# 

# A Report On

Prediction of Pneumonia using X-Ray Images

By

Vadaparthi Sai Sanjana

Indian Institute of Technology, Roorkee

Mechanical Engineering

18117111

saisanjanavadaparthi@gmail.com

# Abstract

The main objective of this project is to predict pneumonia disease more accurately by passing X-ray images as an input to a deep learning model. This model not only helps doctors but also patients to verify whether they have pneumonia or not. A convolutional neural network model is built from scratch to extract features from a given chest X-ray image and classify it to determine if a person is infected with pneumonia or not. A web application is built where the user can upload the X-ray image and the result is shown on the UI.

[**Abstract**](#_5lmkb6r1hea4)1

[**Introduction**](#_hgqxxkga1t0q)2

[What is AI?](#_9h6iceijh8im) 2

[What is Machine Learning?](#_5bsfowr4jfnc) 2

[The Three Types of Machine Learning](#_mlwhf6hdcee2) 3

[Supervised Learning](#_ik21j1tb9gst) 3

[Unsupervised Learning](#_clvg5uwe94ek) 3

[Reinforcement Learning](#_cjeug4r9bbw5) 3

[What is Deep Learning?](#_cl05ci5iunwb) 4

[**Convolution Neural Networks**](#_eiacnnvlrzu7)4

[**How Convolution Neural Network model is developed in this project**](#_6e1uhqq2tw9f)5

[1. First add method - adding convolution 2D layer](#_2f8992vbz6ne) 6

[2. Second add method - adding Max Pooling Layer](#_dk6rsck2dvbb) 7

[3. Third add method - adding flatten layer](#_pi4xm2s0lpgc) 8

[4. Adding the two dense layers](#_i2lcdksbz1pw) 9

[5. Importing dataset and applying image preprocessing techniques](#_syv6yhs3a39l) 10

[6. Compiling the model](#_a3du6khkvoye) 11

[7. Fitting the model](#_wmk0pxum0e2s) 11

[8. Saving the model](#_h5j7qswy68oh) 12

[**Building a Web Application Using Flask**](#_jngj2dy4ueid)12

[**File structure**](#_2f6v3yitb766)13

# Introduction

## What is AI?

AI means getting the computer to mimic human behaviour in some way. The goal of AI is to get computers to perform tasks regarded as uniquely human: things that require intelligence. It refers to the output of a computer i.e., the computer exhibits intelligence that is artificial. The term AI doesn't say anything about the techniques used. These techniques can be rule-based or expert systems (emulates the decision making ability of a computer to solve problems by reasoning through bodies of knowledge represented mainly as if-then rules), machine learning etc.

## What is Machine Learning?

Machine learning is an application of AI that provides systems with the ability to automatically learn and improve from experience without being explicitly programmed. It focuses on the development of computer programs that can access data and use it to learn for themselves. Machine learning outsources the rule defining to the computer by allowing it to find relationships between the input resources and output values. The best-fitting line/curve/boundary is found by iteratively trying algorithms to arrive at the best one. It uses the idea of artificial neurons and neural networks simulated in software for solving complex problems.

## The Three Types of Machine Learning

### **Supervised Learning**

The machine learning algorithm is trained on labelled data. The model is given a small training dataset to work with. The algorithm finds the relationships between the variables in the dataset. In the end, the solution is deployed for use with the final dataset. These supervised algorithms will continue to improve discovering new patterns and relationships as they train themselves on new data.

### **Unsupervised Learning**

This learning holds the advantage of being able to work with unlabeled data. Unsupervised learning does not have labels to work off of resulting in the creation of hidden structures. Instead of defined and set problem statements, unsupervised learning algorithms can adapt to the data by dynamically changing hidden structures.

### **Reinforcement Learning**

[Reinforcement learning](https://it.toolbox.com/article/openais-robot-learns-to-solve-a-rubiks-cube-with-one-hand-peter-welinder-research-lead-openai-shares-insights) directly takes inspiration from how human beings learn from data in their lives. It features an algorithm that improves upon itself and learns from new situations using a trial-and-error method. Favourable outputs are encouraged or ‘reinforced’, and non-favourable outputs are discouraged or ‘punished’. Reinforcement learning works by putting the algorithm in a work environment with an interpreter and reward system. In every iteration, the output result is given to the interpreter, which decides whether the outcome is favourable or not.

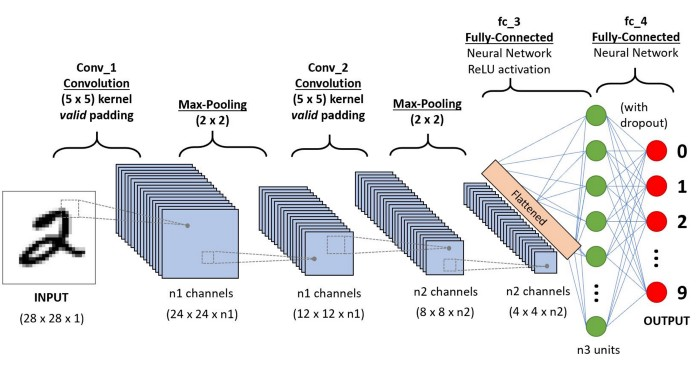
## What is Deep Learning?

Deep Learning is a subset of machine learning that enables computers to solve more complex problems. Put simply, it is all about using neural networks with more neurons, layers and interconnectivity. It mimics the working of the human brain in processing data for use in detecting objects, recognizing speech, translating languages, and making decisions. Deep Learning can learn without human supervision, drawing from data that is both unstructured and unlabeled.

# Convolution Neural Networks

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.



# How Convolution Neural Network model is developed in this project

* Importing the required libraries

*Sequential model* is in **keras.models** package

*MaxPooling2D* is in **keras.layers** package

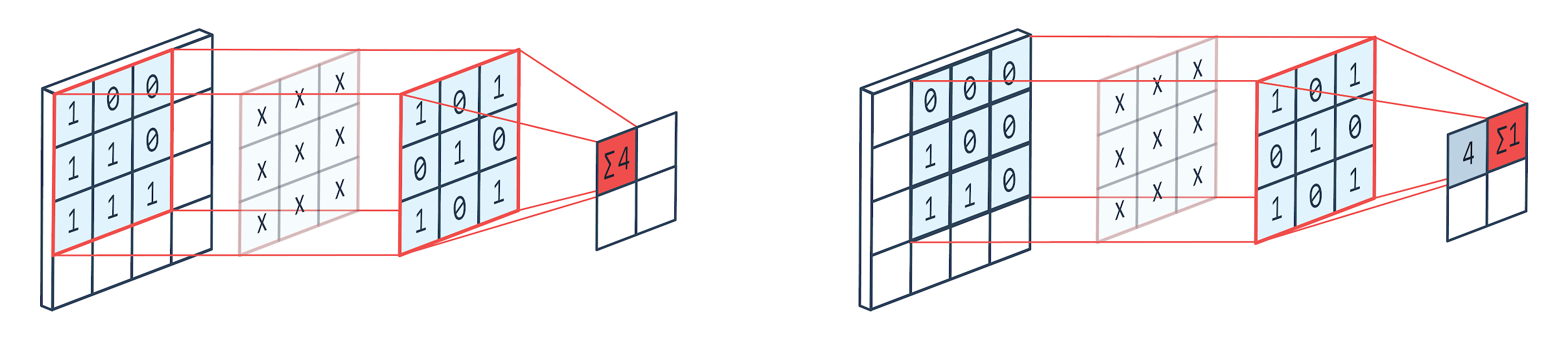
*Dense* is in **keras.layers** package

*ImageDataGenerator* class is in **keras.preprocessing.image** package

* Creating a sequential model incrementally via add methods

## 1. **First add method - adding convolution 2D layer**

*model.add(Convolution2D(32, (3, 3), input\_shape = (64, 64, 3), activation = ‘relu’))*



First parameter ‘32’ is the number of filters. Filters are feature detectors, the number of feature detectors signifies the number of features like edges, lines, object parts etc the network can potentially learn. The number of filters to be selected depends on the complexity of the dataset and depth of the neural network. A common setting to start with is [32, 64, 128] for three layers and if there are more than 3 layers, increasing to [256, 512, 1024], and so on.

The second parameter (3, 3) is the kernel size. It specifies the size of convolutional filters in pixels and it must be an odd integer. If the images are smaller than 128 x 128, smaller filters are considered and hence in the current instance (3, 3) filters are used as the sizes of images are taken as 64 x 64.

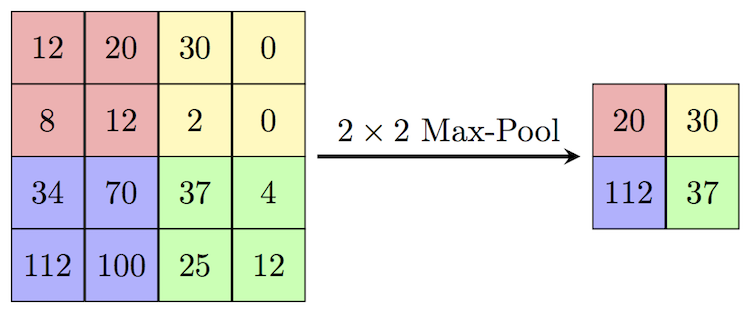
The third parameter is the size of the images. (64,64,3) is sent as a parameter because the images are 64 x 64 RGB images (3-RGB).

Fourth parameter is the **Activation Function** used. Here the Activation Function used is the **Rectified Linear Unit** (ReLu).

## 2. Second add method - adding Max Pooling Layer

*model.add(MaxPooling(pool\_size = (2,2))*

**Max Pooling** is a type of operation that is typically added to CNNs following convolution layers. This reduces the dimensionality of images by reducing the number of pixels in the output from the previous convolution layer. Here the size of the max-pooling layer is taken as 2 x 2, so the first 2 x 2 region is taken and the max value is calculated and stored in the output channel. Since we defined the size of our stride as 2, max is calculated in the next 2 x 2 block, stored in the output and then goes to the next blocks by sliding over by 2 again.



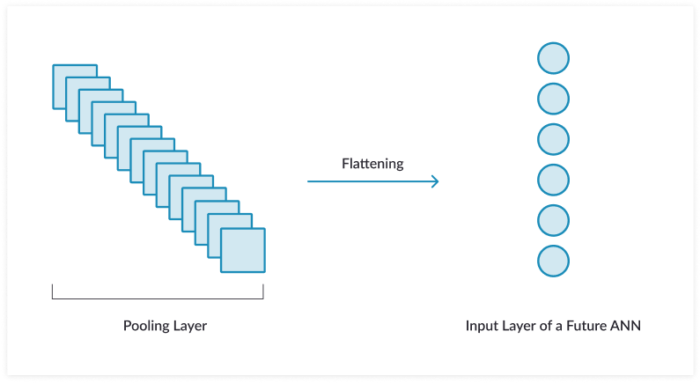
Some of the advantages of max-pooling are :

1. Reducing computational load
2. Reducing overfitting

## 3. Third add method - adding flatten layer

*model.add(Flatten())*

In between the convolutional layer and the fully connected dense layer, there is a **Flatten layer**. Flattening transforms a two-dimensional matrix of features into a vector that can be fed into a fully connected neural network classifier.



## 4. Adding the two dense layers

These fully connected layers work just like the **Artificial Neural Network**’s classification. When we work with ANN we provide features at the input node of the network, it then goes through the hidden layers and spits out some output on the output layer. From the above layers, all the required features are passed as inputs to these layers to get the prediction. Since now we have all the features as an input to the neural network, normal ANN’s work is required i.e., adding dense layers and output nodes at the final layer.

Parameters of Dense layer

1. **Units**

This parameter is a positive integer that denotes the output size of the layer.

1. **Activation**

This parameter sets the element-wise activation function to be used in the dense layer.

1. **Kernel\_initializer**

This parameter tells how to initialize the weights of the layer.

*model.add(Dense(units = 128, kernnel\_initializer = ‘uniform’, activation = ‘relu’))*

Here the parameter ‘units’ is randomly set to 128 since we don't know the exact number of pixels that are considered as inputs.

*model.add(Dense(units = 1, kernel\_initializer = ‘uniform’ activation = ‘sigmoid’))*

Here ‘units’ is set to 1 as the model should predict a single output.

In both the methods above, the parameter *kernel\_initializer* is set to uniform this allows uniform distribution of weights.

## 5. Importing dataset and applying image preprocessing techniques

Keras defines *ImageDataGenerator* class that defines the configuration for image data preparation and augmentation.

Image preprocessing is done to both train and test sets and the results are stored in new variables train\_datagen and test\_datagen.

*train\_datagen = ImageDataGenerator(rescale = 1./255, shear\_range = 0.2,zoom\_range = 0.2, horizontal\_flip = True)*

*test\_datagen = ImageDataGenerator(rescale = 1)*

Rescale parameter is to normalize images i.e, rescale the pixel values from the range of 0-255 to the range of 0-1 which is preferred for neural network models.

shear\_range, zoom\_range, horizontal flip parameters, in this case, allow the current image zoom to 20% and shear by 20% and enable horizontal flip to allow augmentation.

Now for importing the datasets, the datasets are passed into the variables train\_datagen and test\_datagen by:

*x\_train = train\_datagen.flow\_from\_directory(“path of train set”, target\_size = (64, 64), batch\_size = 32, class\_mode = binary)*

*y\_train = test\_datagen.flow\_from\_directory(“path of test set”, target\_size = (64, 64), batch\_size = 32, class\_mode = binary)*

target\_size parameter is set as (64, 64) since the input image size given is 64x64 initially.

Batch\_size is given as 30 since the number of images in the dataset is approximately 6000.

Class\_mode is binary since there are only two cases to be predicted i.e., normal and pneumonia conditions.

## 6. Compiling the model

*model.compile(loss = ’binary\_crossentropy’, optimizer = ‘adam’, metrics = ‘accuracy’)*

Here loss is taken as binary\_crossentropy since the problem is of binary classification. The optimizer used is ‘adam’ optimizer, optimizers are algorithms or methods used to change the attributes of the neural network such as weights and learning rate to reduce losses. Metrics used is accuracy, metrics are used to evaluate the performance of the model. It is similar to the loss function but not used in the training process.

## 7. Fitting the model

*model.fit(x\_train, steps\_per\_epoch = 163, epochs = 100, validation\_data = x\_test, validation\_steps = 20)*

This step is for training the model for a specified number of epochs or exposures to the training dataset. An epoch is referred to as a single pass through the entire training set, followed by the testing of the verification set.

Here the steps\_per\_epoch parameter is set to 163 since batch\_size is given as 20 and the total number of images in the test dataset is5216 (5216/20). Hence for every iteration, the model is trained with 163 images. Same is the case for the validation\_steps.

## 8. Saving the model

Given that deep learning models take hours, days and even weeks to train. It is important to save the model and load it from the disk. The model can be saved by calling the save() function on the model and specifying the filename.

*model.save(“pneumonia\_tester\_model.h5”)*

# Building a Web Application Using Flask

**Flask** is a lightweight python web framework that provides useful tools and features that make creating web applications in python easier. It gives developers flexibility and is a more accessible framework for new developers since the web application can be built quickly only using a single python file. Flask is also extensible and doesn't force a particular directory structure or require complicated boilerplate code before getting started. Flask uses Jinja template engine to dynamically build HTML pages using familiar python concepts such as variables, lists, loops and so on.

In the file, *app.py* flask object is imported and this object is used to create an instance of the flask application. Once the app instance is created, it is used to handle incoming web requests and send responses to the user.

*@app.route* is a decorator that turns a regular python function into a flask view function, which converts the function’s return value into an HTTP response to be displayed by an HTTP client, such as a web browser.

App.py

The saved CNN model is loaded by importing load\_model from keras.models

Index function

‘/’ is passed as a value to @app.route() to signify that this function will respond to web requests for the URL /, which is the main URL. Hence base.html file which contains the necessary HTML code is passed as a parameter to render\_template.

Upload function

This function is to extract the uploaded image into the uploads folder, to make the image compatible with the CNN model by converting it into the size 64x64 and predict the result using the predict function.

# File structure

* Project
  + test
  + train
  + static
    - css
* main.css
  + - js
* main.js
  + templates
    - base.html
  + uploads
* app.py
* model.h5

# Screenshots for Reference

