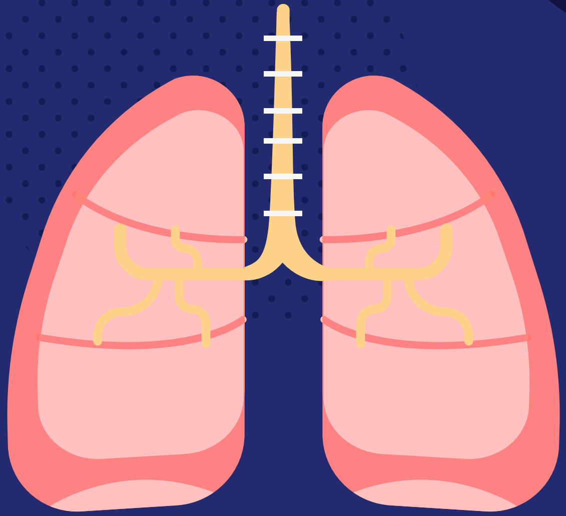


PNEUMONIA DETECTOR

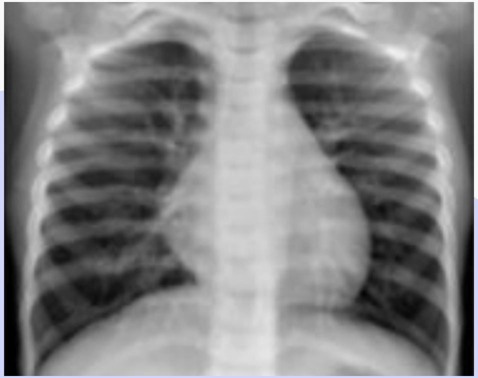
1

WHAT IS PNEUMONIA?

ABOUT THE DISEASE

Pneumonia is an infection that affects the air sacs in one or both lungs. The air sacs fill with fluids causing difficulty with breathing as well as coughing, fever, and chills.

The most common mode of evaluation and diagnosis for the infection is by an x-ray exam. X-ray images use varying colors to display density; patients with pneumonia will have white spots (called infiltrates) in their x-ray scans.



OUR MOTIVATION

1.5 million people were diagnosed with pneumonia back in 2018, and nearly 38.8% of those cases were misdiagnosed. Pneumonia is the 2nd most misdiagnosed condition in the US.

MISDIAGNOSIS

Misdiagnosis often occurs due to incorrect readings of x-ray images. The blacks, grays, and whites, make it extremely hard to spot the white clouds that aid in a physician's diagnosis from an x-ray image.

When undiagnosed/ untreated, the infection can lead to respiratory failure and even death.

MOTIVATION

Our project wanted to test AI accuracy in analyzing x-ray imaging to spot and diagnose pneumonia. A pneumonia detecting AI system, would help perform tasks in reading x-ray imaging. If successful, the project/ AI could decrease the number of misdiagnosis and help save hundreds of lives!

2

Understanding and Visualizing Our Data

Making the foundation of our machine

Data Columns

01 Class

Tells us whether the image shows healthy lungs or lungs with pneumonia (represented as either 0 or 1, binary classification)

02 Split

Tells us whether the data is from our training set or testing set

03 Index

Tells us where in the data set the image is located

	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

Training Vs Testing Data

Training Data

- Training Manual
- Helps Machine learn its task
- Much larger than testing data
- Useful in teaching data how to detect patterns
- “Homework” for the machine

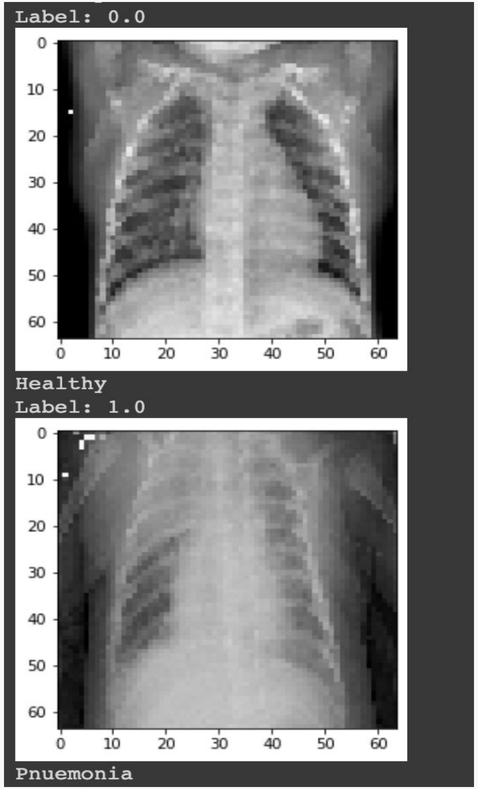
Testing Data

- Test for the machine
- Sees if the machine can recognize patterns outside of the training set
- Problems machine has not seen before
- Is the machine ready to be put to use?

Make Sure Your Data Is Not Overfit!!!!!

What Does Our Data Look Like?

Plotting Our Data

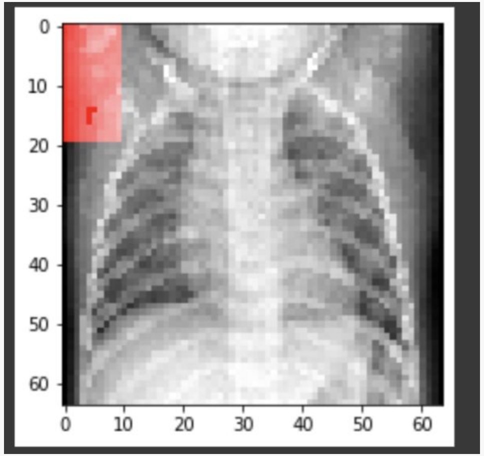


Manipulating Our Data

```
for i in range(20):
  for j in range(10):
    rect_image[i,j,0] = 1
plot_one_image(rect_image)
```

Highlighting certain areas in image

- Rect_image[i,j,0] = 1
  - I =x value of image (what should the width and x-position of rectangle be?)
  - J = y value of image (what should the length and y-position of the rectangle be?)
  - By setting pixel = 1, making the pixel fully red



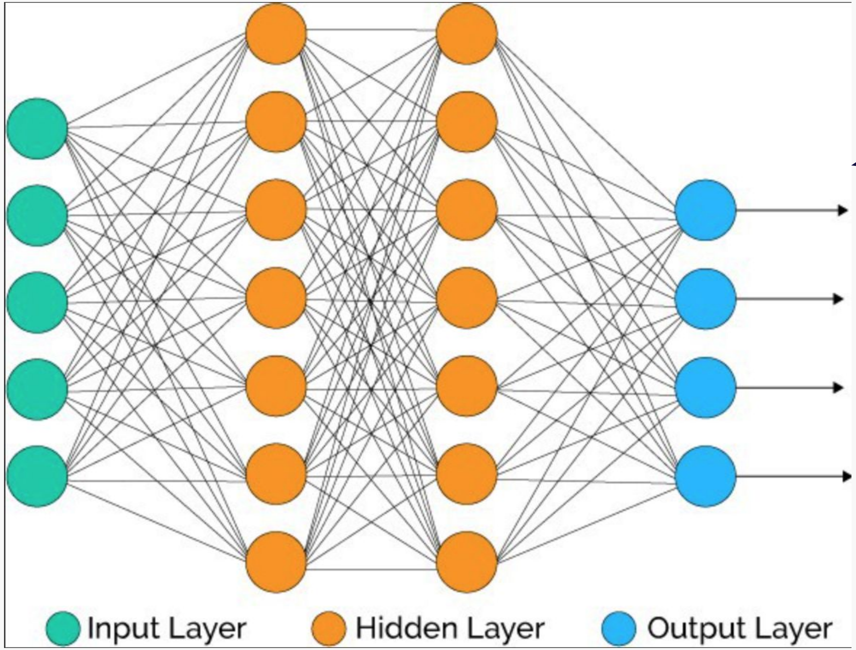
Important uses of function

- Not all of image is necessary for detection
- Zoning in on specific locations makes it easier to detect
- Being able to compare those locations to others
- Makes it easier to detect specific patterns in data

3

Creating and Applying Neural Networks

Neural Networks



Neural Networks look something like this

The neural network is built using the packages known as ‘tensorflow’ and ‘keras’

Sample created and tested neural network

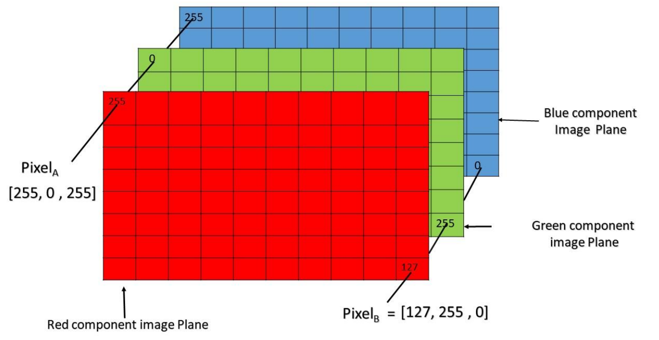
```
#YOUR CODE HERE
from sklearn.neural_network import MLPClassifier

(train_data, train_labels) = get_train_data(flatten = True)
(test_data, test_labels) = get_test_data(flatten = True)
mlp = MLPClassifier(hidden_layer_sizes = 5)
mlp.fit(train_data, train_labels)
storage = mlp.predict(test_data)
print(accuracy_score(test_labels,storage))
#YOUR CODE HERE

0.5
```

Applying Neural Networks to Medical Imaging

- Our model was calibrated through the ‘relu’ and ‘sigmoid’ function
- We are given ‘images’ of shape ‘(64,64,3)’, each assigned a label PNEUMONIA or HEALTHY.

```
dense = DenseClassifier(hidden_layer_sizes = (64,32))
```

```
cnn = CNNClassifier(num_hidden_layers = 1)
```



Pixel of an RGB image are formed from the corresponding pixel of the three component images

Fitting

model_history = model.fit(train_data, train_labels, epochs = 100, validation_data = (test_data, test_labels), shuffle = True, callbacks = [monitor])

Our Model

```
model_history1 = dense.fit(train_data, train_labels, epochs = 100, validation_data = (test_data, test_labels), shuffle = True, call
model_history2 = cnn.fit(train_data, train_labels, epochs = 100, validation_data = (test_data, test_labels), shuffle = True, callba
```

```
Epoch 1/100
63/63 [==============================] - 0s 5ms/step - loss: 0.2739 - accuracy: 0.8950 - val_loss: 0.5445 - val_accuracy: 0.725
Epoch 2/100
63/63 [==============================] - 0s 4ms/step - loss: 0.2523 - accuracy: 0.9015 - val_loss: 0.6019 - val_accuracy: 0.722
Epoch 3/100
63/63 [==============================] - 0s 5ms/step - loss: 0.2518 - accuracy: 0.9045 - val_loss: 0.5639 - val_accuracy: 0.715
Epoch 4/100
63/63 [==============================] - 0s 5ms/step - loss: 0.2574 - accuracy: 0.9040 - val_loss: 0.5502 - val_accuracy: 0.722
Epoch 5/100
63/63 [==============================] - 0s 5ms/step - loss: 0.2604 - accuracy: 0.9040 - val_loss: 0.6377 - val_accuracy: 0.720
Epoch 6/100
63/63 [==============================] - 0s 5ms/step - loss: 0.2578 - accuracy: 0.8970 - val_loss: 0.5987 - val_accuracy: 0.725
Epoch 7/100
63/63 [==============================] - 0s 5ms/step - loss: 0.2563 - accuracy: 0.9020 - val_loss: 0.5533 - val_accuracy: 0.710
Epoch 8/100
63/63 [==============================] - 0s 5ms/step - loss: 0.2587 - accuracy: 0.8960 - val_loss: 0.6787 - val_accuracy: 0.712
Epoch 9/100
63/63 [==============================] - 0s 5ms/step - loss: 0.2444 - accuracy: 0.9000 - val_loss: 0.6323 - val_accuracy: 0.717
Epoch 10/100
63/63 [==============================] - 0s 5ms/step - loss: 0.2515 - accuracy: 0.9040 - val_loss: 0.6306 - val_accuracy: 0.705
Epoch 11/100
63/63 [==============================] - 0s 5ms/step - loss: 0.2259 - accuracy: 0.9030 - val_loss: 0.6115 - val_accuracy: 0.730
Epoch 12/100
63/63 [==============================] - 0s 4ms/step - loss: 0.2651 - accuracy: 0.8975 - val_loss: 0.5783 - val_accuracy: 0.730
Epoch 13/100
63/63 [==============================] - 0s 5ms/step - loss: 0.2415 - accuracy: 0.9035 - val_loss: 0.5673 - val_accuracy: 0.732
Epoch 14/100
63/63 [==============================] - 0s 4ms/step - loss: 0.2511 - accuracy: 0.8980 - val_loss: 0.5311 - val_accuracy: 0.730
Epoch 15/100
63/63 [==============================] - 0s 4ms/step - loss: 0.2662 - accuracy: 0.9105 - val_loss: 0.4920 - val_accuracy: 0.752
Epoch 16/100
63/63 [==============================] - 0s 5ms/step - loss: 0.2456 - accuracy: 0.9095 - val_loss: 0.5995 - val_accuracy: 0.727
```

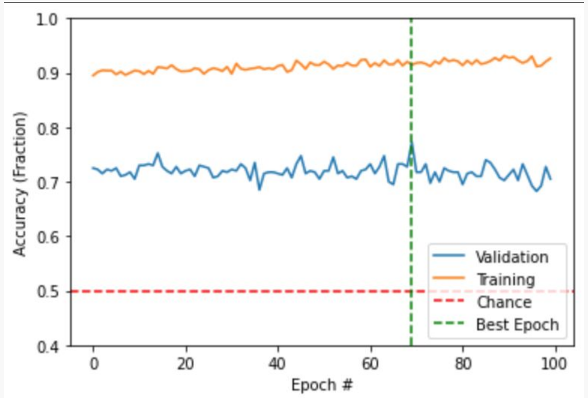
Scoring & Plotting

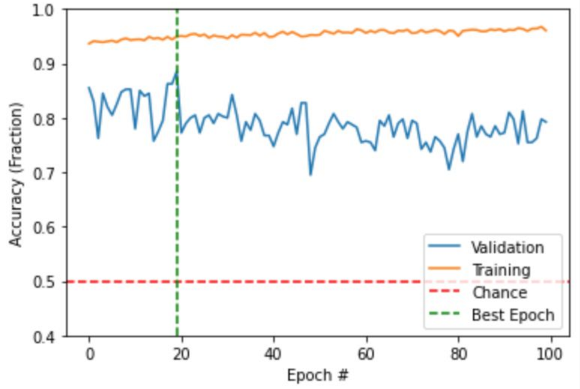
score[0] will be test loss and score[1] will be test accuracy

```
score1 = cnn.evaluate(test_data, test_labels, verbose=0)
score2 = dense.evaluate(test_data, test_labels, verbose=0)
print(score1)
print(score2)
```

```
[1.0759947299957275, 0.8525000214576721]
[0.628600537776947, 0.7250000238418579]
```

Green dotted line is where our model worked best





4 Experimenting With and Augmenting Field Data

Field Data:

Data that is **different** from the **data that you used to build your model** (in your training and test sets). In the context of pneumonia imaging, an example of this could be images that were captured on a different x-ray than the images you built your model with.

73% Accuracy

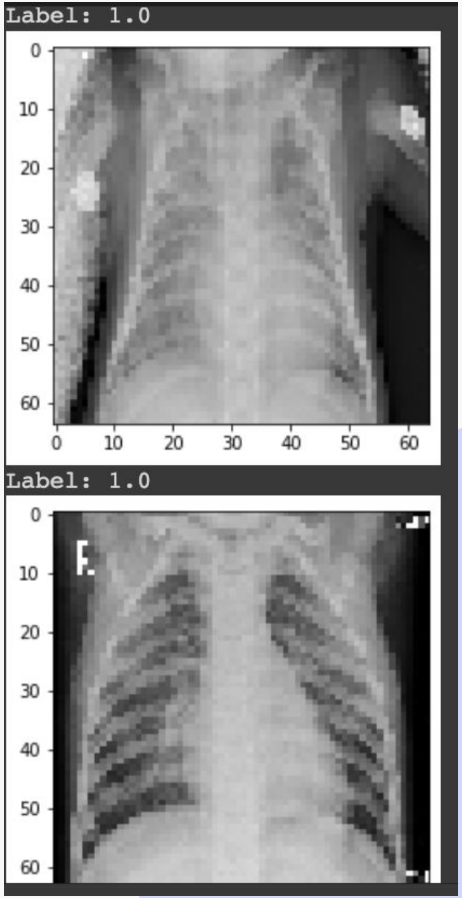
```
y = (cnn.predict(field_data) > 0.5).astype("int32")
print(accuracy_score(field_labels,y))
```

```
0.73
```

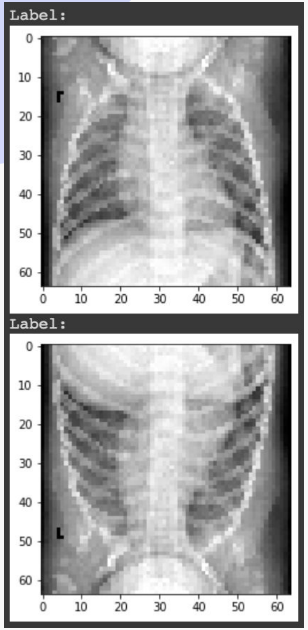
Comparing and Contrasting Data

Function = plot_one_image

Field Data:
Varied in levels of blurriness
Training Data:
Consistent in levels of blurriness



Solution: Augmentation



Accuracy Improved to 80%