Using AI: Pneumonia Detection

Shreya Narayanan, Divina Minocha, Meera Patel, Helena Kromann, Samika Jain & Kevin Xu

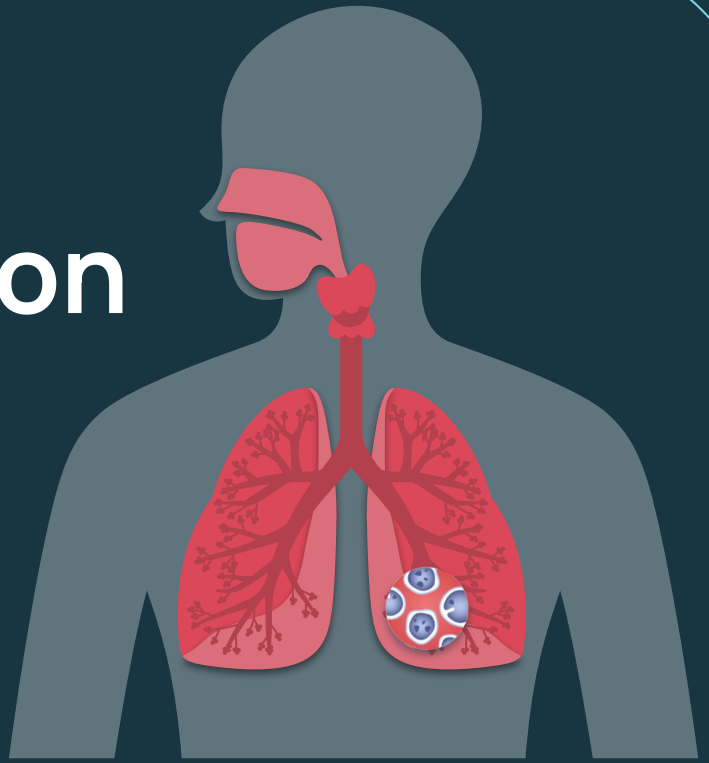
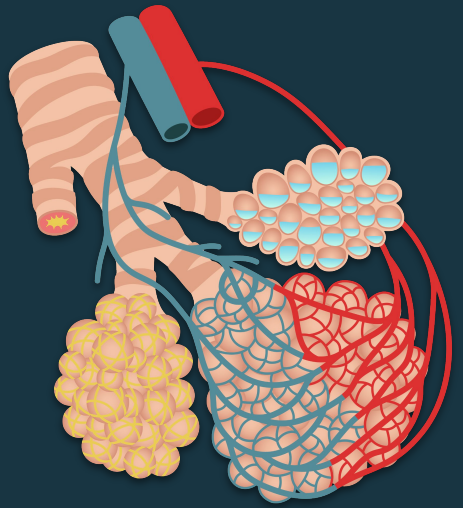


Table of contents

Table of contents

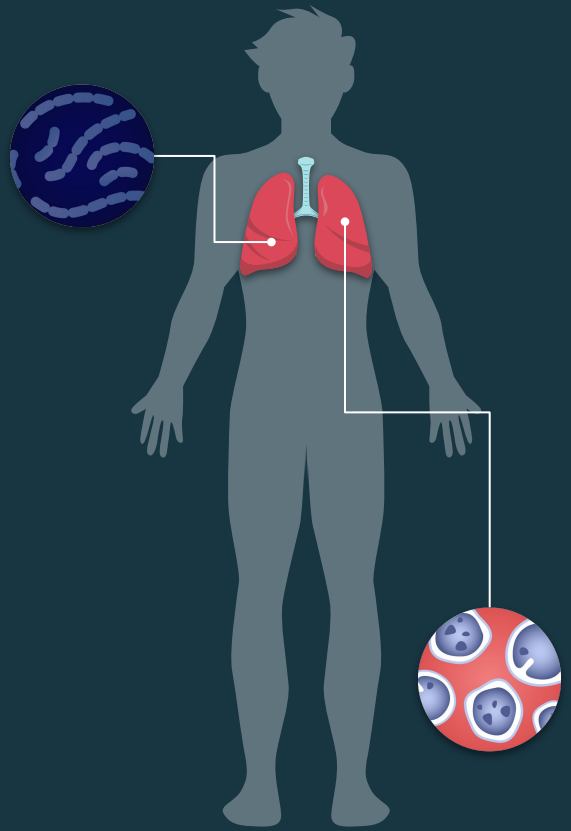
AI in medicine

What have you heard about the applications of AI in healthcare? Let us know in the chat!



What is pneumonia? How do we detect it?

- Inflammation of air sacs
- Fluid in the lungs
- Caused by viruses, bateria, or most recently strains due to COVID-19
- Blood tests
- X-Rays



2,500,000

People die of pneumonia every year
(and those are just the ones that are recorded!)

https://ourworldindata.org/grapher/death-rates-from-pneumonia-and-other-lower-respiratory-infections-vs-gdp-per-capita?xScale=linear&yScale=linear&tab=chart" loading="lazy" style="width: 100%; height: 600px; border: 0px none;" allow="web-share; clipboard-write"

The Coding Process

Building & Training our model

What is Metadata?

Metadata is the information about the images…

- Class: 0 (healthy) and 1 (pneumonia)
- Index: the location in our metadata
- Rows: How many images we have in total

	class	split	index
0	0.0	train	0
1	0.0	train	1
2	1.0	train	2
3	0.0	train	3
4	1.0	train	4
...	...	...	...
2395	1.0	test	2395
2396	0.0	test	2396
2397	0.0	test	2397
2398	1.0	test	2398
2399	0.0	test	2399

2400 rows × 3 columns

Our AI Model

CNN Model

- Combination of neurons
- Processes images like our visual system
- It uses the input images and learns about features using spatial relationships which are processed into patterns to classify the image.

MLP Model

- Flattened
- It processes data one step at a time to understand context and then adjusts to learn patterns and make predictions.

Our Datasets

1. Data Splits

- Training data: Labelled images
- Test data: Used to evaluate the model's performance after training.
- Field data: Real world data

```
X_train, y_train = get_train_data()
X_test, y_test   = get_test_data()
```

Initializing training and testing data

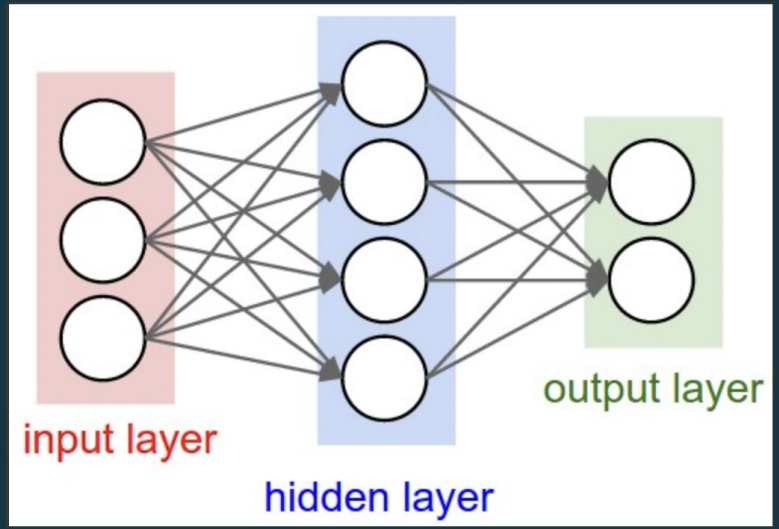
```
X_field , y_field = get_field_data()
```

Initializing field data

Our Datasets

2. Network Architecture

- Layers: Convolutional and Dense Layers
- Convolutional: ReLU activation, convolutional operations to images
- Dense : fully connected, flattened feature maps
- Transfer Learning: Faster training!



Code snippets!

```python
#@title Run this to test if your model is right!
model_1_answer = Sequential()
model_1_answer.add(InputLayer(input_shape=(3,)))
model_1_answer.add(Dense(4, activation = 'relu'))
model_1_answer.add(Dense(2, activation = 'softmax'))
model_1_answer.compile(loss='categorical_crossentropy',
optimizer = 'adam',
metrics = ['accuracy'])

model_1_config = model_1.get_config()

del model_1_config["name"]
for layer in model_1_config["layers"]:
  del layer["config"]["name"]

model_1_answer_config = model_1_answer.get_config()

del model_1_answer_config["name"]
for layer in model_1_answer_config["layers"]:
  del layer["config"]["name"]

if model_1_answer_config == model_1_config:
  print('Good job! Your model worked')
else:
  print('Please check your code again!')
```

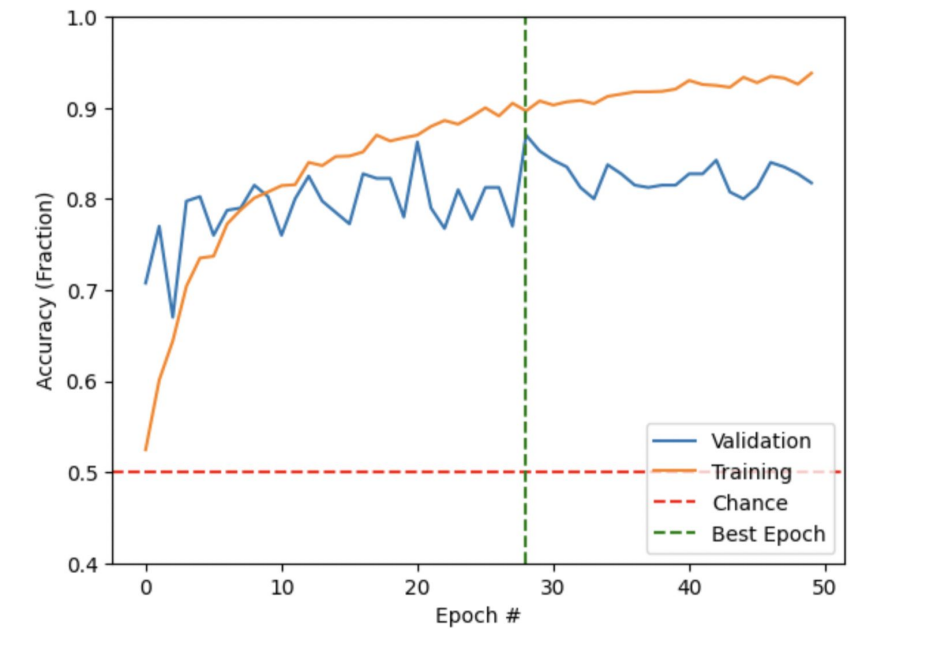
```
Good job! Your model worked
```

Testing

```python
[4]
### YOUR CODE HERE:
model_1 = Sequential()
model_1.add(InputLayer(input_shape=(3,)))
model_1.add(Dense(4, activation = 'relu'))
model_1.add(Dense(2, activation = 'softmax'))
model_1.compile(loss='categorical_crossentropy',
                optimizer = 'adam',
                metrics = ['accuracy'])
### END CODE
```

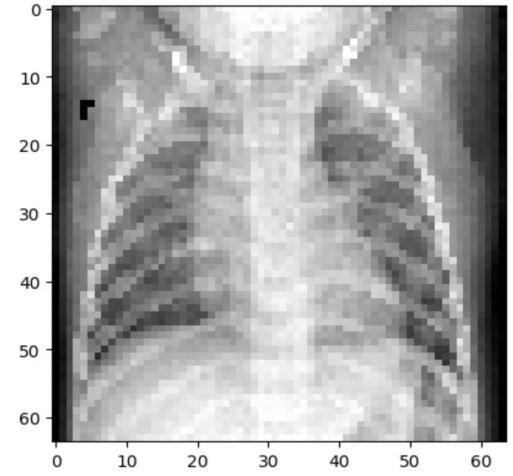
Initializing

Results



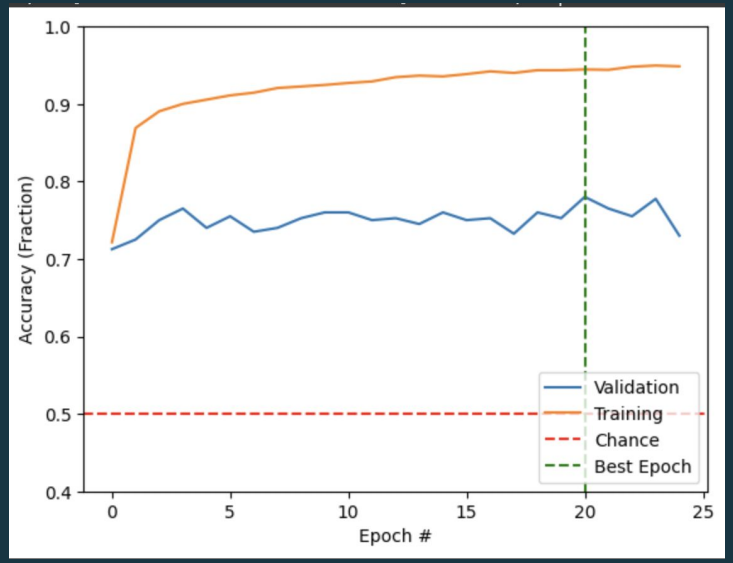

```
1  for i in range (3):
2    plot_one_image(X_train, y_train, i)
3    plot_one_image(X_test, y_test, 1)
4    plot_one_image(X_field, y_field, 1)
```

```
Label: 0.0
```




```
Label: 1.0
```

Code snippets(Transfer Learning)!



Result

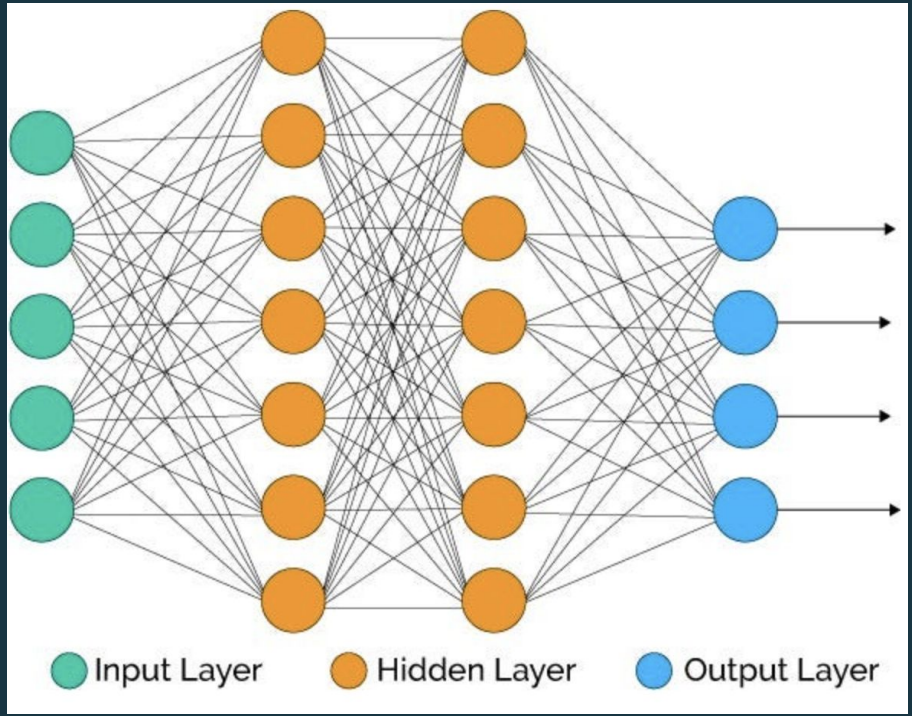
```
1  transfer = TransferClassifier(name = 'VGG16')
2  transfer.fit(X_train, y_train, epochs = 20, validation_data = (X_test, y_test),
   shuffle = True, callbacks = [monitor])
```

Initializing transfer learning

Our Datasets

Training

- Epochs: one complete pass of the training data set through the algorithm
- What does the number of epochs depend on?
- Accuracy: hidden layers, epochs and augmentation



Code snippets!

```
### YOUR CODE HERE

X_train, y_train = get_train_data()
X_test, y_test   = get_test_data()
X_field, y_field = get_field_data()

average_accuracy = 0.0
for i in range(5):
  cnn_temp = CNNClassifier(5)
  cnn_temp.fit(X_train, y_train, epochs = 5, validation_data = (X_test,
  y_test), shuffle = True, callbacks = [monitor])

  y_pred = (cnn_temp.predict(X_field) > 0.5)
  accuracy = accuracy_score(y_field, y_pred)
  print('Accuracy on this run: %0.2f' % accuracy)

  average_accuracy += accuracy / 5.0
print('Average accuracy: ', average_accuracy)
### END CODE
```

Code

```
Accuracy on this run: 0.50
Average accuracy:  0.5005000000000001
```

Accuracy can be very low!

Issues and Solutions

Throughout model-building, we encountered many problems. This is how we solved them:

#1: Field Data

Our friends have provided us some more data!
Hooray!

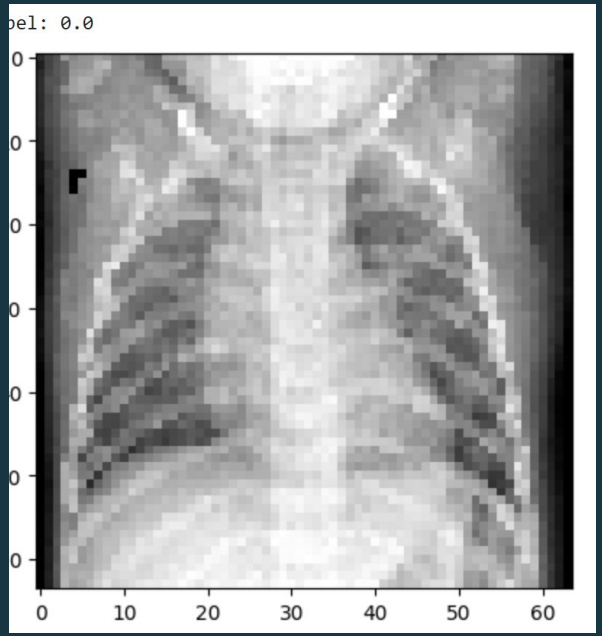
When we run .predict(), the resulting accuracy is only 74% :(

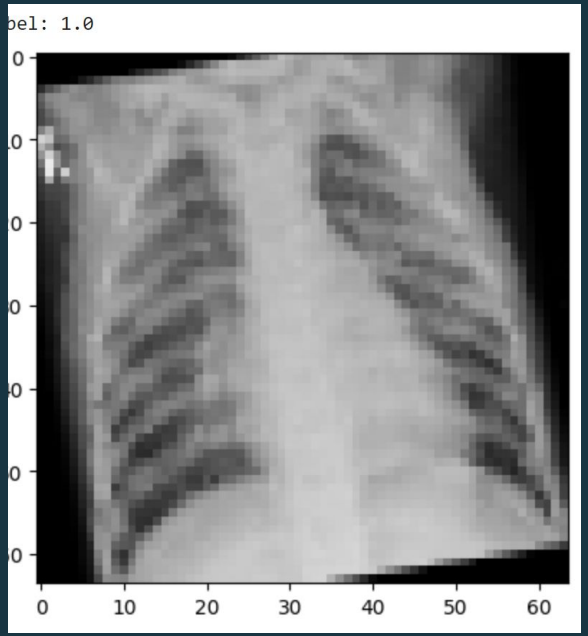
```
y_pred = cnn.predict(X_field) > 0.5
accuracy = accuracy_score(y_field, y_pred)
print(accuracy)
```

```
13/13 [==============================] - 0s 3ms/step
0.74
```

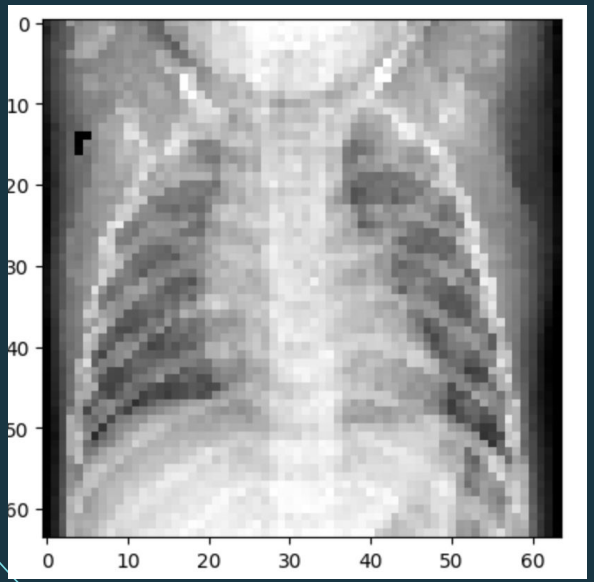
Why?

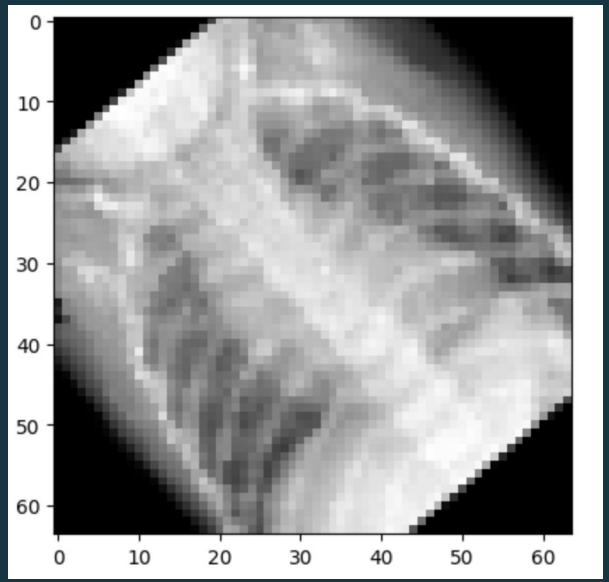
- Train
- Field





Fix - Augmentations





Train dataset

image = X_train

Augmented dataset

new_image = rotate(image, rotate = -40)

Creating More!

```
rotated_L10 = rotate(X_train, rotate=-10)
rotate_R90 = rotate(X_train, rotate=90)
rotate_L90 = rotate(X_train, rotate=-90)
shear_R20 = shear(X_train, shear=20)
shear_L20 = shear(X_train, shear=-20)
sh_ro_R90_R20 = shear(rotate_R90, shear=20)
sh_ro_R90_L20 = shear(rotate_R90, shear=-20)
sh_ro_L90_R20 = shear(rotate_L90, shear=20)
sh_ro_L90_L20 = shear(rotate_L90, shear=-20)
red_train  = remove_color(remove_color(X_train, channel = 1), channel = 2)
```

=

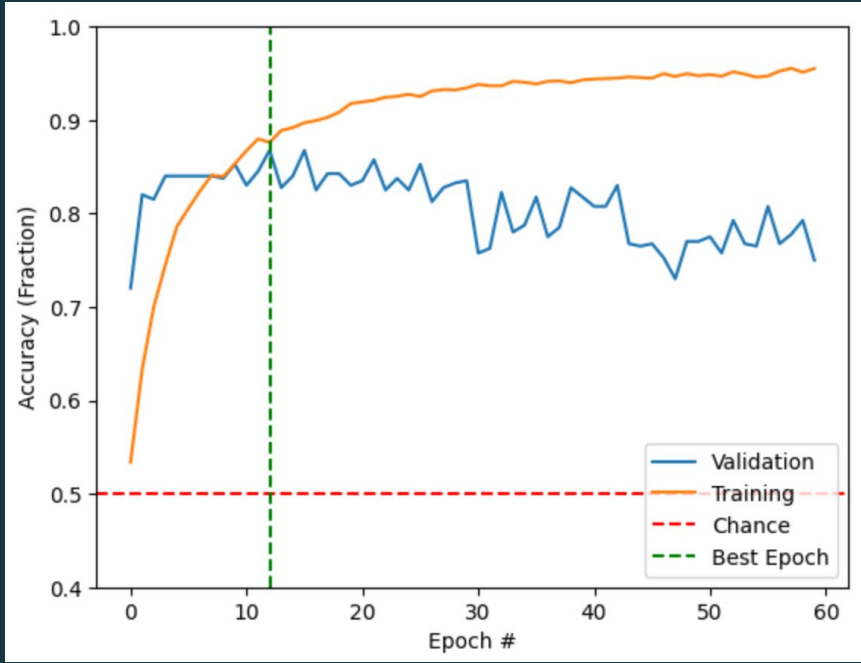
Higher Accuracy!
:)

#2: Overfitting

- Model is too specialized
- “Memorizes” the training data

What is it?

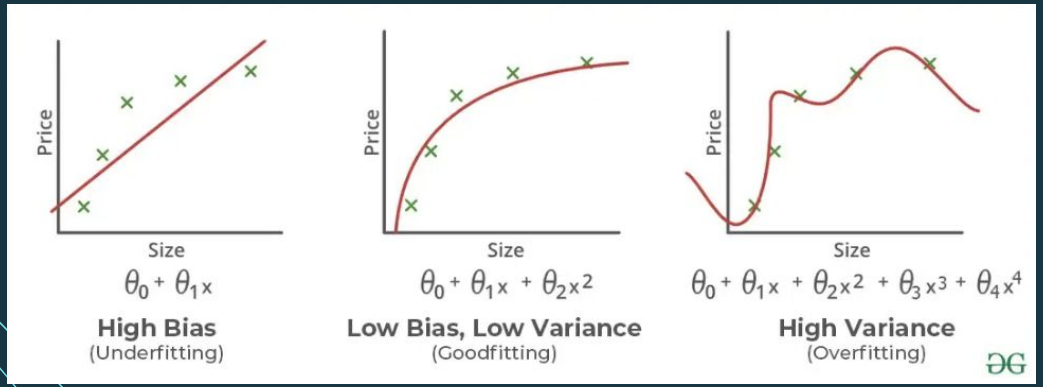
- Performing well on the training set but poorly on the validation set



Fix - Reduce Epochs

```
X_train, y_train = get_train_data()
X_test, y_test   = get_test_data()
cnn = CNNClassifier(num_hidden_layers=2)
history = cnn.fit(X_train, y_train,epochs=50, batch_size=32, validation_data=
(X_test, y_test),
callbacks=[monitor])
plot_acc(history)
```

```
63/63 [==============================] - 0s 8ms/step - loss: 1.2277 - accuracy: 0.9125 - val_loss: 1.3427 - val_accuracy: 0.8375
Epoch 36/50
63/63 [==============================] - 1s 8ms/step - loss: 1.2080 - accuracy: 0.9150 - val_loss: 1.3425 - val_accuracy: 0.8275
Epoch 37/50
63/63 [==============================] - 0s 8ms/step - loss: 1.1986 - accuracy: 0.9175 - val_loss: 1.3463 - val_accuracy: 0.8150
Epoch 38/50
63/63 [==============================] - 0s 7ms/step - loss: 1.1780 - accuracy: 0.9175 - val_loss: 1.3534 - val_accuracy: 0.8125
Epoch 39/50
63/63 [==============================] - 0s 7ms/step - loss: 1.1735 - accuracy: 0.9180 - val_loss: 1.3361 - val_accuracy: 0.8150
Epoch 40/50
63/63 [==============================] - 0s 7ms/step - loss: 1.1639 - accuracy: 0.9205 - val_loss: 1.3401 - val_accuracy: 0.8150
Epoch 41/50
```

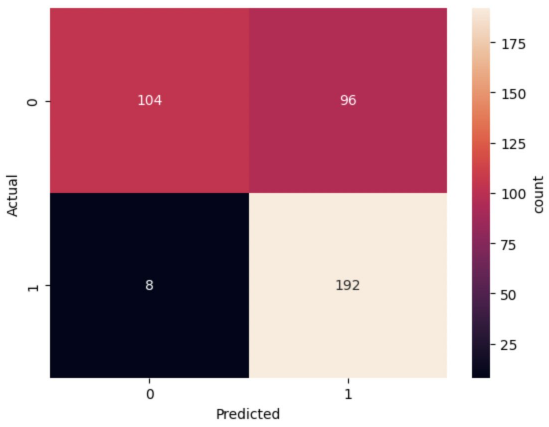


- Too many epochs = possible overfitting
- Not enough epochs = possible underfitting

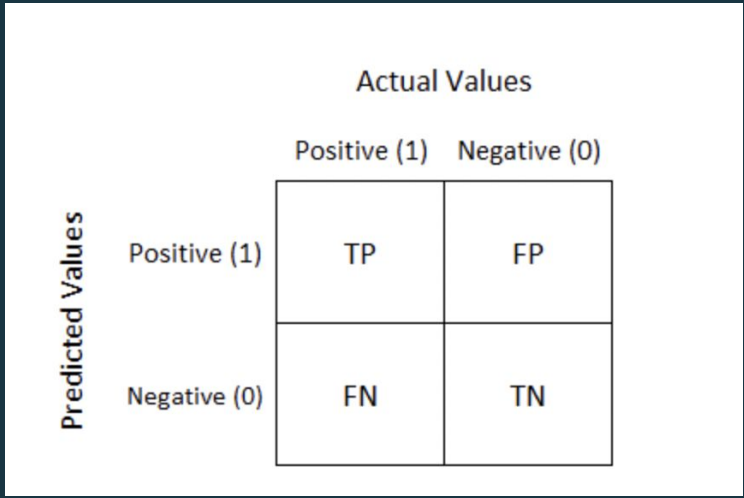
Confusion matrix

```
[ ]  1  # grab our plotting package
     2  import seaborn as sns
     3  import matplotlib.pyplot as plt
```

```
     1  sns.heatmap(confusion, annot = True, fmt = 'd', cbar_kws={'label':'count'});
     2  plt.ylabel('Actual');
     3  plt.xlabel('Predicted');
```



Visualisation of the confusion matrix



Model confusion matrix

Summary

1

Our goal

Pneumonia or healthy from chest X-rays

2

Classification

Images, Labelling and CNN

3

Types of Data

Training, test and field

4

Issues

Overfitting and variation

Real World Application & Integration

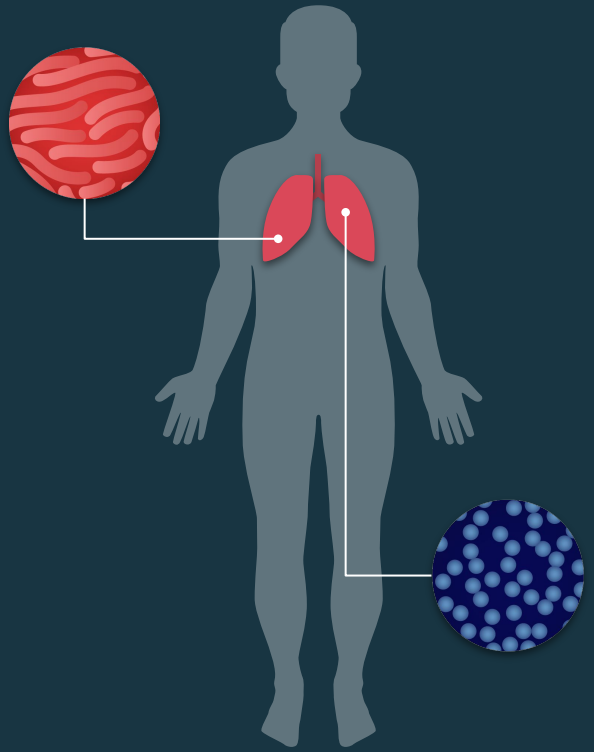
- This would be relatively easy to implement
- Start off by using it alongside doctors
- Develop a high enough accuracy rate to use alone

Benefits of using this model:

Accuracy Levels

Rate of Detection

Economic Factors



Thank you!

We are happy to answer questions-about the process, the coding, or anything else!