Detection of Lung Mass from Chest X-Ray Images using Deep Learning

Arun Prasad Mohan

Department of Computer Science and Engineering, Anna University, College of Engineering (CEG) Campus, Chennai, Tamil Nadu 600025 (arun98.mohan@gmail.com)

Dr. Bhogeswar Borah, Professor

Department of Computer Science and Engineering, Tezpur University, Napaam, Sonitpur, Assam 784028 (bgb@tezu.ernet.in)

Table of Contents

Abstract

1 Introduction

1.1 Lung Cancer

1.2 Artificial Neural Networks (ANN)

1.3 Convolutional Neural Networks (CNN)

2 Literature Review

3 Methodology

3.1 Platform and Hardware

3.2 Dataset

3.3 Limitations of the Experiment

3.4 Data Pre-Processing

3.5 Building the Convolutional Neural Network

4 Experimental Results

5 Future Work

Acknowledgements

References

Source

Detection of Lung Mass from Chest X-Ray Images using Deep Learning

Arun Prasad Mohan

Department of Computer Science and Engineering, Anna University, College of Engineering (CEG) Campus, Chennai, Tamil Nadu 600025 (arun98.mohan@gmail.com)

Dr. Bhogeswar Borah, Professor

Department of Computer Science and Engineering, Tezpur University, Napaam, Sonitpur, Assam 784028 (bgb@tezu.ernet.in)

Abstract

There are more than one million cases of Lung Cancer per year in India alone. Early detection of Lung Cancer is vital in increasing the survival rate and decreasing the treatment costs. This research is aimed at building a Deep Convolutional Neural Network which uses Chest X-Rays to identify Lung Mass, and then make a comparative study by tuning the hyper parameters. A lung mass or a lung tumour can be Malignant or Benign. NIH Chest X-Ray Dataset containing more than 112000 images was used for training and testing. After pre-processing, the data was analyzed and then fed to the designed Convolutional Neural Network. Accuracy of over 96 percent was obtained in all the trials. A comparative study by varying the number of input images and varying the number of hidden layers was carried out. The accuracies obtained were compared and was found that the accuracy increased with the increase in number of hidden layers. A complete product was then ideated which when implemented would be a vital diagnostic tool to detect Lung Mass which can be used in the remote locations of our

country having just X-Ray facilities and no other advanced medical equipment like CT, also it will be able to solve the problem of less availability of expert Radiologists at such locations.

1 Introduction

Lung cancer affects more than one million people in India per year. A lot of research is being carried out to detect and cure cancer. Detecting a Lung cancer initially requires detecting a lung mass which may be benign or malignant. Detecting a lung mass from the chest X-Ray can be challenging at times and often advanced medical imaging techniques like Computed Tomography (CT) is preferred. But this is not always possible for people at remote locations as they may not have access to such sophisticated medical equipment and/or expert Radiologists may not be available. Hence detecting a lung mass from a simple chest X-Ray is necessary. Convolutional Neural Networks (CNN) are preferred for working on Images. A comparative study is carried out by designing a CNN, varying the number hidden layers and the number of input images. The Network is designed, trained and predicted on a simple CPU and neither GPUs (Graphics Processing Unit) nor TPUs (Tensor Processing Units [TPU]) are used.

1.1 Lung Cancer

Lung cancer is the most common cause of cancer related deaths in men and second most common cause in women [WHO 2014]. Lung Cancer is also called as Lung Carcinoma. It is a malignant Lung Tumour or Lung Mass which is characterized by uncontrolled cell growth in the tissues of lungs. A chest radiograph or chest X-Ray is one of the most cost effective and immediate medical imaging technique available and can be used as the first step in diagnosing Lung Cancer. A Chest Radiograph can be used to detect tumours in the lungs and then,

advanced medical imaging techniques like CT and techniques like biopsy may be performed to know if the mass is malignant or benign and to know more about the type and extent of the disease in the victim.

1.2 Artificial Neural Networks (ANN)

More than 50 years ago, in a strong search to understand the Human Brain and mimic its strengths, the development of the Artificial Neural Networks began. A biological neuron contains Dendrites, Soma and Axons. The Biological Neuron receives signals from other neurons through their Dendrites, process in Soma and output it to several other neurons through their axons. A Artificial Neuron is similar to the biological neuron in structure. A simple Artificial Neuron (Fig 1) is an information processing unit which has several links. Each Link has an associated weight and an activation function is applied to its net input to determine its output signal which is then passed on to other neurons. An Artificial Neural Network (Fig 2) is a inter connection of many such Artificial Neurons stacked in various layers. The Network is characterized by the number of (Artificial) Neurons in each layer, the way they are connected with each other and the method by which their weights are determined.

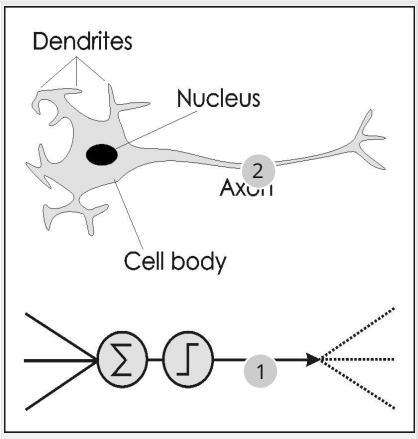


Fig 1 Artificial Neuron vs. Biological Neuron

The advantage of an Artificial Neural Network is that given an set of examples and the relationship between the input and output, the Neural Network learns the relationship between the two.

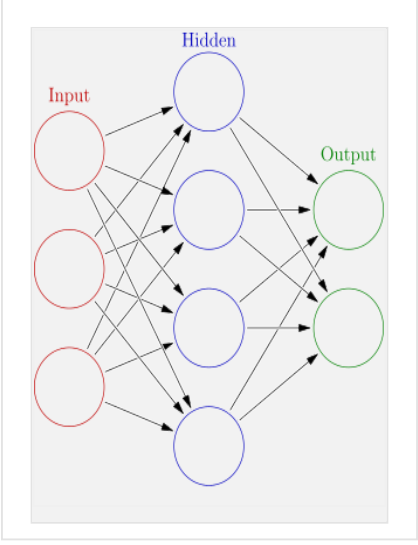


Fig 2 Sample Artificial Neural Network, each Circle represents an Artificial Neuron

Usually, when the Artificial Neural Network has more than 2 or 3 layers, it is said to be a Deep Neural Network. In our research, the standard followed is that the input layer is not taken into account while counting the number of layers.

1.3 Convolutional Neural Networks (CNN)

Considering an image as a big matrix, then a 'kernel' or convolutional matrix can be seen as a tiny matrix that is used for blurring, sharpening, edge detection, and other image processing functions. Essentially, a kernel sits on top of the big image and slides from left to right and up to down, applying a mathematical operation at each (x,y) coordinate in the original image. This operation is called Convolution and this can be used to extract features from the images and build very powerful deep learning systems.

An Convolution operation has the following steps. Select an (x,y) coordinate from the original image. Place the centre of the kernel at this (x,y) coordinate. Then multiply each kernel value by the corresponding image pixel value and take the sum of all multiplication operations. Use the same (x,y) coordinate but this time store the kernel output in the same (x,y) location as the output image. Such a layer is called a convolutional layer.

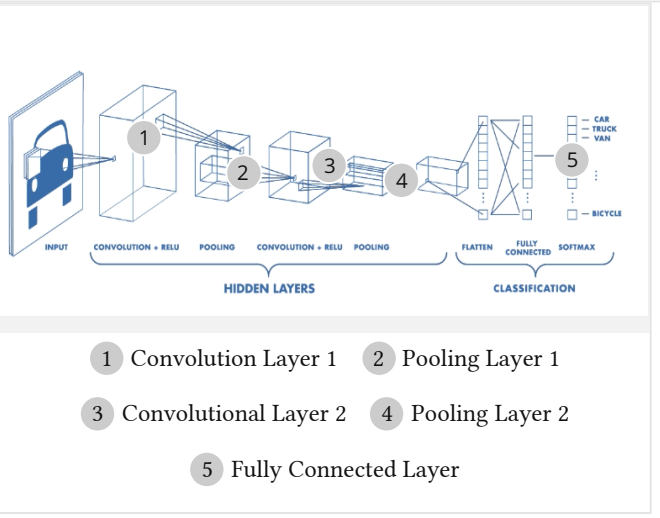


Fig 3 Sample Architecture of a Convolutional Neural Network

A Convolutional Neural Network (Fig 3) is made up of many convolutional layers, pooling and fully connected layers. Convolutional Neural Networks are similar to ordinary Neural Networks but it is assumed that the inputs to CNNs are always images. The Maxpooling Layer, Dense Layer etc... are called non-convolutional layers in the convolutional neural network. The connectivity pattern between neurons resembles the organization of the animal visual cortex.

2 Literature Review

The application of ANNs in medicine started as early as 1980s when the project "Instant Physician" was built which was trained on a set of symptoms, diagnosis and treatments. After training, the ANN could suggest the best diagnosis and treatment based on the symptoms which were given as inputs to it. Stanford University has built ChesXNet, which is an 121 layer deep Convolutional Neural Network to detect Pneumonia from the NIH Chest X-Ray Dataset [CheXNet]. CheXNet could defeat practicing Radiologists in terms of predicting Pneumonia from Chest X-Rays. The article by Kyle O'Brien [Airplane Classification] on

Airplane image classification using keras gives an excellent overview on how to construct an Binary Classifier using Keras with Tensorflow as backend. The CNN Designed in this research is inspired from this article. The other papers referred to and relevant are Thoracic Disease Identification and Boosted Cascaded Convnets.

3 Methodology

The fundamental goal of this research is to build an Convolutional Neural Network which gets Chest Radiographs as inputs and detect the presence of Lung Mass in them and then to make a comparative study by tuning the Hyper Parameters.

The CNN to be constructed will be having 'n' hidden layers with each hidden layer meaning a convolutional layer and an activation function 'relu'. Any number of hidden layers can be added by defining the variable 'n'. At last, after all the hidden layers, a max pooling layer, and an output layer is to be added.

Now, using different number of input images, we increase the number of hidden layers gradually till the maximum limit of the CPU support. Finally we plot graphs with the results obtained and infer from them.

3.1 Platform and Hardware

The entire research was carried out on an 'MacBook Air 13" early 2015' Laptop Computer. The processor is an 1.6 GHz Intel Core i5 processor. It has 4 GB 1600 MHz DDR3 RAM (split into 2 RAMs of 2 GBs each) and an inbuilt graphic card "Intel HD Graphics 6000 1536 MB". The secondary storage capacity is 121.12 GB. The Operating System is "macOS High Sierra Version

10.13.5". The deep Learning libraries used are Keras, Tensorflow and the Programming Language used is Python version 3.6.2rc2.

3.2 Dataset

> NIH Chest X - Ray dataset

The dataset considered for the research is NIH Chest X - Ray dataset [Kaggle Dataset] . The database considered has minute variations over the originally published massive open online database [NIH].

3.2.1 Understanding the Dataset

> 1,12,120 Chest X-Rays from 30,805 unique patients.

The NIH Chest X-Ray dataset is one of the largest publicly available Medical Image Dataset. It contains 1,12,120 Chest X-Rays from 30,805 unique patients. The authors of the dataset text mined the disease labels for the radiographs from the associated radiology reports. Since the labels were text mined, the accuracy is not 100 percent and is expected to be more than 90 percent. There are 15 classes with 'no finding' and 'lung mass' as two of them. The terms radiographs and Chest X-rays are used synonymously in this report. Each radiograph (Fig 4) is of size 1024*1024*3, where the '3' represents the three colour channels RGB. The dataset contains 12 zip files each of size ranging from ~2 GB to ~4GB. Each zip file contains

approximately 10,000 images. A CSV file containing all the patient data and disease labels is available.

The 15 classes are Atelectasis, Consolidation, Infiltration, Pneumothorax, Edema, Emphysema, Fibrosis, Effusion, Pneumonia, Pleural thickening, Cardiomegaly, Nodule, Mass, Hernia and No findings.

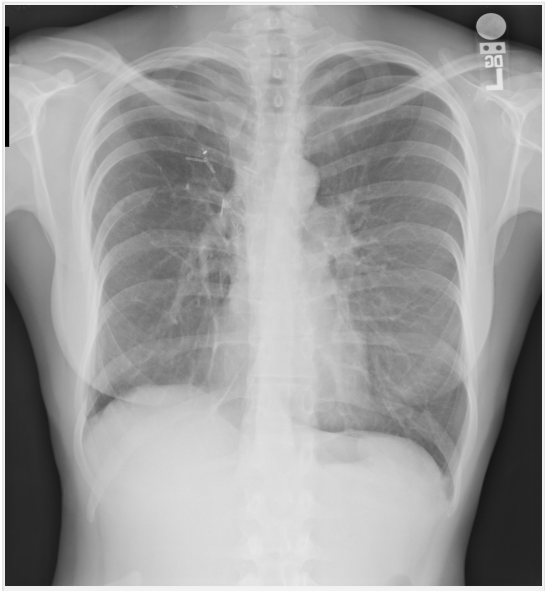


Fig 4 Sample X-Ray Image

3.3 Limitations of the Experiment

Due to hardware constraints and time constraints, the number of images used out of the available 112000+ images, the size of images to which they were reduced to and the number of hidden layers added, had to be restricted. For the purpose of comparison of results, a few hyper parameters of the Convolutional Neural Network were fixed, by taking the fact into

account that a CPU is used and not a GPU or TPU. The **Minimum number of neurons is taken to be 10, Maximum number of neurons is taken to be 50, kernel size is (3,3), Number of epochs is 5 and Batch Size was fixed to be 10 and no padding is used.**

3.4 Data Pre-Processing

3.4.1 Cleaning the CSV File

For the purpose of this research, we do not need all the 15 classes but only lung mass. Hence, all the images having "mass" as one of their labels are re-labelled as "1" and all the other images are labelled "0". This is done with the help of shell scripting (Code 1). Hence, now the problem is converted into a binary classification problem. All the other columns in the CSV file are dropped as they are not required for the problem.

```
#Change the extension of Data_Entry_2017.csv to .txt
sed -e "s/,/ /g" Data_Entry_2017.txt >> test.txt
awk '{print $1,$2}' test.txt>>test2.txt
awk '{print $1}' test2.txt>>c1.txt
awk '{print $2}' test2.txt>>c2.txt
sed "s/.*Mass.*/1/g" c2.txt>>c3.txt
sed "s/.*[^1].*/0/g" c3.txt>>c4.txt
paste -d":" c1.txt c4.txt | sed "s/:/ /g" >> Data.txt
sed "s/ /,/g" Data.txt>>Data_Cleaned.csv
rm test.txt
rm test2.txt
rm c1.txt
rm c2.txt
rm c3.txt
rm c4.txt
rm Data.txt
cut -f2 Data_Cleaned.csv | sort | uniq | wc -l #To check tha
    t we didnt loose any tuple
cut -f2 Data_Entry_2017.csv | sort | uniq | wc -l
```

```
18.                     #Verifying that with the count of the origin
    al file
```

Code 1 Shell Script using sed, awk to clean the CSV file

Now in the CSV File (Fig 5), we have two columns, one with the file names and the other with the binary label (i.e. either 1 or 0).

	A	B
1	Image Number	Class
2	00000001_000.png	0
3	00000001_001.png	0
4	00000001_002.png	0
5	00000002_000.png	0
6	00000003_000.png	0
7	00000003_001.png	0
8	00000003_002.png	0
9	00000003_003.png	0
10	00000003_004.png	0
11	00000003_005.png	0
12	00000003_006.png	0
13	00000003_007.png	0
14	00000004_000.png	1
15	00000005_000.png	0

1 Radiograph File Name 2 Binary Class, 1 indicates presence of Lung Mass

Fig 5 First fifteen tuples after cleaning the CSV

3.4.2 Image Resizing

Since Each Image is of size, 1024 * 1024, the number of parameters it would create will be very high and normal CPUs will take a very large time to learn them. Hence resizing the images is unavoidable for our research. Each Radiograph is resized (Code 2) to 30 * 30 (Fig 6).

```
1. #Before Running the code, put all the original images in a
```

```
new folder
#Delete any previous existing folders
#It may bring trace back errors
from PIL import Image
import os,sys
path = #Path to folder containing the radiographs
dirs = os.listdir(path)
def resize():
 for item in dirs:
 if os.path.isfile(path+item):
 im=Image.open(path+item)
 f,e=os.path.splitext(path+item)
 imResize=im.resize((30,30),Image.ANTIALIAS)
 imResize.save(f+'resized.png','PNG',quality=90)
resize()
```

Code 2 Python Code to resize all images

Fig 6 An Resized 30*30*3 Radiograph

3.4.3 Data Analysis

We find the percentage of X-Rays with Lung Mass and without Lung Mass (Fig 7) from the labels available.

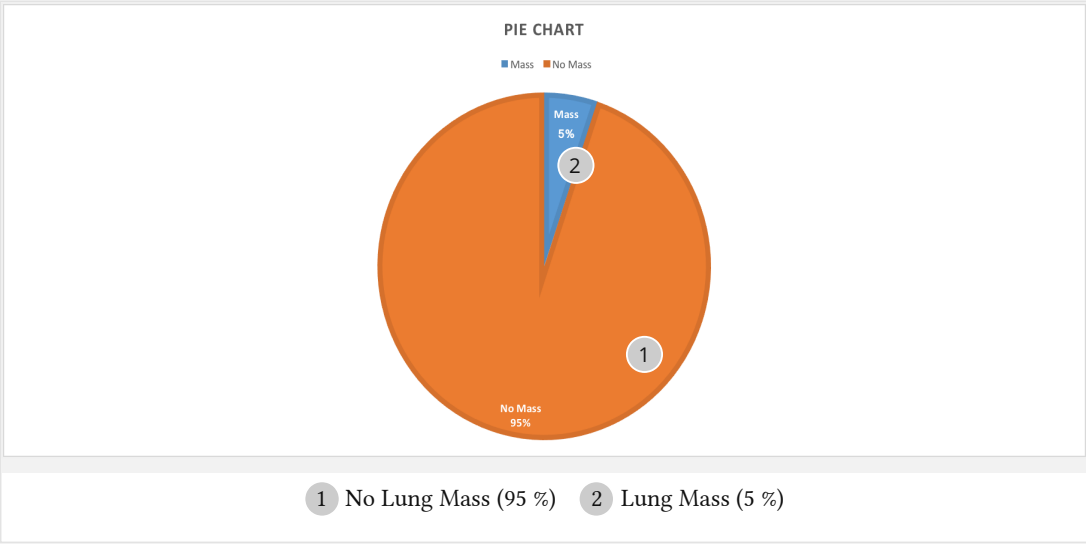


Fig 7 Pie Chart Representing the ratio of chest radiographs with lung mass and without lung mass

3.5 Building the Convolutional Neural Network

3.5.1 Importing Necessary Libraries

```
1. import glob
2. import numpy as np
3. import os.path as path
4. import cv2
5. from keras.models import Sequential
6. from keras.layers import Activation, Dropout, Flatten,
   Dense, Conv2D, MaxPooling2D
7. from sklearn.metrics import accuracy_score, f1_score
```

Code 3 Importing Libraries

(Code 3) The 'glob' module is used to find all the pathnames matching a specified pattern according to the rules used by the Unix shell. But the results are returned in an arbitrary order. The numpy module is the fundamental package in Python for scientific computing. It contains a powerful N-dimensional array object, sophisticated functions and much more. It can also be seen as a general purpose array-processing package. The 'os' module provides an portable way of using operating system dependent functionality. OpenCV or 'Open Source Computer Vision Library' is an open source computer vision and machine learning software library as the name says. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

Keras is a high-level neural networks API, written in Python and capable of running on top of Tensorflow, CNTK or Theano. We use Tensorflow as it helps in fast experimentation and to save time. We import Sequential model, Activation, Dropout, Flatten, Dense, Conv2D and Maxpooling layers from the Keras package.

3.5.2 Reading images

```
1. PATH_ABS = #Path to the folder containing the radiographs
2. file_paths = glob.glob(path.join(PATH_ABS, '*.png'))
3. file_paths.sort()
4. images = [cv2.imread(path) for path in file_paths]
5. images = np.asarray(images)
6. images=images/255
```

Code 4 Reading all the images

(Code 4) The absolute path to the folder containing all the images is taken and then paths to all the files having .png as their extension are stored. Since the paths are returned in an

random order, we sort them with the sort() function so that the paths are now in ascending order, similar to the order of file names in the CSV file containing their binary labels. Now all the images are read using the OpenCV package and then each image is converted into a numpy array using the np.asarray function.

Now we scale each image so that each pixel value lies between 0 and 1. The maximum value of each pixel is 255, hence we divide each pixel by the number 255 (making use of python broadcasting) so that all the resultant pixels lie between 0 and 1.

3.5.3 Reading all the labels

```
1. #Extract only the second column from Data_Cleaned.csv and st
   ore it in a separate file lbls.csv
2. labels = np.zeros(n_images)
3. import csv
4. file=open("#Path to the CSV File","r")
5. reader=csv.reader(file)
6. i=0
7. for line in reader:
8.  labels[i]=line[0]
9.  i=i+1
10. for i in range(n_images):
11.  labels[i]=int(labels[i])
```

Code 5 Reading all the labels

(Code 5) The 'csv' module is imported and is used to import the lbls.csv file. An empty array is created with size equal to the number of images we read. Now using a for loop we read all the labels and then convert each label to an integer.

3.5.4 Splitting Training , Test Data and Visualizing

```
TRAIN_TEST_SPLIT = 0.9
split_index = int(TRAIN_TEST_SPLIT * n_images)
shuffled_indices = np.random.permutation(n_images)
train_indices = shuffled_indices[0:split_index]
test_indices = shuffled_indices[split_index:]
x_train = images[train_indices, :, :]
y_train = labels[train_indices]
x_test = images[test_indices, :, :]
y_test = labels[test_indices]
print("Trying to Visualize a few Images")
import matplotlib.pyplot as plt
def visualize_data(positive_images, negative_images):
 figure=plt.figure()
 count=0
 for i in range(positive_images.shape[0]):
 count+=1
 figure.add_subplot(2,positive_images.shape[0],count)
 plt.imshow(positive_images[i,:,:])
 plt.axis('off')
 plt.title("1")
 figure.add_subplot(1,negative_images.shape[0],count)
 plt.imshow(negative_images[i,:,:])
 plt.axis('off')
 plt.title("0")
 plt.show()
N_TO_VISUALIZE=3
positive_example_indices=(y_train==1)
positive_examples=x_train[positive_example_indices,:,:]
positive_examples=positive_examples[0:N_TO_VISUALIZE,:,:]
negative_example_indices = (y_train == 0)
negative_examples = x_train[negative_example_indices, :, :]
negative_examples = negative_examples[0:N_TO_VISUALIZE, :,
:]
visualize_data(positive_examples, negative_examples)
```

Code 6 Splitting of Training and Test Data, Visualizing Examples

(Code 6) The training data is taken to be 90 % of the input data while the remaining 10 % is considered to be the Test Data on which the accuracy is measured. A few positive and negative examples are then read to check the validity of the code (Fig 8).

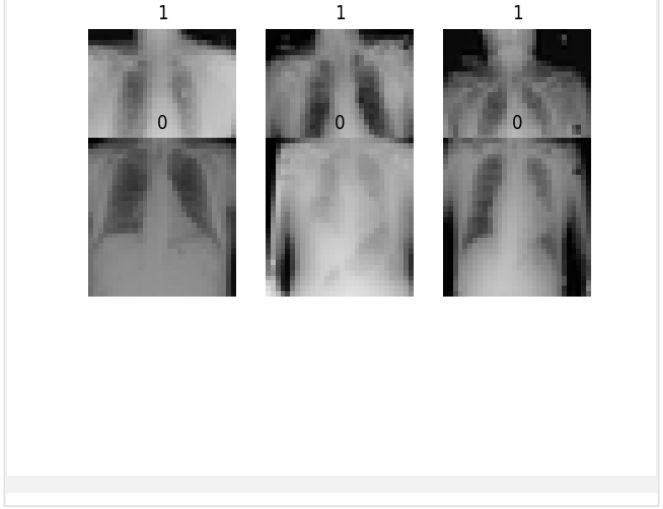


Fig 8 Visualizing positive and negative examples

3.5.5 CNN

```
N_LAYERS= #Insert Number of Layers
def cnn(size,n_layers):
 MIN_NEURONS= #Insert Minimum Number of Artificial Neurons i
n each Layer
 MAX_NEURONS= #Insert Maximum Number of Artificial Neurons i
n each layer
 KERNEL=(_,_) #Define the Kernel Size
 steps=np.floor(MAX_NEURONS/(n_layers+1))
 neurons=np.arange(MIN_NEURONS,MAX_NEURONS,steps)
 neurons=neurons.astype(np.int32)
 model=Sequential()
 for i in range(0,n_layers):
```

```
11.  if i==0:
12.  shape=(size[0],size[1],size[2])
13.  model.add(Conv2D(neurons[i],KERNEL,input_shape=shape))
14.  else:
15.  model.add(Conv2D(neurons[i],KERNEL))
16.  model.add(Activation('relu'))
17.  model.add(MaxPooling2D(pool_size=(2,2)))
18.  model.add(Flatten())
19.  model.add(Dense(MAX_NEURONS))
20.  model.add(Activation('relu'))
21.  model.add(Dense(1))
22.  model.add(Activation('sigmoid'))
23.  model.compile(loss='binary_crossentropy',optimizer='adam',m
    etrics=['accuracy'])
24.  model.summary()
25.  return model
26. model=cnn(size=image_size,n_layers=N_LAYERS)
```

Code 7 CNN

(Code 7) The function 'cnn' gives a general format for building the CNN. We can define the number of hidden layers by N_LAYERS variable. The for loop inside the function creates the required number of hidden layers. Each Hidden layer would have a Convolutional layer, and an activation function, which we have considered to be 'relu' in our project. Finally, we add non-convolutional layers like Maxpooling, Dense and Flatten. We consider the loss function to be 'binary cross entropy' and the optimizer to be 'adam' as it is one of the most stable and trusted optimizer. We measure the performance of the CNN by its accuracy.

3.5.6 Predicting

```
1. EPOCHS = #Enter the Number of Epochs
2. BATCH_SIZE = #Enter the batch size, that is after 'x' number
    of images the weights are updated
3. model.fit(x_train, y_train, epochs=EPOCHS, batch_size=BATCH_
   SIZE, verbose=0)
```

```
4. test_predictions=model.predict(x_test)
5. test_predictions=np.round(test_predictions)
6. accuracy=accuracy_score(y_test,test_predictions)
7. print("Accuracy: "+str(accuracy*100)+"%") #the accuracy is m
   ultiplied with 100 to get in percentage
```

Code 8 Training the model, finding accuracy

(Code 8) The 'model.fit' function uses all the data like the training images, training labels, epochs and batch size to train the model. The model is then evaluated. Firstly, we predict using model.predict and then use accuracy_score to evaluate the model by comparing with the ground truth and print the accuracy.

4 Experimental Results

We consider 4999 images and then 14999 images for training and testing. 90 % of the images is used for training and the remaining 10 % is used for testing. For both 4999 images and 14999 images, we increase the number of hidden layers, starting with 1 hidden layer, and record their accuracies.

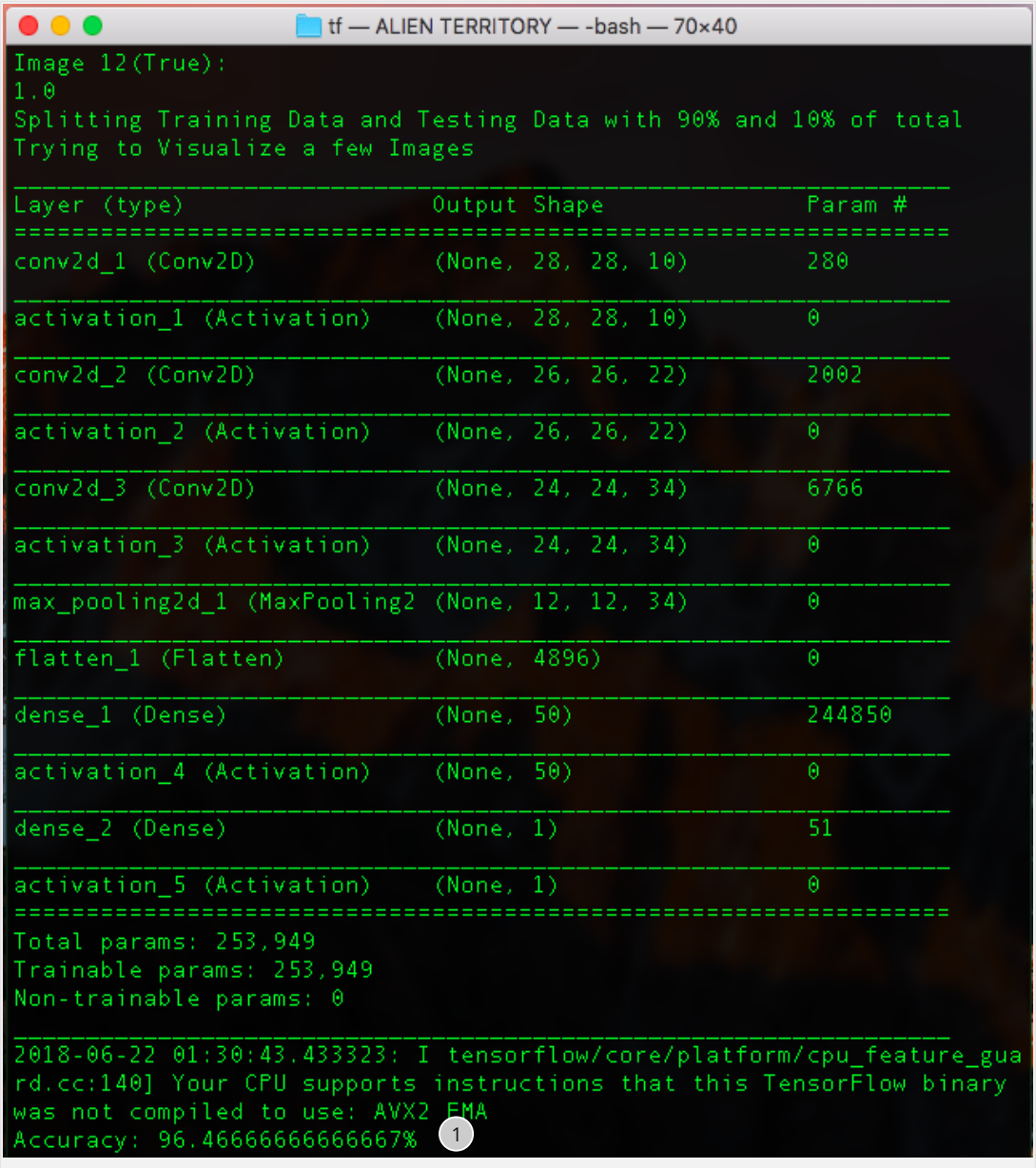
Table 1 Accuracies with different number of Hidden Layers2

	A	B	C
1	Number of Images Used	Number of Hidden Layers	Accuracy (Percentage)
2	4999	1	96
3	4999	3	97.2
4	4999	5	96.6

5	14999	1	96.3
6	14999	3	96.5
7	14999	5	96.6
8	14999	7	97.1

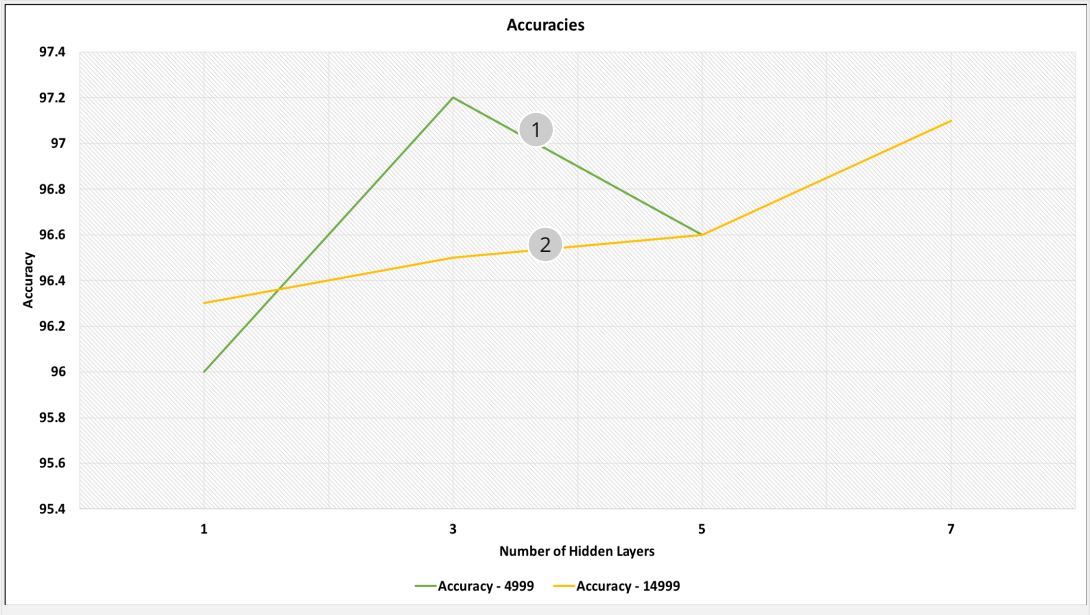
Note Minimum number of neurons = 10, Maximum Number of Neurons = 50, Kernel Size (3,3), Number of Epochs = 5, Batch Size=10

From the above table (Table 1) we can see that for all the trials with different combination of the number of images used and the number of hidden layers added, we get accuracies over 96 %. Initially, 4999 images were used with a single hidden layer and the accuracy for the model was found to be 96 %. Then the number of layers was gradually increased to 3 and then 5, the accuracies were found to be 97.2 % and 96.6 % correspondingly. Then 14999 images were considered and accuracy was calculated with 1 hidden layer. The accuracy was found to be 96.3 %. Then gradually the number of layers was increased to 3,5 and then 7. The corresponding accuracies were 96.5 (Fig 9), 96.6 and 97.1.



1 Accuracy = ~96.5 %

Fig 9 Result showing accuracy of 96.5 %, for 14999 imgaes and 3 hidden layers



1 For 4999 Images 2 For 14999 Images

Fig 10 Comparison of Accuracies for different hyperparameters

> When approximately 15000 Images were used, we can find that, as the ConvNet gets deeper and deeper the accuracy is found to increase steadily.

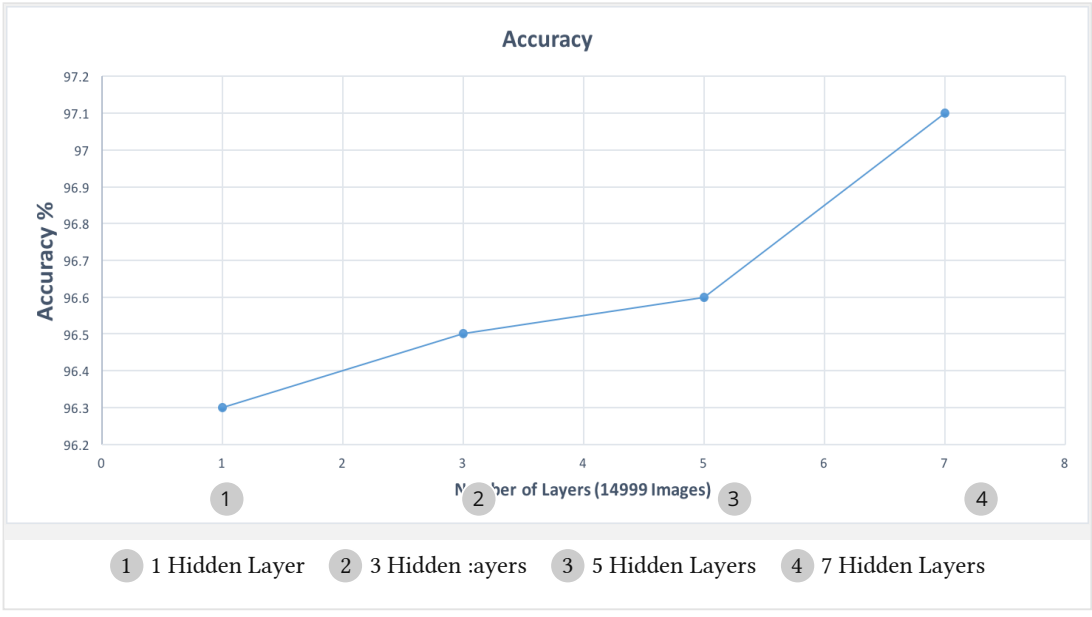


Fig 11 Considering 14999 Images alone

When approximately 15000 Images was used, we can clearly see from the plots (Fig 10 ,Fig 11) that, as the ConvNet gets deeper and deeper the accuracy is found to increase steadily. When moving from 5 to 7 hidden layers we can find that there is a sharp rise in the accuracy from 96.6 % to 97.1 %. Also we can observe that when the number of input images used to train and

test the CNN is increased, the increase in number of hidden layers increased the accuracy steadily without any drop.

5 Future Work

The accuracy can be improved further by increasing the number of input images used to train and test the CNN. Also with the help of GPUs or TPUs, the number of hidden layers added to the CNN can be increased. As seen earlier, Stanford University's CheXNet is an 121 Layer Deep Convolutional Neural Network for the detection of Pneumonia. It is based on the same dataset which is used in our research. The accuracy of CheXNet is so high that it's accuracy is better than practicing radiologists. When the number of layers in our CNN is increased to such an extent, we strongly believe that our ConvNet can also reach such an accuracy in predicting Lung Mass from Chest X-Rays.

Further, an complete product can be developed as follows. A Raspberry Pi like microprocessor can be used and connected to the Cloud Server. The Cloud Server should be used to run the Network. After training the network once, Chest X-Rays can be uploaded anytime to the cloud from the Raspberry Pi using APIs and results can be obtained within seconds. The drawback of such an approach would be that, the hospitals and health centres in the remote locations will have to be having internet facilities and X-Ray machines. We strongly believe that continued research on this will definitely yield an product which can save a lot of lives in the near future.

Acknowledgements

Firstly, I would like to thank my parents, brother and the almighty for being with me always and helping me finish the research successfully. Without their prayers, unconditional love, affection and belief it would have been impossible to complete this research.

I would like to thank the National Science Academies (Indian Academy of Sciences, Bangalore., Indian National Science Academy, New Delhi and The National Academy of Sciences, Allahabad) for providing me with this fabulous Research Fellowship. This has been a great platform for me to learn, enjoy and grow, and has helped me understand what research is all about.

I would like to express my deepest gratitude to my guide, Dr. Bhogeswar Borah for his excellent guidance, care, patience and providing me with an excellent atmosphere and freedom for carrying out my research.

I express my sincere thanks to Professor Dr. V. Vetriselvi, Department of Computer Science and Engineering, College of Engineering Guindy, Anna University, for having faith in me and providing the Science Academies with a Recommendation Letter.

My gratitude towards Tezpur University for providing me with the permission and support to carry out this research in the University premises, and providing me with the hostel and dining facilities.

I express my heartfelt thanks to Mr. Arun Sriraman, Data Scientist, Astrazeneca India Private Ltd. for supporting and motivating me throughout the project.

It was a gift to have such wonderful friends in the University who gave me moral support throughout the research and helped me feel this place home.

References

1. https://en.wikipedia.org/wiki/Tensor_processing_unit
2. *World Cancer Report 2014*. World Health Organization. 2014. pp. Chapter 1.1. ISBN 92-832-0429-8.
3. CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning. Pranav Rajpurkar et. al.
4. https://medium.com/@kylepob61392/airplane-image-classification-using-a-keras-cnn-22be506fdb53
5. Thoracic Disease Identification and Localization with Limited Supervision, Zhe Li1∗, Chong Wang3, Mei Han2∗, Yuan Xue3, Wei Wei3, Li-Jia Li3, Li Fei-Fei31Syracuse University, 2PingAn Technology, US Research Lab, 3Google Inc.
6. Boosted Cascaded Convnets for Multilabel Classification of Thoracic Diseases in Chest Radiographs, Pulkit Kumar*Paralleldots, Inc. et. al.
7. https://www.kaggle.com/nih-chest-xrays/data
8. https://www.nih.gov/news-events/news-releases/nih-clinical-center-provides-one-largest-publicly-available-chest-x-ray-datasets-scientific-community

Source

1. Fig 1: http://www.webpages.ttu.edu/dleverin/neural_network/neural_networks.html
2. Fig 2: https://en.wikipedia.org/wiki/Artificial_neural_network
3. Fig 3: https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html