AI4ICPS HACKATHON

AI for Classifying Chest X-Ray

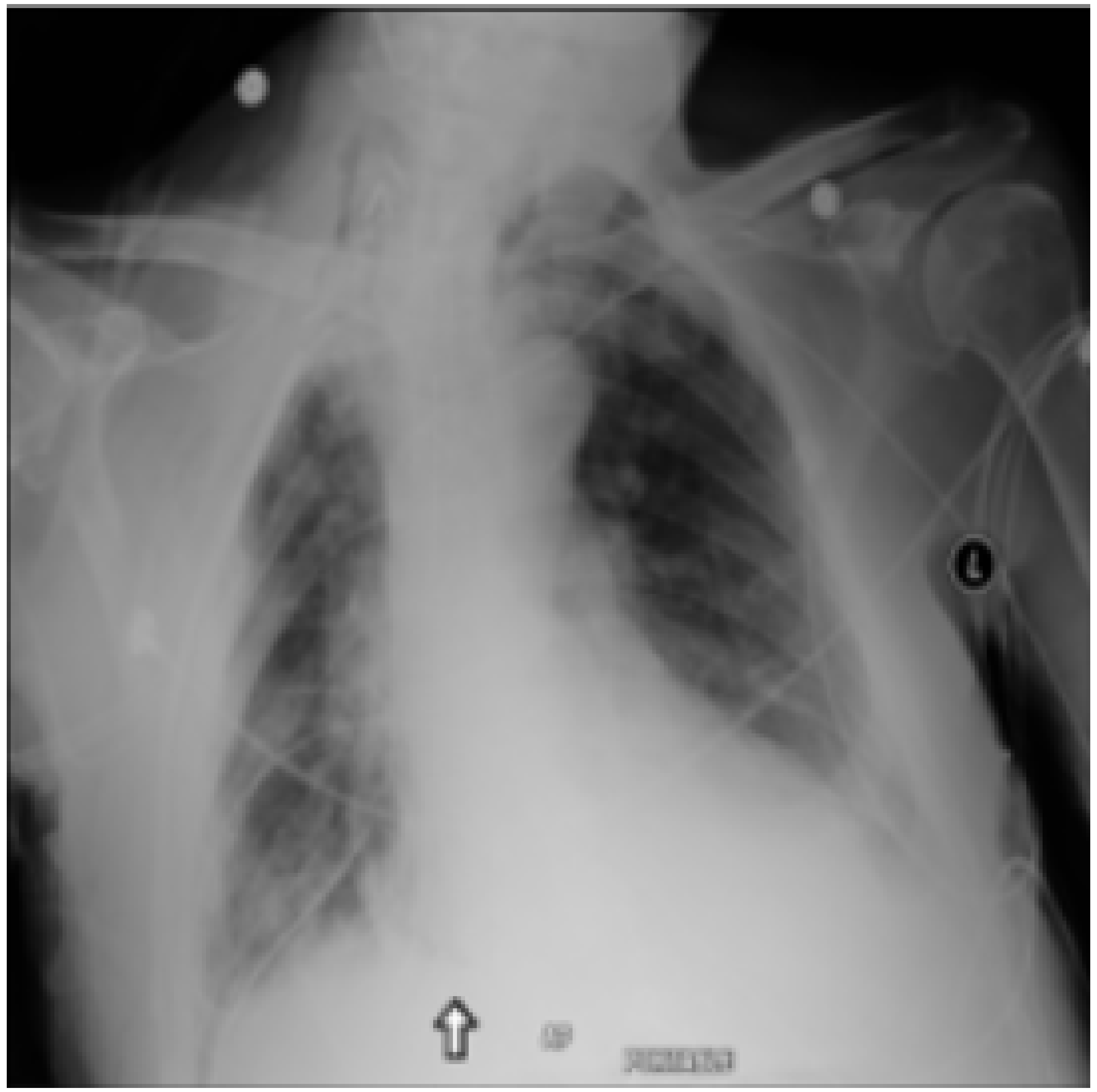
AI4ICPS

RISHABH MISHRA SAYAN GHOSH

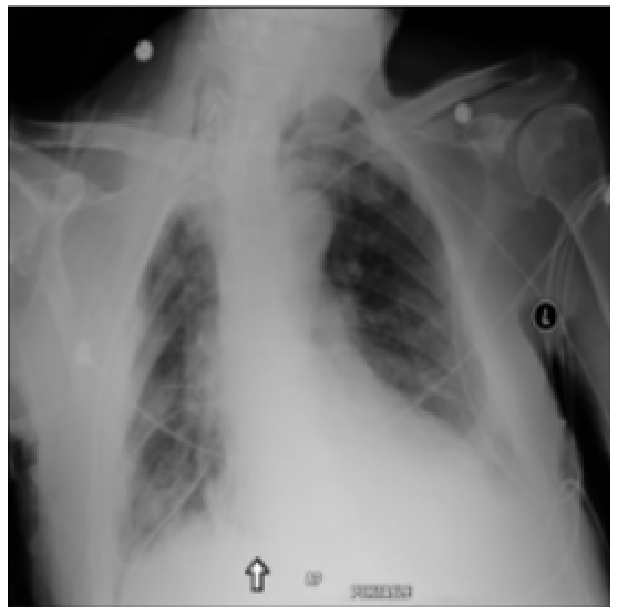
INDEX

Image Preprocessing

- Images were of 1024 x 1024 pixels with 3 channels.
- Resized the images to 256 x 256 keeping the number channels same.
- There was some information loss, so used Data Augmentation to reduce it.
- Normalized the pixel values since CNNs train better within [0,1]



1024x1024



256x256

Data Exploration

	Image Index	Finding Labels	Patient ID	Patient Age	Patient Gender	View Position
0	00000248_005.png	Atelectasis|Effusion|Mass	248	87	M	AP
1	00000248_006.png	Atelectasis|Infiltration	248	87	M	AP
2	00000248_007.png	Atelectasis|Infiltration	248	87	M	AP
3	00000248_008.png	Atelectasis	248	87	M	AP
4	00000248_009.png	Atelectasis	248	87	M	AP

- Along with the training images file, we are provided with a csv file with respective image file name and various other features related to that x-ray image.
- Dropped the Patient ID column as it contributed not much to the output label

- Generated dummy variables for the Patient's Gender and the View Position of the x-ray images as they are the categorical features.
- We are requested to find only if the x-ray images shows some problems or not. Since this is now a binary classification problem, using simple numPy function we converted the Finding Label (output variable) column into 0's and 1's for 'no finding' or 'found anything' respectively.

Data Exploration

```
# Define mapping dictionary
mapping_dict = {'No Finding': 0}

# Use replace() method to replace "no finding" values with 0
truth_1['Finding Labels'] = truth_1['Finding Labels'].replace(mapping_dict)

# Use apply() method to map all other values to 1
truth_1['Finding Labels'] = truth_1['Finding Labels'].apply(lambda x: 1 if x != 0 else x)
```

	Image Index	Patient Age	PA	M	Label
0	00000248_005.png	87	0	1	1
1	00000248_006.png	87	0	1	1
2	00000248_007.png	87	0	1	1
3	00000248_008.png	87	0	1	1
4	00000248_009.png	87	0	1	1

- Label encoding the 'Finding Labels' to the requirements of the Problem Statement.
- The Dataset was balanced with ~60,000 '0' labels and ~50,000 '1' labels.

After all those data manipulation the final csv file looks like this which is then fed to the model along with the images for the prediction.

Model Architecture - Multi-Input CNN

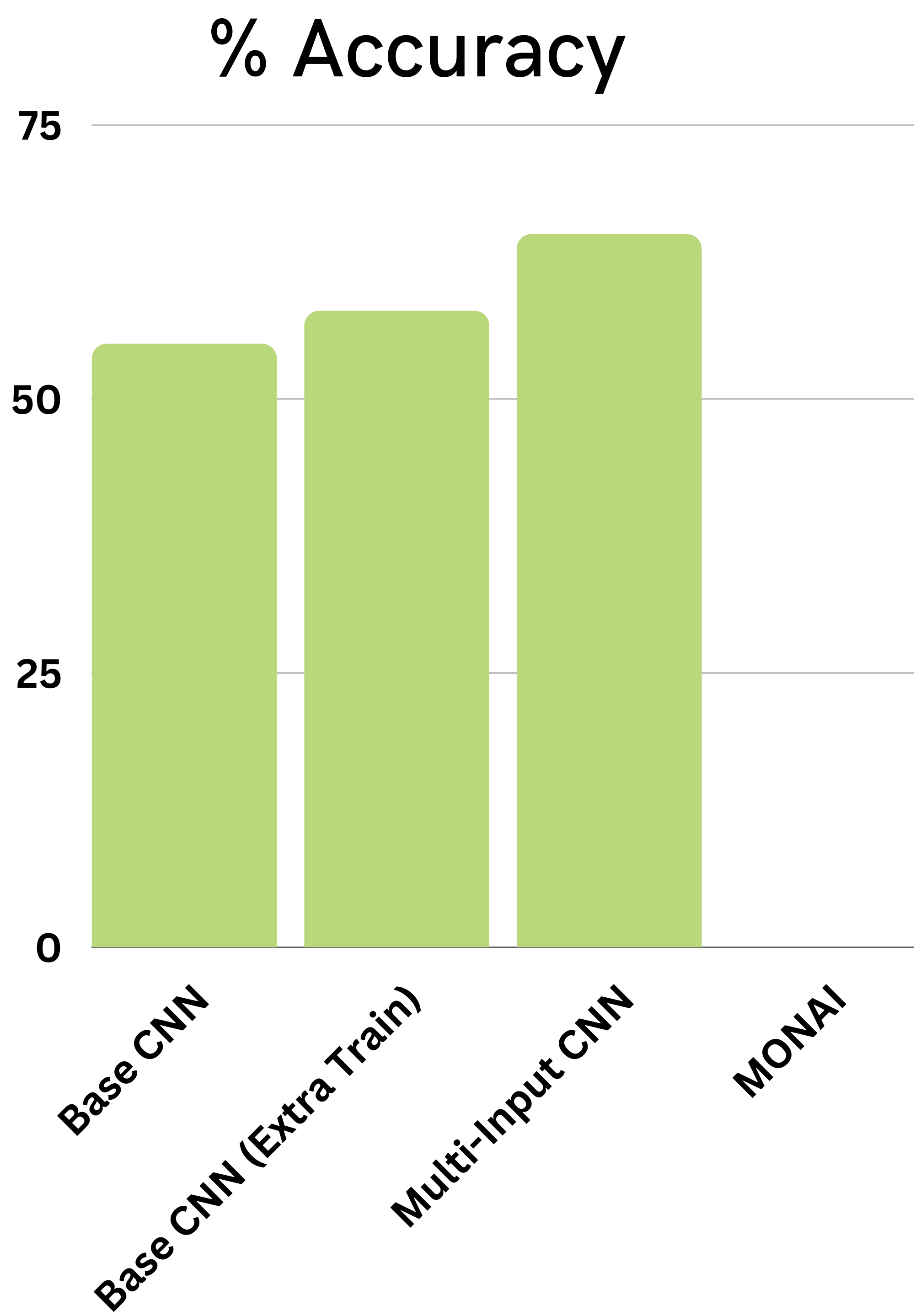
- Defined a CNN for a image processing, a neural network for the structured data.
- The CNN part consists of 3 Conv2D with maxpooling2D which reduce the spatial dimensions of the images .
- The two networks are then merged using the Concatenate layer and passed through a final Dense layer with sigmoid activation function.
- Multi-Input CNNs can leverage both image and structured data to improve the model's performance.

Layer (type)	Output Shape	Param #	Connected to
conv2d_14_input (InputLayer)	[(None, 256, 256, 3)]	0	[]
conv2d_14 (Conv2D)	(None, 254, 254, 32)	896	['conv2d_14_input[0][0]']
max_pooling2d_14 (MaxPooling2D)	(None, 127, 127, 32)	0	['conv2d_14[0][0]']
conv2d_15 (Conv2D)	(None, 125, 125, 64)	18496	['max_pooling2d_14[0][0]']
max_pooling2d_15 (MaxPooling2D)	(None, 62, 62, 64)	0	['conv2d_15[0][0]']
conv2d_16 (Conv2D)	(None, 60, 60, 128)	73856	['max_pooling2d_15[0][0]']
dense_20_input (InputLayer)	[(None, 3)]	0	[]
max_pooling2d_16 (MaxPooling2D)	(None, 30, 30, 128)	0	['conv2d_16[0][0]']
dense_20 (Dense)	(None, 64)	256	['dense_20_input[0][0]']
flatten_6 (Flatten)	(None, 115200)	0	['max_pooling2d_16[0][0]']
dense_21 (Dense)	(None, 32)	2080	['dense_20[0][0]']
concatenate_2 (Concatenate)	(None, 115232)	0	['flatten_6[0][0]', 'dense_21[0][0]']
dense_22 (Dense)	(None, 1)	115233	['concatenate_2[0][0]']

Model Architecture - Alternative

- MONAI stands for Medical Open Network for AI. It is an open-source framework used in deep learning specially for medical image analysis. It supports various types of medical imaging like CT,MRI, Ultrasound, etc.
- It provides pre-built components and workflows for data preprocessing, augmentation, training, and inference of deep learning models.
- MONAI integrates seamlessly with libraries such as PyTorch. However to effectively use MONAI, we need lightning framework of PyTorch which helps in making the model easy to run, robust, and scalable during production.

Results And Conclusions



Multi-Input CNN gave the best performance

Increasing the number of epochs overfitted the model

Base CNN gave good training accuracy but dind't work well on test

Resizing the Images made the computation easier

Information loss was minimal due to data augmentation

Converting the problem to a binary classification problem helped the model

THANK YOU