Anaya Malik, Maya Ashok, Antara Rajgopal, Katie Engel, Serena Kher
Instructor: Ethan Mehta

pneumonia & TB DETECTION

Meet the team...

Maya Ashok - CA

Katie Engel - NY

Serena Kher - CA

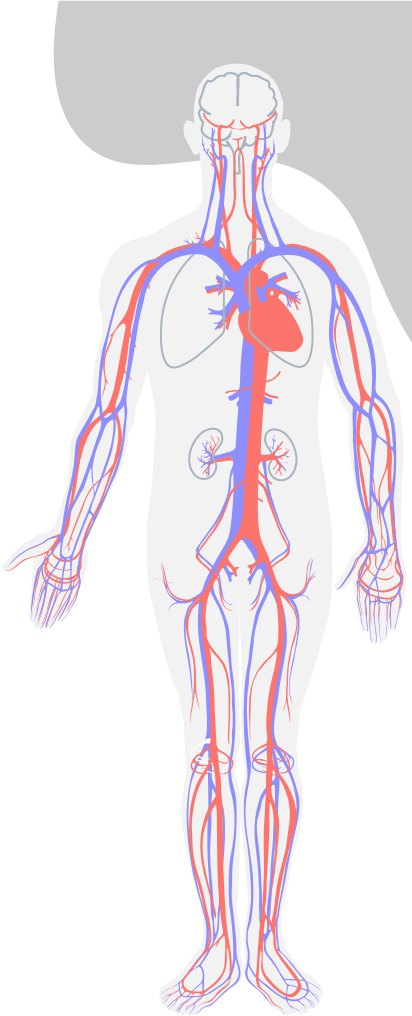
Antara Rajgopal - NJ

Anaya Malik - ATX

Introduction

Pneumonia, Tuberculosis, and AI.

Affecting more than 3 million people, pneumonia is a disease that fills lungs with fluid, making it difficult to breathe and reducing oxygen intake. This can be caused by viruses, bacteria and fungi. To extend the abilities of this pneumonia project, we decided to also analyze if a patient has Tuberculosis (TB). TB is a bacterial infection that impacts the lungs and can even impact the the brain, kidneys and spine. If left untreated, the patient can die so it's crucial to diagnose TB as soon as possible. This can be done through the help of our AI.

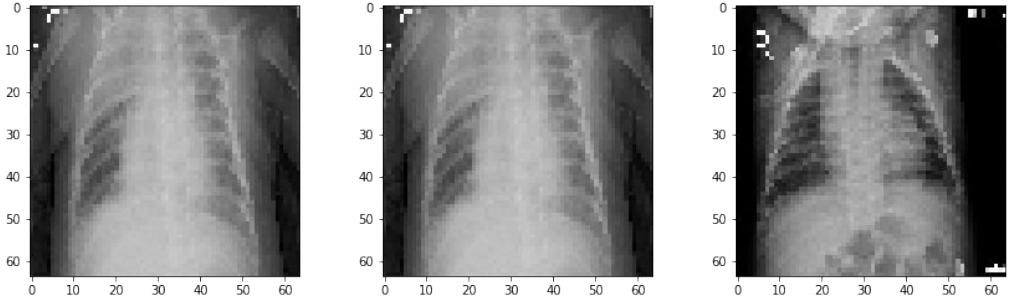


IMPACT

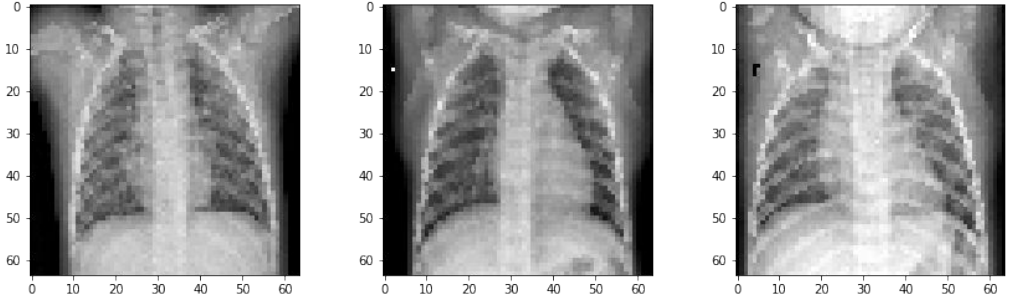
With this AI, we are able to successfully diagnose patients with Pneumonia and Tuberculosis through X-ray scans. By launching our code into an application as well, we are able to increase the availability of easy “health check-in” technology in order to ensure better health for all patients. Patients who may not have easy access to doctors or other forms of healthcare can benefit from easy to access healthcare technology like the one we have created through this project.

Data (Preliminary Data Analysis)

Pneumonia



Healthy



From these X-Rays, a doctor might be able to tell the difference by the cloudiness and density of visible bones. The A.I. will need to take this into account when analyzing for results.

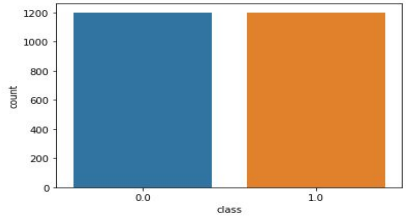
Data (Preliminary Data Analysis)

```
# grab our seaborn visualization toolbox!
import seaborn as sns

### YOUR CODE HERE
metadata.groupby(['class']).count()
sns.countplot(x = 'class', data = metadata)

### END CODE
```

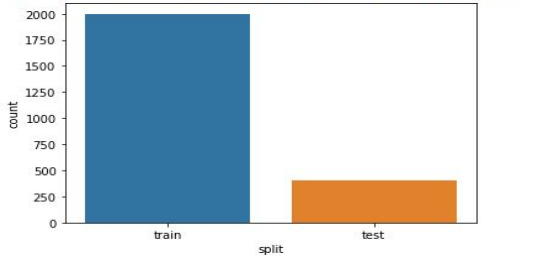
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f54cf2c3d90>
```



This image shows how with the given data the number of samples with pneumonia is the same as the number of healthy samples. It's important for our preliminary data to be equal in 0.0 and 1.0 to ensure equality when determining future diagnosis data.

```
### YOUR CODE HERE
metadata.groupby(['split']).count()
sns.countplot(x = 'split', data = metadata)
### END CODE
```

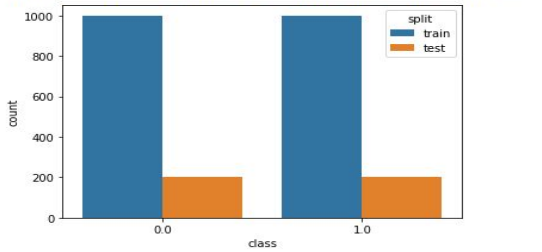
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f54cf3fe250>
```



This image demonstrates the rate of training data to testing data – where training data is significantly higher than testing. The A.I. is taking in information to train its systems. As it learns from each image, it can apply its self learning algorithms to new images.

```
### YOUR CODE HERE
metadata.groupby(['class', 'split']).count()
sns.countplot(x = 'class', hue = 'split', data = metadata)
### END CODE
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f54cecbbc50>
```



When grouped by class and whether the sample was pneumonia/healthy, the data shows that there's more training data but the distribution of healthy vs pneumonia is around the same

METHODS WE USED (MACHINE LEARNING)

01 Tested multiple machine learning models including: k nearest neighbors, logistic regression, and decision trees

02 Utilized Neural Networks and Convolutional Neural Networks

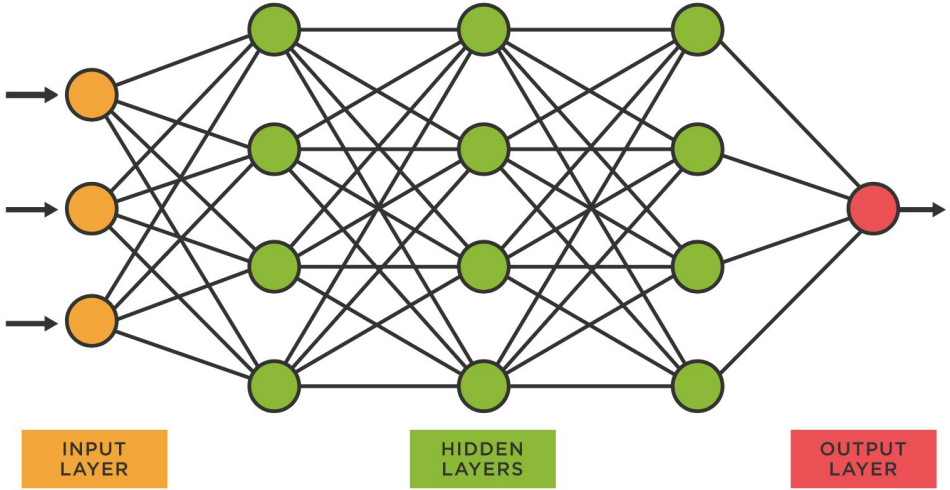
03 Trained and tested the data

04 Utilized Transfer Learning (VGG16)

05 Used image augmentation in our data sets to better our predictions

Neural Networks

- Contains input layer, hidden layer, and output layer
- Hidden and output layers have a specific number of neurons and an activation function
- Sequential model with dense layers



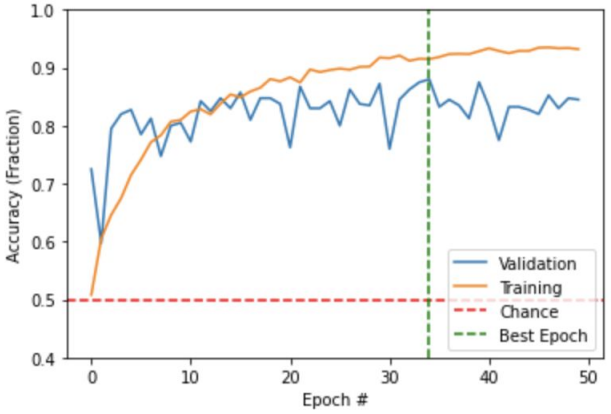
Utilizing CNNs

- Inputs = x-ray images of shape (64,64,3)
- Outputs = label of Pneumonia or Healthy
- Fit both a vanilla neural network (Dense Classifier) and convolutional neural network (CNN Classifier) with our training data
- CNN Classifier did better because of higher test accuracy score
- CNNs are considered to be the best to use for image classification

Dense vs. CNN Accuracy Score

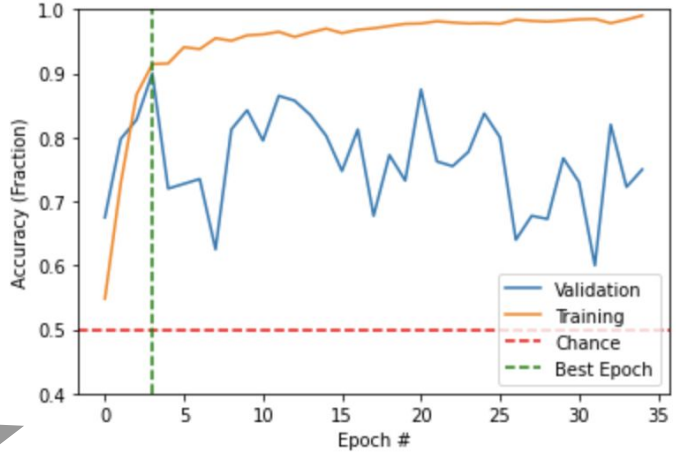
`0.75 < 0.8450000286102295`

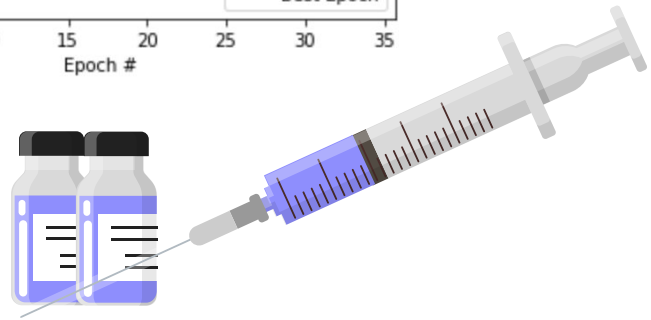
CNN History Graph



Utilizing Transfer Learning

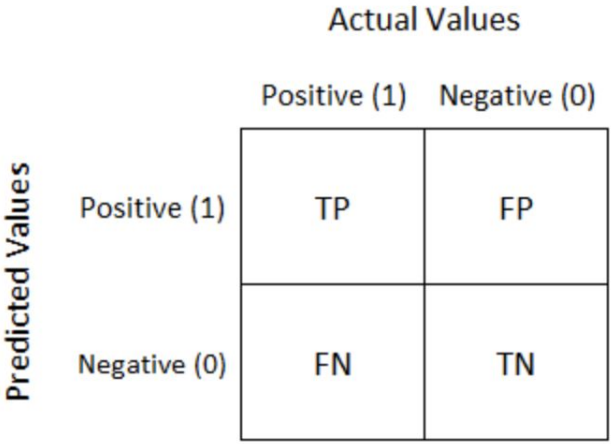
- Problem: Our training data is too small
- Solution: Using experts from ImageNet Classification problem
- VGG16 has studied 14 million images 74 times
- Used expert model VGG16 to train our x-rays
- Graph for fitting using transfer learning with VGG16

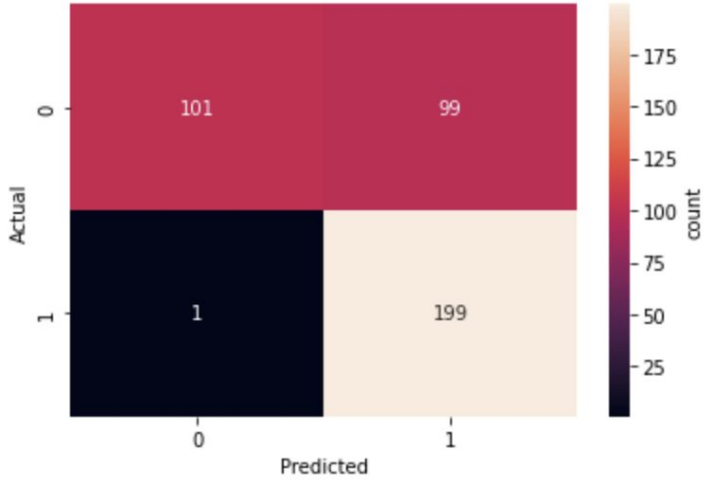




Confusion Matrix & Accuracy Score

- Used confusion matrix to evaluate our transfer learning model's performance along with accuracy score
- **`Accuracy Score: 75.0`**





Data Augmentation

- Problem: Model not performing well because field data (x-ray images are crooked) is different from training and testing data
- Solution: Create augmented datasets to combine with the original
- Augment using shearing and flipping

```
train_data_sheared_20 = shear(train_data, shear=20)
train_data_flip = flip_left_right(train_data_sheared_20, prob = 1)
```

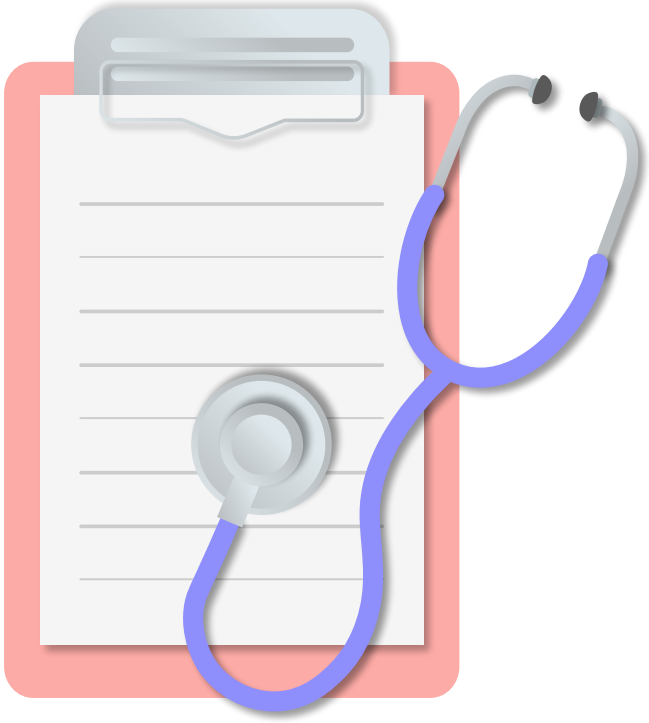
- Combine all data

```
all_data, all_labels = combine_data([train_data_flip, train_data], [train_labels, train_labels])
```

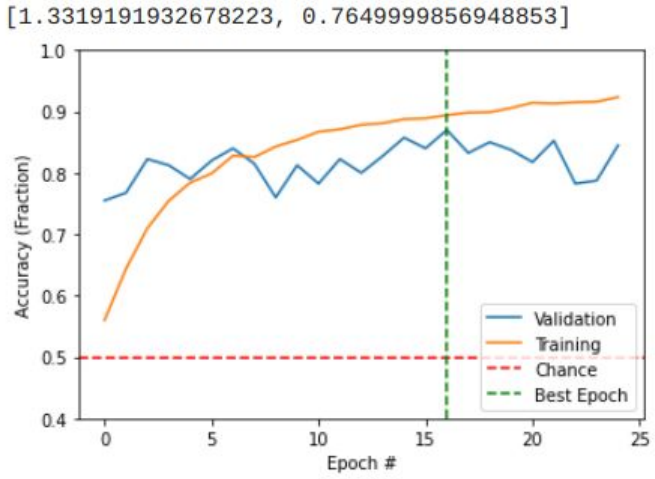
- Train CNN on all data, load model, and score it on field data

```
cnn = CNNClassifier(num_hidden_layers = 2)
cnn_history = cnn.fit(all_data, all_labels, epochs = 25,
                      validation_data = (test_data, test_labels),
                      shuffle = True, callbacks = [monitor])
cnn_score = cnn.evaluate(field_data, field_labels, verbose=0)
print(cnn_score)
plot_acc(cnn_history)
```

Results



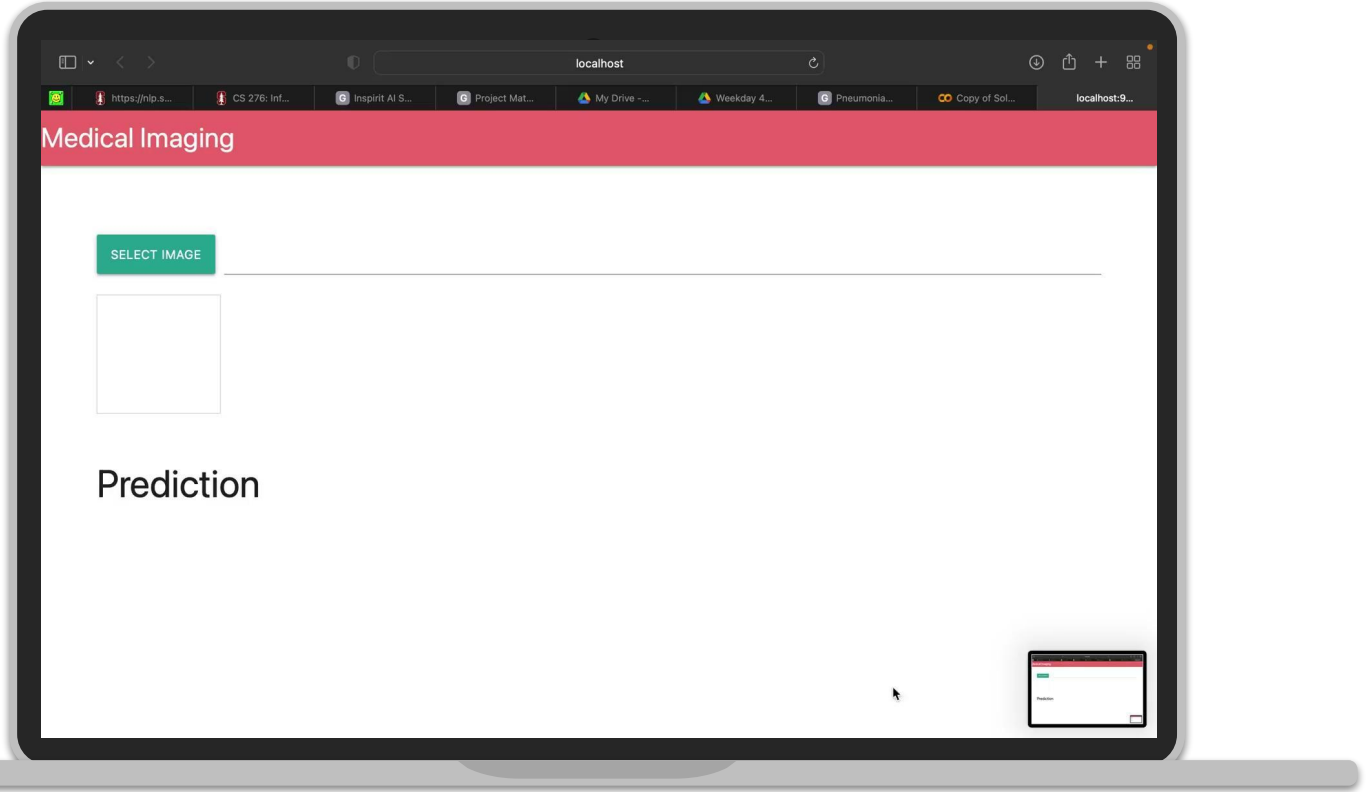
After taking all field data images of pneumonia vs. normal, there is a reported 76% chance of field accuracy. This took all the X-Rays and let the A.I. determine chance of pneumonia.



Extension Project #1: ML App Deployment

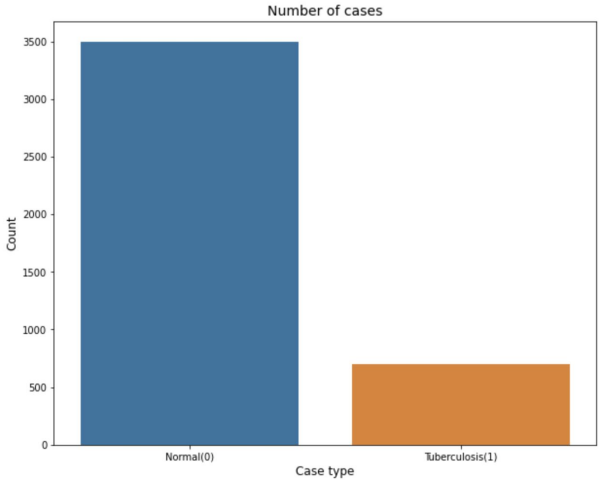
Along with detecting pneumonia, we decided to deploy it via an app. We used a combination of HTML and Javascript to produce the app and deployed it onto a website locally.

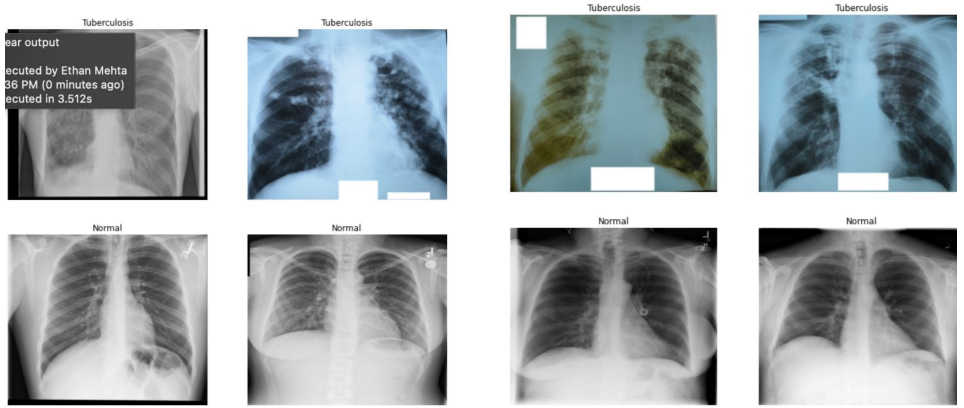
APP DEPLOYMENT VIDEO

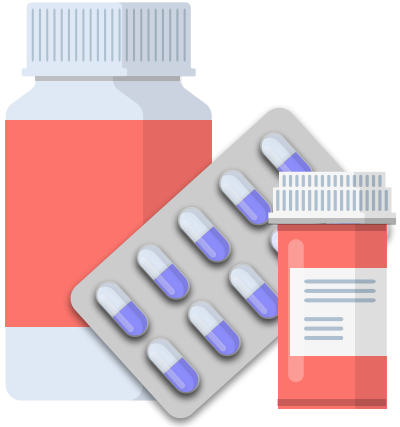


Extension Project #2: Tuberculosis + Pneumonia Detection

Along with detection of pneumonia, we extended the project onto being able to detect tuberculosis. To do this, we downloaded data of X-Rays from patients who did have tuberculosis and processed it through our A.I. This data was downloaded from Kaggle and would be a part of a future development!







Thank you for listening! Any questions?

```python
from flask import Flask, render_template, request
from flask import Flask, send_file, request, Response,
jsonify
from flask import send_file

from flask_cors import CORS, cross_origin
import requests

import numpy as np
import tensorflow as tf
import cv2

import codecs
import base64
import jsonpickle

import zipfile

app = Flask('host_model')


webpage = "templates/webpage.html"
html = codecs.open(webpage, 'r').read()

@app.route("/")
def home():
   return html

#@title Instructor Solution { display-mode: "form" }
@app.route('/predict_diagnosis', methods=['POST'])
@cross_origin(headers=['Access-Control-Allow-Origin'])
def predict_diagnosis():
   try:
     print("Received Image")
     base64_img = base64.decodebytes(request.data)
     nparr = np.frombuffer(base64_img, np.uint8)
     img = cv2.imdecode(nparr, cv2.IMREAD_COLOR)
     img = cv2.resize(img,(64,64)).astype('float32')
   #   cv2.imshow(img)

     print("Processed Image")

     img = np.reshape(img,(1,64,64,3))
   #   pred = cnn.predict(img)[0]
     print("Predicted Diagnosis")

     diagnoses = ["No Pneumonia","Pneumonia"]
   #   pred_diagnosis = diagnoses[np.argmax(pred)]
     pred_diagnosis = diagnoses[0]
     return jsonify({'diagnosis': pred_diagnosis})
   except Exception as e:
     print(e)
     return "Invalid Request", 400
```

```javascript
function onFileSelected(event) {
   try {
       var selectedFile =
event.target.files[0];
       var reader = new FileReader();

       imgtag.title = selectedFile.name;

       reader.onload = function(event) {
           imgtag.src = event.target.result;

       var xhr = new XMLHttpRequest();
       xhr.open("POST",
"/predict_diagnosis")

xhr.send(event.target.result.split(',')[1]);
       xhr.onreadystatechange = function() {
           if (xhr.readyState == 4)
           if (xhr.status == 200) {
           var response =
JSON.parse(xhr.responseText)
               predicted_diagnosis.innerHTML
= response["diagnosis"]
               console.log(response)
           }
       }
       }

       reader.readAsDataURL(selectedFile);
   }catch (error){
       alert("Error Reading Image")
   }

}
```

```html
   <!-- Import Google Icon Font -->
   <link
href="https://fonts.googleapis.com/icon?family=Material+Icons"
rel="stylesheet">
   <!-- Import CSS for Materialize -->
   <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/css/mat
erialize.min.css">

    <!--Optimize Viewport for Mobile Devices-->
    <meta name="viewport" content="width=device-width,
initial-scale=1.0" />

</head>
<body>

   <!-- Title Bar -->
   <nav>
       <div class="nav-wrapper">
         <a href="#" class="brand-logo">Medical Imaging</a>
       </div>
     </nav>

   <!-- Contain for all other HTML Elements -->
   <div style="padding:5%;">

       <!-- Image File Input -->
       <div class="file-field input-field">
           <div class="btn">
             <span>Select Image</span>
             <input type="file" accept="image/*"
onchange="onFileSelected(event)">
           </div>
           <div class="file-path-wrapper">
             <input class="file-path validate" type="text">
           </div>
         </div>

       <!-- Image to be Classified -->
       <img id="image" width="142" height="136"></img>

       <!-- Add New Lines -->

       <br/>
       <br/>

       <!-- Text Fields for the Prediction -->
       <h3>Prediction</h3>

       <!-- Diagnosis Predicted -->
       <h4 id="predicted_diagnosis"></h4>

   </div>

   <!-- Import JS Libraries -->
   <script
src="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/js/mater
ialize.min.js"></script>

   <script src="static/js/medical_imaging_script.js"></script>
   <script>

   var imgtag = document.getElementById("image")
   var predicted_diagnosis =
document.getElementById("predicted_diagnosis")

   </script>
```