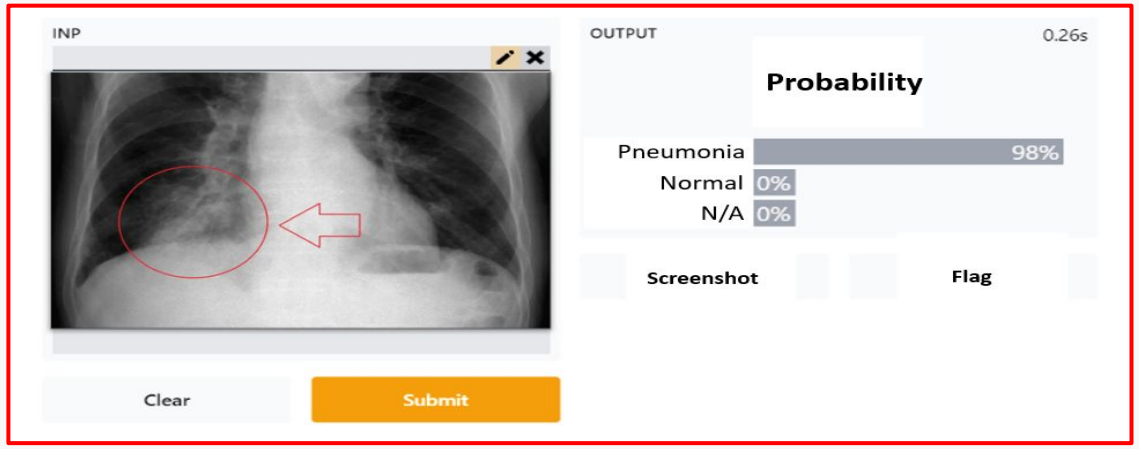
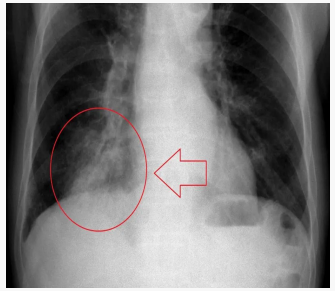
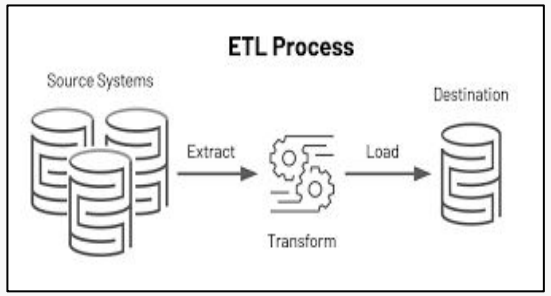
Project Team Members:
Kudzanai, Emmanuela, Noshaad, Maxwell and Chisimnulia

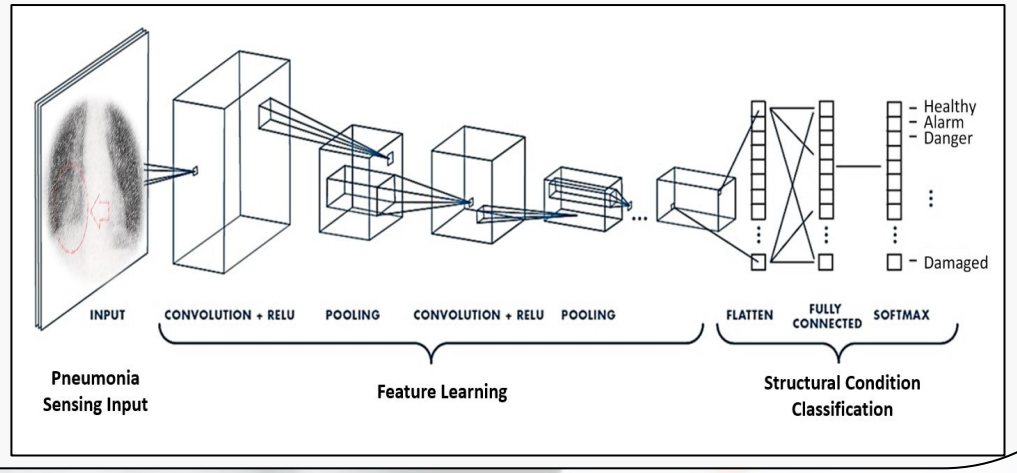
YOU HAVE PNEUMONIA!

Table of Contents







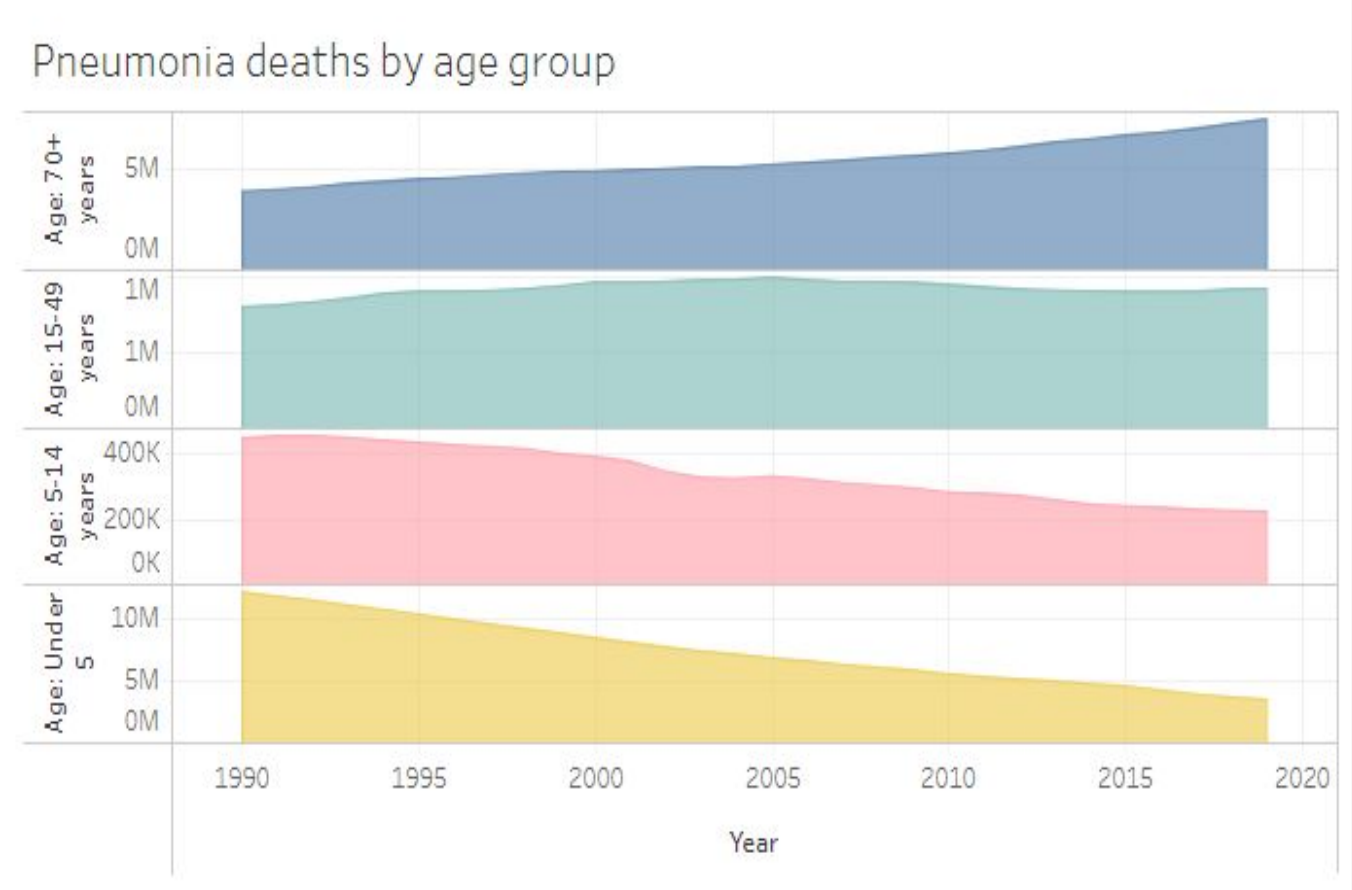


Emmanuela

What is Pneumonia?

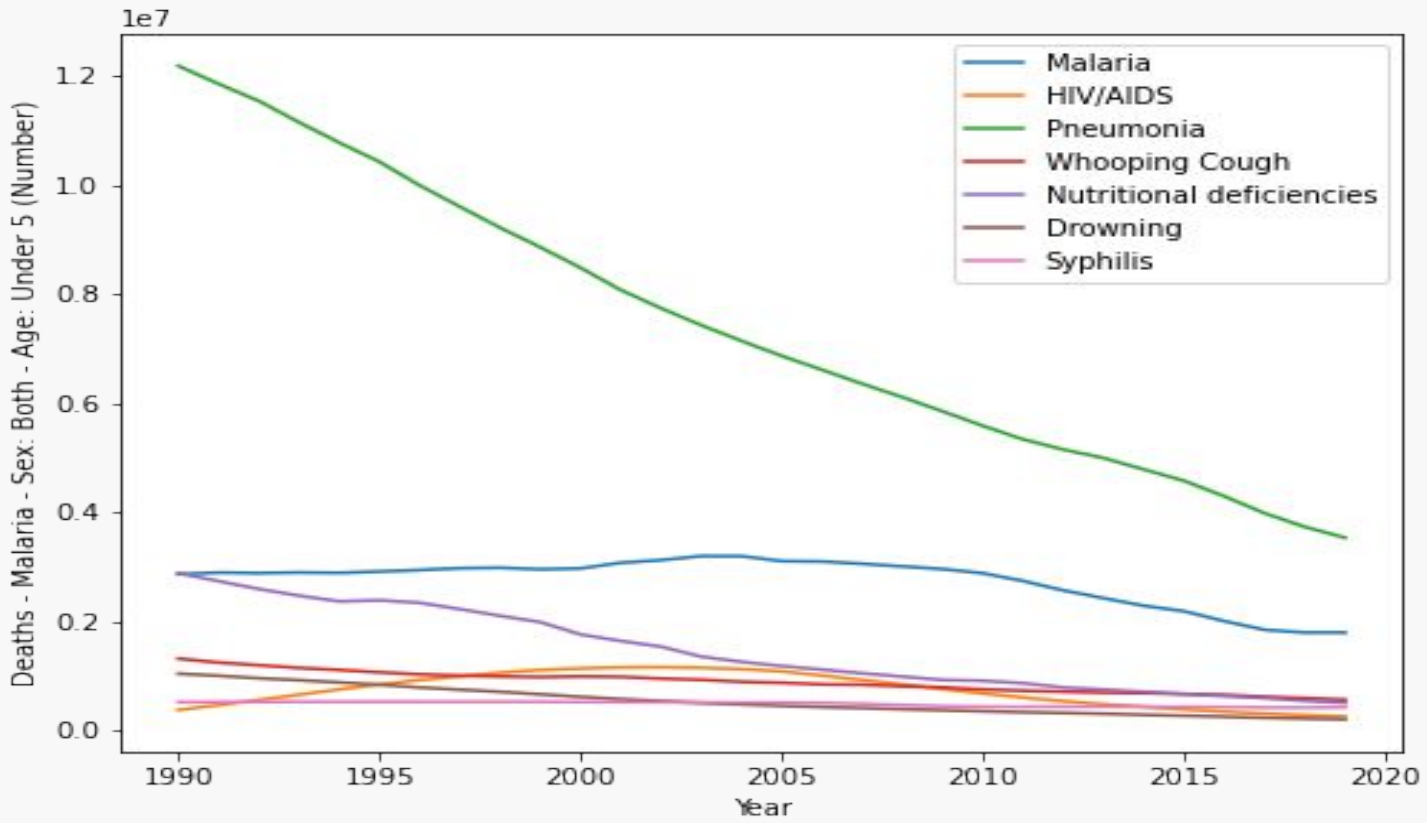
Pneumonia is a disease, usually caused by an infection, that affects the lungs and it is usually diagnosed by physicians using radiological imaging to determine the infectious agent that causes the disease.

The age group that is most affected by this disease is children under 5 and adults over 70. Since this diagnosis is required a lot of resources in a lot of cases it is not done.



The graph shown is generated with Tableau

Did you know…



The graph shown is generated with Matplotlib

15% of child deaths were caused by Pneumonia in 2017 making it the leading cause of death in children under 5yrs old.

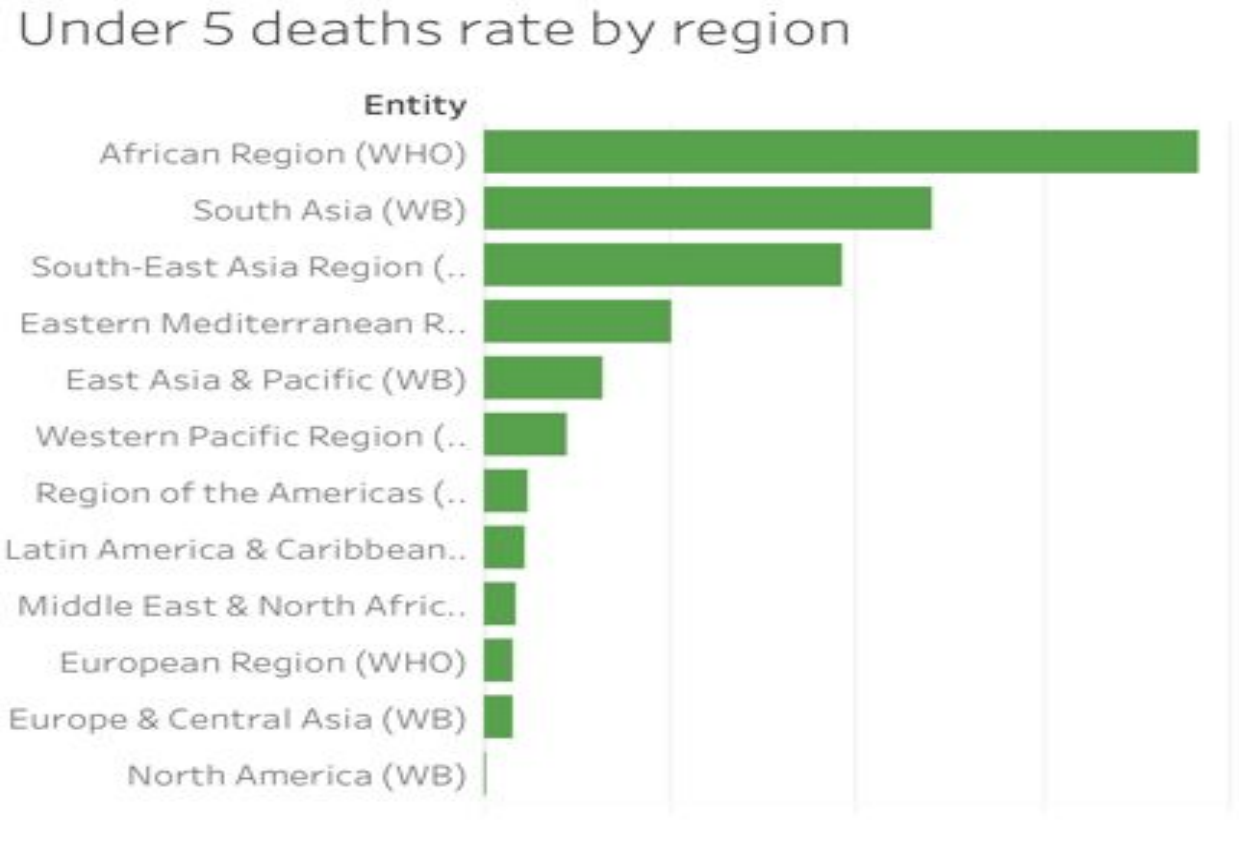
From the 90s till now the number of deaths has decreased substantially.

Although we have seen improvements in the major risk factors, Pneumonia still remains a disease to look out for, especially in children.

Did you know…

Again you can see
that the death rate from pneumonia is highest in Africa and Asia. A clear difference between richer and poorer countries .

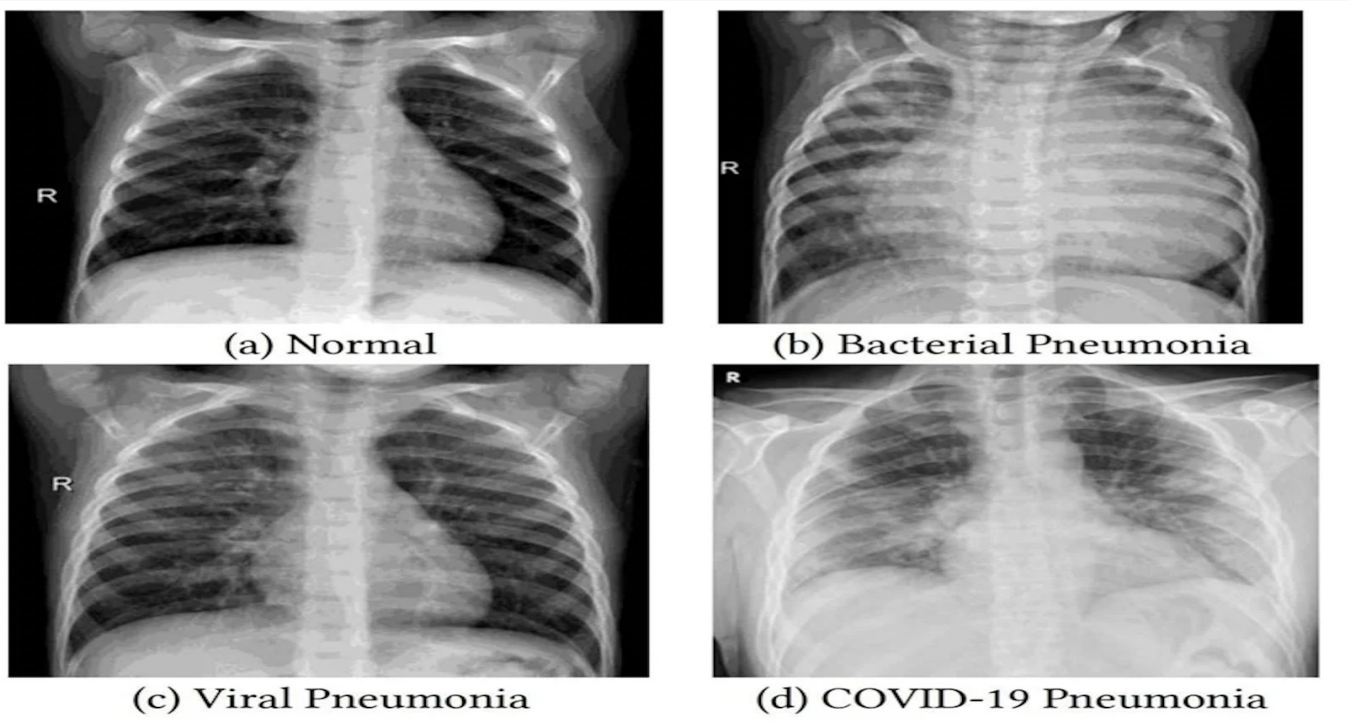
European populations suffer a rate of around 10 deaths per 100,000 while poorer countries see rate of more than 100 deaths per 100,000.



The graph shown is generated with Tableau

Purpose & Goal

- Doctors diagnose Pneumonia by analysing X-ray images of the chest
- However sometimes it is not so clear to the human eye



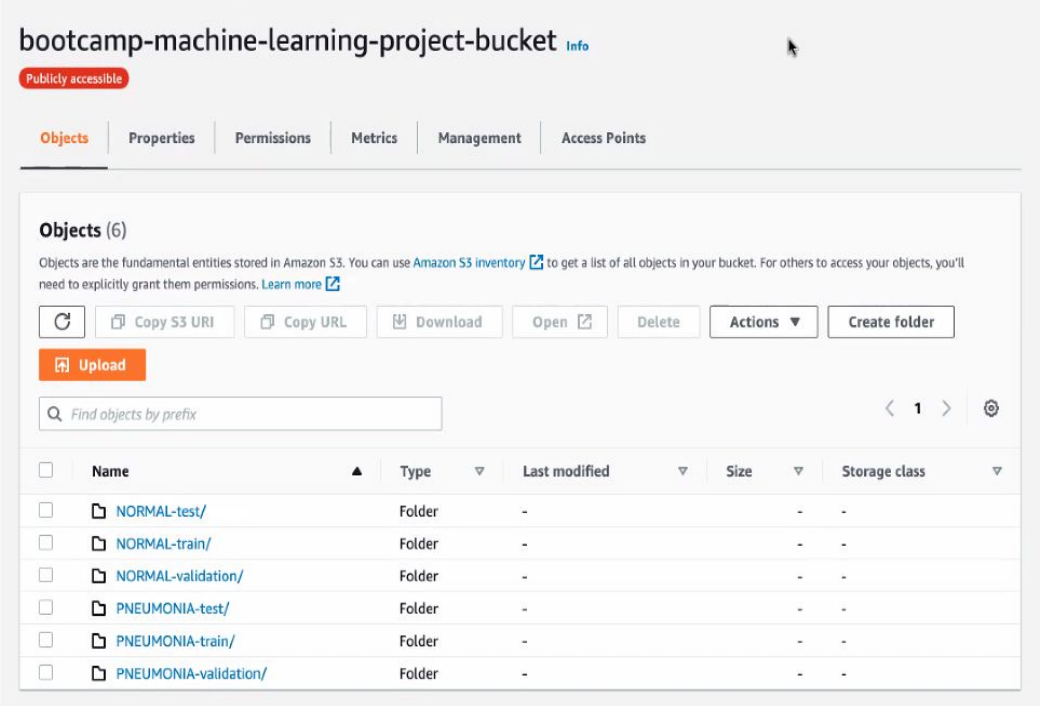
Purpose & Goal

Armed with this information we set out to develop a convoluted neural network (CNN) model that can analyse chest x-ray image and determine whether a person has pneumonia or not.

At the end of this presentation, we hope that our model will be used in the healthcare industry to improve detection of pneumonia and ultimately reduce mortality rate especially in under 5 children.

Data Extraction Transformation & Loading

- Data sourced from kaggle
- Loaded onto AWS into a bucket ready for extraction



Connect to AWS Bucket

```
import boto3
s3_resource = boto3.resource('s3')
```

```
[176] for bucket in s3_resource.buckets.all():
          print(bucket.name)

      bootcamp-machine-learning-project-bucket
```

```
[177] s3_bucket = s3_resource.Bucket(name = 'bootcamp-machine-learning-project-bucket')
```

```
[178] s3_client = boto3.client('s3')
      bucket = "bootcamp-machine-learning-project-bucket"
```

```
#list objects in the bucket
bucket_objects = s3_resource.Bucket(
    name = 'bootcamp-machine-learning-project-bucket')

#for object in bucket_objects.objects.all():
   # if 'train' in object.key:
        #print('train:', object.key)
   # else:
       # print('test',object.key)
```

- Google Colab was used to connect to the bucket

*See **pneumonia_model.ipynb** for details on the ETL*

Data Extraction Transformation & Loading

- Some essential models used for image processing were CV2 and PIL
- Used various modules to create get_data function that processes AWS objects into a (224, 224, 1) Numpy array that is fed into the CNN

```
def get_data(list_length):
  # here we limit the number of images to use in the training data set by changing list_length[:1341]
  #data_list = list_length[:1341]
  # read aws object
  data_list = [s3_client.get_object(Bucket=bucket, Key=i)['Body'].read() for i in list_length ]

  #convert image bytes to jpeg
  data_list = [Image.open(io.BytesIO(i)) for i in data_list]

  # applying grayscale method
  data_list = [ImageOps.grayscale(image) for image in data_list]

  # applying grayscale method
  data_list = [transform(i) for i in data_list]

  # convert image to numpy array
  processed_data = [asarray(image) for image in data_list]

  return processed_data
```

*See **pneumonia_model.ipynb** for details on the ETL*

Data Extraction Transformation & Loading

Challenges faced and resolved

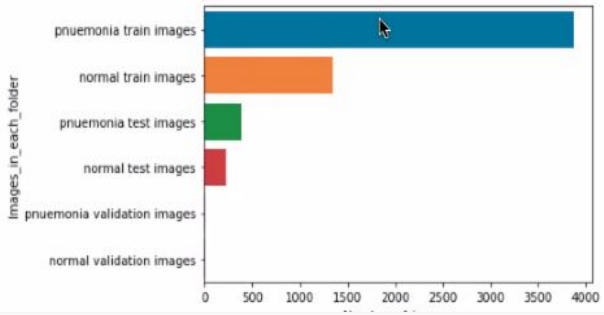
- Image adjustments:
  - Images were changed from bytes to jpeg
  - Changed to greyscale
  - Saved into arrays and finally reduced in size

```
#here we can set/change the size of the images without changing the aspect ratio
# using PIL module
# by changing the basewidth variable
def reduce_image_size(images):
  basewidth = 200
  wpercent = (basewidth / float(images.size[0]))
  hsize = int((float(images.size[1]) * float(wpercent)))
  img = images.resize((basewidth, hsize), Image.ANTIALIAS)
  return img
```

- Data imbalance resolved using class weights

	Images_in_each_folder	Number_of_images
0	pnuemonia train images	3875
1	normal train images	1341
2	pnuemonia test images	390
3	normal test images	234
4	pnuemonia validation images	8
5	normal validation images	8

```
data_set_figure = sns.barplot(x="Number_of_images", y="Images_in_each_folder", data=data_set_structure)
```



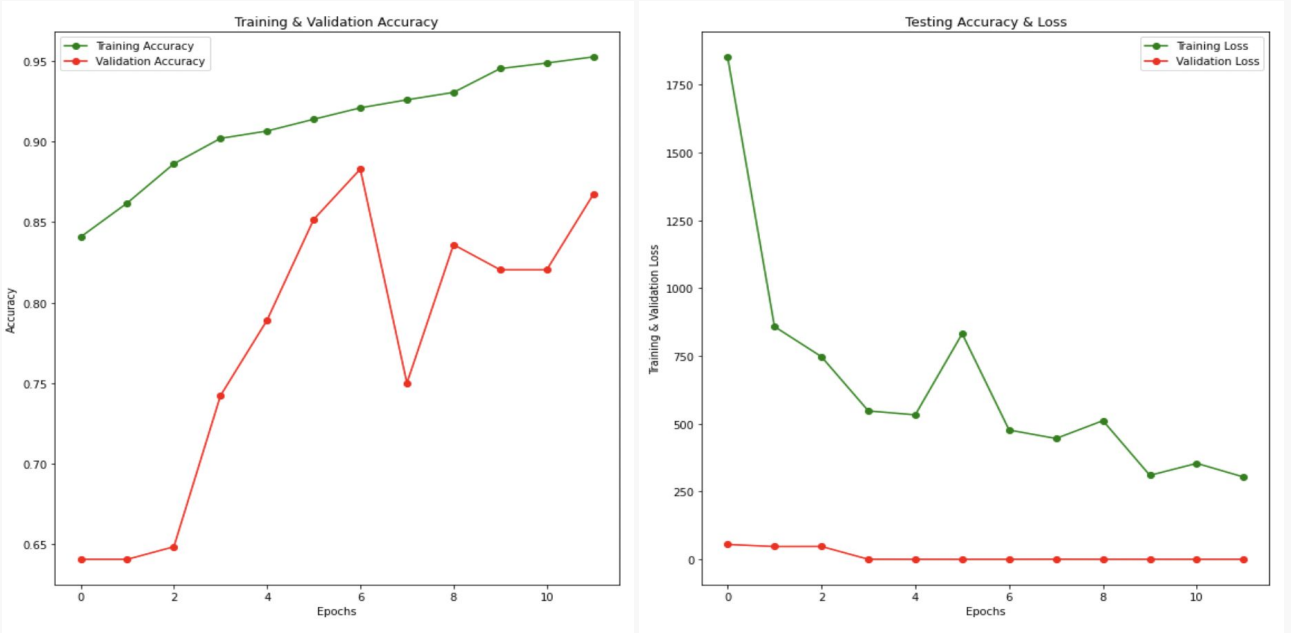

```
cw = pd.Series(y_train[:,0]).value_counts().to_dict()

[56] cw=dict(zip([0,1], cw.values()))

[57] history = model.fit(datagen.flow(x_train,y_train, batch_size = 32) ,epochs = 12 , validation_data = datagen.flow(x_val, y_val),
                         callbacks = [learning_rate_reduction], class_weight =cw)
```

*See **pneumonia_model.ipynb** for details on the ETL*

Machine Learning

- In the train data there is 3875 positive data points and 1341 negative datapoints
- The model accuracy is 88%
- Validation data is appears to be overfitting

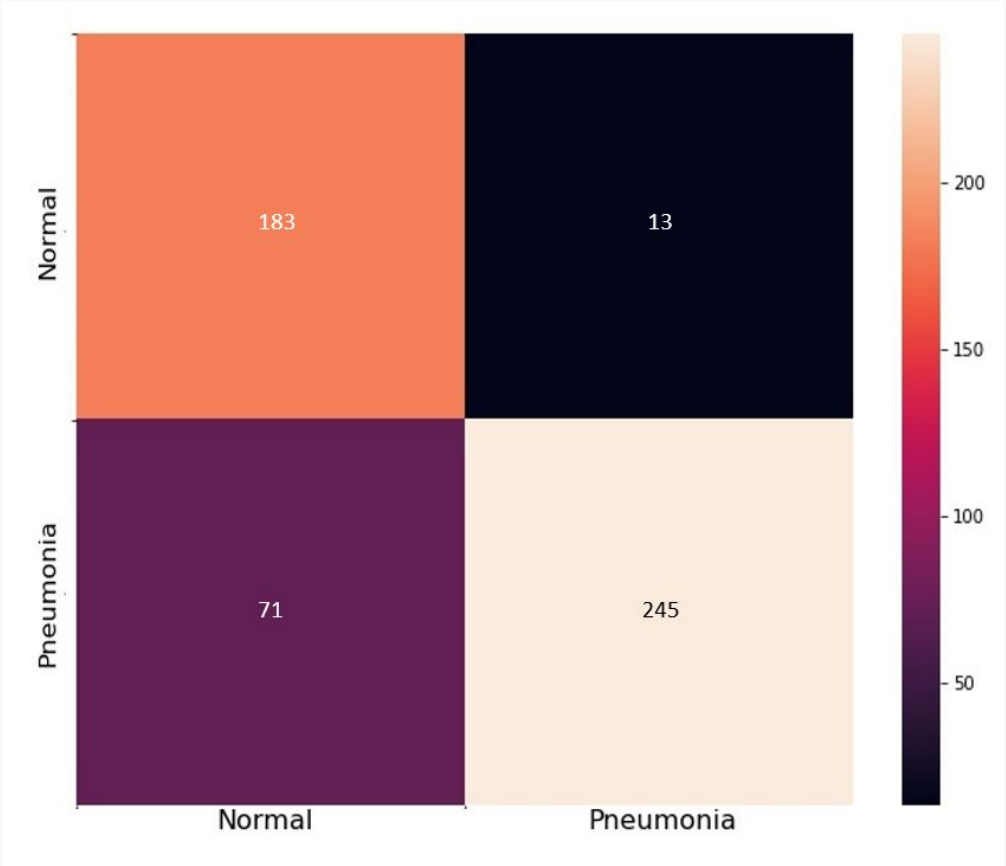
- Used data augmentation to increase the data
  - 30 degree image rotation
  - Split images horizontally etc



Model Performance

- Recall is more important than accuracy for our product
  - At 88% accuracy (threshold value) the model wrongly predicts 22% of the positive instances

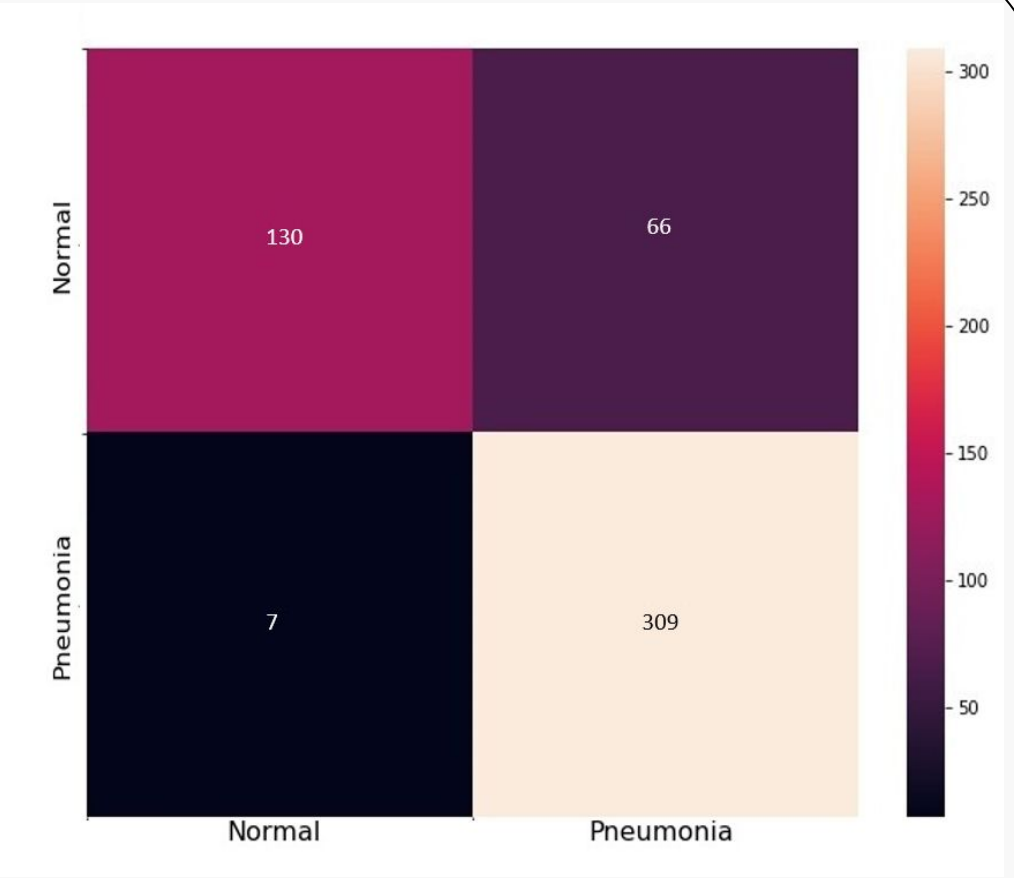
	precision	recall	f1-score
0	0.72	0.93	0.81
1	0.95	0.78	0.85
accuracy			0.84
macro avg	0.84	0.85	0.83
weighted avg	0.86	0.84	0.84



Model Performance

- At 30% accuracy (threshold value) the model wrongly predicts 2% of the positive instances and 33% of the negative instances
- We need to physically check the images of the false positives at 83% accuracy

	precision	recall	f1-score
0	0.95	0.66	0.78
1	0.82	0.98	0.89
accuracy			0.86
macro avg	0.89	0.82	0.84
weighted avg	0.87	0.86	0.85



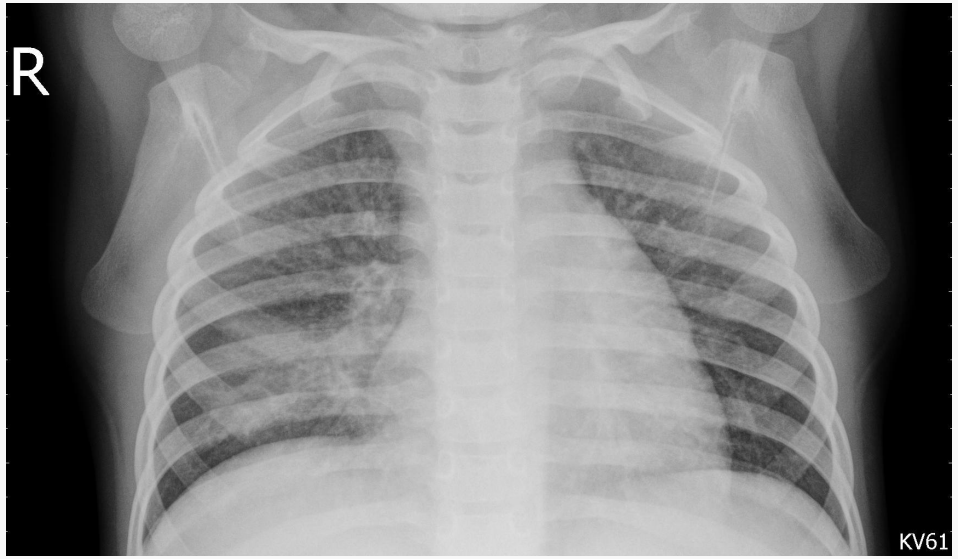
The APP

- The plan for the future is to build a Flask application to help doctors diagnose pneumonia and gather data.
- Doctor uploads image for patient Z using an application
- Application uploads image into our AWS bucket.
- Doctor gets diagnosis from the get_data_function.

```
def get_data_single_image(url):
  threshold = 0.9
  # here we limit the number of images to use in the training data set by
  #changing list_length[:1341]
  url
  # read aws object
  response = requests.get(url)

  #convert image bytes to jpeg
  img = Image.open(io.BytesIO(response.content))

  # applying grayscale method
  img = ImageOps.grayscale(img)

  # applying grayscale method
  img = transform(img)

  # convert image to numpy array
  img_array = asarray(img)

  y_pred = model.predict(img_array)[:,0]

  if y_pred >= threshold:
      print (y_pred)
      print(f"The likelyhood of the patient having pnuemonia is very high")
      print("please perform further tests")

  elif y_pred < threshold:
      print(y_pred)
      print("our model predicts that the patient has no pnuemonia")

  return (y_pred, img_array)
```

The APP

patient_z positive and (patient_j) negative. (Threshold for positive diagnosis is 0.9)



(patient_z)

```
patient_z = get_data_single_image(url)
```

```
[0.9988407]
 The likelyhood of the patient having pnuemonia is very high
please perform further test
```

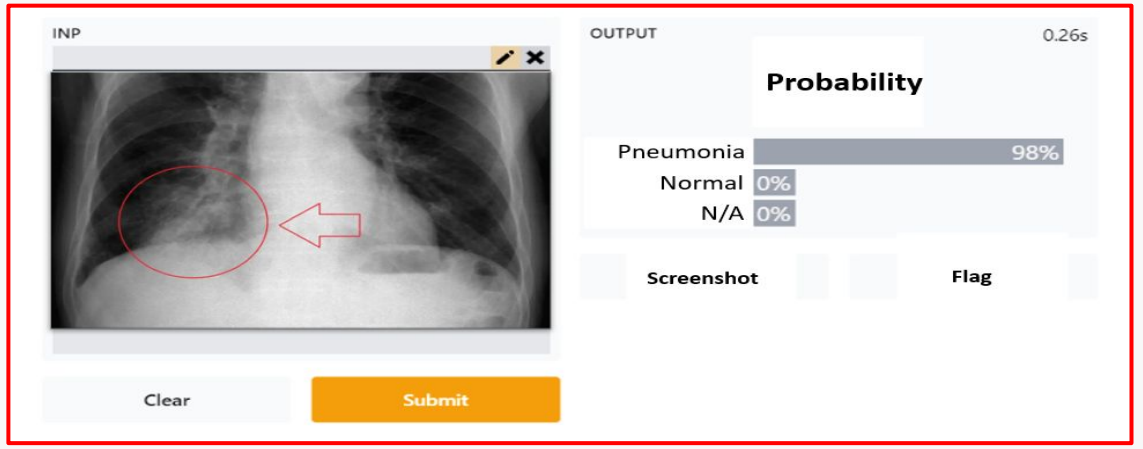
(patient_j)

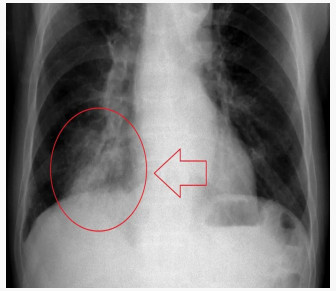
```
patient_j = get_data_single_image(urlJ)
```

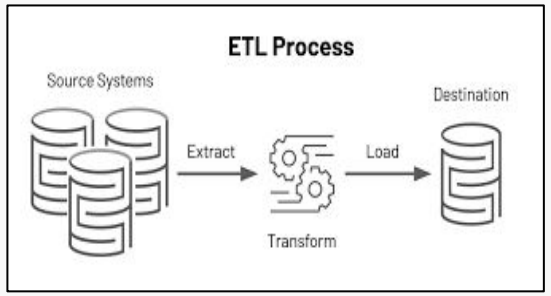
```
[0.57156914]
our model predicts that the patient has no pnuemonia
```

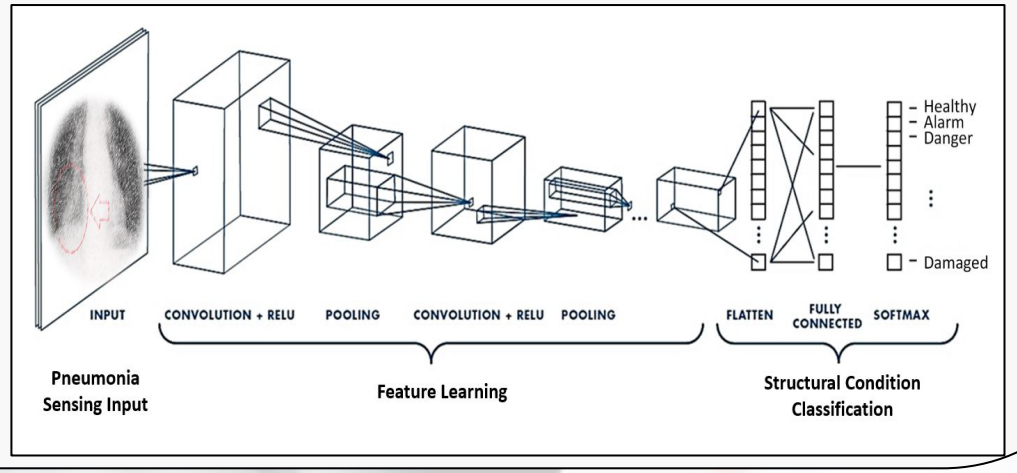
Summary


ANY
QUESTIONS
?


INP
OUTPUT
0.26s
Probability
Pneumonia
98%
Normal
0%
N/A
0%
Screenshot
Flag
Clear
Submit




ETL Process
Source Systems
Extract
Transform
Load
Destination


Healthy
Alarm
Danger
Damaged
INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
Pneumonia Sensing Input
Feature Learning
Structural Condition Classification