Chapter 1

Introduction

1.1 Introduction to Pneumonia

Pneumonia is an infection which is caused in one or both sections of the lungs. Pneumonia becomes more serious when children under the age of five, adult over the age of 65 & people have some conditions like heart failure, diabetes or HIV/AIDS where immune system of the people is weak & they would not able to recovery. Pneumonia is always caused due to weak immunity or weak immune system.

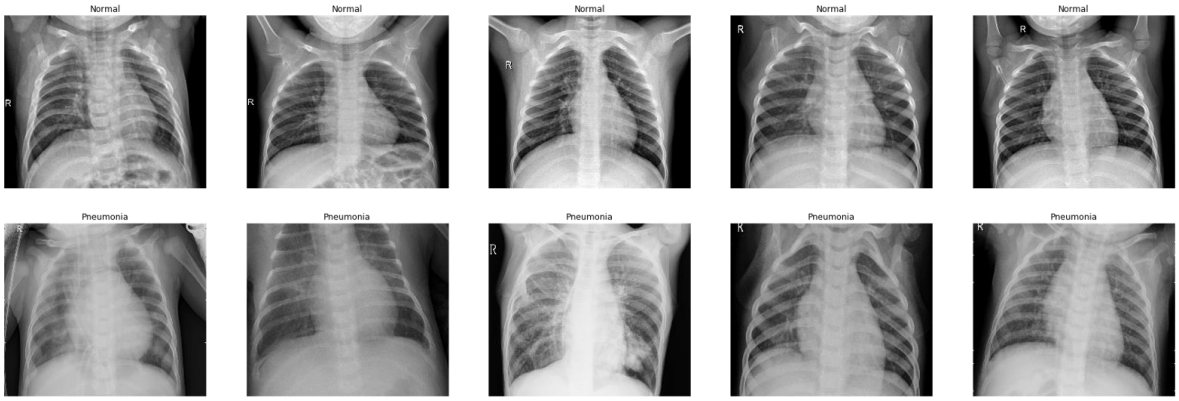

Fig 1.1 Chest X-ray

Fig 1.1 Different type of pneumonia

Majorly pneumonia is categorized into two types:

- Lobar pneumonia: It is also known as non-segmental pneumonia/ focal non-segmental pneumonia. It is a type of bacterial pneumonia. Lobar pneumonia will affect one or more sections of the lungs. In lobar pneumonia, people are generally suffered due to severe pain in lungs, coughing.
- Bronchial pneumonia: It is also known as Bronchopneumonia. Bronchial pneumonia is a type of pneumonia or a condition that causes redness, swallowing & warmness of the lung. This type of pneumonia or condition is very common in children & the one of the leading cause of death in children under 5 years.

1.1.1 Symptoms:

i. Trouble in Breathing
ii. High or moderate Fever
iii. Coughing with phlegm
iv. Chills
v. Sweating
vi. Low energy & fatigue
vii. Headache
viii. Dizziness

ix. Appetite loss
x. Chest pain with difficulty in breathing

1.1.2 Causes:

i. Bacterial pneumonia: Bacterial pneumonia is very common type of pneumonia in adults. But it can affect person from any age, & increase the risk because when abuse of alcohol, smoke cigarettes, have diabetes or weak immune system.
ii. Viral pneumonia: Viral pneumonia is cause due to virus. It is common type of pneumonia in children. Viral pneumonia also includes flu & almost responsible for one-third cases of pneumonia worldwide.
iii. Fungal pneumonia: Fungal pneumonia is the infection caused due to fungi. It is a rare pneumonia. Fungal pneumonia always affect the patients with congenital or acquired defects in the host immune system.

1.1.3 Pneumonia Diagnosis

At the time of diagnosis doctors are majorly asking about the symptoms mentioned above. They also do some physical examination like listening your lungs sound using stethoscope or giving order for Chest X-ray. On the basis of your condition doctors may also order for more than one type of tests like:

i. Blood Test: It is an oxygen level test, in which we check for the amount of oxygen in bloodstreams.
ii. Sputum Test: In this test, pathologist test for the mucus which come out of the lungs.
iii. Pulse oximetry:
iv. Urine Test (Testing for the bacteria in urine)
v. CT Scan (a detailed Chest X-ray)
vi. Fluid sample (Taking sample from ribs & testing on the sample)
vii. Bronchoscopy (Testing for the airways in the lungs)

1.2 Machine Learning & Deep Learning

1.2.1. **Machine learning**

Machine Learning is one of five parts of Artificial Intelligence, where we provides computer system the ability to learn automatically & improve themselves through experience. Machine learning is always focuses on developing computer programs that can import data & learn through that data.

Primarily machine learning is categorized into 3 parts:

- Supervised Machine Learning: Supervised machine learning can be only applied where we have some prior knowledge about data, or we have information to be extracted from data. In supervised machine learning, we are always trying to create a decision boundary between the data itself. Supervised machine learning problems can be further divided into Classification problems & Regression problems.

Classification problems are the problem where we have to select one category from a set of categories for e.g. when we see a picture of Cat, our brain automatically recognizes that the animal in the picture is a cat. Similarly, we have regression problem, where we are dealing with numerical values. For e.g. when we going to buy some product we always approximate or estimate the price of the product before purchasing & make decisions accordingly. All these kind of prediction & classification in our brain, we can do using our prior knowledge about the thing we are dealing with. To tackle such problems we have different algorithms like SVM (Support Vector Machine), Logistic Regression, Linear Regression, Decision Trees, Regularized Linear Regression, Naïve Bayes Classification etc.

- Unsupervised Machine Learning: In contrary, unsupervised machine learning is used when we don't have knowledge about final results, but we want to infer some information from the data. & make decisions on the basis of the inference made. In Unsupervised machine learning, we trying to figure out the hidden structures from the data which is not labelled. Unsupervised machine learning problems are further categorized into Clustering problems & Association Analysis problems. Clustering problems are the problems where we have to create clusters of similar data. For e.g. a departmental store created a system to identify different products to store them separately. That system have no prior experience with the product but once it starts finding unique patterns to separately identify the product. It will create separate cluster for different type of products. Similarly, in associative analysis problems are the problems in which we discover some rules that describes large section of data. Associative analysis majorly is used in recommendation systems & market analysis. For e.g. A departmental store wants to increase the monthly sales. So, they will arrange the products according to the previous choice of the customers. To tackle such problems we have different algorithms like PCA, K-Means, Apriori, F-tree etc. Unsupervised Machine learning also use in reducing dimension of data. At the time of data collection, we collect many features, all are necessary but when we trying to solving the problem we get some values which become constant or not playing a huge role in the operation. To reduce the dimension of the data we majorly use Principle Component Analysis (PCA).
- Semi-supervised Machine Learning: This type of machine learning problem fall between supervised & unsupervised learning. Semi-supervised machine learning algorithms use both labelled & unlabelled data for training. Semi-supervised machine learning algorithms use small proportion of labelled data & large proportion of unlabelled data. Semi-supervised machine learning uses in speech analysis, protein sequence classification etc.
- Reinforcement Machine Learning: It is a type of machine learning which also known as adaptive learning. In reinforcement learning, the system (model) will produce actions, find & analyse errors. Reinforcement learning deals which behaviour of the task & produce actions accordingly.

1.2.2 **Deep learning**

Deep learning is a subset of machine learning, which is aspired by the structure & functionality of the brain. Our brain having neurons which are connected completely with

each other. Similarly, we are creating artificial neurons, & the connection of these neurons known as artificial neural network. Different type of Neural Networks:

- Convolutional Neural Network: A convolutional network is a class of deep learning which most commonly used in visual imagery. Convolutional Neural network having a wide range application in image recognition, video recognition, recommender system & also use in Natural language processing.
- Recurrent Neural Network: A recurring network is a class of deep learning where we use the output from previous epoch to feed as the input to the current epoch. A recurrent network is follow a traditional approach of learning from experience.

Similarly, we are dealing with data we always divide data into 3 parts:

- Training Data: This is the largest proportion of data. We use it to train the model.
- Testing Data: This data is the small proportion of whole data.
- Validation Data: This is the data use to checking the parameters.

The proportion of data majorly like Training data (60%), testing data (20%) & validation data (20%)

1.3 System Architecture

The general architecture of any Machine Learning algorithm

My model is an 18 layer convolutional neutral networks that inputs one x – ray image of chest of a pre-defined size of 224 x 224 size & returns a probability of every class. I trained on a recently released Chest X-Ray dataset which was procured from Data Science Platform Kaggle which currently contains 5000 front view chest X-Ray images, which are individually classified with 3 different labels (Normal, Bacterial pneumonia & Viral Pneumonia).

Architecture of any Neural Network can be broadly divided into three major sub categories:

1) Input Layer
2) Hidden or /middle Layer
3) Output Layer

Input Layer basically pre-processes the data into chunks that can be easily broken down & processed without the loss of valuable information & the data is then fed into the next layer which is namely. Hidden layer or Middle Layer. This layer further contains hundreds of layers which are made up of graph & nodes whose main purpose is to extract features in each step, basically the data from the Input layer is processed & it moves through the hidden layers, new features being extracted at each stage & at the finally at the last layer , all the features are extracted & compressed into a feature matrix to be fed into output layer (i.e next layer). Output layer then analyses the feature matrix after which the model makes a calculated decision with probability which dictates whether the input x-ray image belongs to either of the resultant classes. (Pneumonia detected or not)

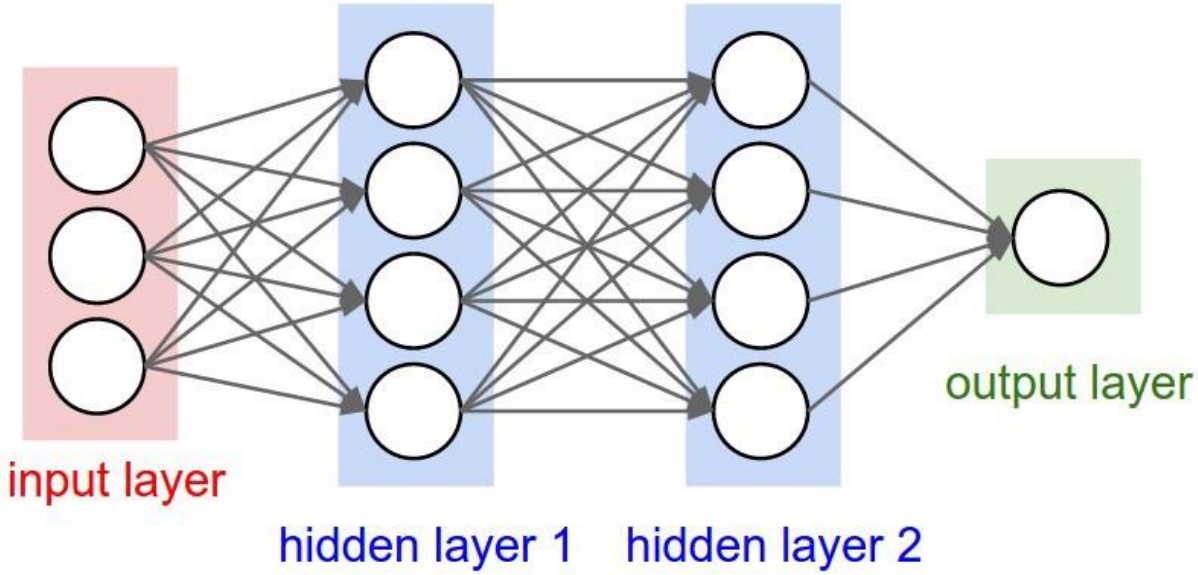

Fig 1.2 Architecture of Deep Neural Network

1.3.1 Merits & Demerits of the Proposed Architecture.

Models inherently do not possess any disadvantages or problems, actual issue arises due to the data & the type of data that is being fed into the system. This causes a bias in the system, generally classified into a high & low bias which dictates how the model will perceive your input & accommodate a result based on the input. If the data being fed into the system is wrong or incomplete this will result in high bias & if the data is incomplete or which the model has never seen before, or doesn't know what to do with it will result in low bias.

This bias can be partially or completely removed by properly normalizing the data, reduce the features to be extracted etc. an equilibrium can be achieved with a trade between high variance dataset which would result in a low error margin for the result as the model have an easier time computing the feature extractions.

Extreme fluctuations in data set would model to get in a situation called as under-fitting &/ or overfitting. A model results in under-fitting when the variance is low & the bias is hig in the underlying data set which results in model being confused where to bisect the training & testing error into parts efficiently, over fitting is correspondingly high variance & low bias where model works too good for training data & not great for the testing data & thus hampers the end result.

1.4 Problem Statement

Pneumonia is the most rampant lung disease that has engulfed people of all ages & genders since a millennia. With recent developments in technology it has become fairly easy to diagnose & treat pneumonia, since the most third world counties have populations with access to a smartphone more than they have to a healthcare, this project comes down to giving user access & a 96 percent accuracy margin to check if a person has pneumonia or not. Saving user, time, money & an overall great precautionary measure.

Pneumonia is swelling (inflammation) of a tissue in one or both lugs. It's usually caused by a bacterial infection. At the end of the breathing tubes in your lungs are a group of tiny air pockets. If you happen to have pneumonia, these little pockets would become inflamed & filled up with fluids.

These symptoms move over a quite a time of from 24 to 48 hours, but slowly they tend to come over the next several days. It can affect anyone, but can be lethal for anyone who happens to be very young or very elderly, who happen to be the core audience for this project. They would need immediate hospital treatment if they develop pneumonia.

The Risk groups:

Some particular groups happen to have an increased risk of developing pneumonia:

1) Infants
2) Elderly people
3) People with other terminal health conditions
4) People with smoking habits
5) People with weakened immune systems

Pneumonia can be difficult to diagnose because it shares many symptoms with other common conditions, such as common cold, asthma etc. if you have breathlessness or at a faster pace than usual, colour of the mucus, if the pain I swore while inhaling or exhaling.

An X-ray is an excellent way of providing fast access way of procuring knowledge if you have pneumonia or not. The key is to scan the x ray image with the android app associated with the

Chapter 2

Background

Computer aided diagnostics tools have the main aim to aid the clinical process of decision making, they add to the components of Computer Vision & A with radio magnetic imaging & processing to recognize patterns. Much of the established literature described in most machine learning papers are that most of the algorithms are knitted to prepare for very specific problems. In recent years though deep learning, methods are chosen to avoid the issues with the h&crafted features through end-to-end feature extraction & classification.

Basic Working: (work needed)

1) INPUT
2) CONV LAYER
3) RELU
4) POOL
5) FC

In this process, the convolutional network transforms the input image layer by layer from original pixel values to the final class scores. Each layer inputs a 3d volume & transforms into an output 3d volume through a differentiable function. Each layer may or may not have parameters for instance Conv & Fully Connected Layers do have parameters, Relu an Pool layers do not possess any parameters.

2.1 Sequential Model:

A sequential CNN model is one of the class of deep, feed-forward artificial neural networks that are commonly applied to visual recognition. It is a called linear stack of convolutional, pooling & dense layer. We optimized the sequential CNN architecture & its hyper parameters for the datasets under study through the Bayesian learning. This process uses a Gaussian processing model of an objective function & its evaluation to optimize the network depth, learning rate, momentum, & variable regularization. The objective function accepts these arguments inputs, & which trains, validates & saves the optimal network that gives the minimum possible optimal solution for the network that gives the minimum classification error on the test data.

2.2 Convolutional Layer:

CNNs, like neural networks, are made up of neurons with learnable weights & biases. Each neuron receives several inputs, takes a weighted sum over them, pass it through an activation function & responds with an output. The whole network has a loss function & all the tips & tricks that we developed for neural networks still apply on CNNs.

To understand the convolutional part of the convolutional neural network we have to understand what ‘convolution’ means exact when an image goes through the convolutional part of the image filter for e.g. a filter of dimensions 5x5x3 which is then slid over the complete image & along the way dot product is taken to quantify a result which is chunks of the input image. Each dot product results in a scalar value. So when we ‘convolve’ the complete image with this filter, it results in a 28x28x1 image.

Convolution layer makes up the main building block of a convolutional neural network, the convolutional layer comprises of a bunch of nondependent filters, & each of these filter is independently convolved with the input image & it results in feature maps sizes 28x28x1.all these filters are initialized randomly & become the parameters which will be 'learned' by the our CNN subsequently.

ConvNet is a sequence of layers, & every layer of a ConvNet transforms one volume of activations to a different one through a series of a differentiable functions. We mainly use the main types of layers to build convolutional architectures namely, Convolutional Layer, Pooling layer & a fc layer (Fully Connected Layer), these are stacked side by side in sequence to result in what we call a Convnet architecture.

2.3 Pooling Layer:

It has been a popular practice to insert a pooling layer in between two consequent conv layers in a convent architecture, its main function is to actively reduce the spatial size of the representation to reduce the amount of parameters & computations needed in the network & thus controlling overfitting in the model. Thus the pooling layer operates independently on every deep part of the input & resizes it spatially, using MAX operation.

- Accepts a volume of size W1×H1×D1W1×H1×D1
- Requires two hyper parameters:
  - their spatial extent FF,
  - the stride SS,
- Produces a volume of size $W2 \times H2 \times D2W2 \times H2 \times D2$ where:
  - $W2 = (W1 - F)/S + 1W2 = (W1 - F)/S + 1$
  - $H2 = (H1 - F)/S + 1H2 = (H1 - F)/S + 1$
  - $D2 = D1D2 = D1$
- Introduces zero parameters since it computes a fixed function of the input
- For pooling layers, it is not common to pad the input using zero-padding.

It is worth noting that there are only two commonly seen variations of the max pooling layer found in practice: A pooling layer with $F = 3, S = 2F = 3, S = 2$ (also called overlapping pooling), & more commonly $F = 2, S = 2F = 2, S = 2$. Pooling sizes with larger receptive fields are too destructive.

2.3.1 Types of Pooling

Before getting into the details, you should know that there are several types of pooling. These include among others the following:

- Mean pooling
- Max pooling
- Sum pooling

Our main focus here will be max pooling.

i. Max pooling:

Max-pooling can be called as a down sampling strategy in Convolutional Neural networks.

Pooling is a very simple operation which sub-sample or down-sample the input shape by replacing a very small sub-array (input array) with a single value & that single value is usually the summary statistics of that small sub-array. We can use any summary statistics to replace it with a single number & that can be done by taking the maximum value from the sub-array, known as Max-Pooling, or we can be taking the average of values of the sub-array, known as Average-Pooling. We will make use of the pooling operation to down sample the size of input volume but there's one more interesting thing which pooling does also. In pooling, we are replacing small sub-array with a single value means we are not that much focusing on the exact location of the result of the feature detector operation. Instead, we are zooming out a little bit & asking for an approximate result of that sub-array. This will make our image classifier robust to **local translation invariant**.

$$\max(x_1, x_2, \dots, x_n) = d \to +\infty\left(\sum i = 1n x_{di}\right)_{1d}$$

In more mathematical terms we can call max-pooling a sample based discretization process. The main objective is to down sample an input representation ad further reducing its dimensionality & allowing it to become freer in its approach. Then we can make assumptions about the features contained in the sub region binned.

Let's say we have a 4x4 matrix representing our initial input. Let's say, as well, that we have a 2x2 filter that we'll run over our input. We'll have a **stride** of 2 (meaning the (dx, dy) for stepping over our input will be (2, 2)) & won't overlap regions.

For each of the regions represented by the filter, we will take the **max** of that region & create a new, output matrix where each element is the max of a region in the original input.

2.4 Backpropagation:

Backpropagation is the central mechanism by which neural networks learn. It is the messenger telling the network whether or not the net made a mistake when it made a prediction.

To propagate is to transmit something (light, sound, motion or information) in a particular direction or through a particular medium. When we discuss backpropagation in deep learning, we are talking about the transmission of information, & that information relates to the error produced by the neural network when they make a guess about data.

Untrained neural network models are like new-born babies: They are created ignorant of the world, & it is only through exposure to the world, experiencing it, that their ignorance is slowly revised. Algorithms experience the world through data. So by training a neural network on a relevant dataset, we seek to decrease its ignorance. The way we measure progress is by monitoring the error produced by the network.

The knowledge of a neural network with regard to the world is captured by its weights, the parameters that alter input data as its signal flows through the neural network towards the net's

final layer that will make a decision about that input. Those decisions are often wrong, because the parameters transforming the signal into a decision are poorly calibrated; they haven't learned enough yet. A data instance flowing through a network's parameters toward the prediction at the end is forward propagation. Once that prediction is made, its distance from the ground truth (error) can be measured.

So the parameters of the neural network have a relationship with the error the net produces, & when the parameters change, the error does, too. We change the parameters using an optimization algorithm called gradient descent, which is useful for finding the minimum of a function. We are seeking to minimize the error, which is also known as the loss function or the objective function.

Output layer error: We need to find a derivative to calculate the error at output layer. The derivative is the difference between true output & predicted output which later multiply with activation function.

$Cos(V_0) = (\hat{y} - y)R'(V_0)$

Where, $\hat{y}$ is the predicted result

y is true result

$$R'(Z) = \begin{Bmatrix} 0 & Z < 0 \\ 1 & Z > 0 \end{Bmatrix}$$

Hidden layer error: To get the error of hidden layer we have to calculate the value of derivative

$C'(Zh)=(y-y)\cdot R'(Zo)\cdot Wo\cdot R'(Zh)$

Calculated errors

Output Layer Error	$E_o = (O - y) \cdot R'(Z_o)$
Hidden Layer Error	$E_h = E_o \cdot W_o \cdot R'(Z_h)$
Cost-Weights Deriv	$LayerError \cdot LayerInput$

2.5 FC Layer:

The output from the convolutional layers represents high-level features in the data. While that output could be flattened & connected to the output layer, adding a fully-connected layer is a (usually) cheap way of learning non-linear combinations of these features.

Essentially the convolutional layers are providing a meaningful, low-dimensional, & somewhat invariant feature space, & the fully-connected layer is learning a (possibly non-linear) function in that space.

We can divide the whole network (for classification) into two parts:

- **Feature extraction**: In the conventional classification algorithms, like SVMs, we used to extract features from the data to make the classification work. The convolutional layers are serving the same purpose of feature extraction. CNNs capture better representation of data & hence we don't need to do feature engineering.
- **Classification**: After feature extraction we need to classify the data into various classes, this can be done using a fully connected (FC) neural network. In place of fully connected layers, we can also use a conventional classifier like SVM. But we generally end up adding FC layers to make the model end-to-end trainable

2.6 Gradient Descent

A gradient is a slope whose angle we can measure. Like all slopes, it can be expressed as a relationship between two variables: 'y over x', or rise over run. In this case, the y is the error produced by the neural network, & x is the parameter of the neural network. The parameter has a relationship to the error, & by changing the parameter, we can increase or decrease the error. So the gradient tells us the change we can expect in y with regard to x.

To obtain this information, we must use differential calculus, which enables us to measure instantaneous rates of change, which in this case is the tangent of a changing slope expressed the relationship of the parameter to the neural network's error.

Obviously, a neural network has many parameters, so what we're really measuring are the partial derivatives of each parameter's contribution to the total change in error.

What's more, neural networks have parameters that process the input data sequentially, one after another. Therefore, backpropagation establishes the relationship between the neural network's error & the parameters of the net's last layer; then it establishes the relationship between the parameters of the neural net's last layer those the parameters of the second-to-last layer, & so forth, in an application in chain rule of calculus.

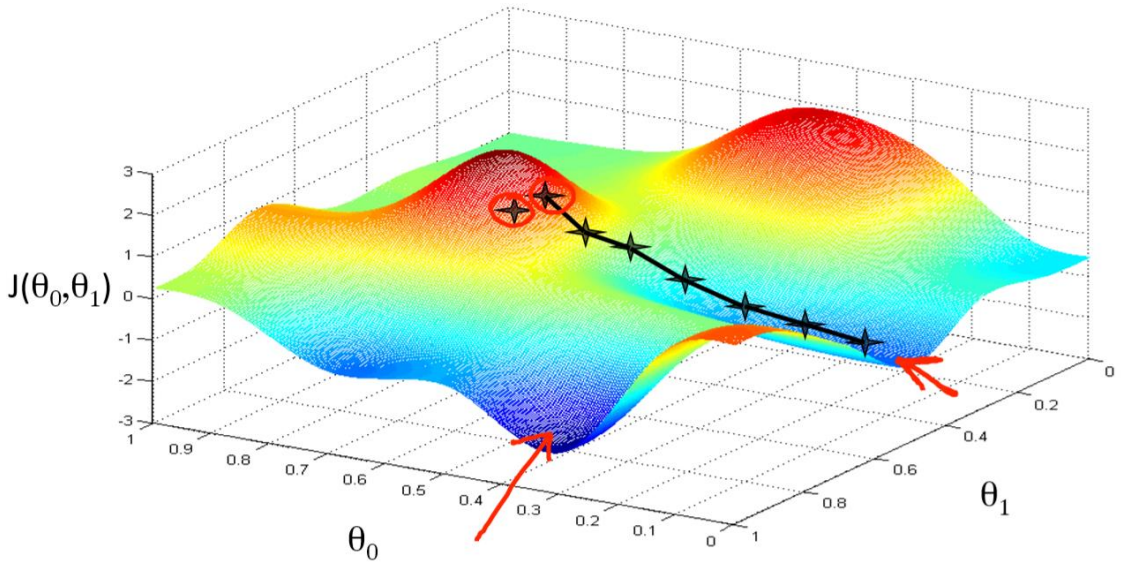

Fig 2.1 learning Model of Gradient Descent

2.7 Batch normalization in Neural Networks

We normalize the input layer by adjusting & scaling the activations. For example, when we have features from 0 to 1 & some from 1 to 1000, we should normalize them to speed up learning. If the input layer is benefiting from it, why not do the same thing also for the values in the hidden layers, that are changing all the time, & get 10 times or more improvement in the training speed.

To facilitate learning, we typically normalize the initial values of our parameters by initializing them with zero mean & unit variance. As training progresses & we update parameters to different extents, we lose this normalization, which slows down training & amplifies changes as the network becomes deeper.

Batch normalization re-establishes these normalizations for every mini-batch & changes are back-propagated through the operation as well. By making normalization part of the model architecture, we are able to use higher learning rates & pay less attention to the initialization parameters. Batch normalization additionally acts as a regularize, reducing (& sometimes even eliminating) the need for Dropout.

In this layer, we are performing the simple operation of converting each element into the z-score.

$$Z = \frac{(x - u)}{\sigma}$$

This function will be applied on the whole batch for normalizing the output of previous layer.

Batch normalization reduces the amount by what the hidden unit values shift around covariance shift. In other words, if an algorithm learned some X to Y mapping, & if the distribution of X changes, then we might need to retrain the learning algorithm by trying to align the distribution

of X with the distribution of Y. Also, batch normalization allows each layer of a network to learn by itself a little bit more independently of other layers. We can use higher learning rates because batch normalization makes sure that there's no activation that's gone really high or really low. & by that, things that previously couldn't get to train, it will start to train. It reduces overfitting because it has a slight regularization effects. Similar to dropout, it adds some noise to each hidden layer's activations. Therefore, if we use batch normalization, we will use less dropout, which is a good thing because we are not going to lose a lot of information. However, we should not depend only on batch normalization for regularization; we should better use it together with dropout.

2.8 Dropout Regularization:

When the neurons are dropped out, the network is somehow forced to learn several independent representations of the patterns with identical input & output. This thus has a big effect on generalizability, improving in a great matter.

Simply put the dropout process can be called as to ignoring neurons during the training phases of certain set of neurons which is chosen at random, ignoring is used here in the sense that these will not be used during a particular forward or backward pass.

Dropouts can be called as the process of shutting down parts of a neural networks, which comes in handy with the result. As we know that eh fully connected layer occupies most of the parameters, & hence neurons develop a certain sense of co-dependency amongst each there during the training period which curbs the individual power of each neuron leading to over fitting of training data. Regularization is way to prevent over-fitting. Regularization reduces over-fitting by adding a penalty to the loss function.

By adding this penalty, the model is trained such that it does not learn interdependent set of features weights. Dropout forces a neural network to learn more robust features that are useful in conjunction with many different random subsets of the other neurons.

Dropout roughly doubles the number of iterations required to converge. However, training time for each epoch is less.

We can drop 20%-50% of the real values. And 20% can be a great start. Because if we drop more than that without understanding the data, we may lead to under fitting.

Dropout layer can provide good result over the visible and hidden layers.

Using a larger momentum can help in increasing the learning rate.

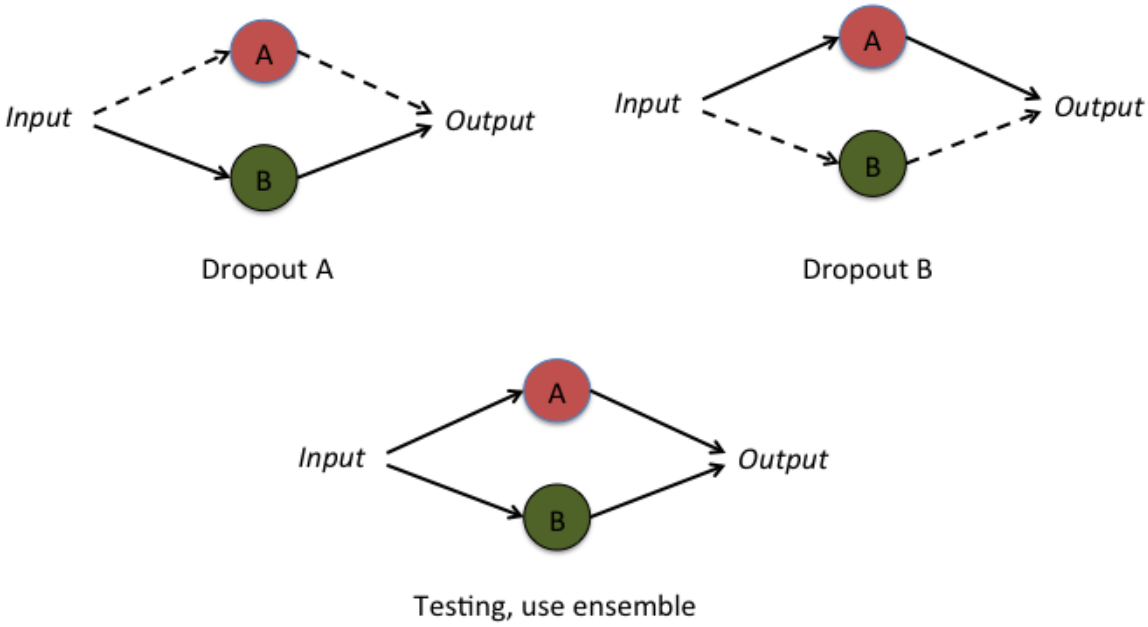

Fig 2.2 Dropout Regularization Workflow

2.9 Adam Optimization:

Adaptive Moment Estimation (Adam) is another method that computes adaptive learning rates for each parameter. In addition to storing an exponentially decaying average of past squared gradients $vtvt$ like Adadelta & RMSprop, Adam also keeps an exponentially decaying average of past gradients $mtmt$, similar to momentum. Whereas momentum can be seen as a ball running down a slope, Adam behaves like a heavy ball with friction, which thus prefers flat minima in the error surface .We compute the decaying averages of past & past squared gradients $mtmt$ & $vtvtrespectively$ as follows:

$$mt = \beta 1mt - 1 + (1 - \beta 1)gtvt = \beta 2vt - 1 + (1 - \beta 2)g2tmt$$
$$= \beta 1mt - 1 + (1 - \beta 1)gtvt = \beta 2vt - 1 + (1 - \beta 2)gt2$$

mtmt & vtvt are estimates of the first moment (the mean) & the second moment (the uncentered variance) of the gradients respectively, hence the name of the method. As mtmt & vtvt are initialized as vectors of 0's, the authors of Adam observe that they are biased towards zero, especially during the initial time steps, & especially when the decay rates are small (i.e. β1β1 & β2β2 are close to 1).

They counteract these biases by computing bias-corrected first & second moment estimates:

$$mt = mt1 - \beta t1_{vt} = vt1 - \beta t2m_t = mt1 - \beta 1tv_t = vt1 - \beta 2t$$

They then use these to update the parameters just as we have seen in Adadelta & RMSprop, which yields the Adam update rule:

$$\theta t + 1 = \theta t - \eta\sqrt{vt} + \epsilon mt\theta t + 1 = \theta t - \eta v_t + \epsilon m_t.$$

The authors propose default values of 0.9 for β1β1, 0.999 for β2β2, & 10−810−8 for ϵϵ. They show empirically that Adam works well in practice & compares favorably to other adaptive learning-method algorithms.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$
$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t - \eta w_t \theta_t$$

How it works:

1. First, it calculates an exponentially weighted average of past gradients, & stores it in variables V_{dW} & V_{db}(before bias correction) & $V_{dWcorrected}$ & $V_{dbcorrected}$ (with bias correction).
2. Then it calculates an exponentially weighted average of the squares of the past gradients, & stores it in variables S_{dW} & S_{db} (before bias correction) & $S_{dWcorrected}$ & $S_{dbcorrected}$ (with bias correction).
3. Finally updates parameters in a direction based on combining information from '1' & '2'.

2.9 Softmax :

Softmax, also known as normalized exponential function, softargmax is a function whose main purpose is to take an input vector of k real numbers, & normalize it into a probability distribution of k probabilities each. Way before softmax is applied, some of the vectors might have a wide range of values from negative, or greater than one & also might not sum up to 1. But after applying softmaxeach component will have values ranging in the interval (0,1) & the components will sum up to 1, so they can be used & interpreted as probabilities. Moreover, the larger input components will correspond to larger probabilities. Softmax is used to map non – normalized output of a network provavility distribution over predicted output classes.

$$\frac{\partial}{\partial q_k}\sigma(\mathbf{q}, i) = \cdots = \sigma(\mathbf{q}, i)(\delta_{ik} - \sigma(\mathbf{q}, k))$$

Such functions commonly use the neural network, logistic regression multiclass classification, provide a variation of non-linear logistic regression.

$$P(y = j \mid \mathbf{x}) = \frac{e^{\mathbf{x}^\mathsf{T}\mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^\mathsf{T}\mathbf{w}_k}}$$

The above function is the composition of linear function. This operation is equal to applying the linear dot product of the weight vector w and feature vector x. The same soft max function also use in reinforcement learning.

2.10 Proposed Model:

The model currently contains of a dataset of 5000 images of pneumonia & non-pneumonia affected lungs, these images are what is fed into the model to make the algorithm understand the subtle nuances of a lung infected with pneumonia. the system is trained with volume of data with features extracted in such a niche degree that when a test image is passed the complexities of Convolutional Neural network jumps into action to found out if the given test image has any resemblance to the pneumonia affected lungs thus giving us some accurate representation whether the image is of affected patient or not, thus saving time & money to an extensive degree.

Chapter 3

Literature Survey

[1] In this paper, they designed an algorithm that can detect pneumonia & other 14 chest related diseases using Chest X-Rays. They created an application to take the X-ray data & transfer that image directly to cloud servers, where there algorithm called "CheXNet" works on that image data & give percentage of chances of having pneumonia. They are using a 121- layered CNN for detecting the pneumonic symptoms. There accuracy on pneumonia is "0.7680".

[2] In this paper, they created an algorithm to identify the pneumonic patterns. They used three different neural networks. Where first network is use to take Image input, transform the image & give output to next network. They used R-CNN with LSTM with 16 to 32 layers. Model accuracy for pneumonia recognition is "0.713".

[3] They created an algorithm which will analyse a 3D CT scan. In the algorithm they used CNN with bidirectional LSTM network. They use 3 DenseNet Networks to analyse the image by using slice index content. Output of every layer is a segmented image. Every DenseNet is a 40-layered architecture. In the final layer, output of all global pool layers is passes through a single bidirectional LSTM before the generating final output at connection layer. Accuracy of RADNet is "81.82%".

[4] In this paper, they created multiple models for predicting the mortality of a pneumonia patient. They have a dataset with the 63 feature for each patient & total of 15000 patients. They used 8 different algorithms like Logistic Regression, HME (Decision Tree), K2MB (Bayesian Network), PCLR (Bayesian Network), Simple Bayesian Model, RL/OP (Rule-based model), Neural Network, K-Nearest Neighbours. Where they found that Neural Network, HME (Decision Tree) & Logistic regression provide a very lowest error rate for each fps level.

[5] In his thesis, he approach to the problem using two MLPs (Multi Layered Perceptrons) because he have very small dataset. So, to approach the problem he used Machine Learning's algorithm like Random Forest with naïve approach & got an accuracy of 61.8%. For MLP-a he got the accuracy of 44.26%. He performed these models on ECG scans.

[6] In this paper, they introduced a new chest X-ray database, named "ChestX-ray8" which have total of 108948 X-ray images of total 32717 unique patients. They used deep convolutional neural network for classification & diseases localization framework. It provided 63.33% of accuracy on pneumonia detection.

[7] In this paper, they planned a calculation that can identify pneumonia and other 14 chest related sicknesses utilizing Chest X-Rays. They made an application to take the X-beam information and exchange that picture legitimately to cloud servers, where there calculation called 'CheXNet' takes a shot at that picture information and give level of odds of having pneumonia. They are utilizing a 121-layered CNN for identifying the pneumonic indications. There precision on pneumonia is '0.7680'.

[8] In this paper, they made a calculation to distinguish the pneumonic examples. They utilized three diverse neural systems. Where first system is use to take Image input, change the picture and offer yield to next system. They utilized R-CNN with LSTM with 16 to 32 layers. Show precision for pneumonia acknowledgment is '0.713'.

[9] They made a calculation which will dissect a 3D CT output. In the calculation they utilized CNN with bidirectional LSTM organize. They utilize 3 DenseNet Networks to investigate the picture by utilizing cut record content. Yield of each layer is a portioned picture. Each DenseNet is a 40-layered design. In the last layer, yield of all worldwide pool layers is goes through a solitary bidirectional LSTM before the creating last yield at association layer. Precision of RADNet is '81.82%'.

[10] In this paper, they made different models for foreseeing the mortality of a pneumonia understanding. They have a dataset with the 63 include for every patient and aggregate of 15000 patients. They utilized 8 unique calculations like Logistic Regression, HME (Decision Tree), K2MB (Bayesian Network), PCLR (Bayesian Network), Simple Bayesian Model, RL/OP (Rule-based model), Neural Network, K-Nearest Neighbors. Where they found that Neural Network, HME (Decision Tree) and Logistic relapse give a least blunder rate for every fps level.

[11] In his theory, he way to deal with the issue utilizing two MLPs (Multi Layered Perceptrons) on the grounds that he have little dataset. In this way, to approach the issue he utilized Machine Learning's calculation like Random Forest with innocent methodology and got a precision of 61.8%. For MLP-a he got the exactness of 44.26%. He played out these models on ECG filters.

[12] In this paper, they presented another chest X-beam database, named 'ChestX-ray8' which have aggregate of 108948 X-beam pictures of all out 32717 one of a kind patients. They utilized profound convolutional neural system for arrangement and sicknesses confinement structure. It gave 63.33% of precision on pneumonia location.

Chapter 4:

System Design & Methodology:

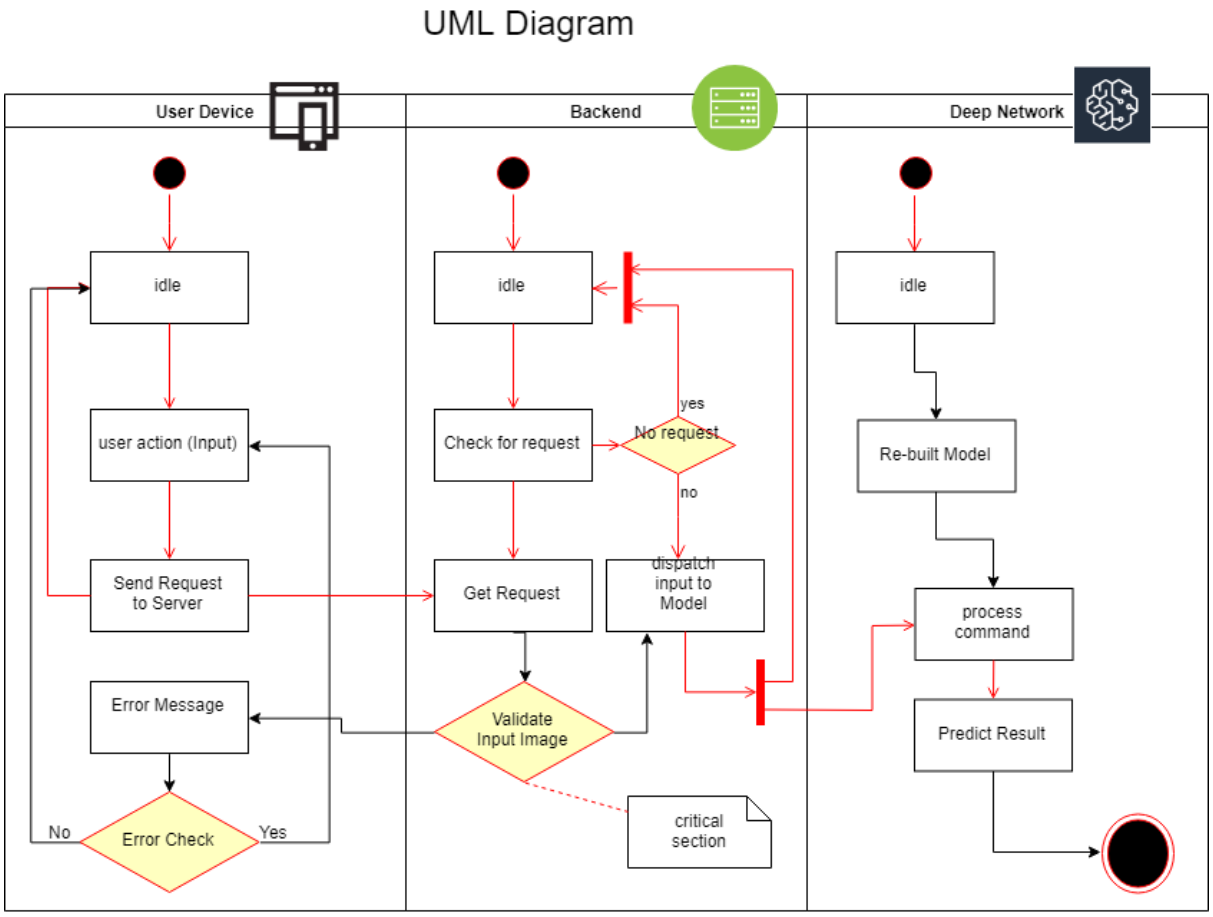

Fig 4.1 UML Diagram of RJxNet

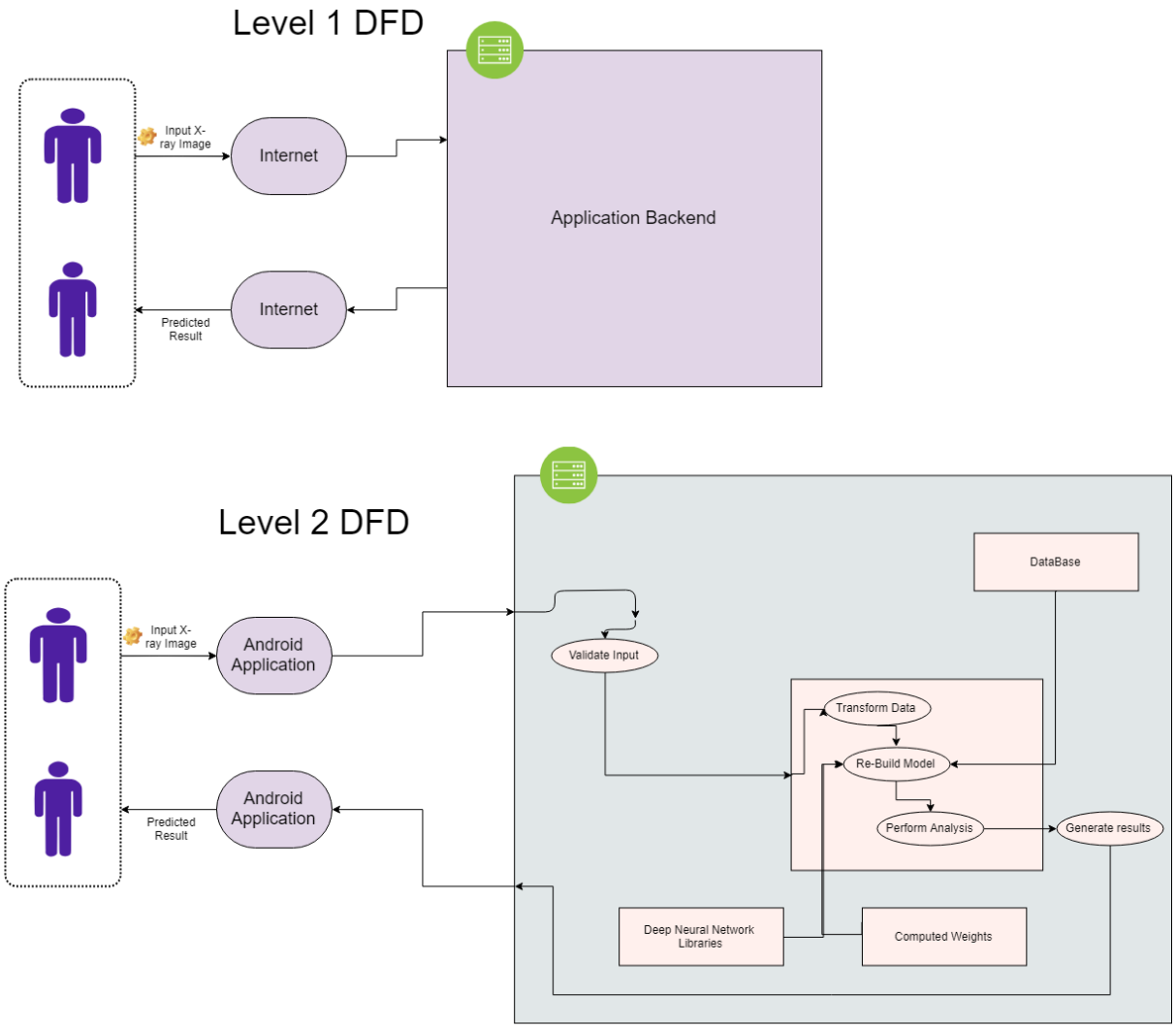

Fig 4.2 Data Flow Diagram of RJxNet

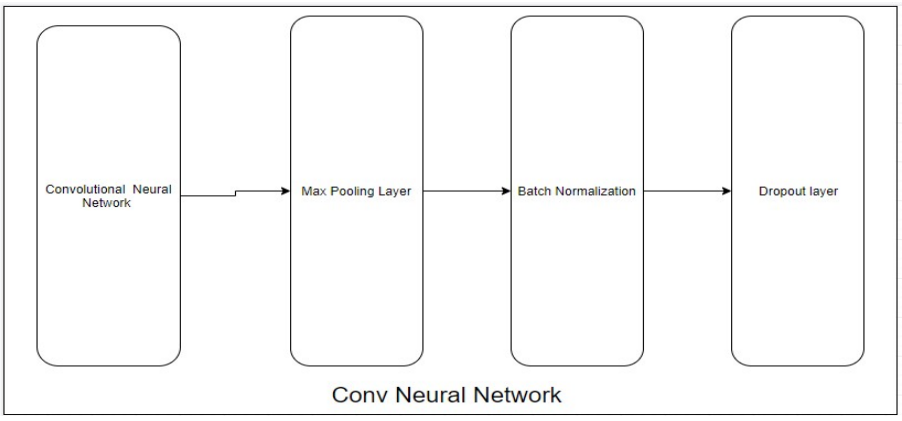

Fig 4.3 Convolutional Neural Network

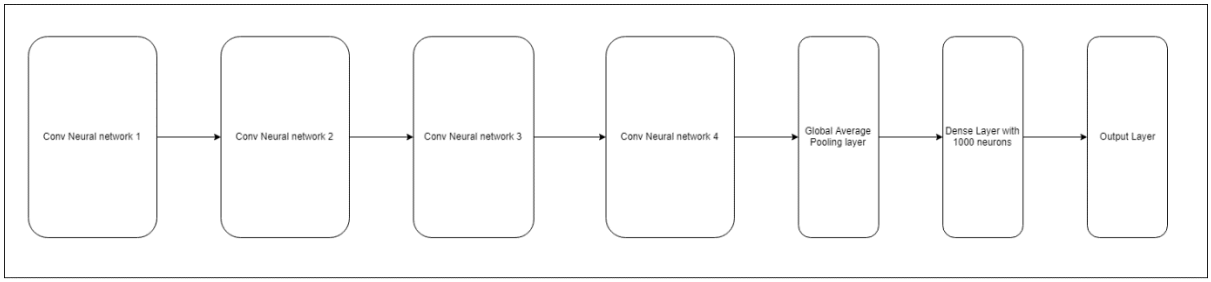

Fig. 4.4 Deep Neural Network Model

Most of this project is divided into three big chunks:

1) Front End (this module mostly consists of an interface which the common user can access to use & retrieve results)
2) Back End = which consists of a frozen model of the highly specific high result yielding neuron model of the current model which results in maximum percentage of accuracy)
3) A pipeline between the two which processes the information across the two modules.

4.1 Front End:

the front end is largely consisted of an android app which consists of an interface with the power for the user to log in information where there easy plugin for users to take a live picture or take a picture from the gallery from which the system will scan the x ray image & send it onto the back end. The application is regulated by the android material design mettle design which helps in better user interface for any level of user to use the app in an easier way. Be it a novice or a seasonal user, the usability of the application is of the highest priority as the target audience for this project is a layman user.

Fig 4.4 Android Application Main Page

4.2 Back End:

The Back End is comprised of the frozen model of the highest accuracy yielding combination of neurons with the weights assigned to them by constant forward & back propagation resulting in a model which gives the upmost accuracy margin. Back end is written in the high level most popular language of python, with certain packages from keras & tensorflow used to validate & deploy different complex machine learning algorithms, different activation functions, kernels, filters, matrices were used to make an equilibrium of neural network which yields accurate results most of the times.

4.3 Pipeline:

Network Layer interaction & the passing of arguments of between the above mentioned modules were handled by the high level language of kotlin supported by the android Studio 11 & above, its interoperability helped in leaps & bounds for the seamless inter connection between native & binary code. The code in python was compiled & frozen in the system while the kotlin code carried arguments on to the function to fetch results & carry it back to the front end to give users an account of the resultant matrix.

From an abstract point of view we can further divide the process into sub categories:

1) Setting up the system
2) Data extraction
3) Pre processing
4) Filtering
5) Classification
6) Neural network
7) Result

1) Setting up the System:

This project was made into mind that the making future updates to this project would be fairly easy, the kotlin back end code base supports dynamic updation of core logic remotely from a central server. This makes the updation of model when the possibility of improved analysis is possible it would be easier to port over to new code. The user need not do anything & the app will make known the possibilities of a false positive, which happens when the model isn't sure about the result & recommends a physical check-up from a trained professional or a doctor.

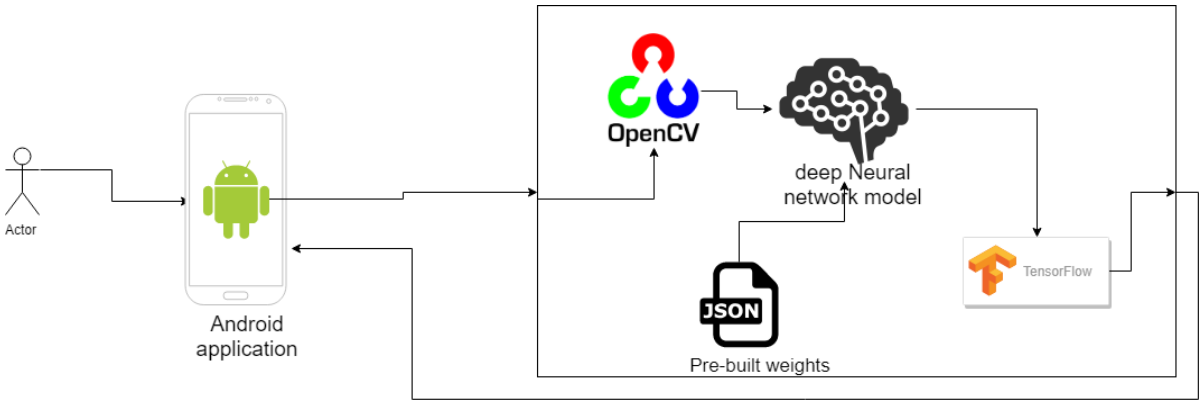

Fig 4.5 System architecture of RJxNEt

2) Data Extraction

This project requires the dataset which is related with pneumonia. We are using the chest X-rays to create the classification model. Because chest X-rays can provide a very clear picture about the symptoms of pneumonia. We used the global dataset of Chest X-ray images which is available on kaggle & other data source websites.

Our data is divide into the three class one is normal chest X-ray, effected by Virus chest X-ray & effected by bacteria chest X-ray. We have dataset splitted into training & testing dataset.

Type	Training Data	Validation Data	Testing Data	Total
Normal X-ray	546	137	66	749
Pneumonia X-ray	3100	775	390	4265
Total	3646	912	456	5014

Table 4.1 Dataset Distribution table

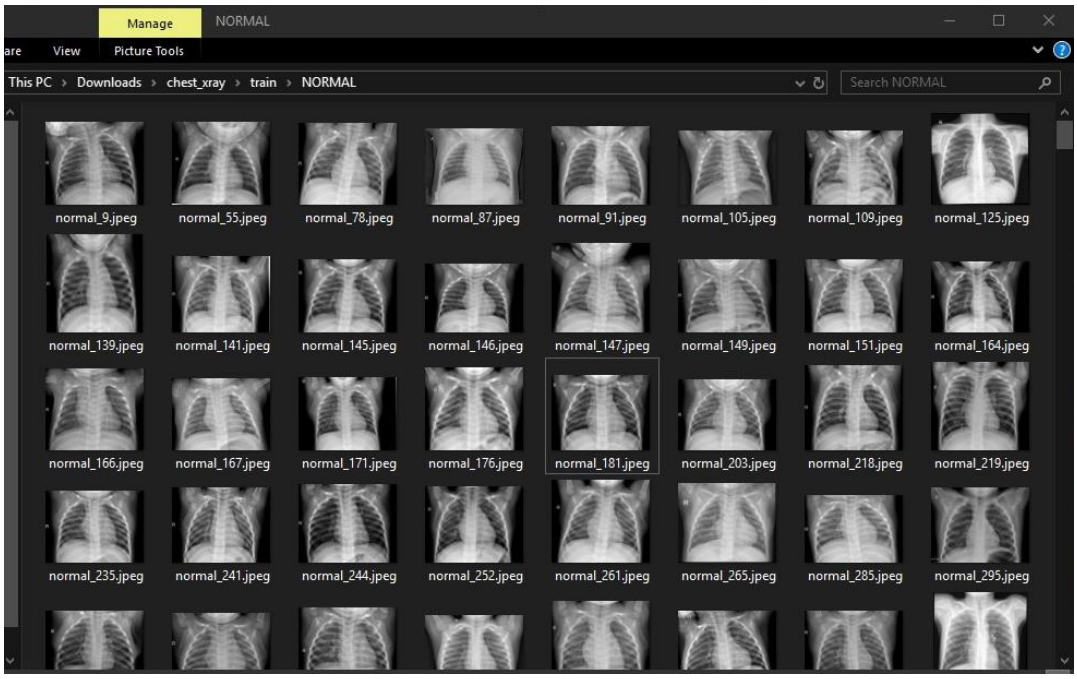

Fig 4.6a Images of Normal Patients Chest X-rays

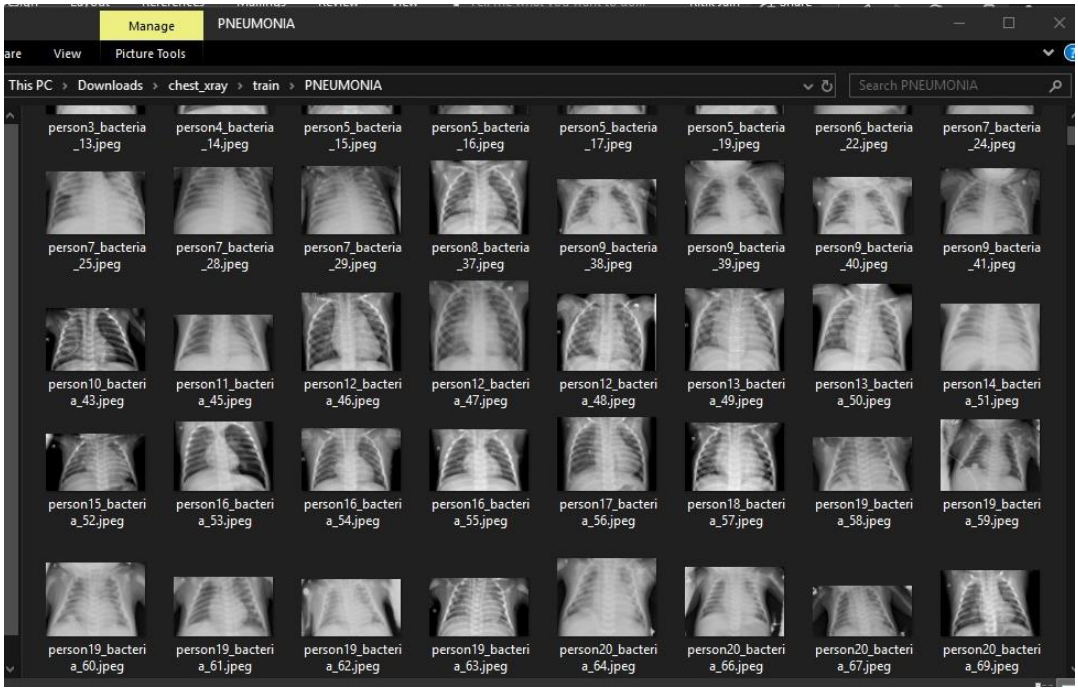

Fig 4.6b Chest X-ray of pneumonia infected patients

```
'chest_xray/train\\NORMAL\\normal_987.jpeg',
'chest_xray/train\\NORMAL\\normal_989.jpeg',
'chest_xray/train\\NORMAL\\normal_990.jpeg',
'chest_xray/train\\NORMAL\\normal_991.jpeg',
'chest_xray/train\\NORMAL\\normal_992.jpeg',
'chest_xray/train\\NORMAL\\normal_993.jpeg',
'chest_xray/train\\NORMAL\\normal_994.jpeg',
'chest_xray/train\\NORMAL\\normal_995.jpeg',
'chest_xray/train\\NORMAL\\normal_996.jpeg',
'chest_xray/train\\NORMAL\\normal_997.jpeg',
'chest_xray/train\\NORMAL\\normal_998.jpeg',
'chest_xray/train\\NORMAL\\normal_999.jpeg',
'chest_xray/train\\PNEUMONIA\\person1000_bacteria_2931.jpeg',
'chest_xray/train\\PNEUMONIA\\person1000_virus_1681.jpeg',
'chest_xray/train\\PNEUMONIA\\person1001_bacteria_2932.jpeg',
'chest_xray/train\\PNEUMONIA\\person1002_bacteria_2933.jpeg',
'chest_xray/train\\PNEUMONIA\\person1003_bacteria_2934.jpeg',
'chest_xray/train\\PNEUMONIA\\person1003_virus_1685.jpeg',
'chest_xray/train\\PNEUMONIA\\person1004_bacteria_2935.jpeg',
'chest_xray/train\\PNEUMONIA\\person1004_virus_1686.jpeg',
'chest_xray/train\\PNEUMONIA\\person1005_bacteria_2936.jpeg',
```

Fig. 4.7 List of Image paths

3) Pre Processing

This project we have to extract all data from the directories & folders into the notebook. I am using OpenCV to read data from the folders & we are transforming the data from 3D matrices into 224x224 square matrix. & stack up the data into an array. By which we can use it for later processes.

We are using inter cubic interpolation while resizing the image from any size to 224x224.

```
1 im = plt.imread('chest_xray//test\\NORMAL\\normal_100.jpeg')
2 plt.imshow(im)
```

```
<matplotlib.image.AxesImage at 0x27529940240>
```

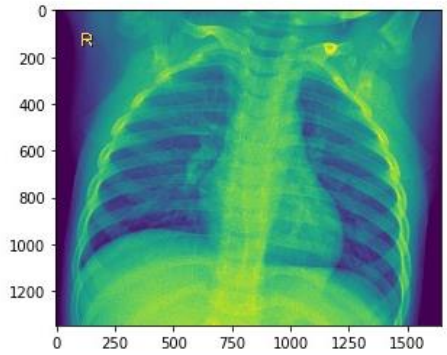

Fig. 4.8 Chest X-ray image before pre-processing

```
:  1 plt.imshow(x[0]) ## Transformed Image
```

```
:  <matplotlib.image.AxesImage at 0x27520bbd748>
```

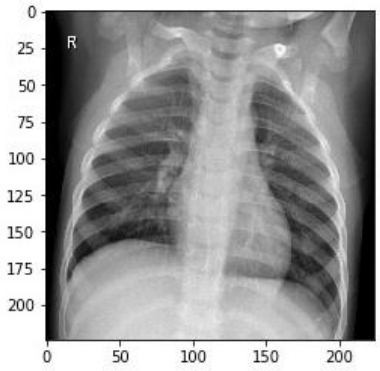

Fig 4.9 Chest X-ray image after pre-processing

4) Filtering

The above image represents the process of pre-processing the input image to fit the model from which credible assumptions will be made as to whether the image contains pneumonic traces or not.

Here are the three elements that enter into the convolution operation:

- Input image
- Feature detector
- Feature map

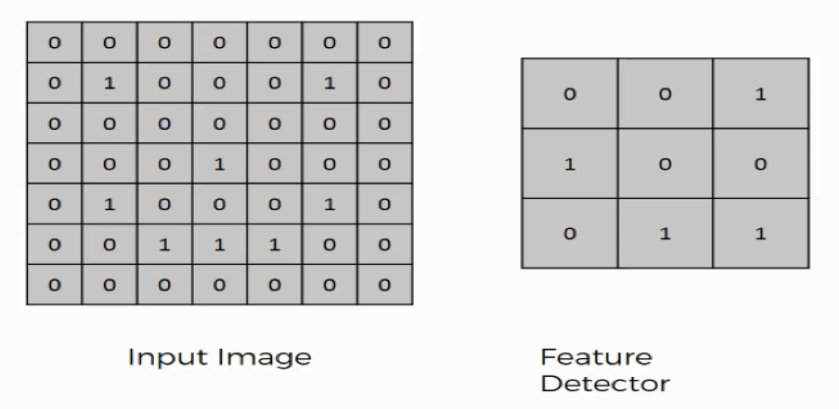

Fig 4.10 Feature Detection from input image

As you can see, the input image is the same smiley face image that we had in the previous tutorial. Again, if you look into the pattern of the 1's & 0's, you will be able to make out the smiley face in there.

Sometimes a 5×5 or a 7×7 matrix is used as a feature detector, but the more conventional one, & that is the one that we will be working with, is a 3×3 matrix. The feature detector is often referred to as a 'kernel' or a 'filter,'

You can think of the feature detector as a window consisting of 9 (3×3) cells. Here is what you do with it:

- You place it over the input image beginning from the top-left corner within the borders you see demarcated above, & then you count the number of cells in which the feature detector matches the input image.
- The number of matching cells is then inserted in the top-left cell of the feature map.
- You then move the feature detector one cell to the right & do the same thing. This movement is called & since we are moving the feature detector one cell at time that would be called a stride of one pixel.
- What you will find in this example is that the feature detector's middle-left cell with the number 1 inside it matches the cell that it is standing over inside the input image. That's the only matching cell, & so you write '1' in the next cell in the feature map, & so on & so forth.
- After you have gone through the whole first row, you can then move it over to the next row & go through the same process.

It's important not to confuse the feature map with the other two elements. The cells of the feature map can contain any digit, not only 1's & 0's. After going over every pixel in the input image in the example above, we would end up with these results:

There are several uses that we gain from deriving a feature map. These are the most important of them: Reducing the size of the input image, & you should know that the larger your strides (the movements across pixels), the smaller your feature map.

One thing to keep in mind is that the depth dimension of the weight would be same as the depth dimension of the input image. The weight extends to the entire depth of the input image. Therefore, convolution with a single weight matrix would result into a convolved output with a single depth dimension. In most cases instead of a single filter (weight matrix), we have multiple filters of the same dimensions applied together.

The output from the each filter is stacked together forming the depth dimension of the convolved image. Suppose we have an input image of size 32*32*3. & we apply 10 filters of size 5*5*3 with valid padding. The output would have the dimensions as 28*28*10.

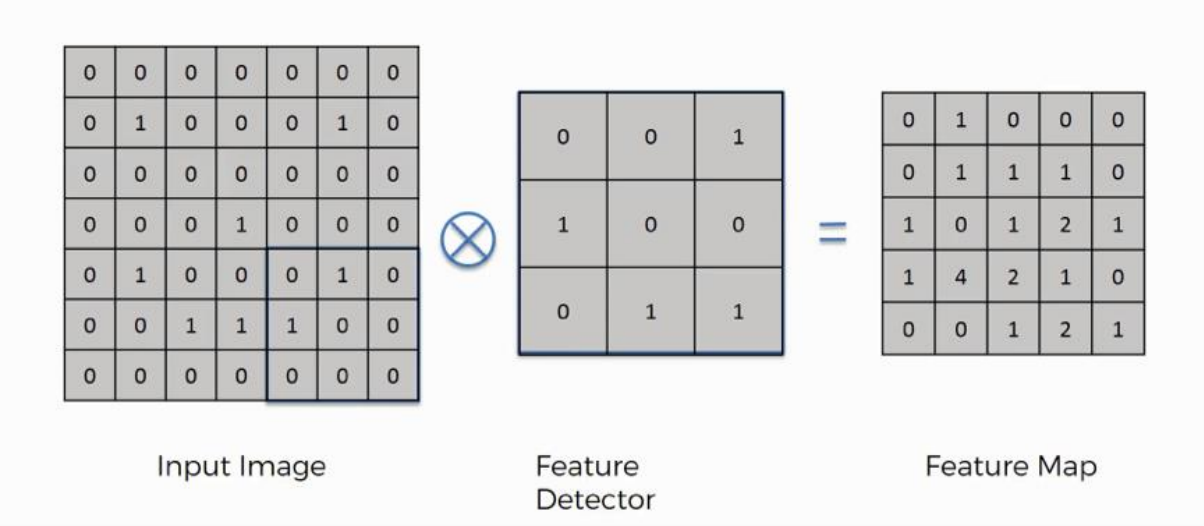

Fig 4.11 Feature mapping and filtering

Pooled Feature Map

The process of filling in a pooled feature map differs from the one we used to come up with the regular feature map. This time you'll place a 2x2 box at the top-left corner, & move along the row.

For every 4 cells your box stands on, you'll find the maximum numerical value & insert it into the pooled feature map. In the figure below, for instance, the box currently contains a group of cells where the maximum value is 4.

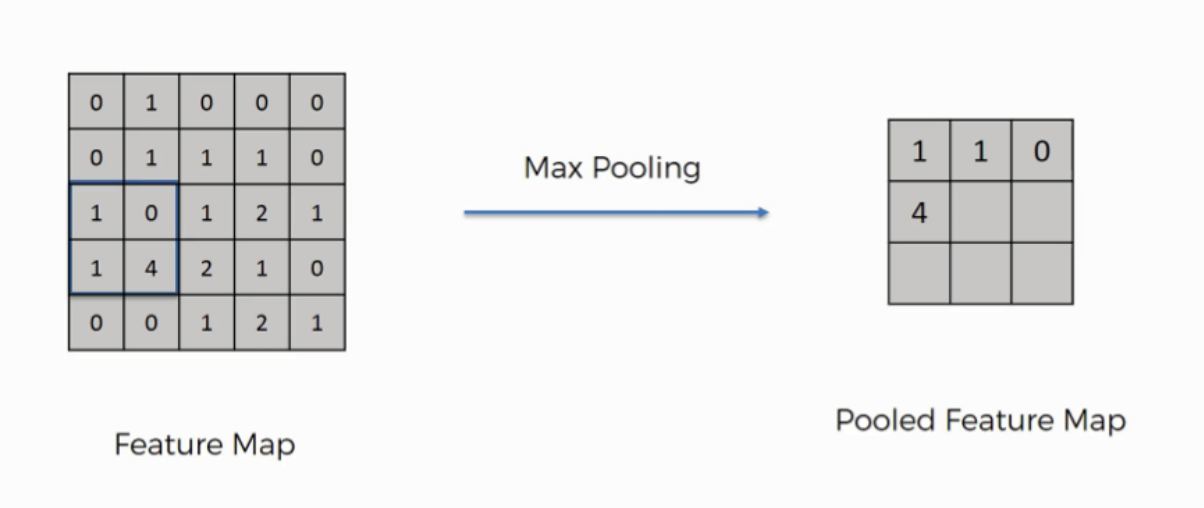

Fig 4.12 Feature Mapping with the pooling

The creation of the pooled feature map also makes us dispose of unnecessary information or features. In this case, we have lost roughly 75% of the original information found in the feature map since for each 4 pixels in the feature map we ended up with only the maximum value & got rid of the other 3. These are the details that are unnecessary & without which the network can do its job more efficiently.

The reason we extract the maximum value, which is actually the point from the whole pooling step, is to account for distortions. Let's say we have three cheetah images, & in each image the cheetah's tear lines are taking a different angle.

The feature after it has been pooled will be detected by the network despite these differences in its appearance between the three images. Consider the tear line feature to be represented by the 4 in the feature map above.

Imagine that instead of the four appearing in cell 4x2, it appeared in 3x1. When pooling the feature, we would still end up with 4 as the maximum value from that group, & thus we would get the same result in the pooled version.

This process is what provides the convolutional neural network with the ‘spatial variance’ capability. In addition to that, pooling serves to minimize the size of the images as well as the number of parameters which, in turn, prevents an issue of ‘overfitting’ from coming up.

Overfitting in a nutshell is when you create an excessively complex model in order to account for the idiosyncrasies we just mentioned.

Again, this is an abstract explanation of the pooling concept without digging into the mathematical & technical aspects of it.

Flattening:

After finishing the previous two steps, we're supposed to have a pooled feature map by now. As the name of this step implies, we are literally going to flatten our pooled feature map into a column like in the image below.

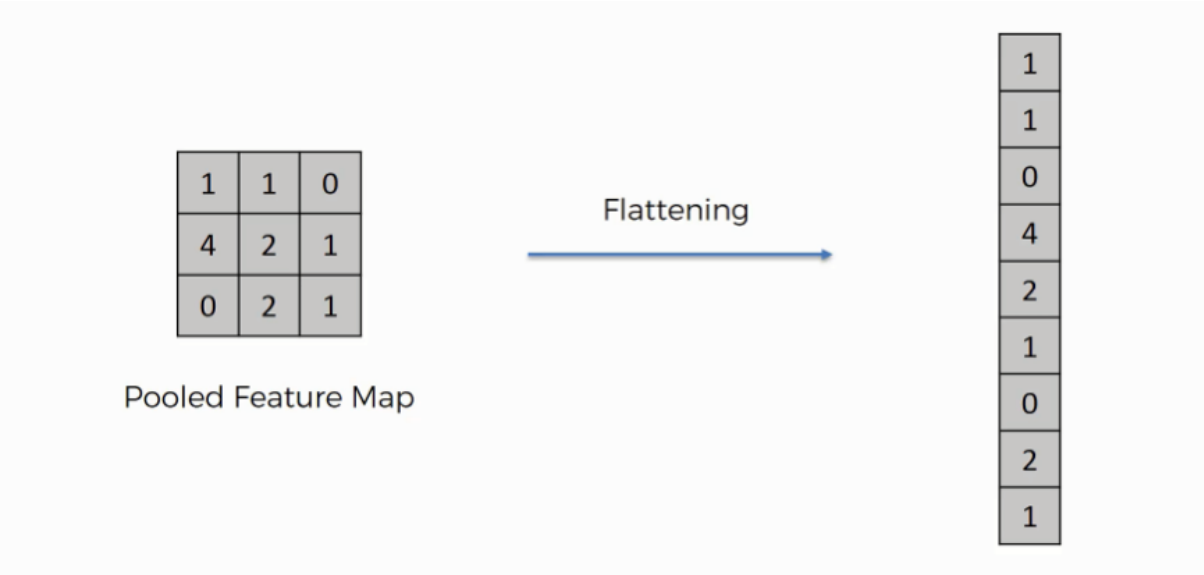

Fig 4.13 Mapped image flattening

As you see in the image above, we have multiple pooled feature maps from the previous step.

What happens after the flattening step is that you end up with a long vector of input data that you then pass through the artificial neural network to have it processed further.

To sum up, here is what we have after we're done with each of the steps that we have covered up until now:

- Input image (starting point)
- Convolutional layer (convolution operation)
- Pooling layer (pooling)
- Input layer for the artificial neural network (flattening)

5) Classification

Classification of any thing is a process which is relates to categorization. The main idea of classification is to recognize the class of the particular data. We can have lot of data from different fields. But we can only able to use the data which is related to solve a particular problem. If that data is available for classification or categorization. Then we create the model for performing particular task of classification. There are many type of classification related to their respective fields for e.g. In media technology we can do classification document, Image, motion videos and many more. In Science, we can do classification for chemicals, taxonomic, biological organisms, medical classification and many more. In other fields like business we can make classification for defective products, security, job analysis and many more.

In our project, we are focused on Image classification

In image classification, we are extracting the important feature from the particular image for e.g. in pneumonia classification/ detection, we are extracting the features of the lung image. Initially, we are separating the lung image from the whole chest X-ray and then we are moving forward to classify the symptoms of pneumonia.

To classify the pneumonia image from the normal image we have to train the model with training dataset and then we can make prediction or classification for the pneumonic infected patient.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 218, 218, 32)	4736
max_pooling2d_1 (MaxPooling2	(None, 109, 109, 32)	0
batch_normalization_1 (Batch	(None, 109, 109, 32)	128
dropout_1 (Dropout)	(None, 109, 109, 32)	0
conv2d_2 (Conv2D)	(None, 105, 105, 64)	51264
max_pooling2d_2 (MaxPooling2	(None, 52, 52, 64)	0
batch_normalization_2 (Batch	(None, 52, 52, 64)	256
dropout_2 (Dropout)	(None, 52, 52, 64)	0
conv2d_3 (Conv2D)	(None, 50, 50, 128)	73856
max_pooling2d_3 (MaxPooling2	(None, 25, 25, 128)	0
batch_normalization_3 (Batch	(None, 25, 25, 128)	512
dropout_3 (Dropout)	(None, 25, 25, 128)	0
conv2d_4 (Conv2D)	(None, 23, 23, 128)	147584
max_pooling2d_4 (MaxPooling2	(None, 11, 11, 128)	0
batch_normalization_4 (Batch	(None, 11, 11, 128)	512
dropout_4 (Dropout)	(None, 11, 11, 128)	0
global_average_pooling2d_1 (	(None, 128)	0
dense_1 (Dense)	(None, 1000)	129000
dense_2 (Dense)	(None, 2)	2002

Total params: 409,850
Trainable params: 409,146
Non-trainable params: 704

Fig 4.14 Model summary

6) Neural Network

Every neural network is a circuit of many neurons which are connected together and make some computation which help in final prediction. A neuron is similar to the biological neuron which are continuously running in human brain.

A neuron having two parts

- Transfer function
- Activation function

Transfer function is a maximum likelihood estimated function which can have one input and one output. It can take a scalar value, vector or a matrix and do multiplication and throws the results to activation function.

Activation function is a special type of function which allows neuron participate in further operation. It take the result of transfer function and put that value into a derived function. Result of the activation function is depends on the type of activation function we are using but in most of the times it is either 0 or 1.

There are many type of activation function like:

- Regularized Linearity (ReLU)
- Leaky Regularized Linearity(Leaky ReLu)
- Sigmoid Function

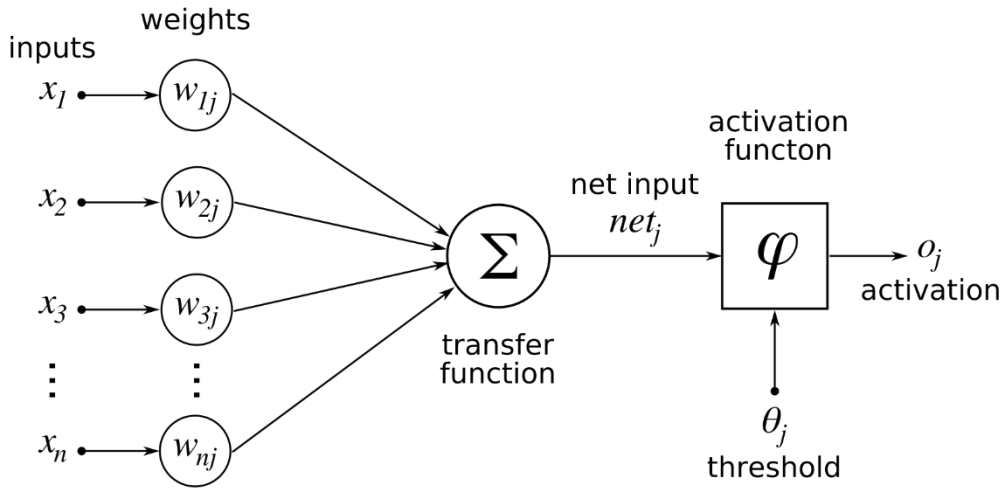

Fig. 4 15 Activation function working

The connections of all the neurons create a neural network. There can be thousands of different neurons on a single layer.

A neural system (NN), on account of counterfeit neurons called fake neural system (ANN) or recreated neural system (SNN), is an interconnected gathering of characteristic or fake

neurons that utilizes a scientific or computational model for data handling dependent on a connectionist way to deal with calculation. Much of the time an ANN is a versatile framework that changes its structure dependent on outside or inner data that moves through the system.

In increasingly down to earth terms neural systems are non-direct measurable information displaying or basic leadership apparatuses. They can be utilized to display complex connections among sources of info and yields or to discover designs in information.

A counterfeit neural system includes a system of straightforward preparing components (fake neurons) which can display complex worldwide conduct, dictated by the associations between the handling components and component parameters. Fake neurons were first proposed in 1943 by Warren McCulloch, a neurophysiologist, and Walter Pitts, a rationalist, who previously teamed up at the University of Chicago.

One traditional sort of fake neural system is the intermittent Hopfield organize.

7) Result

Result of our project is binary class and the model generate the probability for each class in our model. Highest class probability of any class will be used as the classification result. The probability will tell about the chance of happening for particular class. In our model, the result will be either 0 or 1. Where 0 means no pneumonia detected and 1 means pneumonia detected. That result will reflect on our android application. According to the results, application will provide the prescription for patients.

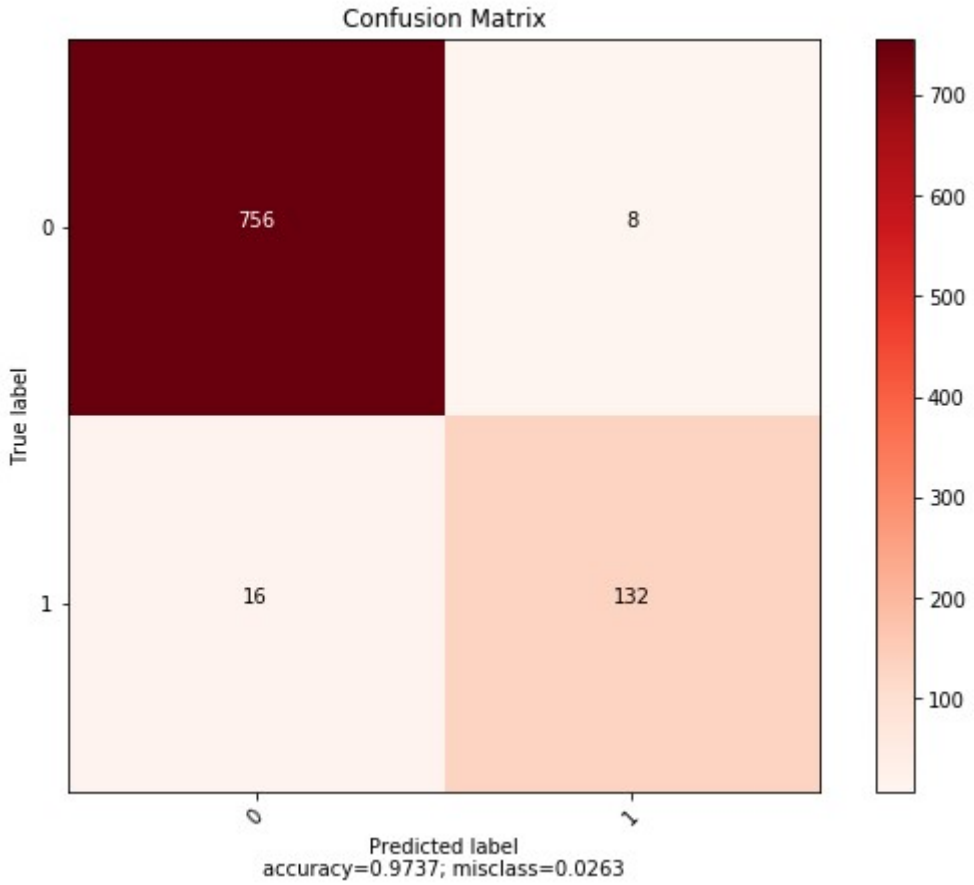

Fig 4.16 Confusion Matrix of the model

	precision	recall	f1-score	support
0	0.98	0.99	0.98	764
1	0.94	0.89	0.92	148
micro avg	0.97	0.97	0.97	912
macro avg	0.96	0.94	0.95	912
weighted avg	0.97	0.97	0.97	912

Fig. 4.17 Precision, recall matrix of model

Chapter 5

Implementation and Experimental Analysis

5.1 Testing and Training

In the initial level, we provide random weights all the parameters to the each layer which is by default selected by the keras framework. In the training phase, we use the backward and forward propagation to calculate and optimize the weights for each layer. We pre-process the data before feeding into the model. The data will be fed in the model in the standard size of 224x224 in the chunks of 32x32 matrix. Which means we feed the image in 7 separate chunks. The model starts convolution with a filter of 7x7 and extract the features from the model.

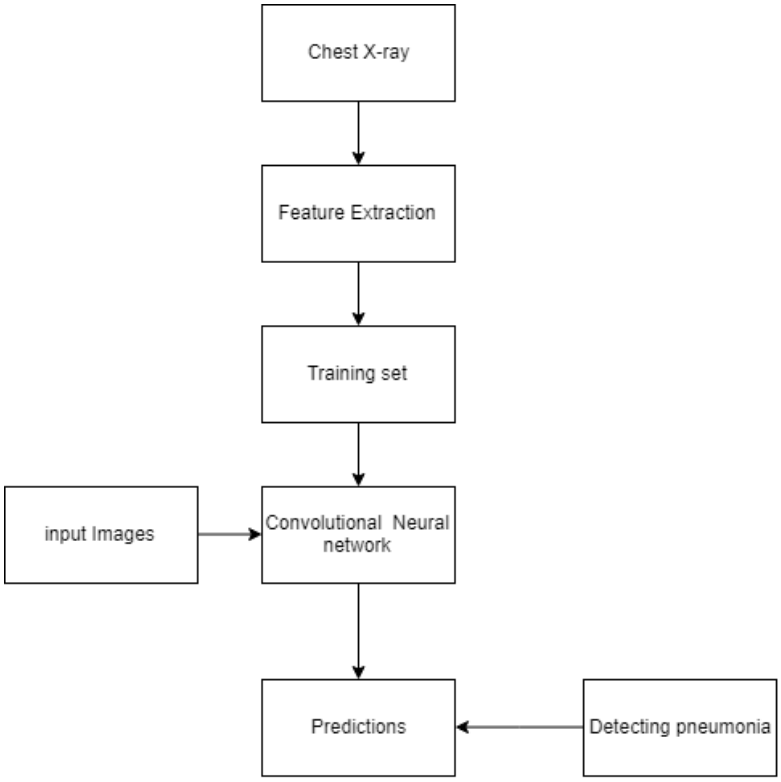

Fig 5.1 Work flow of model training

5.2 Network training

In our model, we applied a complete network 4 times with different feature vectors. Initially we used a convolutional layer with the filter of 7x7 to extract the major features from the image. For filtration of data we applied a max pooling layer with a filter size of 2x2 to extract the maximum value from the image matrix. To reduce the dependency of data over any previous layer we are using the batch normalization layer. Batch normalization will convert the image matrix into zero mean one standard deviation. TO overcome the problem of overfitting we are using dropout layer with the threshold value of 0.15. It will remove the transferred results of neuron where participation is less than the threshold value. Same procedure repeated 4 time but every time only weights of Convolutional layer will change. On second iteration Convolutional layer taking an image matrix of 64x64 with the filter vector of 5x5. On 3rd iteration, Convolutional layer takes 128x128 matrix with the filter size of 3x3 to extract fine details from the images. Than all the weights will transfer to Global Average pooling Layer with default filter size. The extracted feature vector will be flattened and transfer to dense (fully connected layer) of

1000 neurons for the classification and result of dense layer transfer to another dense layer which works as the output layer and having only 2 neurons.

Training model with 40 epochs

```
: from keras.callbacks import ModelCheckpoint
  mcp = ModelCheckpoint(filepath='model_check_path.hdf5',monitor="val_acc", save_best_only=True, save_weights_only=False)
  hist = model.fit(x_train,y_train,batch_size = 32, epochs = 40, verbose=1,  validation_split=0.2, callbacks=[mcp])

  Train on 2916 samples, validate on 729 samples
  Epoch 1/40
  2916/2916 [==============================] - 32s 11ms/step - loss: 0.2554 - acc: 0.8896 - val_loss: 2.2878 - val_acc: 0.8573
  Epoch 2/40
  2916/2916 [==============================] - 28s 10ms/step - loss: 0.1981 - acc: 0.9180 - val_loss: 2.1149 - val_acc: 0.8573
  Epoch 3/40
  2916/2916 [==============================] - 28s 10ms/step - loss: 0.1554 - acc: 0.9352 - val_loss: 2.2994 - val_acc: 0.8573
  Epoch 4/40
  2916/2916 [==============================] - 28s 10ms/step - loss: 0.1242 - acc: 0.9516 - val_loss: 1.1737 - val_acc: 0.8573
  Epoch 5/40
  2916/2916 [==============================] - 28s 10ms/step - loss: 0.1438 - acc: 0.9468 - val_loss: 2.2994 - val_acc: 0.8573
  Epoch 6/40
  2916/2916 [==============================] - 28s 10ms/step - loss: 0.1101 - acc: 0.9582 - val_loss: 2.0684 - val_acc: 0.8573
  Epoch 7/40
  2916/2916 [==============================] - 28s 10ms/step - loss: 0.1125 - acc: 0.9568 - val_loss: 0.2093 - val_acc: 0.9204
  Epoch 8/40
  2916/2916 [==============================] - 28s 10ms/step - loss: 0.0880 - acc: 0.9667 - val_loss: 1.5729 - val_acc: 0.8573
  Epoch 9/40
  2916/2916 [==============================] - 28s 10ms/step - loss: 0.0768 - acc: 0.9739 - val_loss: 0.3031 - val_acc: 0.9067
  Epoch 10/40
```

Fig. 5.2 Training model verbose

5.3 Error Rate

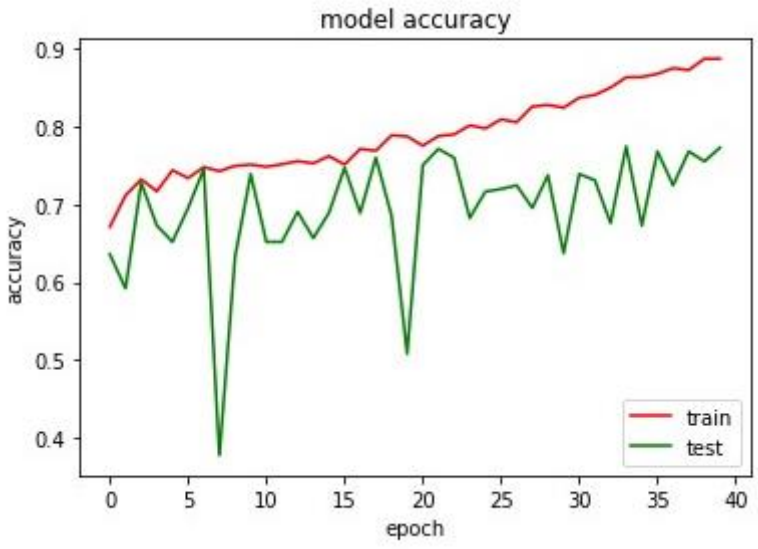

Fig 5.3 Error rate between training and validation

5.4 System Configuration

5.4.1 System requirement

For Developer:

Hardware Requirement:

CPU: Intel I3 5th Gen minimum, Intel I5 5th Gen or above highly recommended.

Storage: 20GB minimum, 30GB recommended.

Memory: 6GB Minimum, 8GB recommended.

Display: 1280x800 minimum screen resolutions.

Graphics Card: NVidia GTX 950M minimum, GTX 1050 or above recommended.

Software Requirement:

Operating System: Windows (64-bit Architecture), Linux or Mac

Software Tools: Android Studio, Anaconda 3, Jupyter Notebook, JetBrains PyCharm, NVidia CUDA Toolkit(v9.0 recommended) , NVidia cuDNN v4.0 minimum (v5.1 recommended), opencv

Language Support: Python 3.5 or above, JAVA/Kotlin, XML.

Python Library Support: TensorFlow, Keras, OpenCV, matplotlib.

For User:

Hardware Requirement:

Device: Android smart device

Android Version: Marshmallow (v6) or above.

RAM: 1GB minimum, 2GB or above recommended.

Camera: 8 mega-pixel rear-camera (optional).

Software Requirement:

Have to install the android application file.

5.4.2 Software tools and configuration

Windows: Microsoft windows is a gathering of graphical UIs which are created, sold and showcased by Microsoft. Windows is a working framework which make an interface between the client and framework equipment. For our purpose, we use 64bit operating system architecture because we required the data path pipelines to be large enough to store and transfer data large values.

1. **Android Studio:**

Android Studio is the expert Integrated Development Environment (IDE) for Android application progression, in light of IntelliJ IDEA. Over IntelliJ's incredible code supervisor and

designer apparatuses, Android Studio offers significantly more highlights that upgrade your efficiency when building Android applications, for example,

- An adaptable Gradle-based form framework
- A quick and highlight rich emulator
- A brought together condition where you can create for all Android gadgets
- Moment Run to push changes to your running application without structure another APK
- Code layouts and GitHub coordination to enable you to manufacture normal application highlights and import test code
- Broad testing apparatuses and structures
- Build up apparatuses to get execution, ease of use, rendition similarity, and different issues
- C++ and NDK support

Coding Assistance:

The IDE provides several features like code completion by analysis of context, code refactoring and many options which fix inconsistencies of code via suggestions, code navigations which allows directing the class or function or declaration in the code files.

Built in tools and integration

The IDE provides integration of various tools like grunt, gradle, SBT. It also support various git, database connectors, android virtual emulator, debugger, rich layout editor and firebase cloud messaging. These tools are used to provide accessibly to the user. For example git is the tools for version control where when we are developing any project and we have to update scripts into the github then it always create a checkpoint for the last values. And update the latest.

Plugin ecosystem

Android studio have a great support with plugins. Android studio provide a store to download and install all the plugins. There is a repository of plugins which can access. Plugin is a very small package that allow the ide or any tool to perform an specific task which can help the user for their ease. Some most famous plugins like Markdown Navigator, Mirror, Android material designer icon generator, DTO generator, key promotor, ADB Idea, Butterknife Zelenzy, Android Methods Count, Code Galance and many more.

Support Language

IDE supports all the language which are preferred by Google to develop android application. Google mainly use the langugaes like C++, JAVA, Kotlin, XML and many more. They provide assistance service for all the languages.

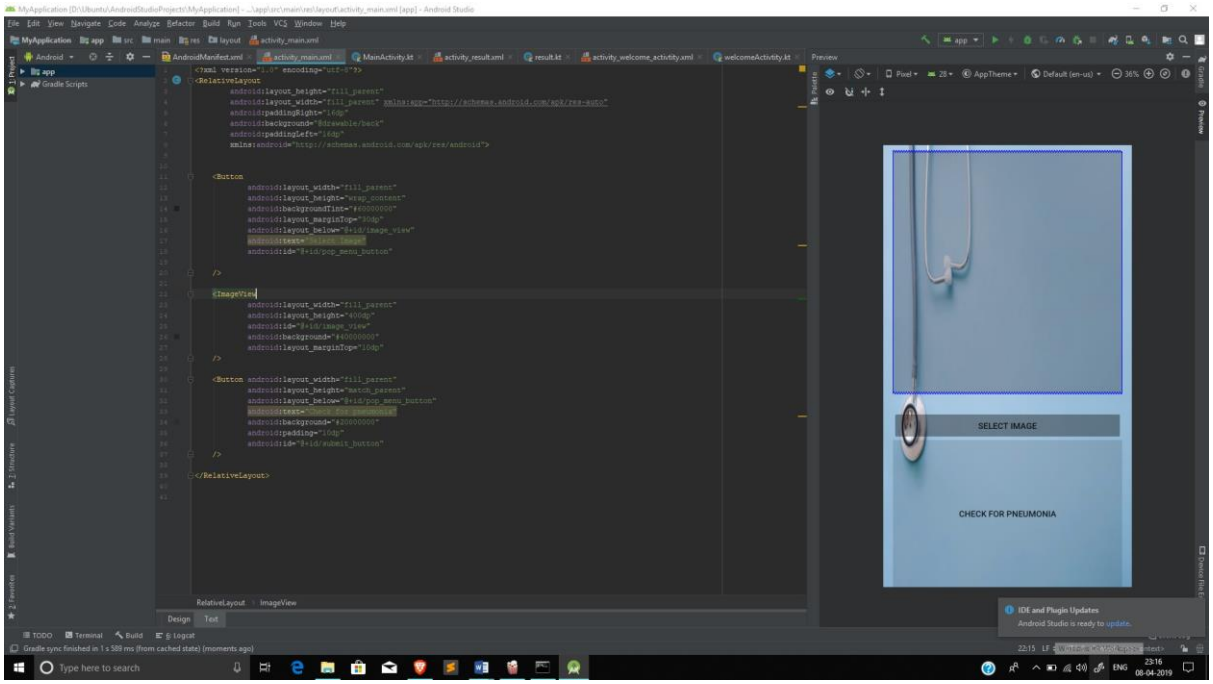

Fig 5.4 Android Studio

2. OpenCV (Open Source Computer Vision)

OpenCV is a very popular computer vision library started by Intel in 1999. The cross-platform library sets it focuses on real-time image processing tasks and some patent free computer vision algorithms which are used in today's world. In 2008, willow Garage took over support of the tool and create openCV 2.3.1 now comes with a programming interface to C, C++, python, JAVA and Android. OpenCV is released under a BSD license so it is used in academic projects and commercial products.

OpenCV 2.4 comes with very fast, optimized and real-time face recognizer class for face recognition. Latest version of opencv introduces support for GPU calculation using CUDA and OpenCL, Android platform, iOS platform and linux embedded platforms.

An image is an array of (m x n x d). Where m is the number of rows, n is number of columns and d is the dimensions. Majorly an image is in form of 3d array. Each dimension represents one colour in the image. For e.g. we always having the image in the basic form of RGB colours where R is Red, G is Green and B is Blue colour.

We are using openCV to read the images from the directories and converting the image in format which can be readable by my model. We are using geometric image transformation. OpenCV provide several types of transformation techniques like:

- Affine Transformation (cv.GetAffineTransform(src, dst, mapMatrix))
- Perspective Transformation (cv.GetPerspectiveTransform(src, dst, mapMatrix))
- Rectangle sub pixel transformation (cv.GetRectSubPix(src, dst, center))
- Invert Affine Transformation (cv2.invertAffineTransform(M[, iM]))
- Linear polar Transformation
- Log polar Transformation

3. Pycharm: IDE for python programmers

Pycharm is an Integrated Development Environment developed for the python programmers. It is available in Community, professional an ultimate version. This IDE is totally free, open source. It provide support to all the language related to python programming. It provides the project level customized configuration for setting up the server.

A rich, record based HTTP customer for mechanized testing.

Backing for the web2py Python system: devoted run/investigate arrangement type, support for layout language, route among perspectives and formats, and a web2py venture format.

Backing for Google App Engine (GAE): design GAE settings from a board, transfer applications from a devices menu, see log documents, devoted support for running appcfg.py directions, and a GAE venture layout.

Language infusion to insert SQL support into Python strings, with autocomplete on SQL directions as well as on the mapping of your characterized datasource.

Fig 5.5 Pycharm introduction page

Code Assistance:

This IDE having several features which are very useful for the programmer like code completion by analysis of the context, Code refactoring which allows the user to check for the uses of the module which is going to be deleted or modifying, Source Code navigation which allows the access the prefixed class files, declaration and function used the programs.

Built in tools and Integration

IDE provide the integration of many tools related to the system like Web browser, File Watchers, External tools, Terminal, Database Connector, Git, SSH terminal, Diagrams, Diff & merge, Vim Emulation, setting repository, Remote SSH external tools and debuggers.

Plugin ecosystem

Pycharm have a great plugin repository. We can select the values plugins directly from the repository. Pycharm also provide support to create your own plugin to make thinks more accessible and clear to the use.

Support Language:

Pycharm have a great language support with code assistance for all the language. Pycharm provide support for programming languages like python, JavaScript, CoffeeScript. Markup langugaes like XML, HTML, YAML, CSS, Sass, Stylus.

Framework support: Django > 1.3, Google App Engine, Flask, Buildout, Web2py, Pyramid, Maya, SQLAlchemy and also provide support for many templates.

4. Anaconda 3

The open source anaconda distribution is the demanding way to approach the data science, information science and AI problems through python/R on Windows, Linux and Mac OS X. It becomes the business standard for creating, testing and preparing on a solitary machine empowering singular information researchers to:

Rapidly download 1,500+ Python/R information science bundles

Oversee libraries, conditions, and situations with Conda

Create and train AI and profound learning models with scikit-learn, TensorFlow, and Theano

Break down information with adaptability and execution with Dask, NumPy, pandas, and Numpy

Picture results with Matplotlib, Bokeh, Datashader, and Holoviews

5. Anaconda Navigator

Anaconda Navigator is a work area graphical UI (GUI) incorporated into Anaconda circulation that enables clients to dispatch applications and oversee conda bundles, conditions and channels without utilizing order line directions. Guide can scan for bundles on Anaconda Cloud or in a neighbourhood Anaconda Repository, introduce them in a situation, run the bundles and update them. It is accessible for Windows, macOS and Linux.

Anaconda navigator provides a set of pre-built applications:

- JupyterLab: JupyterLab is the cutting edge UI for Project Jupyter. It offers all the commonplace structure squares of the exemplary Jupyter Notebook (scratch pad, terminal, word processor, document program, rich yields, and so forth.) in an adaptable and amazing UI. The main stable discharge was reported on February 20, 2018,[16] and in December 2018 it was received as the essential interface for the cloud-based Jupyter administration Jupyo

- QtConsole: The Qt support is a lightweight application that generally feels like a terminal, however gives various upgrades just conceivable in a GUI, for example, inline figures, legitimate multi-line altering with linguistic structure featuring, graphical calltips, and considerably more. The Qt comfort can utilize any Jupyter portion.
- Spyder: Spyder is an open source cross-stage incorporated advancement condition (IDE) for logical programming in the Python language. Spyder coordinates with various unmistakable bundles in the logical Python stack, including NumPy, SciPy, Matplotlib, pandas, IPython, SymPy and Cython, just as other open source software.[3][4] It is discharged under the MIT license.
- Glueviz: It provide multi-dimensional linked data exploration. It is a python library to explore the co-relation between the data. It is a multi-disciplinary tool.
- Orange: It is an open source data visualization, data mining and machine learning tool kit. It provides a visual programming front end analysis and exploration of data.
- RStudio: RStudio is a free and open-source coordinated advancement condition (IDE) for R, a programming language for factual processing and illustrations. RStudio was established by JJ Allaire, maker of the programming language ColdFusion. Hadley Wickham is the Chief Scientist at RStudio.
- Visual Studio Code: Visual Studio Code depends on Electron, a structure which is utilized to send Node.js applications for the work area running on the Blink design motor.

Conda

Conda is an open source, language-agnostic, cross platform, bundle chief and condition the executives system that introduces, runs and updates bundles and their dependencies. It was made for python programmers, however it can bundle and disseminate programming for any language (e.g. R), including multi-language projects. The conda bundle and condition supervisor is incorporated into all form of anaconda, miniconda and anaconda repository.

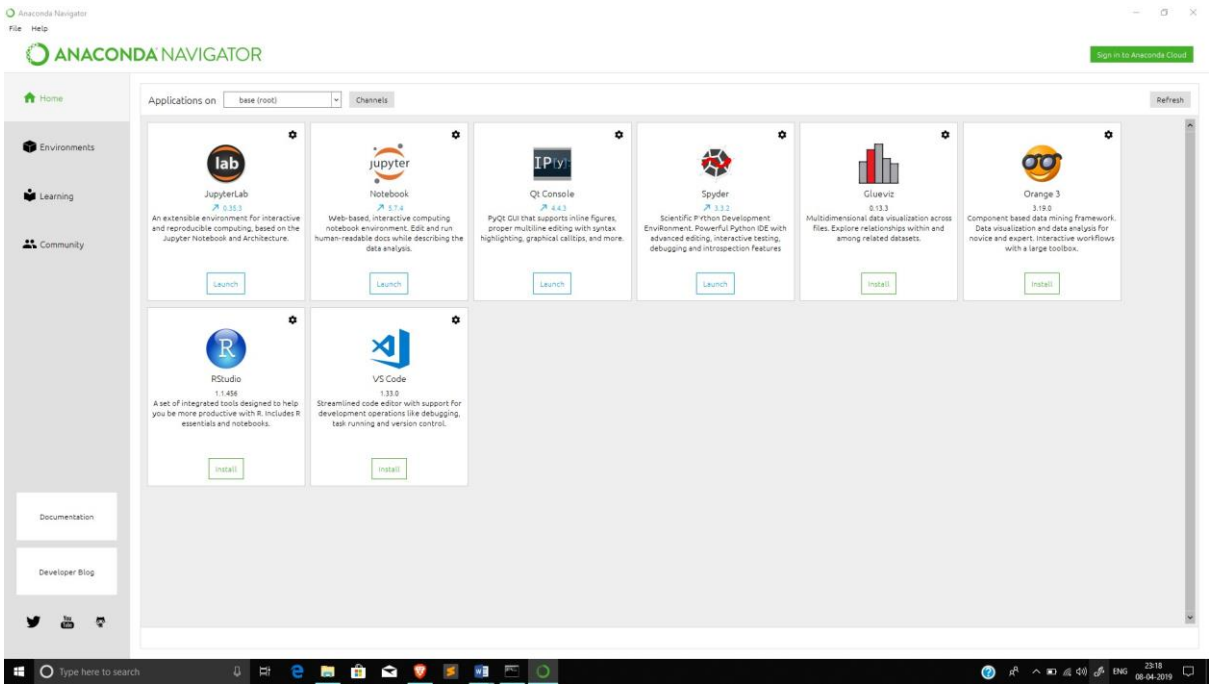

Fig 5.6 Anaconda Navigator Welcome page

Anaconda cloud

Anaconda Cloud is a bundle the board administration by Anaconda where you can discover, access, store and offer open and private scratch pad, conditions, and conda and PyPI bundles. Cloud has helpful Python bundles, note pads and situations for a wide assortment of uses. You don't have to sign in or to have a Cloud account, to scan for open bundles, download and introduce them.

You can manufacture new bundles utilizing the Anaconda Client order line interface (CLI), at that point physically or consequently transfer the bundles to Cloud.

6. Nvidia CUDA Toolkit

The NVIDIA CUDA Toolkit gives an improvement domain to making elite GPU-acceleration applications. With the CUDA Toolkit, you can create, improve and convey your applications on GPU-quickened implanted frameworks, work area workstations, endeavor server farms, cloud-based stages and HPC supercomputers. The toolbox incorporates GPU-quickened libraries, investigating and advancement apparatuses, a C/C++ compiler and a runtime library to convey your application.

GPU-quickened CUDA libraries empower drop-in speeding up over different spaces, for example, direct polynomial math, picture and video preparing, profound learning and diagram examination. For creating custom calculations, you can utilize accessible incorporations with generally utilized dialects and numerical bundles just as very much distributed advancement APIs. Your CUDA applications can be conveyed over all NVIDIA GPU families accessible on reason and on GPU cases in the cloud. Utilizing worked in abilities for appropriating

calculations crosswise over multi-GPU designs, researchers and analysts can create applications that scale from single GPU workstations to cloud establishments with a huge number of GPUs.

Fig 5.7 Nvidia CUDA Toolkit

To begin, peruse through web based beginning assets, enhancement guides, illustrative precedents and work together with the quickly developing designer network.

CUDA toolkit provide a wide range of features for acceleration of performance. CUDA Toolkit provides:

Specific libraries for specific domains

Development of application in one system and deploy in many system

Language support integration:

CUDA toolkit provide good support for different languages. It provide support for C, C++, FORTAN and python. For every language NVIDIA corp. created separate toolkit for e.g. for python they created pyCUDA, FORTAN they created cudaFORTAN, C/C++ they created PGI CUDA C/C++ Compiler and many more.

7. Nvidia cuDNN

Nvidia cuDNN is the deep neural network library which is created to give GPU acceleration for the primitive deep neural networks. It provide high tuning implementation for the standard routes such as backward- forward convulation, pooling, normailzation and activation layers.

It allows data science researcher and frameworks developers to focus on the training Deep neural networks. Developing software application related to Machine Learning.

NVIDIA cuDNN accelerates all the widely used deep neural network or deep learning framework suck as tensorflow, keas, MxNet, Matlab, Caffe, Caffe2, pyTorch and many more. To access the cuDNN framework or NVidia optimized deep learning frameworks, nvidia created NVIDIA GPU CLOUD for direct access.

Fig 5.7 cuDNN network

Features of NVIDA cuDNN:

- Gathered convolutions currently support NHWC inputs/yields and FP16/FP32 figure for models, for example, ResNet and Xception
- Expanded convolutions utilizing blended accuracy Tensor Core tasks for applications, for example, semantic division, picture super-goals, denoising, and so on
- TensorCore increasing speed with FP32 data sources and yields (recently confined to FP16 input)
- RNN cells bolster various use cases with choices for cell cutting and cushioning veils
- Naturally select the best RNN execution with RNN seek API
- Subjective measurement requesting, striding, and sub-districts for 4d tensors methods simple combination into any neural network

cuDNN is provide support for windows, linux and MacOS system with different GPU system architecture such as Volta, Pascal, Kepler, Maxwell Tegra K1, and many more.

Language Support for application

cuDNN provide support for many language such as python, C++, FORTAN, Matlab etc.

5.4.3 Programming language

I. Python

Python is a high-level programming. Python is an general purpose programming language which is created by Guido Van Rossum. Python features a automation memory management and dynamic type system. It supports paradigm or architecture of multiple programming language which includes imperative, functional, object-oriented and procedural.

In our project, we are using python 3.7.1 which includes many libraries for data processing, creating model, plotting data, and extracting directories.

II. Kotlin

Kotlin is a statically typed, cross-platform, general purpose programming with type inference. Kotlin is designed to work conjunction to JAVA. It supports the JVM version of libraries which comes under his standards. It majorly target to compile with JVM but also able to compile with JavaScript or native codes. Kotlin is designed in sponsorship by jetbrains and google.

In our project, we are using kotlin to create the android application for giving one user interface for our model to get input from the user and display results. It is providing me the back end support to write the functionality of the application.

III. XML

Extensible markup language is a markup language which defines some set of rules for encoding the document format. It is human readable and machine readable. The main goal to create such type of markup language is to focuses on simplicity, generality and uses over the internet. It mainly use in creating or writing API to aid the processing of XML data. XML is too similar to hypertext mark-up language.

In our project, we are using XML to create the graphical user interface for the android application. It uses to define the properties of the each element in the application. XML also use for creating animations, transitions and graphics designing.

Library and framework Support

- Numpy: It is a library in python programming language which use to dealing with massive amount of data, multi-dimensional arrays along with performing operation broadcasting by which the operation is applied on every element on one call.
- OS: This library helps in dealing with operating system commands and process. We can go to any directory through this library.
- Glob: This library provide support to deal with element paths and return the paths available in particular folder into a list.
- Keras: It is open source deep learning framework written in python. It have the capability of running on the top of tensorflow framework, Theano, PlaidML or Microsoft Cognitive Toolkit. It design to create faster, easy and user-friendly deep learning models. We are using to import the layers from the models
- Tensorflow: Tensorflow is the open source software library designed by google. It use for data flow and differentiable programming over a wide range of tasks. It provide support of GPU acceleration to give use the efficiency on estimating the hyper parameters.
- Matplotlib: It is a data plotting or data visualization library for python programming language. It use to provide the mathematical graphs and image visualization. Also help in reading the images and image processing.
- Tqdm: It is a python library for providing progress of the project model.
- re: re is the library which use to search the patterns from any string and extract useful information form the library.

Outputs

```
def predict(model, img_path):
    if type(img_path) == str:
        img_path = [img_path]

    x = []
    for img in img_path:
        im = cv2.imread(img)
        x.append((cv2.resize(im, (224,224), interpolation = cv2.INTER_CUBIC)))
    x = np.array(x)
    return model.predict(x)
```

```
predict(model, norm_path.format('test') + '/normal_9.jpeg')
```

```
array([[0.5239477 , 0.47605234]], dtype=float32)
```

Fig 5.8 Prediction for input image

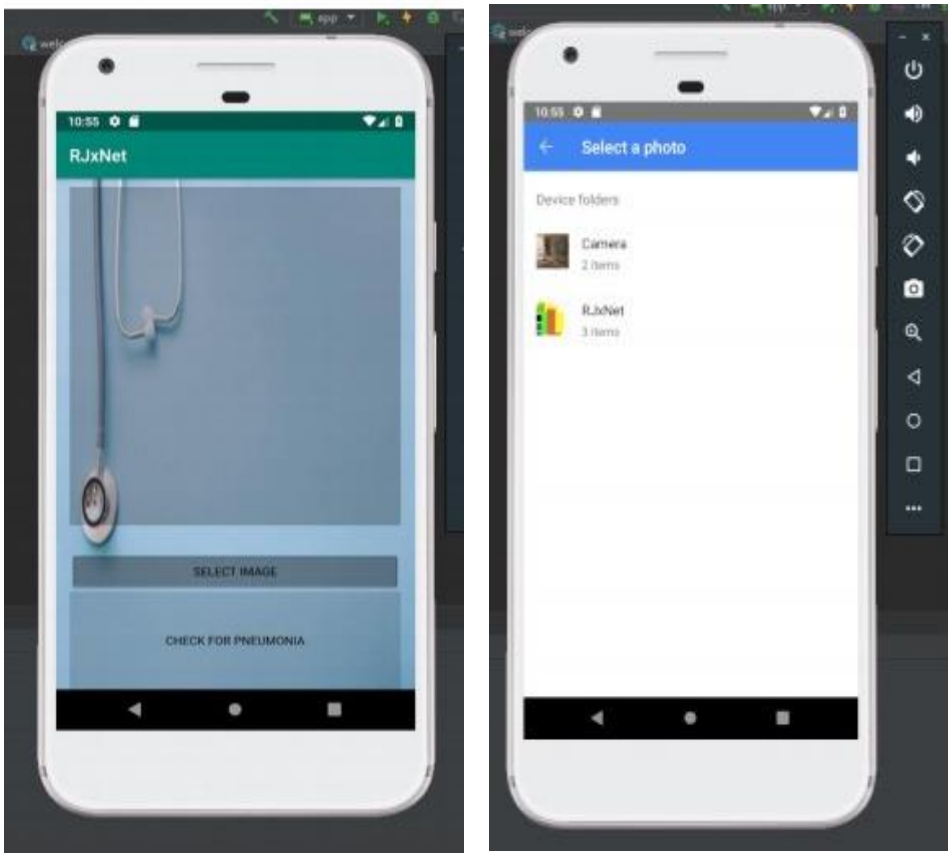

Fig 5.9 Android Application

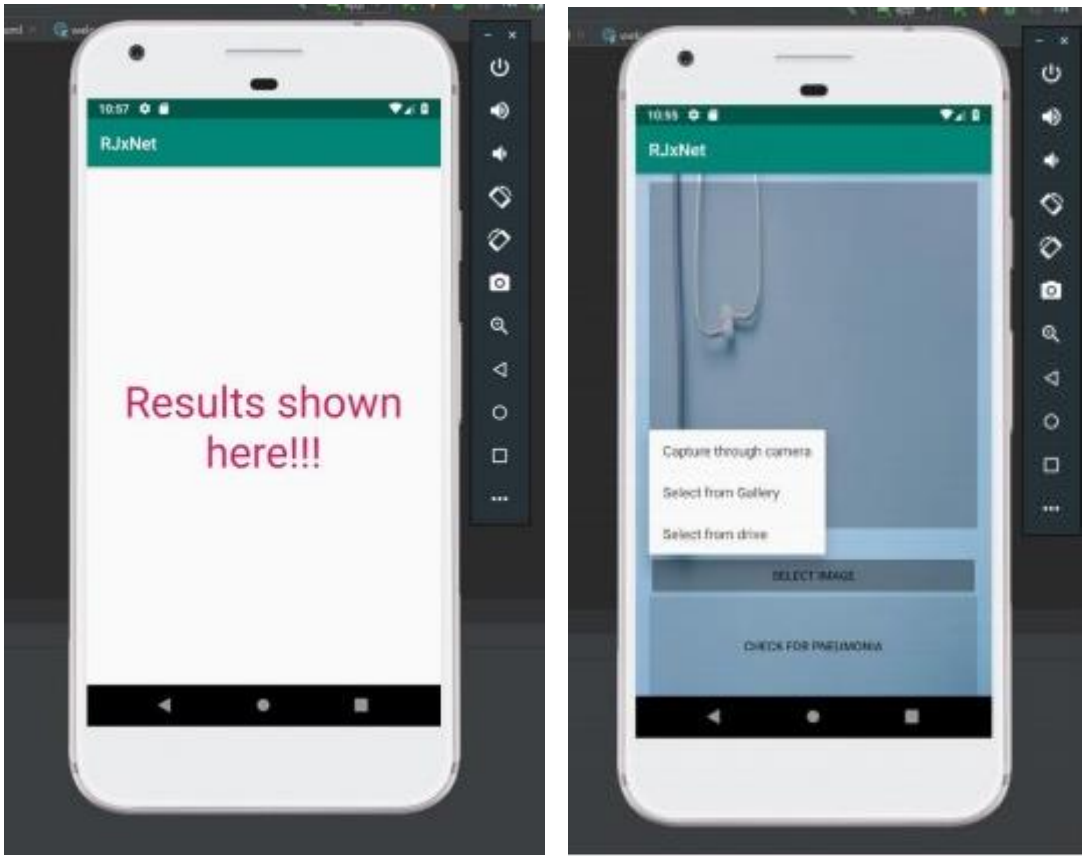

Fig 5.10 Application Output

Chapter 6:

Conclusion and Future Work

Pneumonia is severe type of infection in lungs. It is most common problem in the children of age 5 or less, and the older people ages more than 65 years. Because of pneumonia millions of children are dead in last 5 years. My project is based on a deep neural network which can helps us in detection of pneumonia by inputting Chest X-rays. This project can help persons who are suffering from some chest related problem and want to check whether it is pneumonia or not. This project can help doctor in providing the prescription as soon as pneumonia is being detected. We used different tools and techniques in developing this application. We created the android application. As we know in today's world, everyone using an android smartphone, so our application will be more accessible for the patients and doctors. We embed the pneumonia detection model into the android application by which people from remote places also can use. They don't have to use internet for detection. Also, we provide the android application at a very basic level to provide accessibility and usability to the people who don't have any knowledge with advance android applications. We able to achieve accuracy of 96.7% in our small neural network.

In future work, we will optimize our model to give high performance on real time data and try to reduce size of the android application. We will try to create another model which will classify the pneumonia caused by virus, bacteria and fungi. We are trying to provide the prescription on some basis and recommend the patients for pneumonia specific medicines.