

Pneumonia Detection from X-ray images using Deep Learning Neural Networks

Javier Herbas

Presentation Outline

Pneumonia background

Data Gathering

Convolutional Neural Networks

Model Evaluation

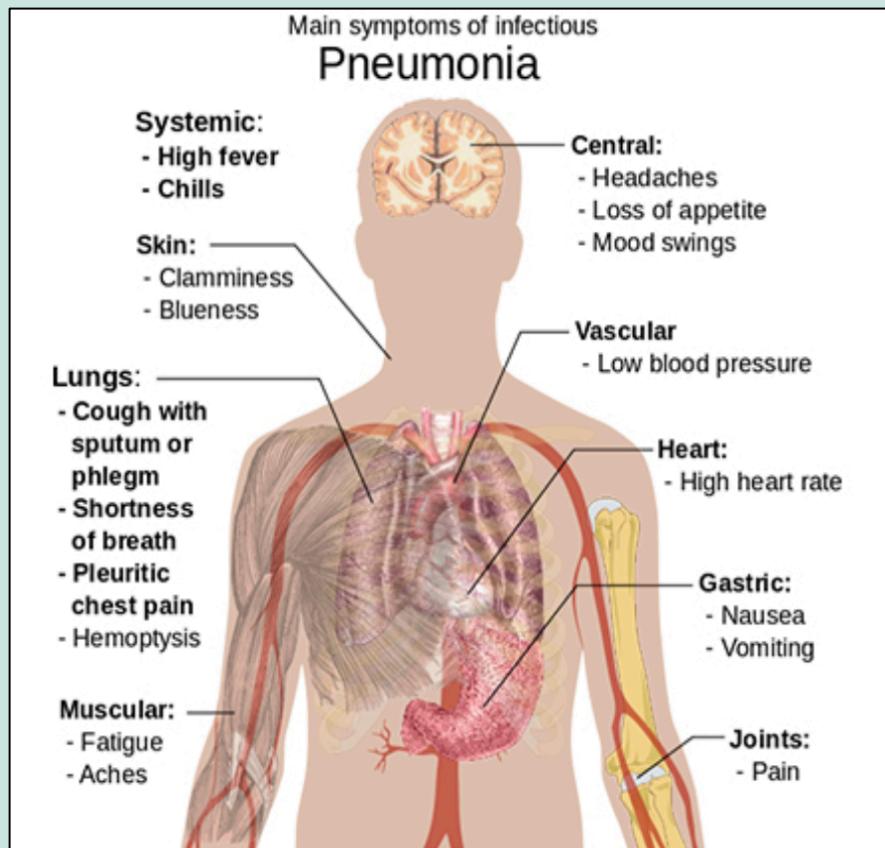
Summary and Recommendations

Way Forward

Pneumonia Background

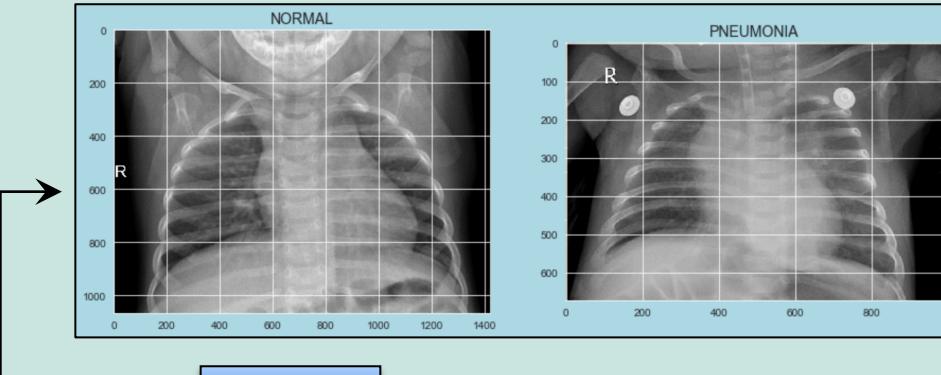
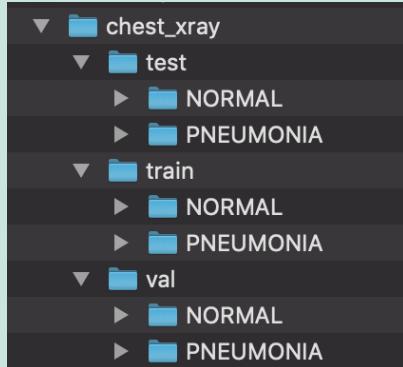
What is Pneumonia?

- Is a lung infection that can be caused by different types of microorganisms including bacteria, viruses, and fungi
- It causes the lung's air sac, or alveoli, to become inflamed and filled up with fluid and/or pus, making breathing more difficult
- Severe pneumonia will inflame the lungs to a point where they cannot take enough oxygen or expel enough carbon dioxide
- Continuous deprivation of oxygen can damage organs such as the kidney and the heart
- Today the most common diagnosis of severe Covid-19 is severe Pneumonia

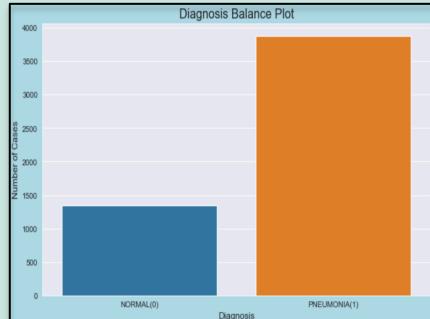


Data Gathering and QC

- Source: Kaggle
- X-rays: 5856
- Nested Folder Structure:
 - Train subset: 5216 (89.07%)
 - Test subset: 624 → 524 (8.95%)
 - Val subset: 16 → 116 (1.98%)

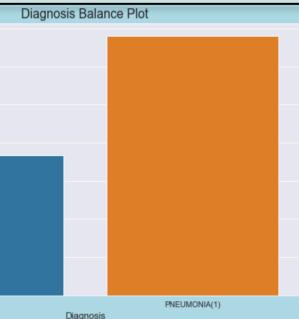


Train Subset



Looking at the data distribution

Test Subset



- Images were resized to 256x256 pixels (0-255)
- Train subset was augmented modifying 4 parameters

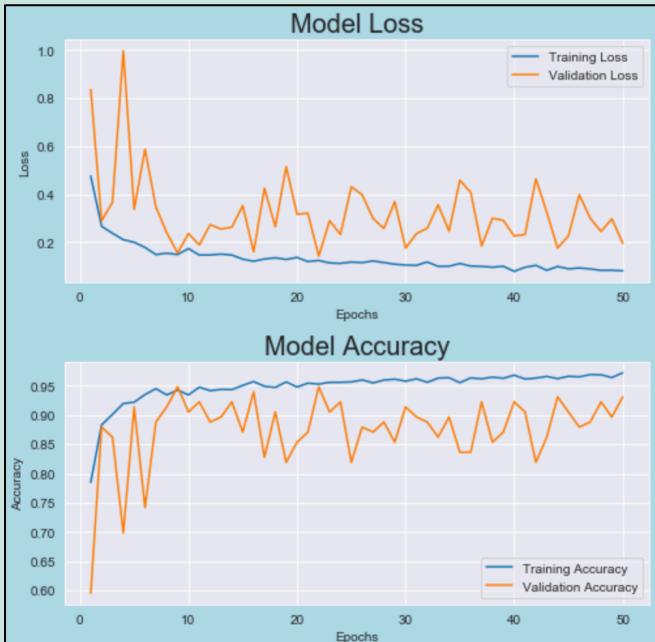
Convolutional Neural Network Model

- 15 different model architectures were tested
- Simple 2-layer models ➡ more complex 5-layer models.
- Tests were done with 5, 10 and 15 epochs. If validation accuracy > 70% then model was fitted with 30 and 50 epochs

```
1 model_5 = models.Sequential()                                         Model_5
2
3 # 1st Convolution
4 model_5.add(layers.Conv2D(32, (3, 3), activation='relu',
5                      input_shape=(150, 150, 3)))
6 model_5.add(layers.MaxPooling2D((2, 2)))
7
8 # 2nd Convolution
9 model_5.add(layers.Conv2D(64, (3, 3), activation='relu'))
10 model_5.add(layers.MaxPooling2D((2, 2)))
11
12 # 3rd Convolution
13 model_5.add(layers.Conv2D(128, (3, 3), activation='relu'))
14 model_5.add(layers.MaxPooling2D((2, 2)))
15
16 # 4th Convolution
17 model_5.add(layers.Conv2D(128, (3, 3), activation='relu'))
18 model_5.add(layers.MaxPooling2D((2, 2)))
19
20 # Flattened the layer
21 model_5.add(layers.Flatten())
22
23 # Fully connected layers
24 model_5.add(layers.Dense(64, activation='relu'))
25 model_5.add(layers.Dense(2, activation='softmax'))
26
27 model_5.summary()
```

```
1 model_15 = models.Sequential()                                         Model_15
2
3 # 1st Convolution block
4 model_15.add(layers.Conv2D(16, (3, 3), activation='relu', padding='same',
5                         input_shape=(150, 150, 3)))
6 model_15.add(layers.Conv2D(16, (3, 3), activation='relu'))
7 model_15.add(layers.MaxPooling2D((2, 2)))
8
9 # 2nd Convolution block
10 model_15.add(layers.Conv2D(32, (3, 3), activation='relu', padding='same'))
11 model_15.add(layers.Conv2D(32, (3, 3), activation='relu'))
12 model_15.add(layers.MaxPooling2D((2, 2)))
13
14 # 3rd Convolution block
15 model_15.add(layers.Conv2D(64, (3, 3), activation='relu', padding='same'))
16 model_15.add(layers.Conv2D(64, (3, 3), activation='relu'))
17 model_15.add(layers.MaxPooling2D((2, 2)))
18
19 # 4th Convolution block
20 model_15.add(layers.Conv2D(96, (3, 3), dilation_rate=(2,2), activation='relu',
21                         padding='same'))
22 model_15.add(layers.Conv2D(96, (3, 3), activation='relu'))
23 model_15.add(layers.MaxPooling2D((2, 2)))
24
25 # 5th Convolution block
26 model_15.add(layers.Conv2D(128, (3, 3), dilation_rate=(2, 2), activation='relu',
27                         padding='same'))
28 model_15.add(layers.Conv2D(128, (3, 3), activation='relu'))
29 model_15.add(layers.MaxPooling2D((2, 2)))
30
31 # Flattened the layer
32 model_15.add(layers.Flatten())
33
34 # Fully connected layers
35 model_15.add(layers.Dense(64, activation='relu'))
36 model_15.add(layers.Dropout(0.4))
37 model_15.add(layers.Dense(2, activation='softmax'))
38
39 model_15.summary()
```

Model Evaluation



Model 5:
Epoch 50/50
Train Accuracy: 0.9722
Train Loss: 0.0803
Val. Accuracy: 0.9310
Train Loss: 0.1954

Model 15:
Epoch 50/50
Train Accuracy: 0.9631
Train Loss: 0.1026
Val. Accuracy: 0.9224
Train Loss: 0.2479

test accuracy: 91.22
test loss: 0.39

Almost identical



test accuracy: 90.84
test loss: 0.3

Model Evaluation

- For unbalanced data the accuracy is not the best metric

		Predicted	
		Negative	Positive
Actual	Negative	True Negative	False Positive
	Positive	False Negative	True Positive

As a reminder:

$$\text{Precision} = \text{TP}/(\text{TP} + \text{FP})$$

$$\text{Recall} = \text{TP}/(\text{TP} + \text{FN})$$

$$\text{F1 score} = (\text{TP} + \text{TN})/\text{Total}$$

TP = True Positive

FP = False Positive

FN = False Negative

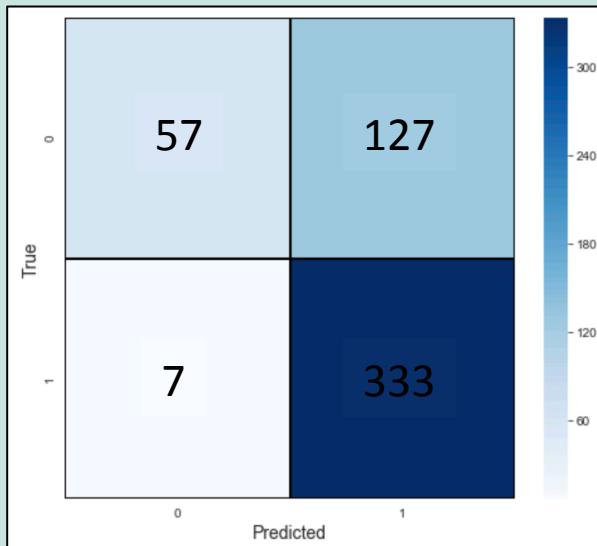
TN = True Negative

Model 5

Precision score: 72.39 %

Recall score: 97.94 %

F1 score: 83.25%

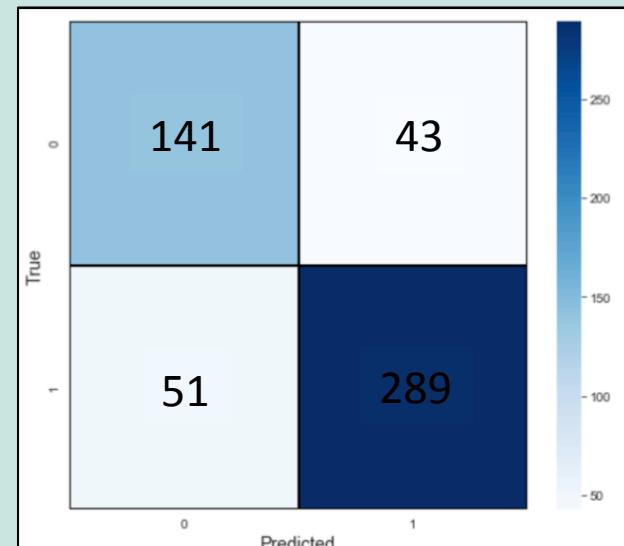


Model 15

Precision score: 87.05%

Recall score: 85.00%

F1 score: 86.01%



Summary & Recommendations

The following recommendations are based on my short experience with Deep Learning Neural Network, and they represent the challenges I faced during this project.

- Check your data's distribution so that you can properly interpret your model after training it. An unbalanced dataset will not be properly evaluated with just accuracy but with other metrics such as precision and recall. Confusion Matrix are very useful
- Don't over complicate the model's architecture as it won't necessarily translate as better. Start simple and add layers as you progress
- Don't limit yourself to just one model. Try different options so that you can compare results and reduce uncertainty

Way forward

- Train both selected models with a minimum of 100 epochs to try to reach convergence
- Redo the selected models with a split of 80, 20, 20% for train, test and validation subset respectively, and compare to current split used
- Redo this exercise with the original dataset from the Mendeley site (June 2018)

01110100
01101000
01100001
01101110
01101011
00100000
01111001
01101111
01110101

