

# **Pneumonia Detection** from Chest X-Ray Images

A Project Report

Submitted in fulfillment of the requirements for the award of  
The Certificate for

**Internship Program 2019**

By

**SMARTBRIDGE**

By

**A Jyotishwara Prasad**

(Geetanjali College of Engineering and Technology)

**N Mukesh**

(Geetanjali College of Engineering and Technology)

**E Pravallika**

(JNTUA College of Engineering, Pulivendula)

**P Saileswar Reddy**

(JNTUA College of Engineering, Pulivendula)

**K R G Deepak Teja**

(JNTUA College of Engineering, Pulivendula)

## **ABSTRACT**

With the high rise in popularity of neural networks, researchers and engineers have been able to find many world-changing applications for computer vision. Today, I want to show you an example of this. By acquiring a labeled dataset of chest x-ray scans for pneumonia, I was able to rapidly build and prototype an artificial intelligence model that identified pneumonia cases with 90% accuracy. (for test\_data)

### **Computer Vision & Convolutional Neural Networks**

CNNs are very similar to the traditional feed-forward networks which have come to define deep learning, except that they implement a mathematical operation known as a convolution. Compared to traditional neural networks, they are a lot more efficient at processing image data. By applying convolutions, we can be much more efficient with these operations.

### **Organizing our X-Ray Scans**

Thanks to [Kaggle](#), I was able to obtain this dataset for pneumonia x-ray scans, which already came labeled! There was one folder named “Normal Scans” and another “Pneumonia Scans”. Detecting pneumonia in an x-ray scan is a simple binary classification problem: either we detect pneumonia, or we don’t. Did I give you a good hint with binary?

### **Building Our Model**

A great prototyping library/tool is Keras. Keras is a high-level API which lets you build neural networks quickly, without having to worry too much about their intricacies. Keras uses a more complicated framework on the back-end which you get to define. For this Pneumonia detection example, I used Keras with a Tensorflow backend. Here we also install pillow library for loading, modifying and saving images.

### **Training and Testing the Network**

Let’s train the network. Fundamentally speaking, the process I am automating is very simple. The accuracy achieved on train\_data is 90%.

# CONTENTS

## Problem Statement

<b>1. Introduction</b>	<b>1</b>
a. Artificial Intelligence	
b. Neural Networks	
c. Convolutional Neural Networks	
<b>2. Dataset Details</b>	<b>3</b>
<b>3. Libraries</b>	<b>4</b>
a. Keras	
b. TensorFlow	
c. Pillow	
d. matplotlib	
<b>4. Functions</b>	<b>5</b>
a. image_data_format()	
b. ImageDataGenerator()	
c. flow_from_directory()	
d. fit_generator()	
e. evaluate_generator()	
<b>5. Output</b>	<b>8</b>
<b>6. Conclusion</b>	<b>8</b>
<b>References</b>	<b>8</b>
<b>Appendices</b>	<b>9</b>

**Problem Statement:** To detect Pneumonia clouds in scanned **Chest X-Rays** of pediatric patients.

## 1. Introduction

Finding ways to automate diagnostics from medical images, has continuously been one of the most interesting areas of software development. This model presents an approach for detecting the presence of pneumonia clouds in chest X-rays by using Image processing techniques. For this, we have worked on a dataset containing Chest X-ray images pertaining to Normal and Pneumonia infected patients. The task has been performed using Python and keras with Tensorflow as background because they are free, open source tools and can be used by all, without any legality issues or cost implication.

- Domain : Artificial Intelligence
- Sub-Domain : Image Recognition
- Techniques : Convolution Neural Networks

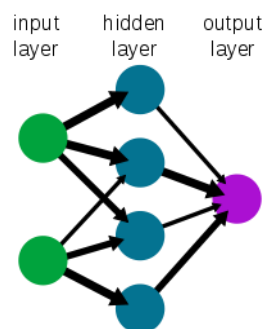
### Artificial Intelligence

**Artificial intelligence** (AI) is the branch of computer science which aims to create intelligence machines. Artificial Intelligence (AI) is the study of how computer systems can simulate intelligent processes such as learning, reasoning, and understanding symbolic information in context.

It is simply a technique which enables computer to mimic human behavior.

### Neural Networks

A **neural network** is a network or circuit of neurons, or in a modern sense, an artificial neural network, composed of artificial neurons or nodes for solving Artificial Intelligence (AI) problems.

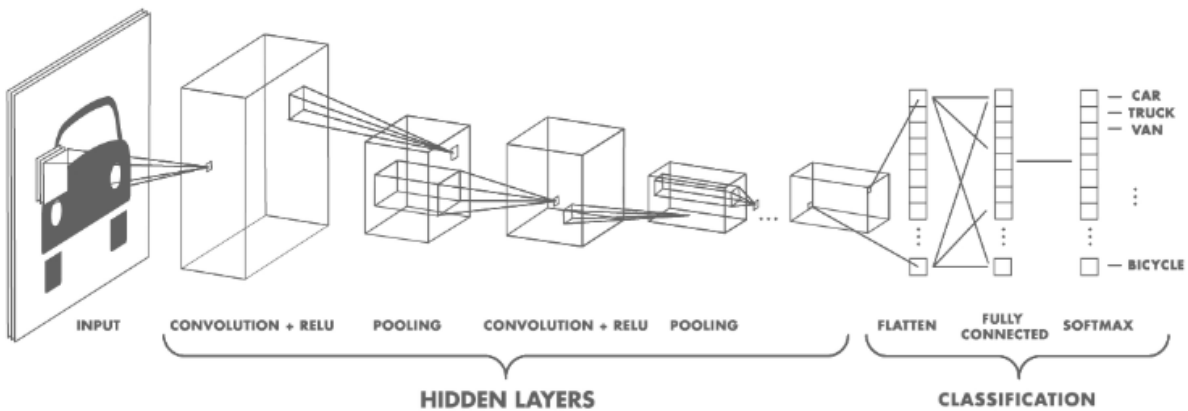


**Fig 1.1 A Simple Neural Network**

## Convolutional Neural Networks

A **Convolutional Neural Network (CNN, or ConvNet)** is a class of deep neural networks, most commonly applied to analyzing visual imagery.

They have applications in image and video recognition, recommender systems, image classification, medical image analysis, and natural language processing.



**Fig 1.2 Convolution Neural Network**

### Building Blocks

- Convolution Layer
- Pooling Layer
- RELU Layer
- Full Connection

### Tools/Libraries

- Languages : Python
- Tools/IDE : Anaconda
- Libraries : keras, tensorflow, pillow, matplotlib

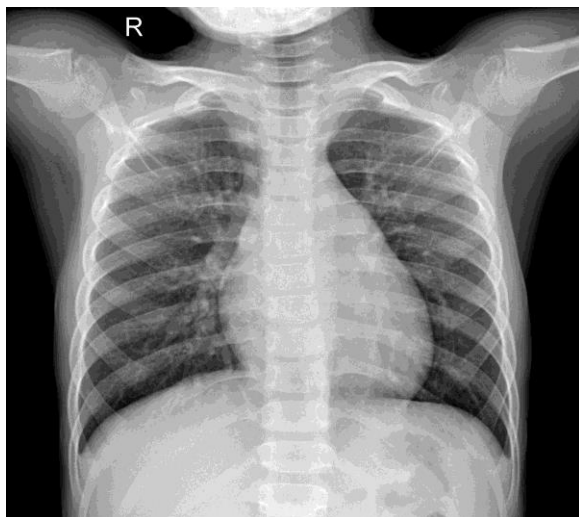
## 2. Dataset Details

- Dataset Name : chest-xray-pneumonia
- Dataset source : <https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia>

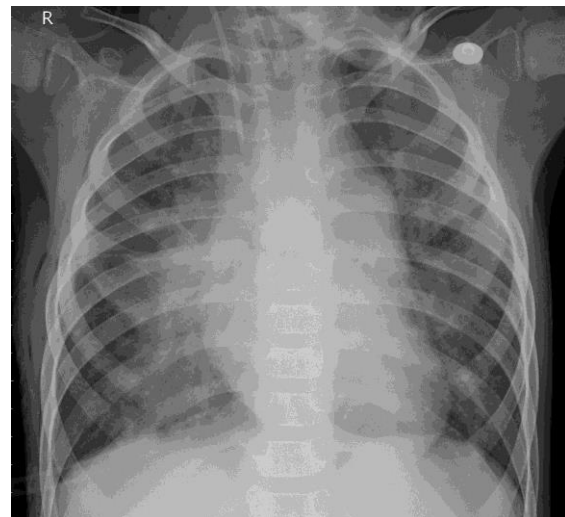
Chest X-ray images of pediatric patients of one to five years old were selected as part of patients' routine clinical care and grouped in this dataset.

The Dataset is organized into three folders (train, test and val) and contains two subfolders (Pneumonia/Normal) for each image category.

- Number of Classes : 2
- Number of Images :
  - Training : 5216
    - Normal : 1341
    - Pneumonia : 3875
  - Testing : 624
    - Normal : 234
    - Pneumonia : 390
  - Validation : 16
    - Normal : 08
    - Pneumonia : 08



**Fig 2.1 X-ray of Normal Person**



**Fig 2.2 X-ray of Pneumonia Infected person**

In total there are 5856 X-Ray images (JPEG) under 2 categories (Pneumonia/Normal).

### 3. Libraries

#### Keras

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, Theano, or PlaidML.

Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.

#### TensorFlow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks.



**Fig 3.1 TensorFlow Logo**

#### Pillow

**Python Imaging Library** (abbreviated as **PIL**) (in newer versions known as Pillow) is a free library for the Python programming language that adds support for opening, manipulating, and saving many different image file formats. It is available for Windows, Mac OS X and Linux.

#### Matplotlib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+.

Matplotlib was originally written by John D. Hunter, has an active development community, and is distributed under a BSD-style license.

## 4. Functions

### `image_data_format()`

The "backend" parameter should either be "tensorflow", "cntk", or "theano". When switching the backend, make sure to switch the "image\_data\_format" parameter too. For "tensorflow" or "cntk" backends, it should be “**channels\_last**”. For “theano”, it should be “**channels\_first**”.

```
"image_data_format": "channels_first",  
"backend": "theano"
```

```
"image_data_format": "channels_last",  
"backend": "tensorflow"
```

### `ImageDataGenerator()`

Generate batches of tensor image data with real-time data augmentation. The data will be looped over (in batches) i.e it behaves like an iterator. It has a special keyword **yield**, which is similar to **return** as it returns some value. When the generator is called, it will return some value and save the state. Next time when we call the generator again, it will resume from the saved state, and return the next set of values just like an iterator.

Thus using the advantage of generator, we can iterate over each (or batches of) image(s) in the large data-set and train our neural net quite easily.

### `flow_from_directory()`

The `ImageDataGenerator` class has three methods `flow()`, `flow_from_directory()` and `flow_from_dataframe()` to read the images from a big numpy array and folders containing images.

The `flow_from_directory()` expects at least one directory under the given directory path.

### `fit_generator()`

In `fit_generator()`, you don't pass the x and y directly, instead they come from a generator.

As it is written in keras documentation, generator is used when you want to avoid duplicate data when using multiprocessing. This is for practical purpose, when you have large dataset.

keras' `fit_generator()` function is used if



- Real-world datasets are often too large to fit into memory.
- They also tend to be challenging, requiring us to perform data augmentation to avoid overfitting and increase the ability of our model to generalize.

In [7]: `model.summary()`

Layer (type)	Output Shape	Param #
=====		
conv2d_4 (Conv2D)	(None, 148, 148, 32)	896
activation_4 (Activation)	(None, 148, 148, 32)	0
max_pooling2d_4 (MaxPooling2D)	(None, 74, 74, 32)	0
conv2d_5 (Conv2D)	(None, 72, 72, 32)	9248
activation_5 (Activation)	(None, 72, 72, 32)	0
max_pooling2d_5 (MaxPooling2D)	(None, 36, 36, 32)	0
conv2d_6 (Conv2D)	(None, 34, 34, 64)	18496
activation_6 (Activation)	(None, 34, 34, 64)	0
max_pooling2d_6 (MaxPooling2D)	(None, 17, 17, 64)	0
conv2d_7 (Conv2D)	(None, 15, 15, 128)	73856
activation_7 (Activation)	(None, 15, 15, 128)	0
max_pooling2d_7 (MaxPooling2D)	(None, 7, 7, 128)	0
flatten_1 (Flatten)	(None, 6272)	0
dense_1 (Dense)	(None, 50)	313650
activation_8 (Activation)	(None, 50)	0
dropout_1 (Dropout)	(None, 50)	0
dense_2 (Dense)	(None, 1)	51
activation_9 (Activation)	(None, 1)	0
=====		
Total params: 416,197		
Trainable params: 416,197		
Non-trainable params: 0		

**Fig 4.1 Convolution and Pooling**

## `evaluate_generator()`

Evaluates the model on a data generator.

```
#Accuracy of test data.
scores = model.evaluate_generator(test_generator, 624/16)
print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))

acc: 90.87%
```

**Fig 4.2 evaluate\_generator()**

```

Epoch 1/20
326/326 [=====] - 118s 363ms/step - loss: 0.4317 - acc: 0.8207 - val_loss:
0.3261 - val_acc: 0.8125
Epoch 2/20
326/326 [=====] - 104s 319ms/step - loss: 0.2619 - acc: 0.8961 - val_loss:
2.0861 - val_acc: 0.6250
Epoch 3/20
326/326 [=====] - 106s 325ms/step - loss: 0.2100 - acc: 0.9214 - val_loss:
0.6142 - val_acc: 0.6250
Epoch 4/20
326/326 [=====] - 104s 319ms/step - loss: 0.1896 - acc: 0.9300 - val_loss:
1.5634 - val_acc: 0.6250
Epoch 5/20
326/326 [=====] - 98s 299ms/step - loss: 0.1703 - acc: 0.9346 - val_loss: 0.6629
- val_acc: 0.6875
Epoch 6/20
326/326 [=====] - 98s 301ms/step - loss: 0.1638 - acc: 0.9427 - val_loss: 2.9174
- val_acc: 0.5625
Epoch 7/20
326/326 [=====] - 101s 311ms/step - loss: 0.1681 - acc: 0.9398 - val_loss:
1.1629 - val_acc: 0.6250
Epoch 8/20
326/326 [=====] - 104s 320ms/step - loss: 0.1642 - acc: 0.9425 - val_loss:
1.9716 - val_acc: 0.6250
Epoch 9/20
326/326 [=====] - 106s 327ms/step - loss: 0.1477 - acc: 0.9465 - val_loss:
2.6345 - val_acc: 0.6250
Epoch 10/20
326/326 [=====] - 100s 307ms/step - loss: 0.1473 - acc: 0.9469 - val_loss:
0.8341 - val_acc: 0.6875
Epoch 11/20
326/326 [=====] - 114s 350ms/step - loss: 0.1568 - acc: 0.9477 - val_loss:
0.9443 - val_acc: 0.6875
Epoch 12/20
326/326 [=====] - 104s 319ms/step - loss: 0.1430 - acc: 0.9503 - val_loss:
2.4098 - val_acc: 0.5625
Epoch 13/20
326/326 [=====] - 108s 332ms/step - loss: 0.1455 - acc: 0.9509 - val_loss:
1.1889 - val_acc: 0.5625
Epoch 14/20
326/326 [=====] - 117s 359ms/step - loss: 0.1310 - acc: 0.9559 - val_loss:
0.4286 - val_acc: 0.8125
Epoch 15/20
326/326 [=====] - 99s 305ms/step - loss: 0.1378 - acc: 0.9544 - val_loss: 0.5606
- val_acc: 0.7500
Epoch 16/20
326/326 [=====] - 108s 333ms/step - loss: 0.1375 - acc: 0.9509 - val_loss:
0.7799 - val_acc: 0.7500
Epoch 17/20
326/326 [=====] - 112s 344ms/step - loss: 0.1309 - acc: 0.9561 - val_loss:
1.3712 - val_acc: 0.5625
Epoch 18/20
326/326 [=====] - 103s 315ms/step - loss: 0.1326 - acc: 0.9534 - val_loss:
0.5645 - val_acc: 0.8125
Epoch 19/20
326/326 [=====] - 100s 307ms/step - loss: 0.1336 - acc: 0.9511 - val_loss:
0.4271 - val_acc: 0.8750
Epoch 20/20
326/326 [=====] - 100s 307ms/step - loss: 0.1314 - acc: 0.9580 - val_loss:
0.6753 - val_acc: 0.6875

```

**Fig 4.3 Epochs**

## Output

- Accuracy for train\_data : 95.80
- Loss for train\_data : 13.14

Accuracy of Test\_Data : 90.31

## Conclusion

Thus a model was built for detecting pneumonia from scanned X-ray images by using CNN with an accuracy of 95% (on train\_data) and an accuracy of 90% (on test\_data) approximately.

### Further Enhancements:

- We can compile the model at more number of epochs.
- Hyper-parameter Tuning(There are a lot of parameters that we can play with).
- Use of much more deeper architectures.

## References

- i. [https://en.wikipedia.org/wiki/Neural\\_network](https://en.wikipedia.org/wiki/Neural_network)
- ii. [https://en.wikipedia.org/wiki/Convolutional\\_neural\\_network](https://en.wikipedia.org/wiki/Convolutional_neural_network)
- iii. <https://en.wikipedia.org/wiki/Keras>
- iv. <https://en.wikipedia.org/wiki/TensorFlow>
- v. [https://en.wikipedia.org/wiki/Python\\_Imaging\\_Library](https://en.wikipedia.org/wiki/Python_Imaging_Library)
- vi. <https://en.wikipedia.org/wiki/Matplotlib>
- vii. image\_data\_format():  
<https://www.codesofinterest.com/2017/09/keras-image-data-format.html>
- viii. fit\_generator():  
<https://datascience.stackexchange.com/questions/34444/what-is-the-difference-between-fit-and-fit-generator-in-keras>
- ix. <https://towardsdatascience.com/>

## Appendices

### Image processing

**Image processing** is a method to perform some operations on an **image**, in order to get an enhanced image or to extract some useful information from it. It is a type of signal processing in which input is an image and output may be image or characteristics/features associated with that image.

### Neuron

A **neuron** (also called nerve cell) is a cell that carries electrical impulses. Neurons are the basic units of the nervous system. Every neuron is made of a cell body (also called a soma), dendrites and an axon. Dendrites and axons are nerve fibers

### Theano

**Theano** is a Python library and optimizing compiler for manipulating and evaluating mathematical expressions, especially matrix-valued ones.