**Project Report**

**Problem Statement**

The COVID-19 pandemic has caused a significant global health crisis, and the rapid spread of the virus has highlighted the need for efficient and accurate diagnosis methods. CT scans have been shown to be a useful tool in detecting COVID-19, as they can provide detailed images of the lungs and reveal abnormalities that may be indicative of the disease.

This project aims to develop a computer-aided diagnosis system that can accurately detect COVID-19 from CT scans. The project aims to use machine learning techniques, such as deep neural networks, to analyze and interpret CT images. The developed system would be able to identify patterns and features in the images that are specific to COVID-19 and distinguish them from other lung diseases or normal lung tissue.

The project uses a UNet model trained on a dataset pf .nii files to segment the lungs in CT scans, which are then used to classify the scans as COVID-19 positive or negative. This classification is performed using segmented lung images from two other datasets taken from kaggle.

In addition to lung segmentation, a separate UNet model was trained to detect the location of the infection in CT scans where COVID-19 is present. The developed system can be used to provide a faster and more accurate diagnosis of COVID-19, which can aid in the early detection and treatment of the disease.

The use of CT scans for COVID-19 diagnosis has the potential to significantly improve the speed and accuracy of diagnosis, which is critical in controlling the spread of the virus. The developed system could assist radiologists and other medical professionals in the diagnosis of COVID-19, especially in areas with limited access to specialized medical resources.

**Datasets**

**a.** **COVID-19 CT scans**

**Link: -**<https://www.kaggle.com/datasets/andrewmvd/covid19-ct-scans>

**Type: -** .nii files

**About:-** CT scans play a supportive role in the diagnosis of COVID-19 and is a key procedure for determining the severity that the patient finds himself in.  
 Models that can find evidence of COVID-19 and/or characterize its findings can play a crucial role in optimizing diagnosis and treatment, especially in areas with a shortage of expert radiologists.  
 This dataset contains 20 CT scans of patients diagnosed with COVID-19 as well as segmentations of lungs and infections made by experts.

**b.** **CT Scans for COVID-19 Classification**

**Link: -**<https://www.kaggle.com/datasets/azaemon/preprocessed-ct-scans-for-covid19>

**Type: -** .jpg files

**About:-** Data were collected from 2 hospitals: Union Hospital (HUST-UH) and Liyuan hospital (HUST-LH) which is describes in detail [in this paper](https://europepmc.org/article/ppr/ppr141530). The authors of this paper classified individual CT images into three types,  
 (I) 5705 non-informative CT (NiCT) images where lung parenchyma was not captured for any judgment,  
 (ii) 4001 positive CT (pCT) images where imaging features associated with COVID-19 pneumonia could be unambiguously discerned, and  
 (iii) 9979 negative CT (nCT) images where imaging features in both lungs were irrelevant to COVID-19 pneumonia.

So the first step of methodologies in the paper was to extract lung parenchyma from CT images. We extracted lung parenchyma by using the methods explained in the paper. Thes extracted images are in the preprocessed CT scans folder where the original images are in the original CT scans folder. All the images are resized to (512x512)

**List of Software Used:**

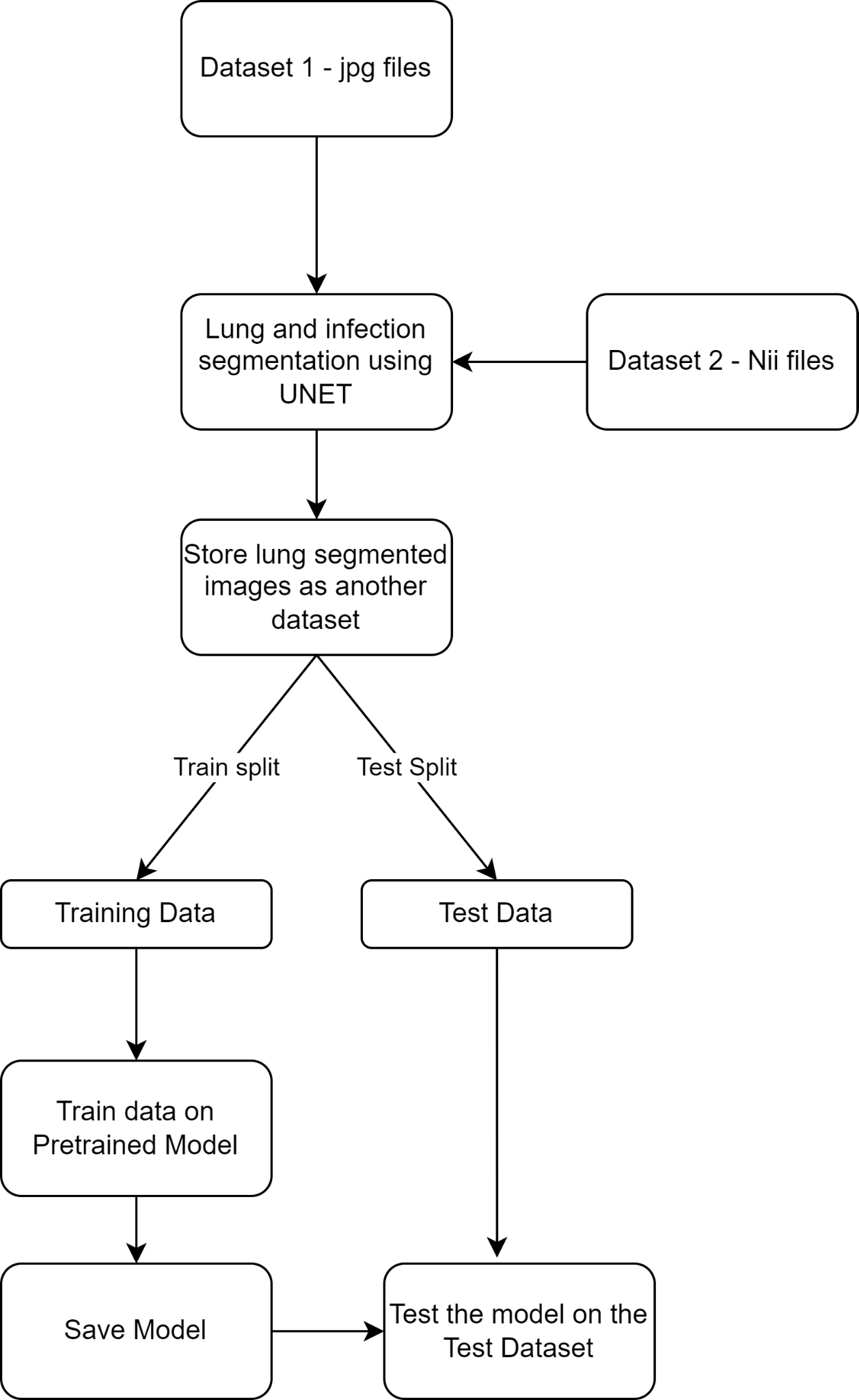
Kaggle GPU, Kaggle notebook, Jupyter Notebook

**Pre-processing methods**

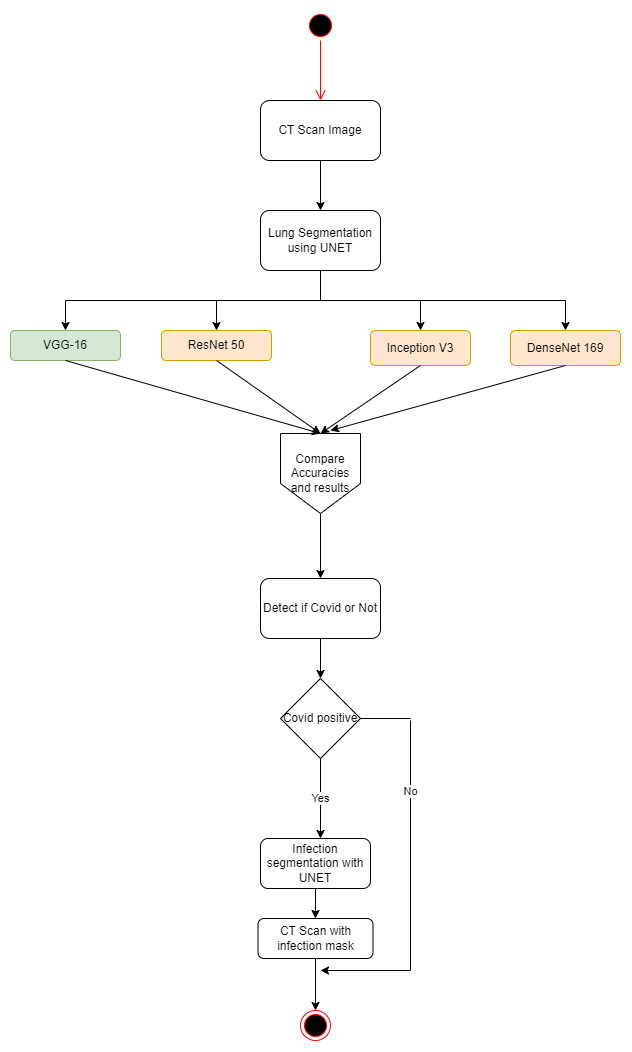
Most of the image pre-processing is not needed as we have unet for getting both the lung mask and the infection mask but we do use histogram equalization and gaussian blur for regularising the model.

**Architecture:**

*Project Workflow*

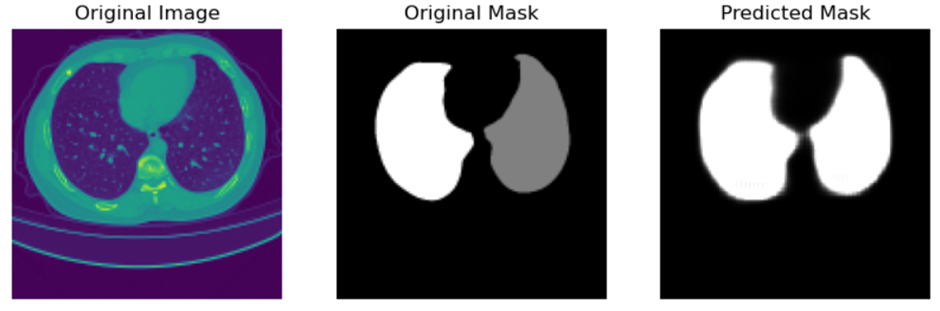
****

*Project Architecture*

****

**Lung Segmentation Using Unet**

In this section, we take 20 nii files with ct-scans and 20 nii files with lung masks of the ct-scans. We convert both types of images to 2d image slices and then into arrays. We also reshape each image array for both the ct-scans and lung mask to (128,128,1). Following this, we create an unet model that can effectively create a lung mask from a CT-Scan that can work from any kind of image and any quality. We train the model on the nii files and test it on the jpg images. After training, we get a 0.8 training dice coefficient and a 0.78 validation dice coefficient. Fig 1 and Fig2 show results of the unet model for images from the same dataset and images from a different dataset.



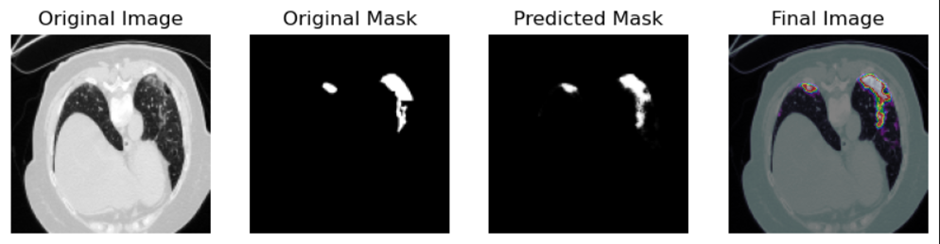
**Fig 1: - Image from the same dataset**

****

**Fig 2: - Image from different dataset**

**Infection Segmentation Using Unet**

20 NII files with CT scans and 20 NII files with infection masks of the CT images are used in this part. Both sorts of photos are transformed into 2D image slices and then into arrays. Moreover, we reshape each picture array for the infection mask and ct-scans to (128,128,1). Then, using an unet model that can function with any type of image and any quality, we efficiently generate a lung mask from a CT-Scan. We use the nii files to train the model and the jpg images to test it. We obtain 0.82 for the training dice coefficient and 0.83 for the validation dice coefficient following training.Fig 3 and Fig 4 show results of the unet model for images from the same dataset and images from a different dataset.



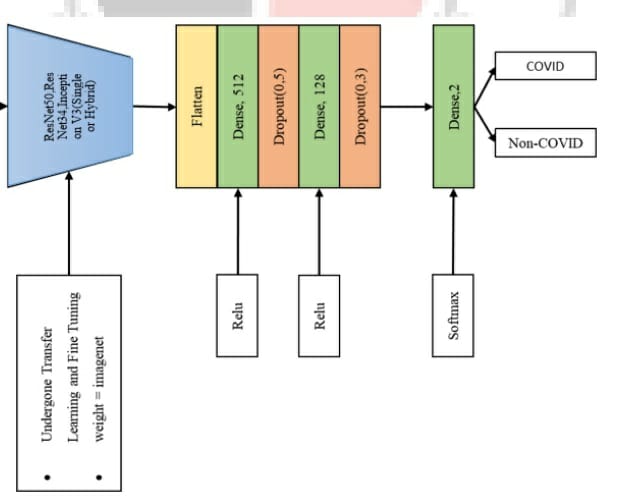
**Fig 3: - Image from the same dataset**

****

**Fig 4: - Image from different dataset**

**Modeling for Covid or Not**

We use the Unet Model to segment the lung for each image from the dataset that has .jpg files and store them in another folder. We then split the dataset into train and test split and make the training data undergo augmentation using ImageDataGenerator. We then train the model on four different models namely Vgg16, ResNet50, Inception V3 and DenseNet169. Each model is constructed using transfer learning and fine-tuning. We then train the model and by using a list of metrics, test the model on testing data and data outside the dataset. After extensive testing, we found the vgg16 to be the most accurate.



**Fig 5:- model structure**

*Unet Architecture*

Model: "model"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param # Connected to

==================================================================================================

input\_1 (InputLayer) [(None, 128, 128, 1 0 []

)]

conv2d (Conv2D) (None, 128, 128, 16 160 ['input\_1[0][0]']

)

conv2d\_1 (Conv2D) (None, 128, 128, 16 2320 ['conv2d[0][0]']

)

max\_pooling2d (MaxPooling2D) (None, 64, 64, 16) 0 ['conv2d\_1[0][0]']

conv2d\_2 (Conv2D) (None, 64, 64, 32) 4640 ['max\_pooling2d[0][0]']

conv2d\_3 (Conv2D) (None, 64, 64, 32) 9248 ['conv2d\_2[0][0]']

max\_pooling2d\_1 (MaxPooling2D) (None, 32, 32, 32) 0 ['conv2d\_3[0][0]']

conv2d\_4 (Conv2D) (None, 32, 32, 64) 18496 ['max\_pooling2d\_1[0][0]']

conv2d\_5 (Conv2D) (None, 32, 32, 64) 36928 ['conv2d\_4[0][0]']

max\_pooling2d\_2 (MaxPooling2D) (None, 16, 16, 64) 0 ['conv2d\_5[0][0]']

conv2d\_6 (Conv2D) (None, 16, 16, 128) 73856 ['max\_pooling2d\_2[0][0]']

conv2d\_7 (Conv2D) (None, 16, 16, 128) 147584 ['conv2d\_6[0][0]']

max\_pooling2d\_3 (MaxPooling2D) (None, 8, 8, 128) 0 ['conv2d\_7[0][0]']

conv2d\_8 (Conv2D) (None, 8, 8, 256) 295168 ['max\_pooling2d\_3[0][0]']

conv2d\_9 (Conv2D) (None, 8, 8, 256) 590080 ['conv2d\_8[0][0]']

up\_sampling2d (UpSampling2D) (None, 16, 16, 256) 0 ['conv2d\_9[0][0]']

concatenate (Concatenate) (None, 16, 16, 384) 0 ['up\_sampling2d[0][0]',

'conv2d\_7[0][0]']

conv2d\_10 (Conv2D) (None, 16, 16, 128) 442496 ['concatenate[0][0]']

conv2d\_11 (Conv2D) (None, 16, 16, 128) 147584 ['conv2d\_10[0][0]']

up\_sampling2d\_1 (UpSampling2D) (None, 32, 32, 128) 0 ['conv2d\_11[0][0]']

concatenate\_1 (Concatenate) (None, 32, 32, 192) 0 ['up\_sampling2d\_1[0][0]',

'conv2d\_5[0][0]']

conv2d\_12 (Conv2D) (None, 32, 32, 64) 110656 ['concatenate\_1[0][0]']

conv2d\_13 (Conv2D) (None, 32, 32, 64) 36928 ['conv2d\_12[0][0]']

up\_sampling2d\_2 (UpSampling2D) (None, 64, 64, 64) 0 ['conv2d\_13[0][0]']

concatenate\_2 (Concatenate) (None, 64, 64, 96) 0 ['up\_sampling2d\_2[0][0]',

'conv2d\_3[0][0]']

conv2d\_14 (Conv2D) (None, 64, 64, 32) 27680 ['concatenate\_2[0][0]']

conv2d\_15 (Conv2D) (None, 64, 64, 32) 9248 ['conv2d\_14[0][0]']

up\_sampling2d\_3 (UpSampling2D) (None, 128, 128, 32 0 ['conv2d\_15[0][0]']

)

concatenate\_3 (Concatenate) (None, 128, 128, 48 0 ['up\_sampling2d\_3[0][0]',

) 'conv2d\_1[0][0]']

conv2d\_16 (Conv2D) (None, 128, 128, 16 6928 ['concatenate\_3[0][0]']

)

conv2d\_17 (Conv2D) (None, 128, 128, 16 2320 ['conv2d\_16[0][0]']

)

conv2d\_18 (Conv2D) (None, 128, 128, 1) 17 ['conv2d\_17[0][0]']

==================================================================================================

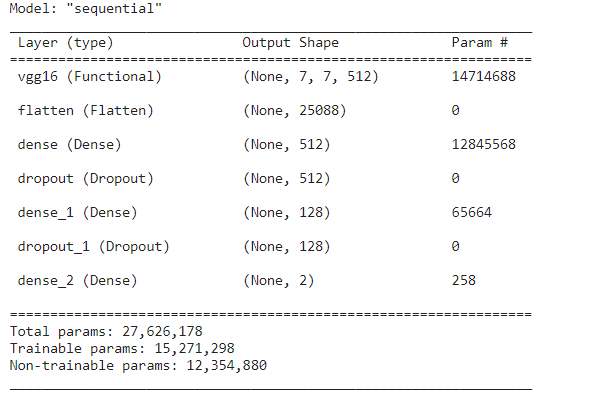
Total params: 1,962,337

Trainable params: 1,962,337

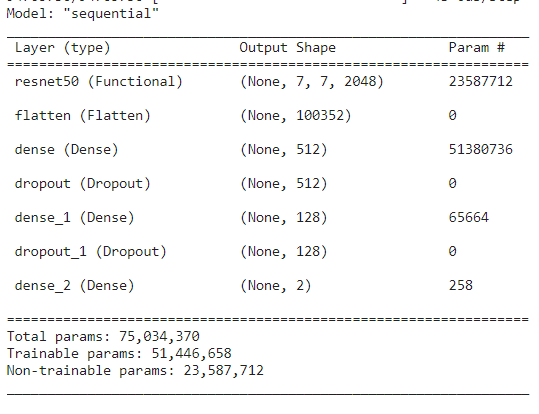
Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

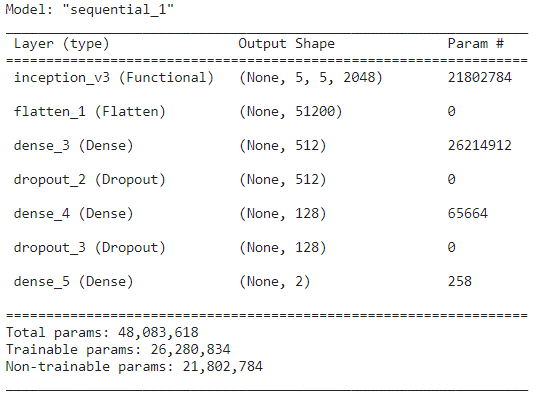
*Vgg16 Architecture*

**

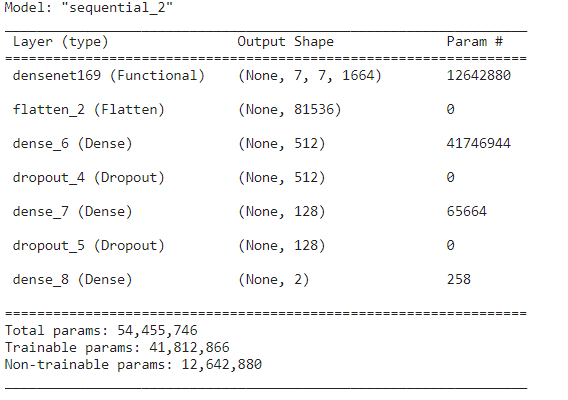
*ResNet50*

**

*Inception V3*

**

*DenseNet169*

**

**Pre Training Models:**

VGG-16, ResNet 50, Inception v3, DenseNet-169

**All Metrics:**

1. **Dice Coefficient:**

The Dice coefficient is a commonly used metric in deep learning, particularly in image segmentation tasks. It is a measure of the similarity between two sets of data, and it is often used to evaluate the accuracy of segmentation models. The Dice coefficient, also known as the Sørensen-Dice coefficient, is calculated as twice the intersection of the predicted and actual segmentation masks, divided by the sum of the number of pixels in both masks. The formula for the Dice coefficient is as follows:

Dice = (2 \* |X ∩ Y|) / (|X| + |Y|)

where X is the predicted segmentation mask and Y is the ground truth segmentation mask. The Dice coefficient ranges from 0 to 1, with a value of 1 indicating a perfect match between the predicted and ground truth masks. A value of 0 indicates that there is no overlap between the two masks.

1. **IOU Coefficient:**

The IOU (Intersection over Union) coefficient is a popular evaluation metric in deep learning used to measure the overlap between the predicted and ground truth bounding boxes or segmentation masks in object detection and segmentation tasks. It is calculated as the ratio of the area of the intersection between the predicted and ground truth boxes/masks to the area of their union. The IOU coefficient ranges from 0 to 1, with a value of 1 indicating a perfect overlap between the predicted and ground truth boxes/masks. The IOU coefficient is often used in combination with other metrics, such as precision and recall, to evaluate the performance of object detection and segmentation models.

1. **Precision:**

Precision is a metric used to evaluate the performance of a model in binary classification tasks. It measures the proportion of true positive predictions over the total number of positive predictions made by the model. In other words, it measures how many of the model's positive predictions are actually correct. Precision is calculated as follows:

Precision = True Positives / (True Positives + False Positives)

A high precision value indicates that the model is making accurate positive predictions, while a low precision value indicates that the model is making many false positive predictions

1. **Recall**

In machine learning, recall is a metric used to evaluate the performance of a model in binary classification tasks. It measures the proportion of true positive predictions over the total number of actual positive samples in the data. In other words, it measures how many of the positive samples in the data are correctly identified by the model. Recall is calculated as follows:

Recall = True Positives / (True Positives + False Negatives)

A high recall value indicates that the model is correctly identifying most of the positive samples in the data, while a low recall value indicates that the model is missing many positive samples.

1. **F1 Score**

In machine learning, the F1 score is a metric used to evaluate the performance of a model in binary classification tasks. It is a weighted average of precision and recall, where the F1 score reaches its best value at 1 and worst value at 0. The F1 score is calculated as the harmonic mean of precision and recall:

F1 Score = 2 \* ((Precision \* Recall) / (Precision + Recall))

The F1 score provides a balance between precision and recall and is often used to evaluate the overall performance of a model in binary classification tasks, particularly when the classes are imbalanced. A high F1 score indicates that the model has a good balance of precision and recall, while a low F1 score indicates that the model is either biased towards precision or recall.

1. **Accuracy:**

In machine learning, accuracy is a metric used to evaluate the performance of a model in classification tasks. It measures the proportion of correctly classified samples over the total number of samples in the dataset. In other words, it measures how many of the model's predictions are correct. Accuracy is calculated as follows:

Accuracy = (Number of Correct Predictions) / (Total Number of Predictions)

A high accuracy value indicates that the model is making accurate predictions, while a low accuracy value indicates that the model is making many incorrect predictions. Accuracy is often used as a primary metric to evaluate the performance of classification models when the classes are balanced.

1. **ROC and AUC Curve:**

In machine learning, the ROC (Receiver Operating Characteristic) curve and AUC (Area Under the Curve) are commonly used metrics for evaluating the performance of binary classification models.

The ROC curve is a plot of the true positive rate (TPR) against the false positive rate (FPR) at different classification thresholds. The TPR, also known as recall or sensitivity, is the proportion of actual positive samples that are correctly classified as positive by the model. The FPR is the proportion of actual negative samples that are incorrectly classified as positive by the model. The ROC curve provides a graphical representation of the trade-off between TPR and FPR for different classification thresholds.

The AUC is the area under the ROC curve and provides a single number that summarizes the overall performance of the model. The AUC ranges from 0 to 1, with a value of 1 indicating a perfect classifier and a value of 0.5 indicating a random classifier. A higher AUC value indicates a better model performance in distinguishing between positive and negative samples.

The ROC curve and AUC are particularly useful for evaluating the performance of models in imbalanced classification tasks, where the classes are unevenly distributed. They provide a robust evaluation of the model's performance across different classification thresholds and can help to identify the best threshold for a particular application.

1. **Binary Accuracy:**

In machine learning, binary accuracy is a metric used to evaluate the performance of a model in binary classification tasks. It measures the proportion of correctly classified samples over the total number of samples in the dataset.

Binary accuracy is calculated as follows:

Binary Accuracy = (Number of Correct Predictions for the Positive Class + Number of Correct Predictions for the Negative Class) / Total Number of Predictions

where the positive class refers to the class of interest, and the negative class refers to the other class. Binary accuracy provides an overall measure of how well the model is performing in terms of correctly classifying samples into their respective classes.

1. **Accuracy Loss:**

In machine learning, accuracy loss is a metric used to evaluate the performance of a model during the training process. It measures the difference between the model's predicted output and the actual output, in terms of accuracy. Accuracy loss is calculated as follows:

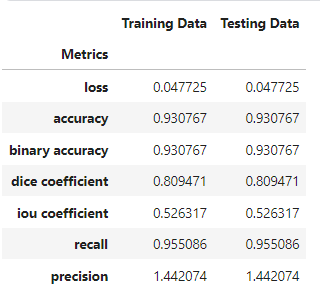
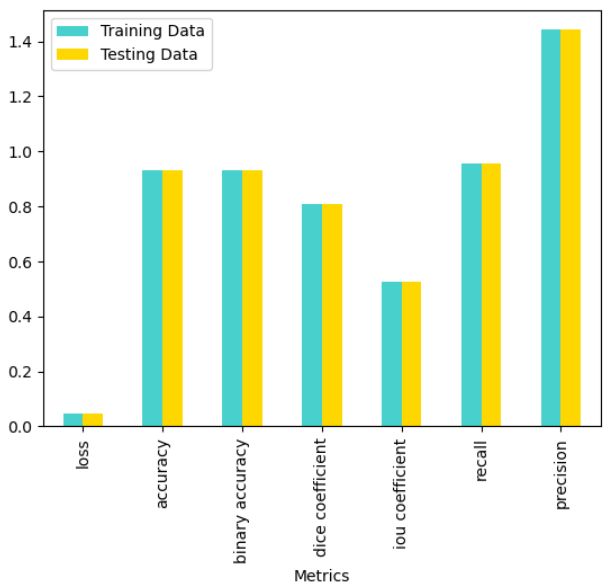
Accuracy Loss = (1 - Accuracy)

where Accuracy is the proportion of correctly classified samples over the total number of samples in the dataset.

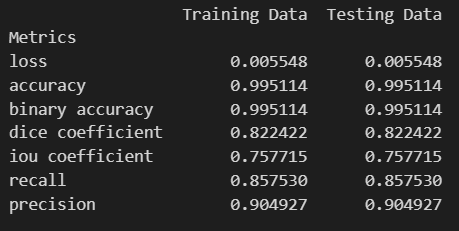
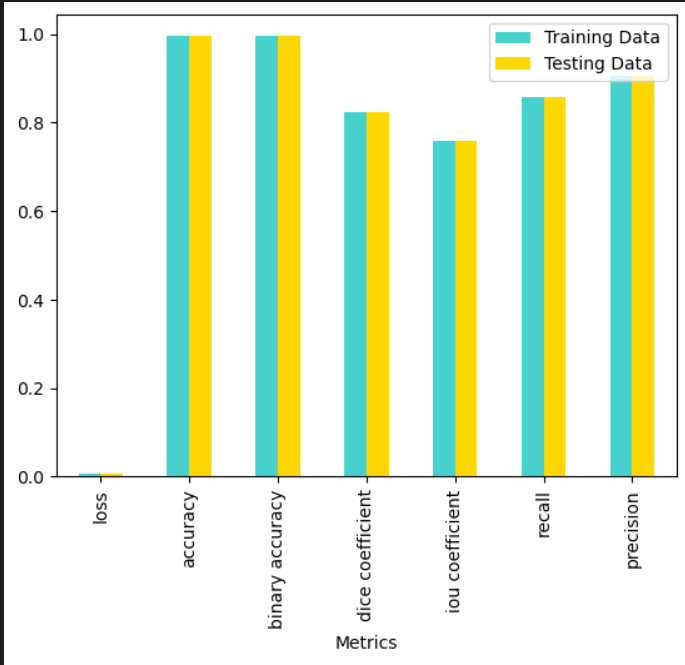
During the training process, the goal is to minimize the accuracy loss by adjusting the model's parameters through optimization algorithms. The accuracy loss metric is used to determine how well the model is performing during training and to guide the optimization process towards improving the model's accuracy. Accuracy loss is commonly used in machine learning algorithms, particularly for supervised learning tasks such as classification and regression. It is a key metric for evaluating the performance of the model and determining when to stop the training process to prevent overfitting.

**Results:**

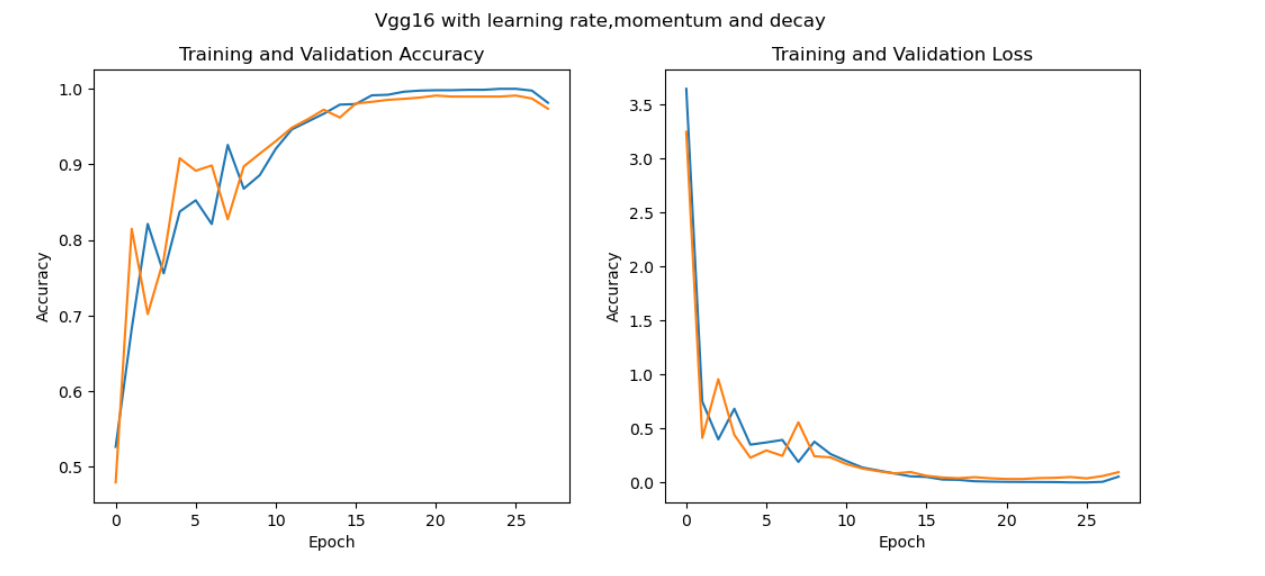
*Unet - Lung Segmentation*

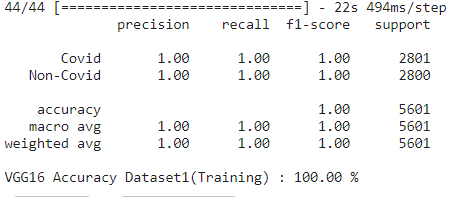


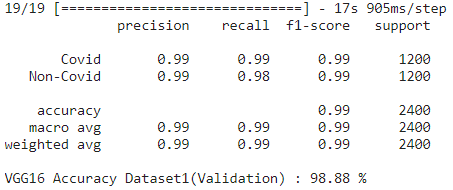
*Unet - Infection Segmentation*

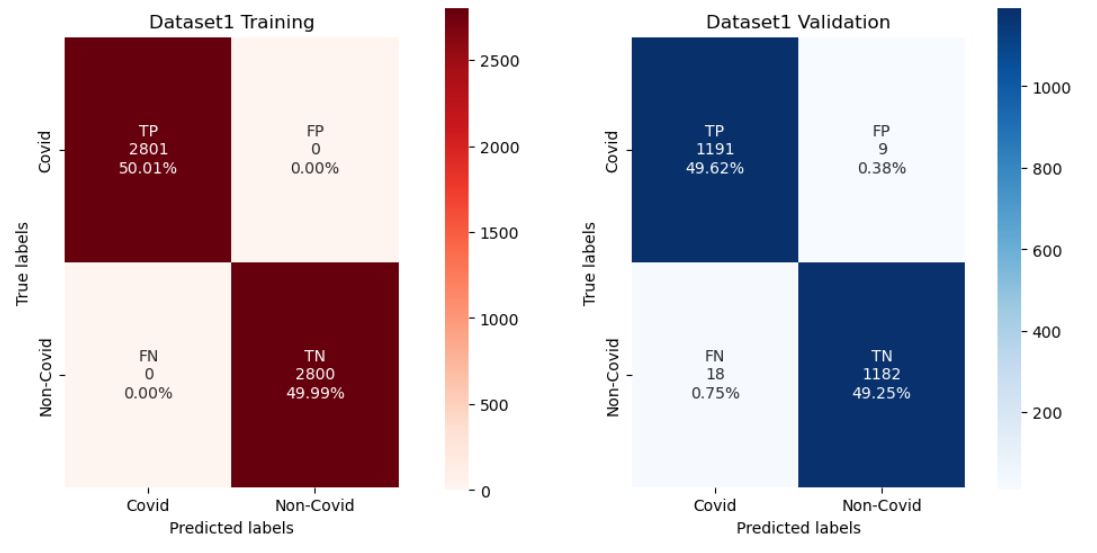


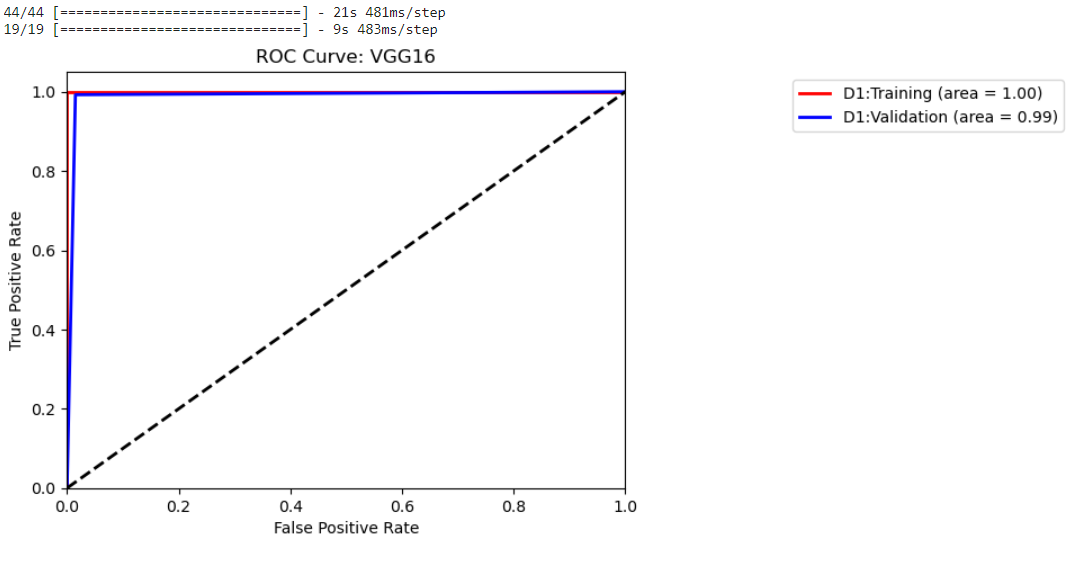
*Vgg16*



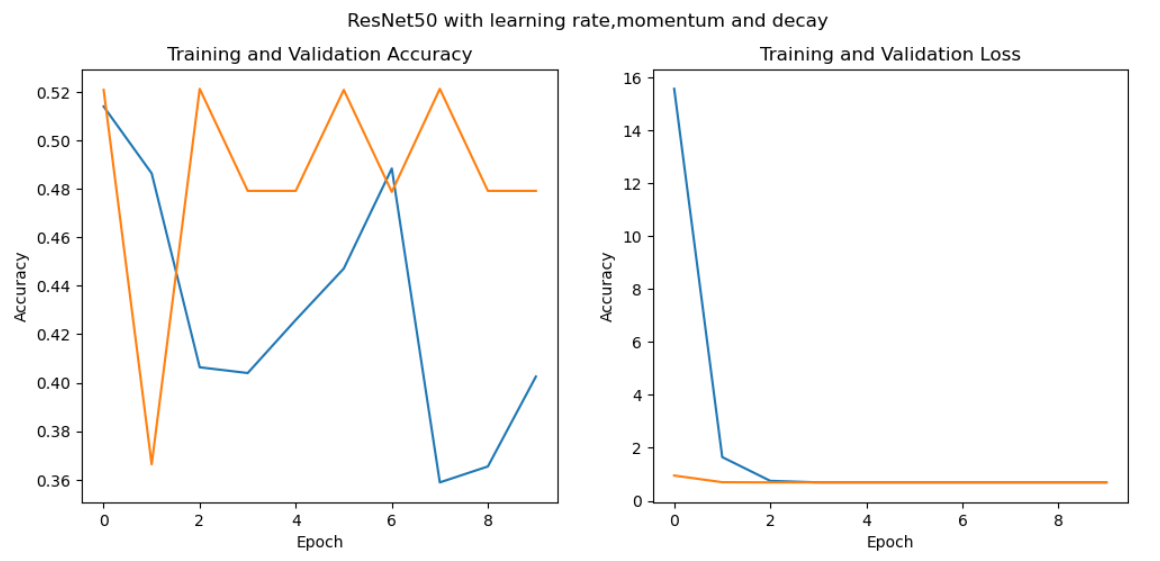


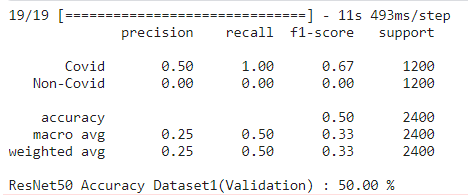


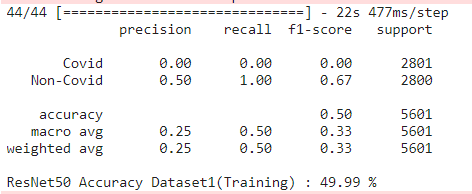


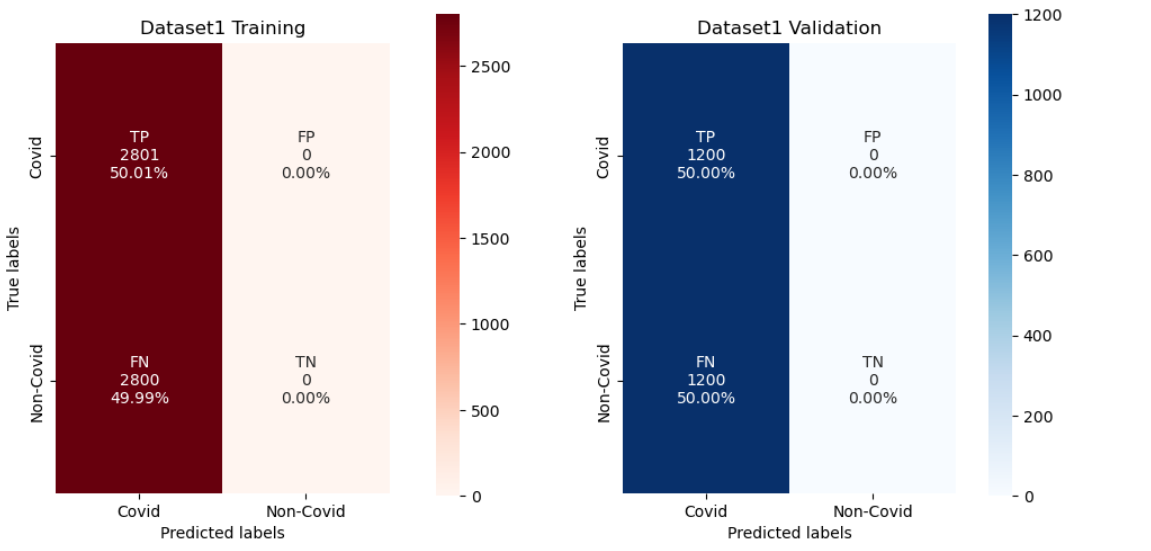


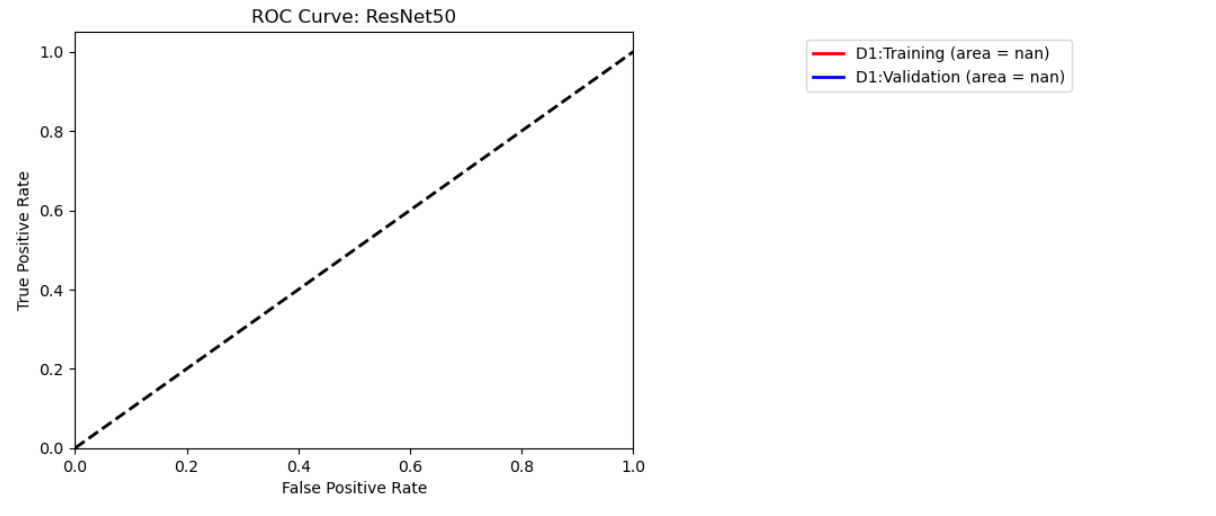
*Resnet 50*



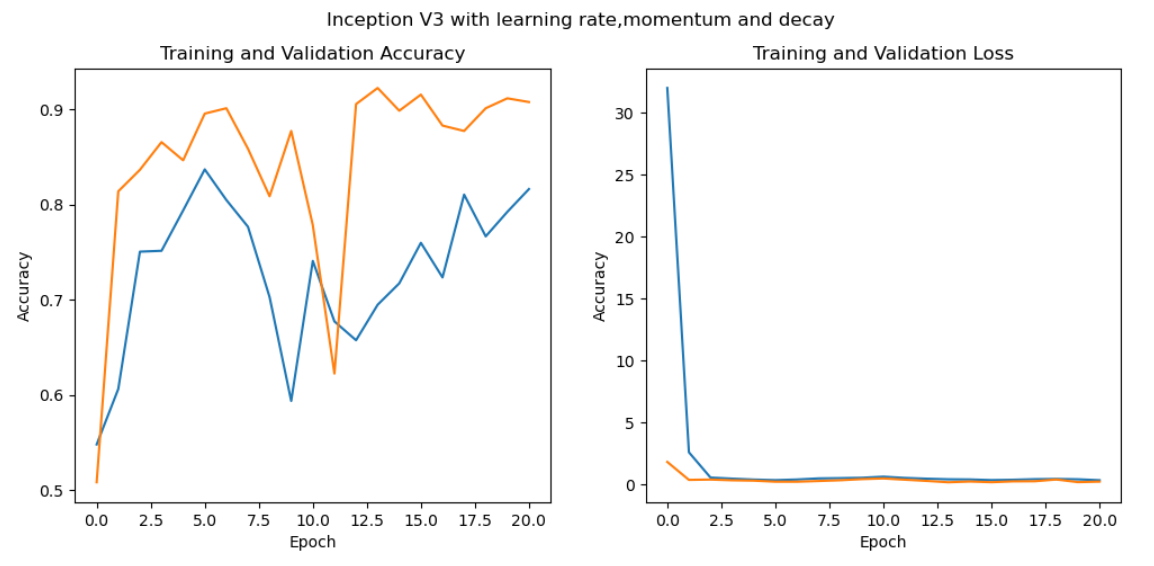


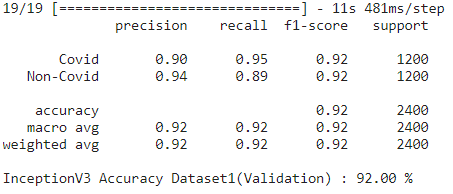


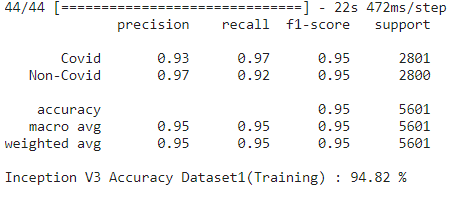


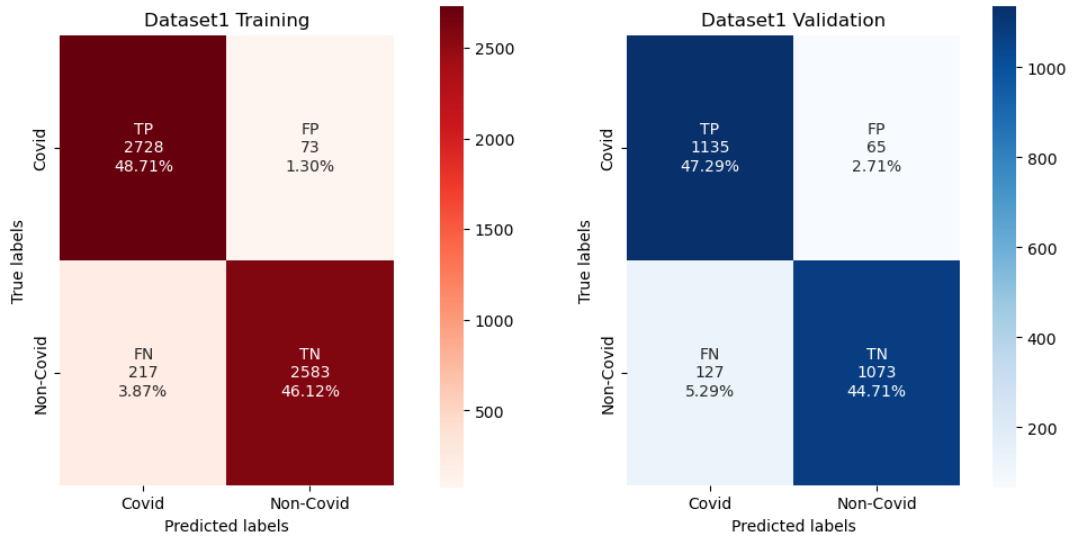


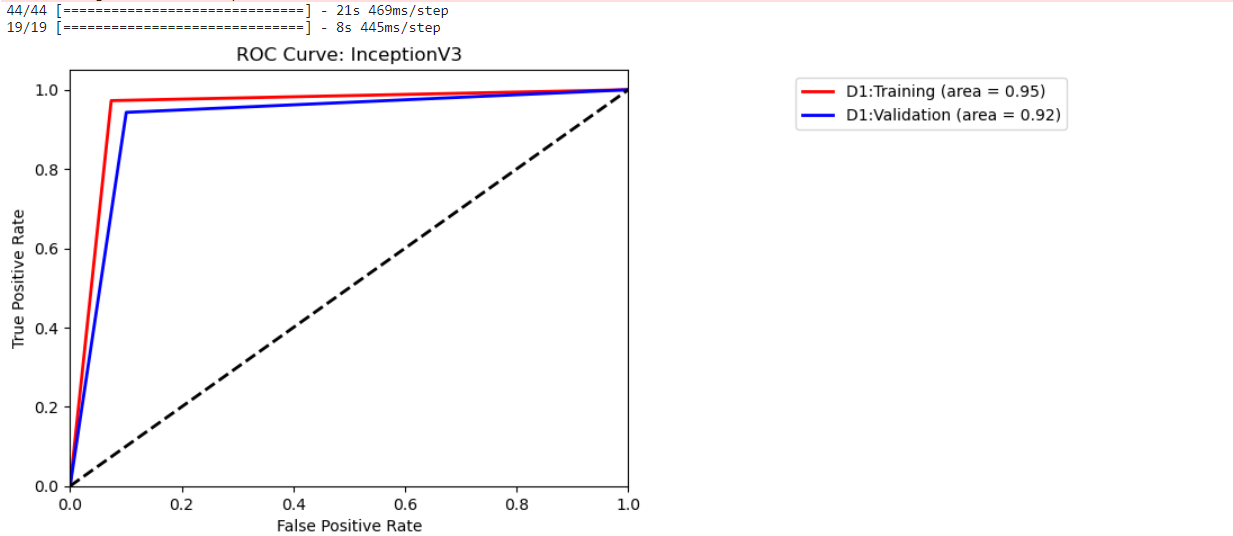
*Inception V3*



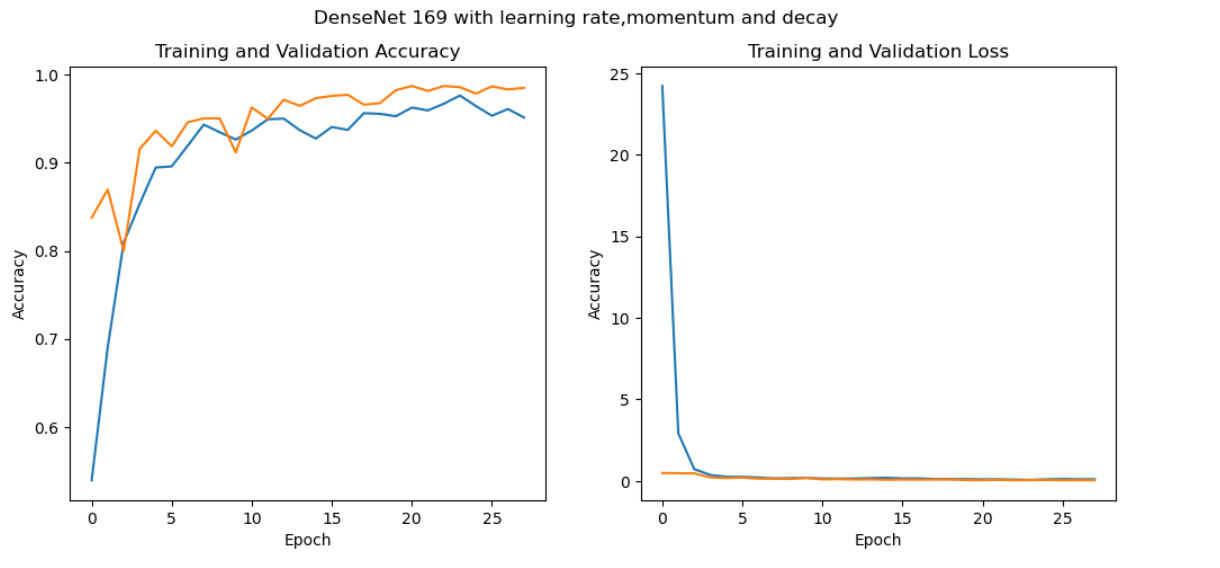


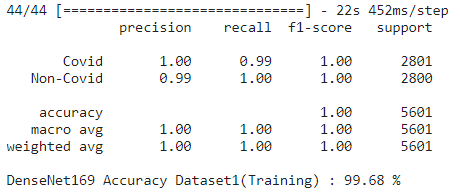


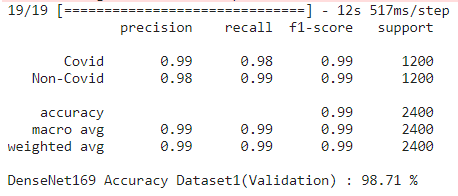


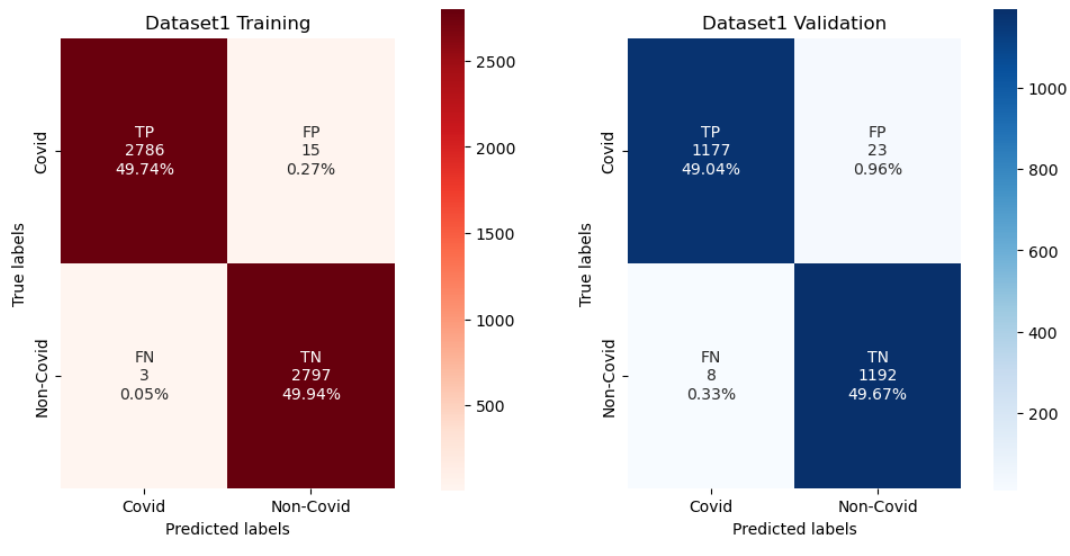


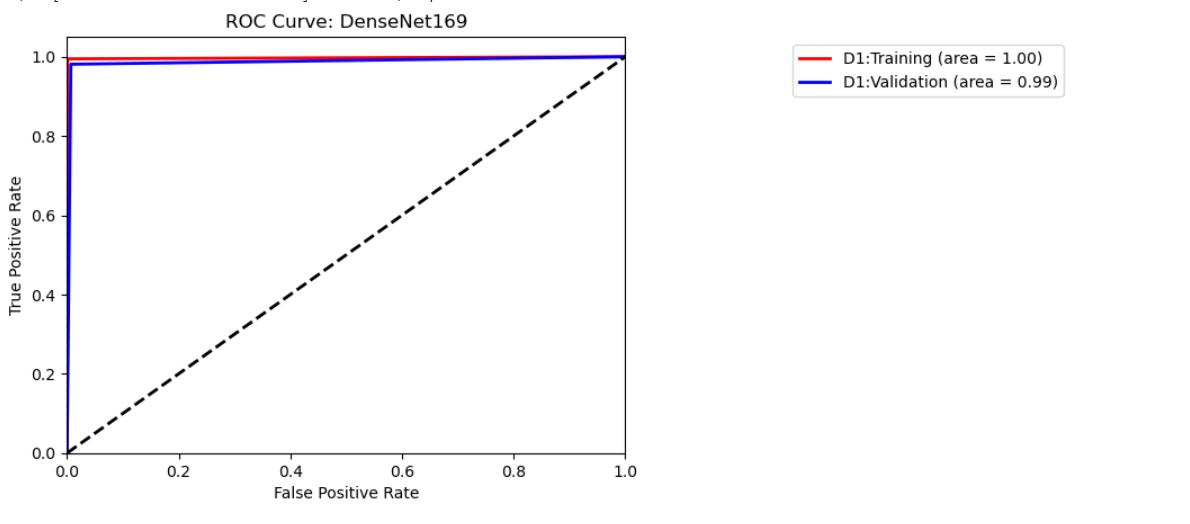
*DenseNet169*











**Limitations and Future Scope:**

· Inter dataset accuracies currently achieved are not as good as we hoped. Improvements could be made in that area

· For training the infection mask model, we used dataset having all covid images. If non covid image is feeded to the model it will not be able to understand.

· For a more accurate diagnosis, the same modeling could be done on a complete high quality medical image dataset.

· A frontend web application could be implemented for the same where any user can enter an image, and the model will return whether the lung has covid or not. If yes, infection mask image could be shown to the user.

**Conclusion:**

In conclusion, the COVID-19 pandemic has emphasized the importance of efficient and accurate diagnosis methods. This project addresses this need by developing a computer-aided diagnosis system that can accurately detect COVID-19 from CT scans. The system uses AI techniques, such as deep neural networks, to analyze and interpret CT images and can identify specific patterns and features in the images that are indicative of COVID-19. The system's ability to segment the lungs and detect the location of the infection can assist medical professionals in making a faster and more accurate diagnosis, which is critical in controlling the spread of the virus. This project's outcomes have significant potential to improve the speed and accuracy of COVID-19 diagnosis, particularly in areas with limited access to specialized medical resources.