on pulmonary CT: Evaluation of the diagnostic accuracy," *Radiology*, vol. 296, no. 2, pp. E65–E71, 2020.

[5] C. M. YeŞİlkanat, "Spatio-temporal estimation of the daily cases of COVID-19 in worldwide using random forest machine learning algorithm," *Chaos Solitons Fractals*, vol. 140, no. 110210, p. 110210, 2020.

[6] F. Shahid, A. Zameer, and M. Muneeb, "Predictions for COVID-19 with deep learning models of LSTM, GRU and Bi-LSTM," *Chaos Solitons Fractals*, vol. 140, no. 110212, p. 110212, 2020.

[7] S. Guhathakurata, S. Kundu, A. Chakraborty, and J. S. Banerjee, "A novel approach to predict COVID-19 using support vector machine," in *Data Science for COVID-19*, pp. 351–364, Elsevier, 2021.