AIML Capstone
Pneumonia Detection Challenge

Final Report Submission

AIML Group
April 2021 A

April 3, 2022

Contents

1. Introduction

1.1 Team Details

Name
Bipul K Ghosh
Laxmipathi Bhat
Moses Antony Durai
Shreyas Sonkusale
Siva Sai Kumar

1.2 Document Version

Date	Document version	Description
March 13, 2022	1.0	Interim Report Submission
April 3, 2022	2.0	Final Report Submission

1.3 Project Description

In this capstone project, the goal is to build a pneumonia detection system, to locate the position of inflammation in an image. As a part of the interim report submission, the team is to cover the following:

1. Summary of problem statement, data and findings
   - Understand the data.
   - Make an abstract or an overview based on your approach.
   - Break the problem into smaller tasks.
   - Discuss among your teammates and share responsibilities.

2. Summary of the Approach to EDA and Pre-processing:
   - Include any insightful visualization.
   - Identification of meaningful features
   - Interactions or summary data

3. Deciding Models and Model Building
   - Selection of suitable algorithm and its justification

4. Summary of Problem Statement, Data and Findings

4.1 Problem Statement

The objective of the project is to build a pneumonia detection system to locate the position of inflammation in an image.

4.2 Data and Findings

Two csv files contain the following columns:
stage_2_train_labels.csv

patientId - A patientId. Each patientId corresponds to a unique image (which we will see a little bit later)

x - The upper-left x coordinate of the bounding box
y - The upper-left y coordinate of the bounding box
width - The width of the bounding box
height - The height of the bounding box
Target - The binary Target indicating whether this sample has evidence of pneumonia or not.

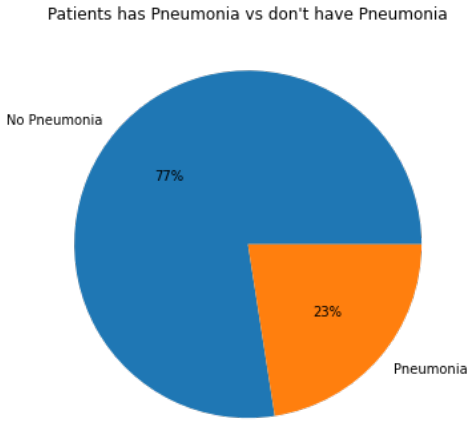
stage_2_test_images.csv
patientId - patient's unique id that is common with other files and dicom images
class - class type of the patitient

Sample Data:
The header of the stage_2_train_labels.csv dataset can be seen below:

	patientId	x	y	width	height	Target
0	0004cfab-14fd-4e49-80ba-63a80b6bddd6	NaN	NaN	NaN	NaN	0
1	00313ee0-9eaa-42f4-b0ab-c148ed3241cd	NaN	NaN	NaN	NaN	0
2	00322d4d-1c29-4943-afc9-b6754be640eb	NaN	NaN	NaN	NaN	0
3	003d8fa0-6bf1-40ed-b54c-ac657f8495c5	NaN	NaN	NaN	NaN	0
4	00436515-870c-4b36-a041-de91049b9ab4	264.0	152.0	213.0	379.0	1
5	00436515-870c-4b36-a041-de91049b9ab4	562.0	152.0	256.0	453.0	1

Data Findings:

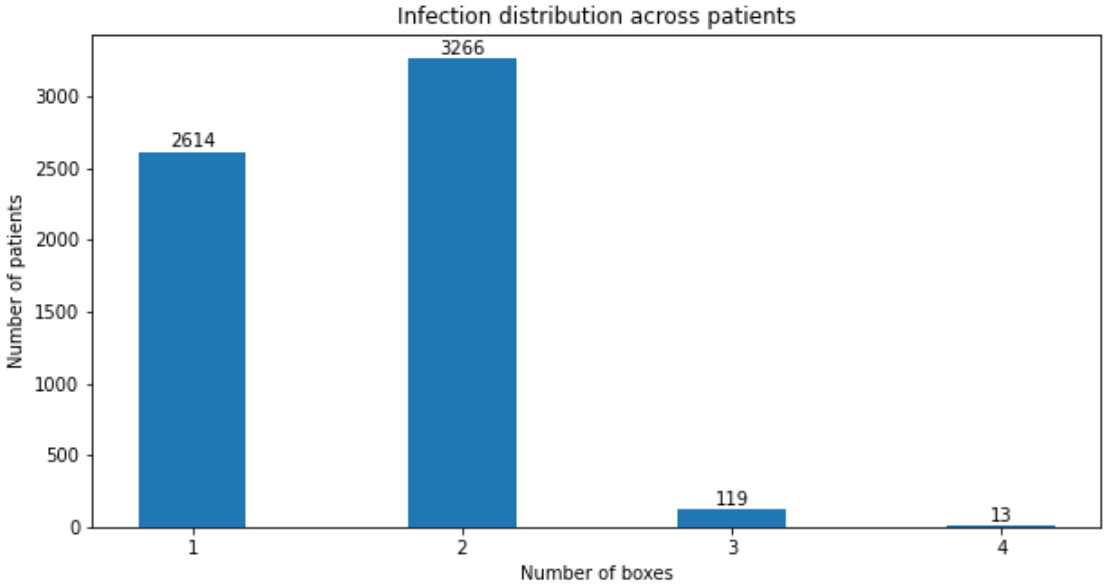
1. Each row in the CSV file contains a patientId (one unique value per patient), a target (either 0 or 1 for absence or presence of pneumonia, respectively) and the corresponding abnormality bounding box defined by the upper-left hand corner (x, y) coordinate and its corresponding width and height. In this particular case, the patient does *not* have pneumonia and so the corresponding bounding box information is set to NaN.
2. The train_labels dataframe has 30227 rows and 6 columns.
3. Number of unique patientId are: 26684
4. No of patients who has Pneumonia: 6012 (i.e., 23.0%)
5. No of patients who don't have Pneumonia: 20672 (i.e., 77.0%)



5. Summary of the Approach to EDA and Pre-processing

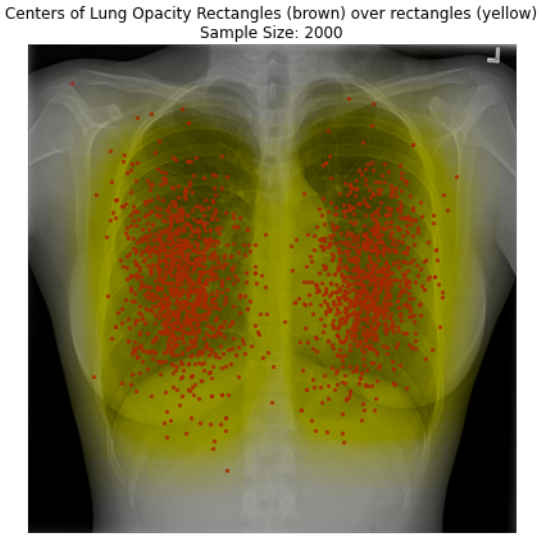
5.1 Exploratory Data Analysis

The EDA and pre-processing done after merging of both the dataset based on our finding from the earlier section.



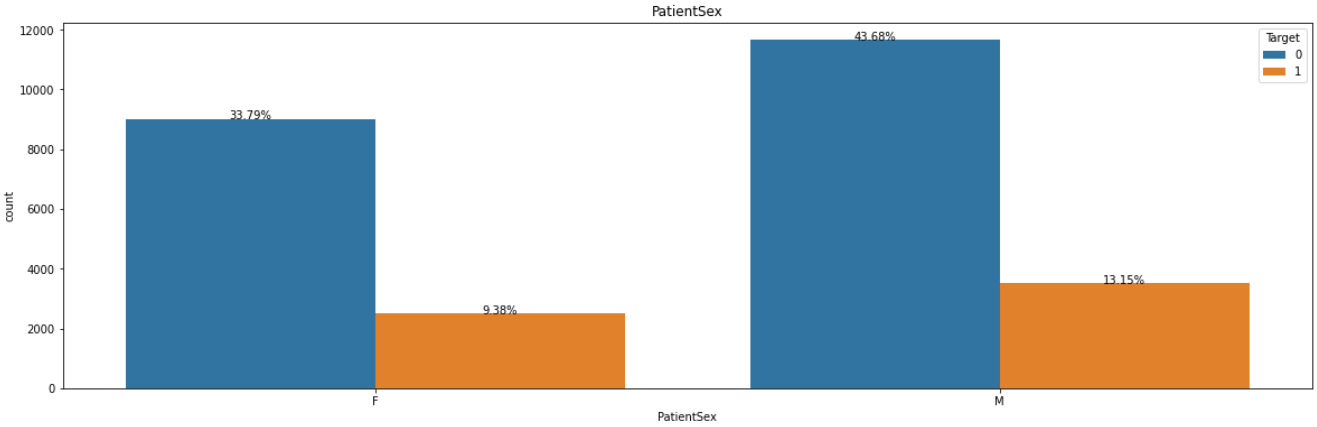
The above barplot represent the number of bounding boxes classes from the entire dataset. After merging the dataset has 37,629 rows and 8 columns and the number of unique patients is 26,684.

Let's look at the area of the infection with the sample of 2000 patients. Make use of the normal DICOM images and apply the scatter plot to approximately visualize the area of inflection. It look like the with the sample size of 2000 both the lung has similar infection rate.

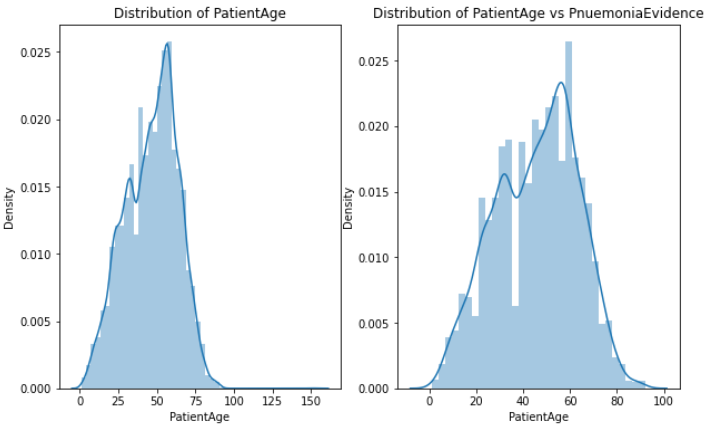


The inflammation spread is even on both the side of the lungs (left & right) with the severity is more at the bottom than the top.
Note: The scatter points are the centroid of each bounding box for the sample patients

There are no huge class imbalance found for patient gender versus target column – the same can found from the below visualisation



The below histogram shows the age wise distribution of the inflection, there are some peaks found with the age 40 and 58.



5.2 Hypothesis testing between PatientSex (categorical) and Target (categorical)

According to five steps process of hypothesis testing:
reference link

1. Define Hypothesis.
2. Build a Contingency table.
3. Find the expected values.
4. Calculate the Chi-Square statistic.
5. Accept or Reject the Null Hypothesis.

Step 1: - Define Hypothesis

Null Hypothesis (H_o): whether PatientSex and Target are independent, i.e. no relationship
Alternative Hypothesis (H_1): whether PatientSex and Target are dependent, i.e. has a relationship
$\alpha = 0.05$

Step 2: - Build a Contingency table

```
data_crosstab = pd.crosstab(training_data['PatientSex'], training_data['Target'], margins=True, margins_name="Total")
data_crosstab
```

Target	0	1	Total
PatientSex			
F	9016	7115	16131
M	11656	9842	21498
Total	20672	16957	37629

Step 3: - Find the expected values

Based on the null hypothesis that the two variables are independent. We can say if A, B are two independent events

$P(A \cap B) = P(A) * P(B)$

Let's calculate the expected value for the first cell (E1) that is those who are Females and are not deducted for Pneumonia.

```
# p = p(no pneumonia) * p(female)
p = p(20672/37629) * p(16131/37629)
p = 0.2355040789
# now, E1 is 37629 * 0.2355040789 = 8861.78298652635
```

```
rows = training_data['PatientSex'].unique()
columns = training_data['Target'].unique()
data = []
for i in columns:
  row_data =[]
  for j in rows:
    E = data_crosstab[i]['Total'] * data_crosstab['Total'][j] / data_crosstab['Total']['Total']
    row_data.append(E)
  data.append(row_data)

# Create the pandas DataFrame
expected_value_df = pd.DataFrame(data, columns = ['F', 'M'])

#Total sum per column:
expected_value_df.loc['Total',:]= expected_value_df.sum(axis=0)

#Total sum per row:
expected_value_df.loc[:,'Total'] = expected_value_df.sum(axis=1)

expected_value_df.head().T
```

Step 4: - Calculate the Chi-Square statistic

```
# significance level
alpha = 0.05

# Calcualtion of Chisquare
chi_square = 0
rows = training_data['PatientSex'].unique()
columns = training_data['Target'].unique()
for i in columns:
  for j in rows:
    O = data_crosstab[i][j]
    E = data_crosstab[i]['Total'] * data_crosstab['Total'][j] / data_crosstab['Total']['Total']
    chi_square += (O-E)**2/E

print(f'We can see Chi-Square is calculated as {chi_square}!')
```

```
We can see Chi-Square is calculated as 10.424176151069577!
```

Step 5: - Accept or Reject the Null Hypothesis

```
# The p-value approach
print("Approach 1: The p-value approach to hypothesis testing in the decision rule")
p_value = 1 - stats.chi2.cdf(chi_square, (len(rows)-1)*(len(columns)-1))
conclusion = "Failed to reject the null hypothesis."
if p_value <= alpha:
    conclusion = "Null Hypothesis is rejected."

print("chisquare-score is:", chi_square, " and p value is:", p_value)
print(conclusion)
```

```
Approach 1: The p-value approach to hypothesis testing in the decision rule
chisquare-score is: 10.424176151069577  and p value is: 0.0012437632740927018
Null Hypothesis is rejected.
```

```
# The critical value approach
print("Approach 2: The critical value approach to hypothesis testing in the decision rule")
critical_value = stats.chi2.ppf(1-alpha, (len(rows)-1)*(len(columns)-1))
conclusion = "Failed to reject the null hypothesis."
if chi_square > critical_value:
    conclusion = "Null Hypothesis is rejected."

print("chisquare-score is:", chi_square, " and critical value is:", critical_value)
print(conclusion)
```

```
Approach 2: The critical value approach to hypothesis testing in the decision rule
chisquare-score is: 10.424176151069577  and critical value is: 3.841458820694124
Null Hypothesis is rejected.
```

Conclusion

We have enough evidence that there is an association between PatientSex and their Target column, at 5% significance level.

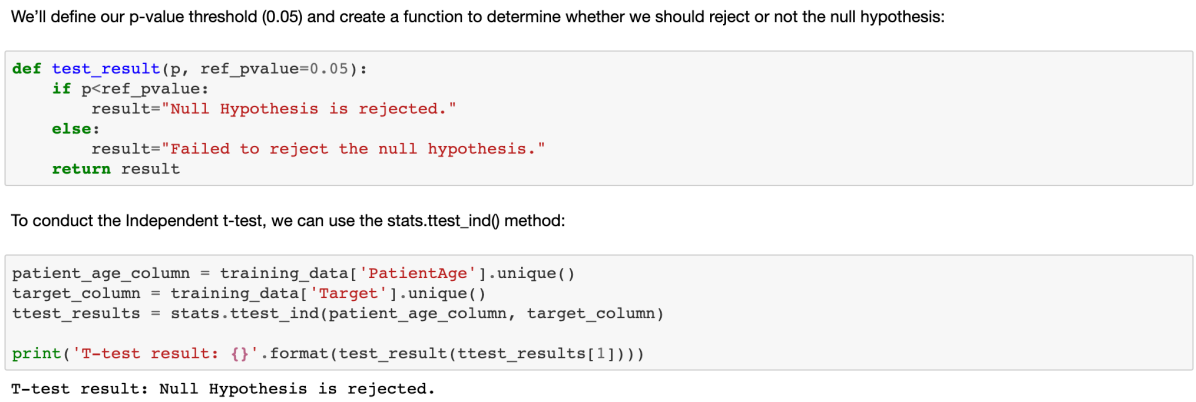
5.3 Hypothesis testing between PatientAge (continuous) and Target (categorical)

This is a useful type of test used to compare one numeric feature (as in this case it is PatientAge) and categorical column (as in this case it is pneumonia exist or not) in a dataset. It determines if there is a significant difference between these two columns, by yielding a p-value, which tells you how likely the difference observed in the sample between those two column could have happened if there was no difference at the population level (that is our null hypothesis). If the p-value is less than a certain threshold (usually 0.05) then there's a statistically significant difference between the two column under investigation.

reference link

Definition of hypothesis
Null Hypothesis (H_0): whether PatientAge and Target are independent, i.e. no relationship
Alternative Hypothesis (H_1): whether PatientAge and Target are dependent, i.e. has a relationship
p-value thershold = 0.05

T-test
We'll define our p-value threshold (0.05) and create a function to determine whether we should reject or not the null hypothesis:



Conclusion
We have enough evidence that there is an association between PatientAge and their Target column, at 5% significance level.

5.4 Pre-Processing

The following pre-processing steps were applied when performing Exploratory Data Analysis on the data set:

1. Performed rows with null values check.
2. Merged both the two CSV files – added class, path of image and patient id columns.
3. There was no duplicate values found.
4. Created new column – to represent the number of bounding boxes for each patient.

5. Converted the images from DICOM (*.DCM) format to JPEG (*.JPG). pydicom library used to load the dicom images into the notebook.
6. Resized the image to fit in the model input parameters.

6. Model Selection and Model Building

6.1 Models

Based on our case study, the comparative performance for different CNNs for two-class classification problem with and without augmentation is shown in Table below. It is apparent from this table that all the evaluated pretrained models perform very well in classifying COVID-19 and normal images in two-class problem. The weighted average performance matrix for eight different networks are very similar whereas small gain can be observed when training was done using image augmentation. Among the networks trained with 338 X-ray images for two-class problem, ResNet18 and CheXNet are equally performing for classifying images while CheXNet and DenseNet201 are performing better than others in case of training with augmented images, although the difference is marginal. CheXNet is producing the highest accuracy of 99.4% and 99.7% for two-class classification without and with image augmentation respectively. Interestingly, CheXNet is performing well in both the cases, with and without augmentation and this can be explained from the fact that CheXNet is the only network which is pre-trained on a large X-ray image database and the network supposed to perform better for X-ray image classification without the requirement of training again on a larger dataset. However, in this classification problem as the COVID-19 images are significantly different from normal images all the tested networks are performing well. This is apparent from the ROC curves of Figure 5 as well. In both the cases (without and with augmentation) for two-class problem, ROC curves are showing comparable performance from all the networks.

Schemes	Models	Accuracy	Precision (PPV)	Sensitivity (Recall)	F1 Scores	Specificity
Without image augmentation	SqueezeNet	95.19	95.27	95.19	95.23	97.59
	MobileNetv2	95.9	95.97	95.9	95.93	97.95
	ResNet18	95.75	95.8	95.75	95.78	97.88
	InceptionV3	94.96	94.98	94.95	94.96	97.49
	ResNet101	95.36	95.4	95.36	95.38	97.68
	CheXNet	**97.74**	**96.61**	**96.61**	**96.61**	**98.31**
	DenseNet201	95.19	95.06	95.9	95.04	97.87
	VGG19	95.04	95.06	95.03	95.04	97.51
With image augmentation	SqueezeNet	95.10	95.18	95.10	95.14	97.17
	MobileNetv2	96.22	96.25	96.22	96.23	97.80
	ResNet18	96.44	96.48	96.44	96.46	97.91
	InceptionV3	96.20	97.00	96.40	96.60	97.50
	ResNet101	96.22	96.24	96.22	96.23	97.80
	CheXNet	96.94	96.43	96.42	96.42	97.29
	DenseNet201	**97.94**	**97.95**	**97.94**	**97.94**	**98.80**
	VGG19	96.00	96.50	96.25	96.38	97.52

6.2 Inference

Hence, we summarize below on how to decide our modelling strategies:

- CNN perform better with a larger dataset than a smaller one. Transfer learning can be used in the training of deep CNNs where the dataset is not large.

- The concept of transfer learning uses the trained model from large dataset such as ImageNet and modify the Softmax and classification layer of the pre-trained networks.
- The pre-trained weights are then used for faster training of the network for an application with comparatively smaller dataset. This removes the requirement of having large dataset and also reduces the long training period as is required by the deep learning algorithm when developed from scratch.
- Eight different pre-trained CNN models were trained, validated and tested in this study. The experimental evaluation of MobileNetv2, SqueezeNet, ResNet18, ResNet101and DenseNet201
- Three comparatively shallow networks (MobileNetv2, SqueezeNet and ResNet18) and five deep networks (Inceptionv3, ResNet101, CheXNet, VGG19 and DenseNet201) were evaluated in this study to investigate whether shallow or deep networks are suitable for this application.
- Two different variants of ResNet were used to compare specifically the impact of shallow and deep networks with similar structure. Performance difference due to initially trained on different image classes other than X-ray images were compared with CheXNet, which is a 121-layer DenseNet variant and the only network pre-trained on X-ray images.
- Eight pre-trained CNN models were trained using stochastic Gradient Descent (SGD) with momentum optimizer with learning rate, α = 10-3, momentum update, β = 0.9 and mini-batch size of 16 images with 20 Back Propagation epochs. Fivefold cross-validation result was averaged to produce the final receiver operating characteristic (ROC) curve, confusion matrix, and evaluation matrices.
- Two different experiments were carried out in this study: i) Two-class image classification using models trained without and with images augmentation, and ii) Three-class image classification using models trained without and with image augmentation.

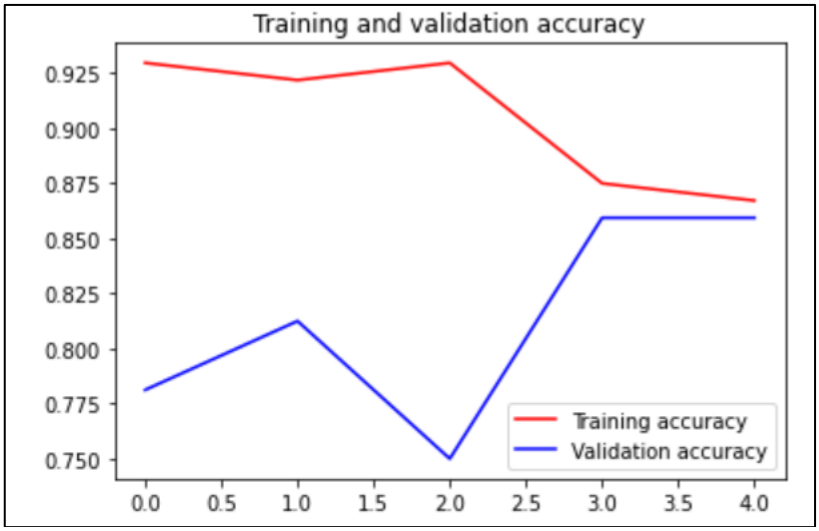
We started working with the following models to build and train using RSNA dataset:

Shallow Network	Deep Network
• MobileNetv2 - More Info	• ResNet101 - More Info • DenseNet121 - More Info • VGG19 - More Info • DenseNet201 - More Info

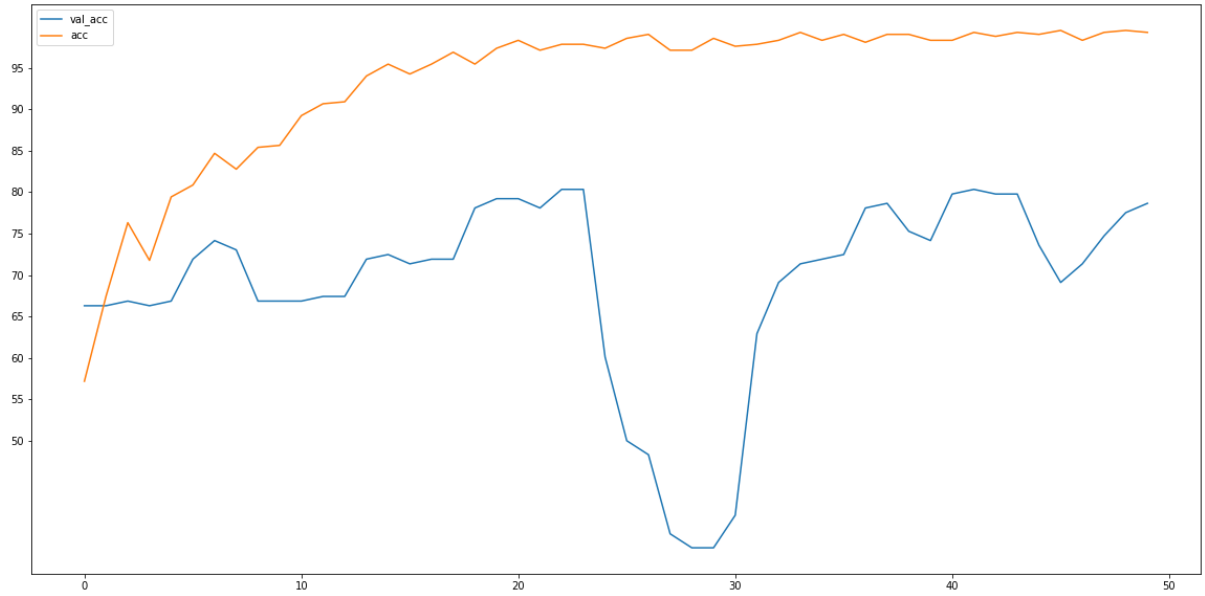
Below are the training and validation accuracy obtained different models:

Sl No	Model Name	Training Accuracy	Val Accuracy	Sample Size
1	ResNet101	0.8672	0.8594	1000
2	VGG19	0.9928	0.8034	1000
3	MobileNetv2	0.9967	0.9900	5000
4	DenseNet201	1.0000	0.7734	1000
5	DenseNet121	079	0.79	1000

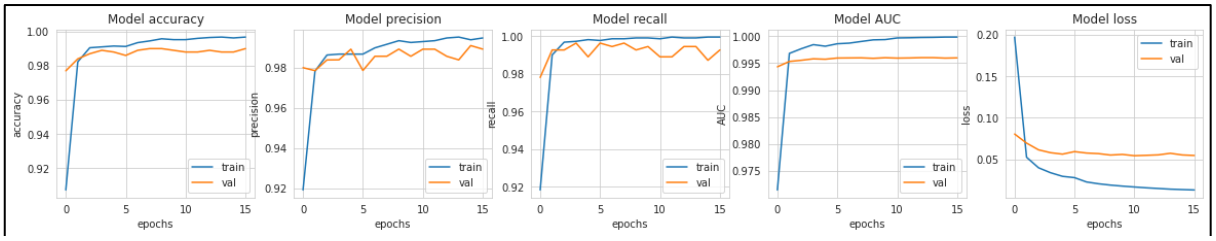
ResNet101 – Training & Validation Accuracy



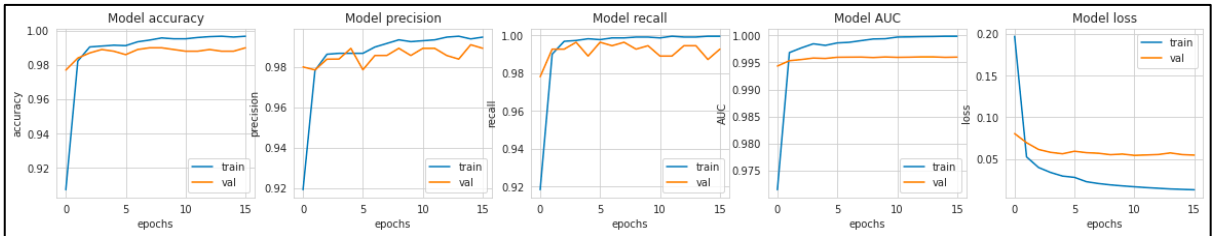
VGG19 – Training & Validation Accuracy



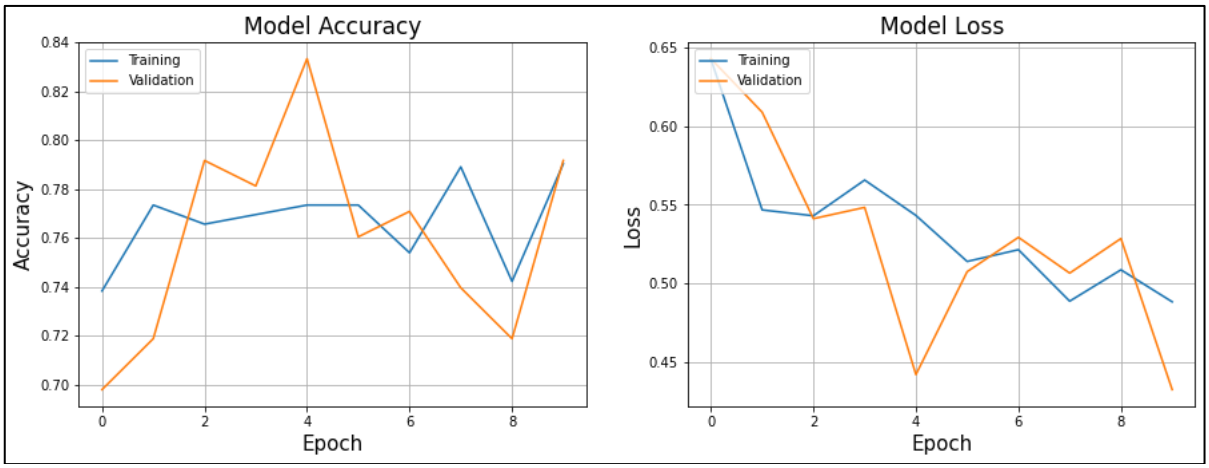
MobileNetv2 – Training & Validation Accuracy



DenseNet201 – Training & Validation Accuracy



DenseNet121 – Training & Validation Accuracy



7. How to improve your model performance?

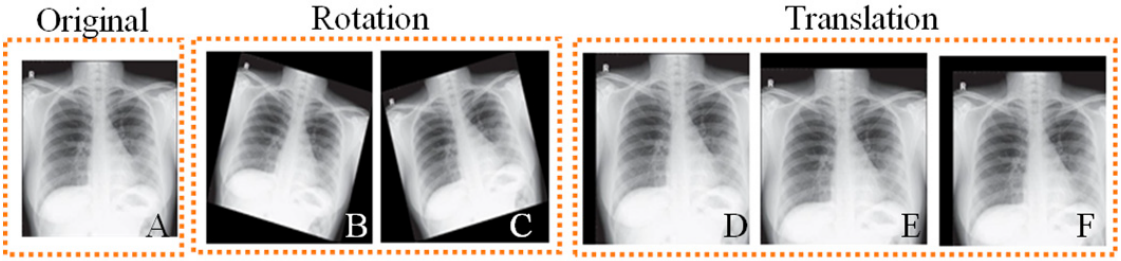
What are the approaches you can take to improve your model? Can you do some feature selection, data manipulation and model improvements.

In order to improve the performance of the models, following two approaches are recommended:

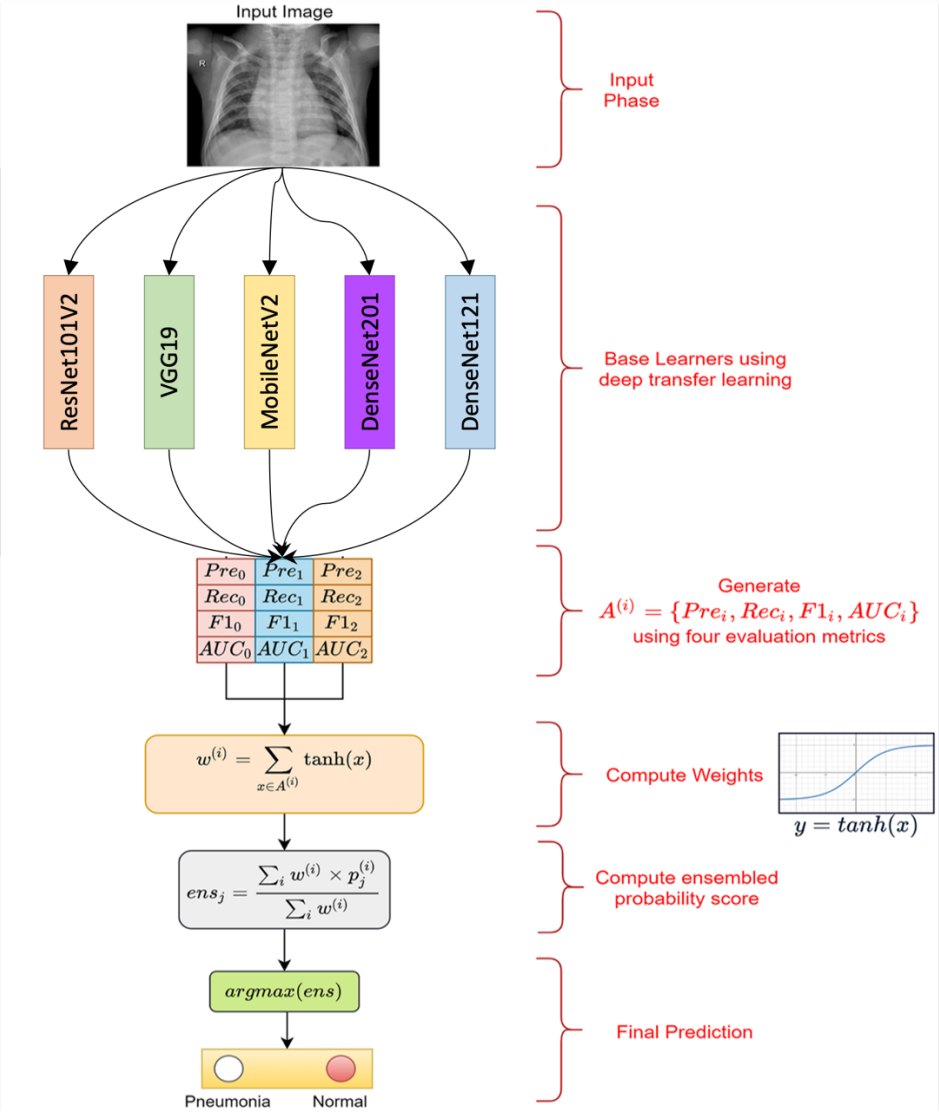
- Image augmentation technique
- Ensemble technique – including optimizers

Using augmentation technique, altering the existing data to create some more data for the model training process. In other words, it is the process of artificially expanding the available dataset for training a deep learning model.

In this picture, the image on the left is only the original image, and the rest of the images are generated from the original training image.



For Ensemble technique, the decisions of multiple classifiers (for our case it is binary classification) are fused to obtain the final prediction for a test sample. It is performed to capture the discriminative information from all the base classifiers, and thus, results in more accurate predictions. Some of the ensemble techniques that were most frequently used in studies in the literature are average probability, weighted average probability, and majority voting. The average probability-based ensemble assigns equal priority to each constituent base learner. However, for a particular problem, a certain base classifier may be able to capture information better than others. Thus, a more effective strategy is to assign weights to all the base classifiers. However, for ensuring the enhanced performance of the ensemble, the value of the weights assigned to each classifier is the most essential factor. Most approaches set this value based on experimental results. In this study, we devised a novel strategy for weight allocation, where four evaluation metrics, precision, recall, f1-score, and area under receiver operating characteristics (ROC) curve (AUC), were used to assign the optimal weight to base CNN models. In studies in the literature, in general, only the classification accuracy was considered for assigning weights to the base learners [8], which may be an inadequate measure, in particular when the datasets are class-imbalanced. Other metrics may provide better information for prioritizing the base learners. The overall workflow of the proposed ensemble framework is presented below.



8. Code and Reference

The colab notebook and html version are attached in the Olympus portal and also available in the GitHub - https://github.com/mosesad/aiml-apr-2021-group-a