

Lung Cancer Detection Using CNN

A Report for the Evaluation of Project 2

Submitted by

Mohammed Tahir

(1613101405 / 16SCSE101692)

in partial fulfillment for the award of the degree

Of

Bachelor of Technology

In

Computer Science and Engineering

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

**Under the Supervision of
Dr.Kuldeep Singh Kaswan
Assoc.Professor**

APRIL / MAY- 2020



**SCHOOL OF COMPUTING SCIENCE AND
ENGINEERING**

BONAFIDE CERTIFICATE

Certified that this project report “LUNG CANCER DETECTION USING CNN”
is the bonafide work of “MOHAMMED TAHIR (1613101405)” who carried out
the project work under my supervision.

SIGNATURE OF HEAD

Dr. MUNISH SABHARWAL,
PhD (Management), PhD (CS)
Professor & Dean,
School of Computing Science &
Engineering

SIGNATURE OF SUPERVISOR

Dr. KULDEEP SINGH KASWAN,
PhD
Assoc Professor,
School of Computing Science &
Engineering

Table of Contents

Abstract	1
1. Introduction	2
2. Background	4
3. Deep Learning	8
3.1 Artificial Neural Network	8
3.2 Convolution Neural Network	11
3.3 Deep Learning Libraries (TensorFlow and Keras)	12
3.4 TensorBoard	12
4. Lung Cancer debrief	15
4.1 Causes	16
4.2 Risk factors	17
4.3 Diagnosis	17
4.4 Treatment	17
5. Methodology	18
5.1 Problem Statement	18
5.2 Data source	18
5.3 Model	19
5.3.1 Model description	20
5.3.2 Model Summary	21
5.3.3 Model Configuration	22
5.3.4 Visualization of the model	28
5.3.5 Hyperparameter Tuning	29
5.3.6 Results and Discussion	32
5.3.7 Confusion Matrix	35
6. Conclusion and Future Scope	36
7. References	37

Abstract

The recent surge of Deep Learning has led to breakthrough advancements in almost every field of its application. A particular deep learning architecture, arguably the most popular one is the Convolution Neural Networks. The interest in convnets has seen an exponential increase due to their effectiveness and scalability. CNNs have become the go-to solution for image data problems and has provided results that are at par with if not better than human standards. The simplicity of the CNN architecture is another big factor of its success. The image processing and classification capabilities of CNN have found great usage in medical field, making it possible to detect and classify diseases as severe as Cancer effectively for the sake of better care. In this project, we've initiated an elaborate study of Convolution Neural Networks, build multiple architectures from scratch and furthered our understanding with the preparation of an elementary dog-cat CNN classifier model followed by a more extensive CNN model for detection of lung cancer in a patient. The project is built on Google's interactive and versatile cloud platform for AI development Google Colaboratory, using the open-source neural network library 'Keras' for model development and libraries such as matplotlib and tensorboard (tensorflow) for result plotting and analysis. Data for training and testing our model was extracted from the ' LUNA2016 medical image database '. The model was tuned using Grid-Search and achieved over 97% test accuracy in its final iterations. To culminate, we have enlisted some future-work prospects like De-convolution/Translated-Convolution, implement one or more named CNN networks like Inception or Alexnet, test the model on larger images etc.

Introduction

The use of Convolutional Neural Networks can be traced back to the nineties for character recognition purposes(Le Cun et al., 1997) but it wasn't until the 2012 AlexNet that it grew to the widespread acclaim that we know of today. In less than a decade, researchers have progressed from single-digit layers in a CNN model to triple-digits, integrating various other data science techniques to create umpteen possible configurations. The architecture and working of a Convolution Neural Network can be understood better by a simple binary image-classification model such as a dog-cat CNN that classifies an image as one of the two preset classes, Dog or Cat. The architecture of CNN models is predominantly similar, beginning with a few convolution layers that apply various convolution filters or kernels or masks to the input image. An image is represented as a matrix of gray-scale or color intensity values at the pixel represented by the matrix cell indices. Each filter/kernel can be thought of as a feature extractor that extracts positions of that particular feature represented by the kernel, on the input image. The convolution operation hence produces a feature map . The convolution layer are each followed by a pooling layer generally that sub-samples an image to provide lower dimensional matrices for better computations. A series of convolution + pooling layers are followed by a number of densely or fully connected layers after flattening the output. The fully connected layers operate as a typical neural network and finally classifies to binary (sigmoid/relu) or multiple classes (softmax). The medical field is a likely ground for machine learning practice and application, as medical regulations allow increased sharing of anonymized data for the sake of better care. The field is pretty young and flourishing at the forefront of technology. This problem is interesting and

promising in that it has impactful implications for the future of healthcare, deep learning applications affecting personal decisions, and computer vision in general. Cancer is a disease that needs no documentation of its perils. It could be present in almost any part of the body like lungs, breast, prostate, skin etc. Lung cancer is the leading cause of cancer-related deaths worldwide. The National Lung Screening Trial (NLST), a randomized control trial in the U.S. including more than 50,000 high-risk subjects, showed that lung cancer screening using annual low-dose computed tomography (CT) reduces lung cancer mortality by 20% in comparison to annual screening with chest radiography. Due to the absence of any true remedy or cure for the disease, detection in the early stage is crucial in preventing its spread. In this project, we are interested to train a convolution neural network and then use the trained model to solve a binary image classification problem to classify as positive or negative results.



Figure 1: 2D CT scan slice containing a small (5mm) early stage lung cancer nodule

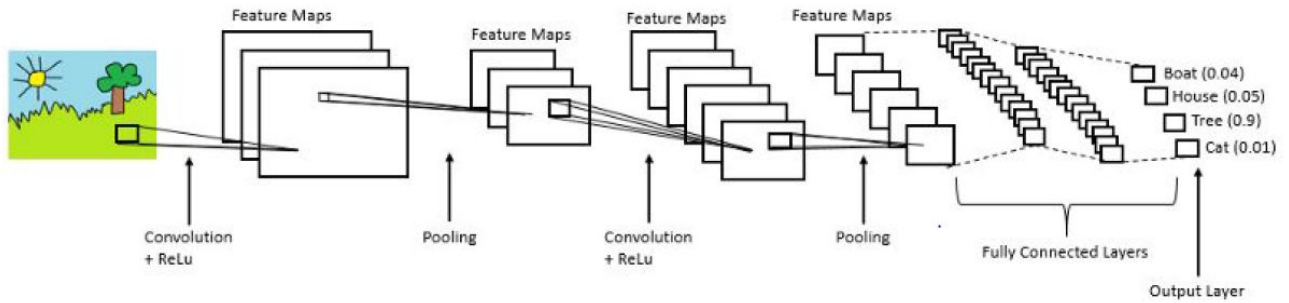
Background

Before getting started with model construction and tuning we gleaned all the necessary knowledge from sources like

- Deeplearning.ai by Andrew Ng Course 1-2
- Deeplearning.ai by Andrew Ng Course 4
- Keras documentation
- Towards-data-science blogs

The deeplearning.ai courses provided an elaborate explanation of Neural Networks and CNN with the development of elementary layers, activations and back-propagation mechanisms without any predefined framework. This helped in getting a robust understanding of the maths involved and the use of each parameter or argument that is used later on. The course then talked about improving neural networks by use of hyperparameter tuning, regularization, and optimization. It explained about the training-validation-test data split, bias and variance, normalization, gradient-checking, types of regularization, weight initialization, overfitting/underfitting and other such significant aspects of model creation and optimization.

Course-4 of the specialization got us introduced with the use of Convolution Neural Networks for computer vision tasks. It got us familiar with the CNN building blocks and vocabulary, talking more about the mathematical operations involved in convolution. We created our first CNN model as a programming assignment and moved on to create another one from scratch, this time a cat-dog image classifier.



The elaborate Keras documentation next helped us to build our first Convolution Neural Network to classify images with a ‘Dog’ or ‘Cat’ label. The model was built using the sequential API in Keras and consists of three convolution-2D layers followed by a max-pooling-2D layer each. The final out is then flattened and fed to a densely connected layer feeding it to the final output layer with one node operating the sigmoid function and providing output:

0 for cat

1 for dog

The model inputs images of dimensions 150 x 150, using 2000 images for training and 800 images for validation. The number of images is further increased by making use of the Image Processing class in Keras called ImageDataGenerator which generates batches of tensor image data with real-time data augmentations such as zooming, flipping, scaling etc. The model was compared on categorical crossentropy and binary crossentropy loss functions and RMSprop optimizer and adagrad optimizers. After getting a good grasp at model creation and tuning, we needed to visualize and analyze the model results and its architecture. The Python libraries such as seaborn and matplotlib were a great help in achieving the purpose. The tensorflow visualization toolkit, TensorBoard opened new horizons however.

It made it easier to understand, debug and optimize the Tensorflow code. It also made it easier to present our model architecture through block diagrams and flow charts and analyze the accuracy and loss through graphs with a large number of other unexplored features which we never seem to end. All this was possible with a few lines of code to enable tensorboard on our already existing model definition and see it all on a locally hosted browser window.

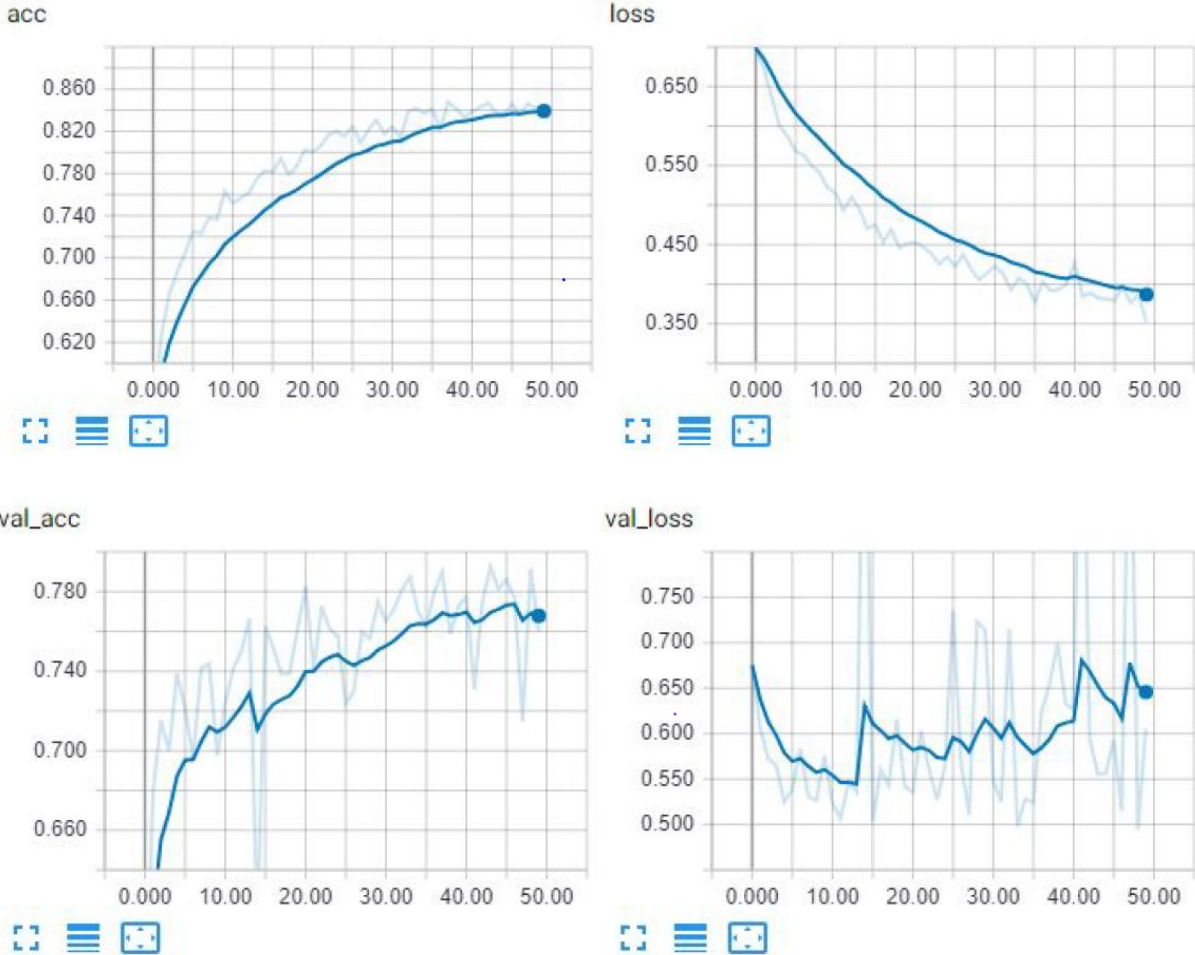
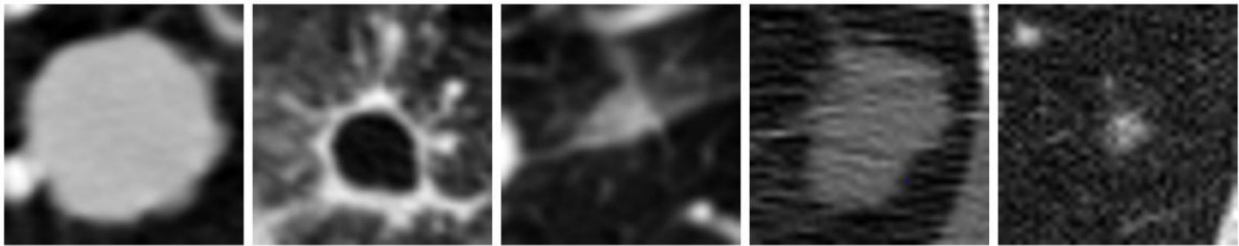
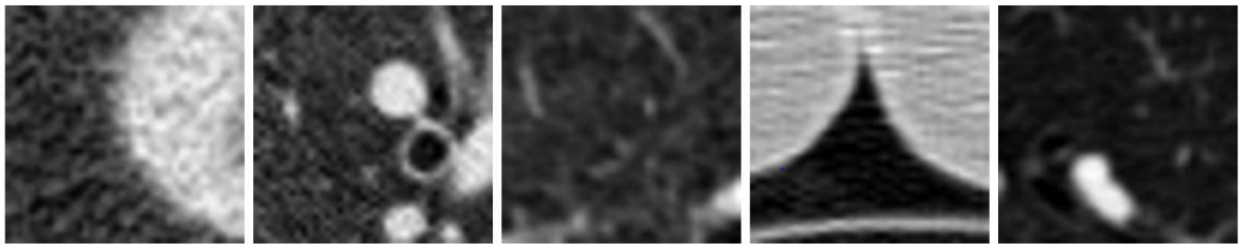


Figure : Tensorboard graphs for training and validation acc/loss of the Cat-Dog CNN model

Our final task was to apply our learnings to create a Cancer prediction CNN model. We indulged ourselves into extensive Research Paper readings and learned as much as we could about Cancer and the application of Deep Learning to the Medical domain. We decided to move forward with the detection of Lung Cancer nodule model as the model was matched our caliber and the data was freely available from the LUNA2016 Medical Image Database.



Examples of cancerous images



Examples of non-cancerous images

Deep Learning

Artificial Neural Network

An ANN mimics the working of a human neural system by operating on hidden layers consisting of neurons that are connected to each other and activate on stimuli from the neighbouring neurons. The various methods involved and terminology related to ANNs have been briefed below.

Preprocessing

Deep neural network (Convolution neural network in this case) is used. The Deep neural network uses multiple hidden layers between input and output layers. Every hidden layer can have different no. of units. In fact, no. of hidden layers and units in each hidden layer are also hyperparameters for the deep learning algorithm. Here 'L' stands for the no. of hidden layers in a neural Network. In deep learning networks, each layer of units trains on a distinct set of features based on the previous layer's output. The further we advance into the neural net, the more complex the features nodes can recognize, since they aggregate and recombine features from the previous layers. Due to this property deep nets can handle very large high dimensional data with large no. of parameters.

Usable Function

Sigmoid function

A Sigmoid function is a mathematical function which has a “S” shaped curve. The sigmoid function is represented by the equation

$$\sigma(t) = 1 / (1 + e^{-t})$$

The sigmoid function has a domain of real numbers and returns the value -1, 1 or 0, 1

depending upon the convention used. The values of

$\sigma(t)$ depends

if $t \rightarrow \infty$ then $\sigma(t) = 1$

if $t \rightarrow -\infty$ then $\sigma(t) = 0$

Hyperbolic tangent function (tanh)

The hyperbolic tangent function is similar to cosine and sine function. It is represented by the equation

$$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

The function has a domain of real numbers and returns a value in [0, 1]

if $h \rightarrow \infty$ then $\tanh(h) = 1$

if $h \rightarrow -\infty$ then $\tanh(h) = 0$

The slope of the function is given by

if $\tanh(h) = g(h)$

$$g'(h) = 1 - \tanh(h)^2$$

ReLU and leaky ReLU activation function

The ReLU activation function is defined by the equation

$$f(z) = \max(0, z)$$

where z is the input to the neuron.

The leaky ReLU activation function is defined by the equation

$$f(z) = \max(0.01z, z)$$

Gradient descent

Gradient descent is a first-order iterative optimization algorithm. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient (or of the approximate gradient) of the function at the current point.

Loss (error) function

The loss or error function is used to calculate the accuracy of the model. It is given by equation

$$L(\hat{y}, y) = -y \log \log(\hat{y}) + (1-y) \log \log(1-\hat{y})$$

Here y is the true label and \hat{y} is the predicted label.

So,

$$\text{if } y=1 \quad L(\hat{y}, y) = -\log \log(\hat{y})$$

this means we want $\log \hat{y}$ to be as big as possible. (\hat{y} is close to 1).

$$\text{if } y=0 \quad L(\hat{y}, y) = -\log \log(1-\hat{y})$$

this means we want $\log(1-\hat{y})$ to be large and \hat{y} should be small. (\hat{y} is close to 0).

Cost function

Cost function is the average of all the values of loss function which means we need smallest possible value of cost function.

It is defined by

$$J(w, b) = \frac{1}{m} \sum L(\hat{y}^i, y^i)$$

$$J(w,b) = -\frac{1}{m} \sum (y^i \log(\hat{y}^i) + (1-y^i) \log(1-\hat{y}^i))$$

We need to define values of w , b such that it minimizes the cost function.

Convolution Neural Network

Convolution Neural Network is a class of Deep Neural Network. It is used on image data and used to analyze the different features of the images. The concept of CNN is derived from biological process. Like neurons are connected with each other, nodes of CNN are connected with each other and transfer values or signals. CNN is highly used in image classification. But the process may need some preprocessing depending on the nature of the dataset.

Tuning is the most important part in CNN, i.e accuracy on getting correct classification. Two very important terms regarding Neural Network are parameter and hyperparameter.

There are difference between them. Parameters are those which the model learns from training data like weights and biases. Prediction process needs the help of parameters. Parameters are the confidence of neural network. These are the keys of machine learning algorithms. On the other hyperparameters are provided to the model, for example number of hidden layers and nodes, activation functions, number of filters kernel size, stride, pooling, learning rate etc. hyperparameters often help to estimate parameters. It's on the hand of the developer to set the hyperparameters carefully. Mainly hyperparameters are tuned to increase the efficiency of the model.

Deep Learning Libraries (TensorFlow and Keras)

Tensorflow

It is an open source software library supporting high performance numerical operations on variety of platforms like CPU,GPU,TPU. AI organization of Google developed this. It provides various support in many scientific domain like machine learning, deep learning etc.

Keras

Keras is a high level API on Neural Network. Which is developed in python language and can run on Tensorflow, Theano and CNTK.

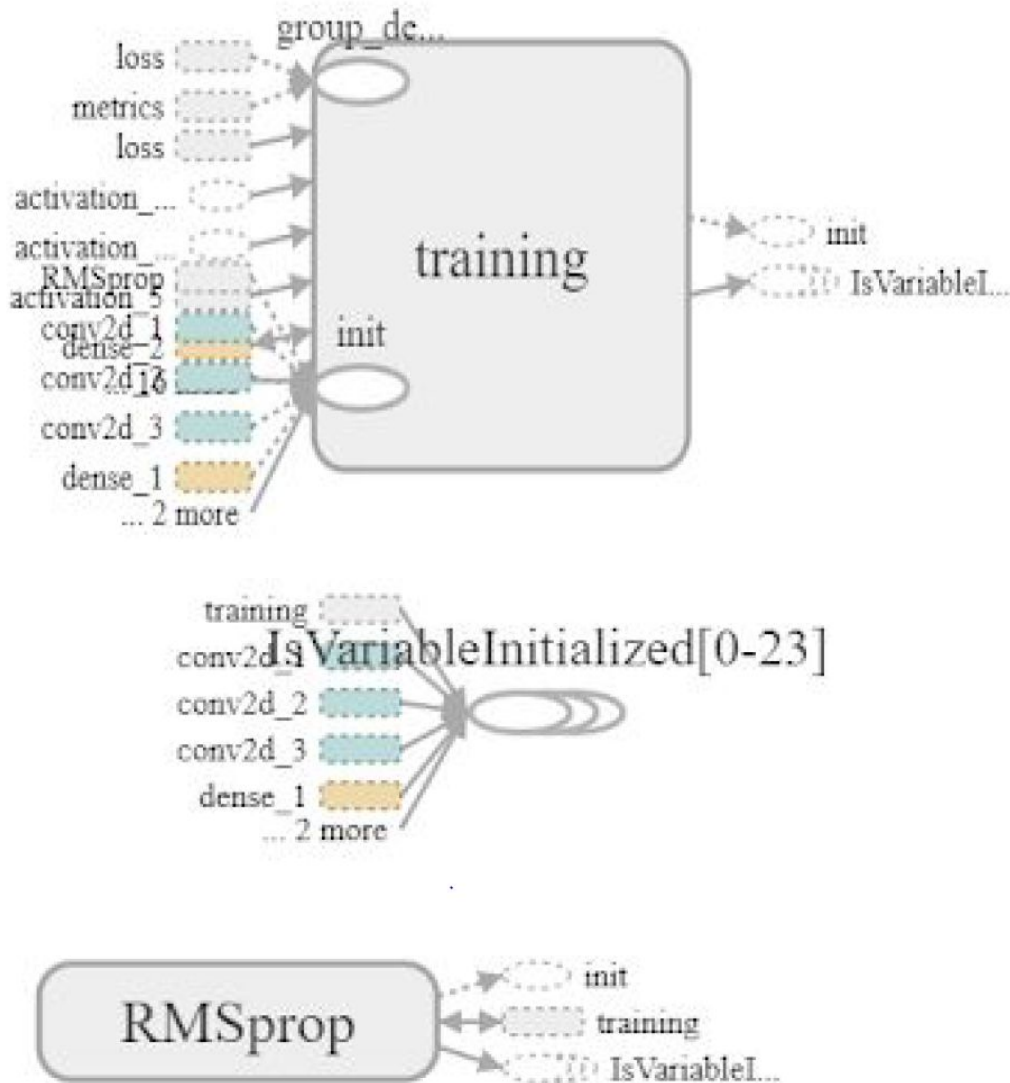
- It is very user friendly
- It is well documented.
- It is scalable.
- It runs on CPU and GPU.
- It allows very fast and easy prototyping.

TensorBoard

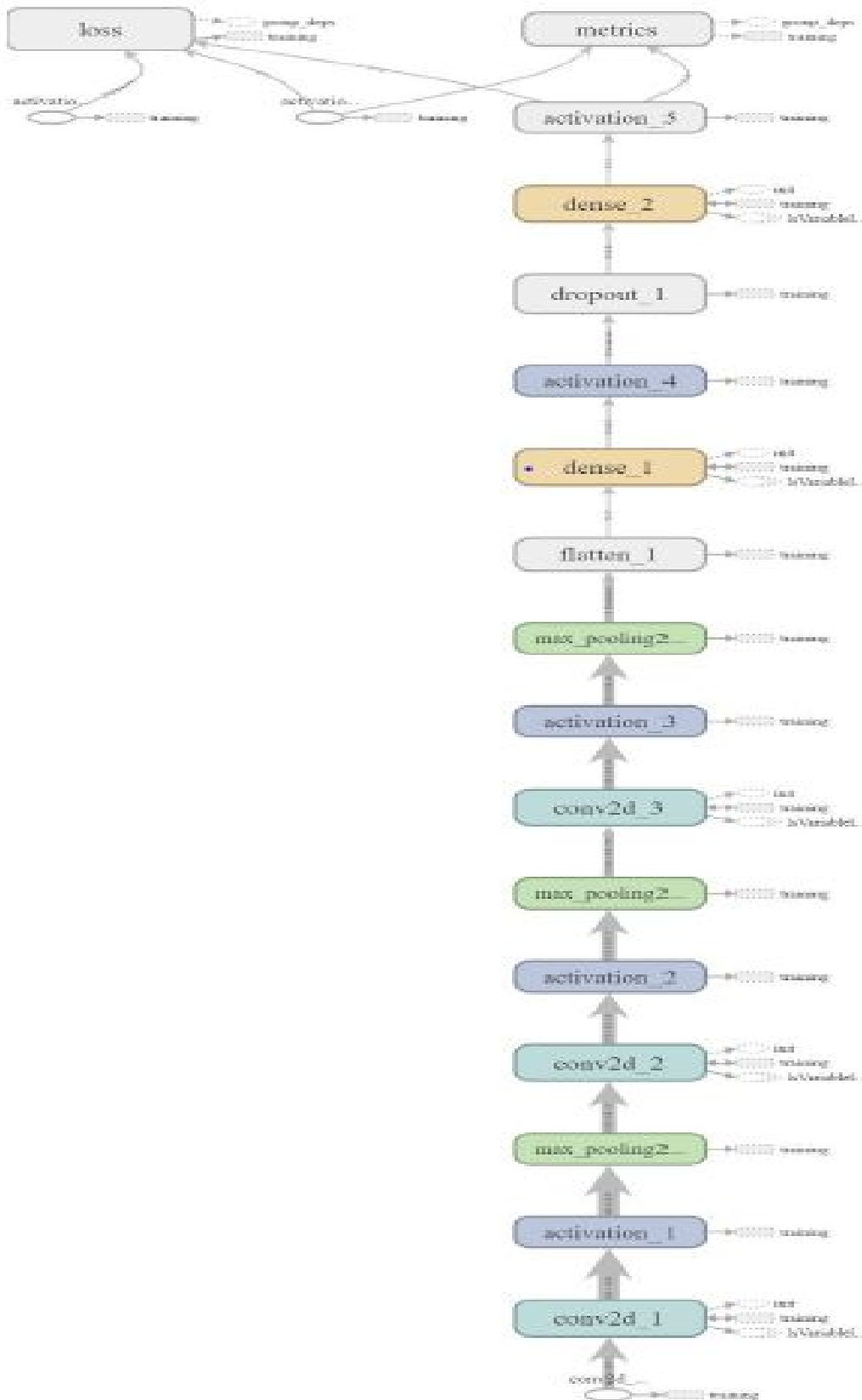
The computations you'll use TensorFlow for - like training a massive deep neural network - can be complex and confusing. To make it easier to understand, debug, and optimize TensorFlow programs, we've included a suite of visualization tools called TensorBoard. You can use TensorBoard to visualize your TensorFlow graph, plot quantitative metrics about the execution of your graph, and show

additional data like images that pass through it. When TensorBoard is fully configured, it looks like this:

The computations of Tensorflow can be sometimes very complex and very confusing. But when it comes to the form of graph i.e when a very complex thing is visualized in certain matter it becomes easy to interpret. Tensorboard offers nothing but visualizing the computations.



Figures above and below : Cat-Dog model visualization through TensorBoard graphs



Lung Cancer debrief

In simple words cancer is abnormal growth of cells, and ultimately to the stoppage of the essential cellular functions of the organism. These cells are generally called 'tumor cells' and they often clump together into lumps to form 'tumors'. They also carry the potential to invade other parts of the body of the organism and have detrimental effects there.

Lung cancer can be divided into four stages depending on severity as following

- Stage 1: Cancer is found in the lung only.
- Stage 2: Cancer is grown to nearby lymph nodes.
- Stage 3: Cancer is in the lung and lymph nodes are grown in the middle of the chest.
- Stage 3A: Cancer is found in lymph nodes only on the same side of the chest where it first started..
- Stage 3B: Cancer has spread to lymph nodes on the opposite side of the chest or to lymph nodes above the collarbone.
- Stage 4: Cancer has spread to both lungs, into the area around the lungs, or to distant organs.

Early symptoms may include

- Bad painful cough
- Cough with phlegm or blood
- Heavy chest pain while deep breathing, laughing, coughing or any other muscular activities near chest
- shortness of breath
- weakness and fatigue
- loss of appetite and weight loss

Causes

About ninety percent of lung cancer cases involve smoking. Smoking causes destruction of lung tissues, lung can repair them but heavy smoking makes lung stopping it's natural behaviour.

A radioactive gas Radon, is the second leading cause, according to the American Lung Association.

Breathing in other hazardous substances can also cause lung cancer if it happens over a long period of time. A type of lung cancer called mesothelioma is almost always caused by exposure to asbestos.

Other substances that can cause lung cancer are:

- nickel
- petroleum products
- uranium
- arsenic
- Cadmium
- Chromium

Inherited genetic mutations may be the reason to develop lung cancer, especially if you smoke or are exposed to other carcinogens.

But above all there may be no specific reason for lung cancer.

Risk factors

Above all the biggest risk factor is smoking. Inhaling toxic substances increases risk of getting lung cancer. Secondhand smoke is also a major risk factor. Other risk factors may include family health history, previous therapy or health history etc.

Diagnosis

- Imaging tests : An abnormal lump can be seen on X-ray. MRI, CT, and PET scans. These scans produce more detail and find smaller lesions.
- Sputum cytology : Microscopic examination of phlegm of cough can determine if cancer cells are present.

Treatment

- Self-care
- Quitting smoking
- Medications
- Chemotherapy and Targeted therapy
- Surgery
- Pulmonary lobectomy and Video-Assisted thoracoscopic surgery
- Medical procedure
- Thoracotomy and Radiation therapy
- Supportive care
- Palliative care
- Specialists

Methodology

Problem Statement

The Goal of the project is to train a Convolution Neural Network for binary image classification i.e detecting whether a person has lung cancer or not , which includes following tasks:

- Fetch and Preprocess the whole dataset.
- Train a Neural network on the training data.
- Optimize the network to improve accuracy.
- Compare the performances of Algorithms used.

Inputs: The inputs to the network are 40*40 pixel image snippets.

Outputs: The output of the network is the class label for the input image.

Data source

Publicly available LIDC/IDRI database is used. This data uses the Creative Commons Attribution 3.0 Unported License . Scans with a slice thickness greater than 2.5 mm were excluded. In total, 888 CT scans are included.Each radiologist marked lesions they identified as non-nodule, nodule < 3 mm, and nodules >= 3 mm.

Data Description :

Total 2948 images are used.

The data includes three datasets:

- Training Data: Contains total 2064 grayscale images
- Validation data: Contains 442 grayscale images
- Testing data: Contains 442 grayscale images

Image size 40*40 pixels

Model

The model is implemented using the Sequential API in Keras. It consists of three Convolution-2D layers with 16, 32 and 64 kernels of size (3,3), (5,5) and (7,7) respectively each followed by a Maxpooling-2D layer of size (2,2) pooling with a stride of 2. Maxpooling layers reduce the dimensionality of data thereby reducing the number of parameters, leading to a reduction in training time and tackling overfitting. Pooling layers downsample each feature map independently. They reduce the height and width, keeping the depth constant.

In the end we have fully connected layers that flatten the last convolution layer's output and connect every node of the present layer with the other node of the next layer. Neurons in a fully connected layer have full connections to all activations in the previous layer, as they have in regular Neural Networks. We need the Flattening layers since the output of both convolution and pooling layers are 3D volumes, but a fully connected layer expects a 1D vector of numbers. The Flattening operation simply involves arranging the 3D volume of numbers into a 1D vector. The flatten output becomes an input to the fully connected layers.

We then implement grid search to find the best values for hyperparameters optimizer, epochs and batch size.

Model description

Convolution Layer 1:

Kernel size (3,3)

Number of filters 16

Strides (1,1)

Padding Same

Activation Relu

Convolution Layer 2:

Kernel size (5,5)

Number of filters 32

Strides (1,1)

Padding Same

Activation Relu

Convolution Layer 3:

Kernel size (7,7)

Number of filters 64

Strides (1,1)

Padding Same

Activation Relu

Model Summary

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 40, 40, 16)	160
max_pooling2d_1 (MaxPooling2D)	(None, 20, 20, 16)	0
conv2d_2 (Conv2D)	(None, 20, 20, 32)	12832
max_pooling2d_2 (MaxPooling2D)	(None, 10, 10, 32)	0
conv2d_3 (Conv2D)	(None, 10, 10, 64)	100416
max_pooling2d_3 (MaxPooling2D)	(None, 5, 5, 64)	0
flatten_1 (Flatten)	(None, 1600)	0
dense_1 (Dense)	(None, 1024)	1639424
dropout_1 (Dropout)	(None, 1024)	0
dense_2 (Dense)	(None, 1)	1025
activation_1 (Activation)	(None, 1)	0
=====		
Total params: 1,753,857		
Trainable params: 1,753,857		
Non-trainable params: 0		
=====		
None		

Model Configuration

```
{'layers': [{'class_name': 'Conv2D',  
'config': {'activation': 'relu',  
'activity_regularizer': None,  
'batch_input_shape': (None, 40, 40, 1),  
'bias_constraint': None,  
'bias_initializer': {'class_name': 'Zeros', 'config': {}},  
'bias_regularizer': None,  
'data_format': 'channels_last',  
'dilation_rate': (1, 1),  
'dtype': 'float32',  
'filters': 16,  
'kernel_constraint': None,  
'kernel_initializer': {'class_name': 'VarianceScaling',  
'config': {'distribution': 'uniform',  
'mode': 'fan_avg',  
'scale': 1.0,  
'seed': None}},  
'kernel_regularizer': None,  
'kernel_size': (3, 3),  
'name': 'conv2d_11',  
'padding': 'same',  
'strides': (1, 1),  
'trainable': True,
```

```
'use_bias': True}},  
{'class_name': 'MaxPooling2D',  
'config': {'data_format': 'channels_last',  
'name': 'max_pooling2d_10',  
'padding': 'same',  
'pool_size': (2, 2),  
'strides': (2, 2),  
'trainable': True}},  
{'class_name': 'Conv2D',  
'config': {'activation': 'relu',  
'activity_regularizer': None,  
'bias_constraint': None,  
'bias_initializer': {'class_name': 'Zeros', 'config': {}},  
'bias_regularizer': None,  
'data_format': 'channels_last',  
'dilation_rate': (1, 1),  
'filters': 32,  
'kernel_constraint': None,  
'kernel_initializer': {'class_name': 'VarianceScaling',  
'config': {'distribution': 'uniform',  
'mode': 'fan_avg',  
'scale': 1.0,  
'seed': None}},  
'kernel_regularizer': None,  
'kernel_size': (5, 5),
```

```
'name': 'conv2d_12',
'padding': 'same',
'strides': (1, 1),
'trainable': True,
'use_bias': True}},
{'class_name': 'MaxPooling2D',
'config': {'data_format': 'channels_last',
'name': 'max_pooling2d_11',
'padding': 'same',
'pool_size': (2, 2),
'strides': (2, 2),
'trainable': True}},
{'class_name': 'Conv2D',
'config': {'activation': 'relu',
'activity_regularizer': None,
'bias_constraint': None,
'bias_initializer': {'class_name': 'Zeros', 'config': {}},
'bias_regularizer': None,
'data_format': 'channels_last',
'dilation_rate': (1, 1),
'filters': 64,
'kernel_constraint': None,
'kernel_initializer': {'class_name': 'VarianceScaling',
'config': {'distribution': 'uniform',
'mode': 'fan_avg',
```

```
'scale': 1.0,  
'seed': None}},  
'kernel_regularizer': None,  
'kernel_size': (7, 7),  
'name': 'conv2d_13',  
'padding': 'same',  
'strides': (1, 1),  
'trainable': True,  
'use_bias': True}},  
{'class_name': 'MaxPooling2D',  
'config': {'data_format': 'channels_last',  
'name': 'max_pooling2d_12',  
'padding': 'same',  
'pool_size': (2, 2),  
'strides': (2, 2),  
'trainable': True}},  
{'class_name': 'Flatten',  
'config': {'data_format': 'channels_last',  
'name': 'flatten_4',  
'trainable': True}},  
{'class_name': 'Dense',  
'config': {'activation': 'relu',  
'activity_regularizer': None,  
'bias_constraint': None,  
'bias_initializer': {'class_name': 'Zeros', 'config': {}}},
```

```
'bias_regularizer': None,  
'kernel_constraint': None,  
'kernel_initializer': {'class_name': 'VarianceScaling',  
'config': {'distribution': 'uniform',  
'mode': 'fan_avg',  
'scale': 1.0,  
'seed': None}},  
'kernel_regularizer': None,  
'name': 'dense_7',  
'trainable': True,  
'units': 1024,  
'use_bias': True}},  
{'class_name': 'Dropout',  
'config': {'name': 'dropout_4',  
'noise_shape': None,  
'rate': 0.2,  
'seed': None,  
'trainable': True}},  
{'class_name': 'Dense',  
'config': {'activation': 'linear',  
'activity_regularizer': None,  
'bias_constraint': None,  
'bias_initializer': {'class_name': 'Zeros', 'config': {}},  
'bias_regularizer': None,  
'kernel_constraint': None,
```

```
'kernel_initializer': {'class_name': 'VarianceScaling',
'config': {'distribution': 'uniform',
'mode': 'fan_avg',
'scale': 1.0,
'seed': None}},
'kernel_regularizer': None,
'name': 'dense_8',
'trainable': True,
'units': 1,
'use_bias': True}},
{'class_name': 'Activation',
'config': {'activation': 'sigmoid',
'name': 'activation_4',
'trainable': True}}],
'name': 'sequential_5'}
```

Visualization of the model

Input → [Conv Layer 1 → ReLU] → Max Pool Layer 1 → [Conv Layer 2 → ReLU] → Max Pool Layer 2 → [Conv Layer 3 → ReLU] → Max Pool Layer 3 → [Fully-Connected Layer 1 → Dropout] → Fully-Connected Layer 2 → Output Classes [0 or 1]

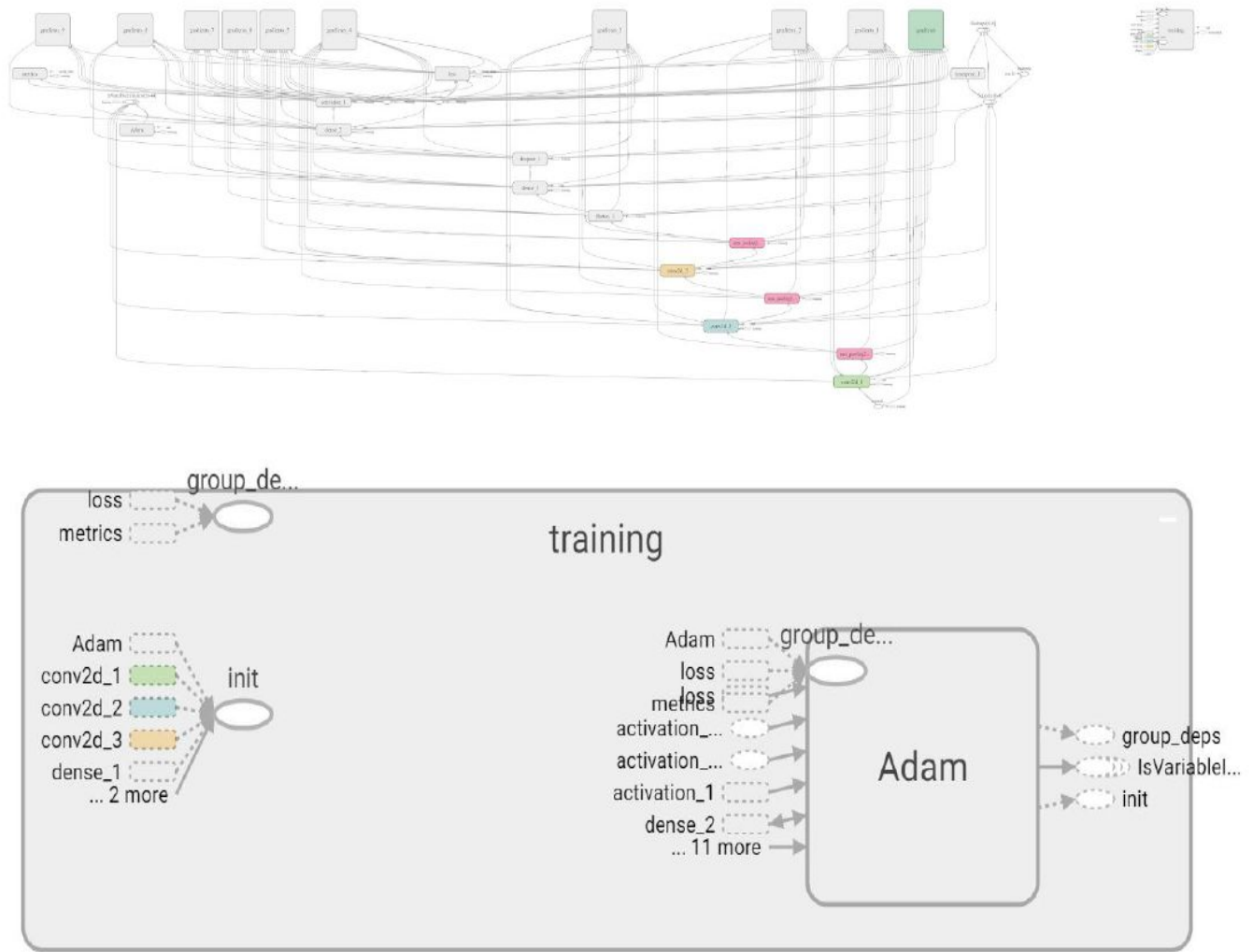


Figure : tensorboard graph representation of model

Hyperparameter Tuning

The model was trained on different combinations of hyperparameters and the performance in each case was recorded as below:

1.

epochs = [10,25]

batches = [50,100,104]

optimizers = ['SGD', 'adam']

Cross validation = 10

Best parameter combination

{'batch_size': 104, 'epochs': 25, 'optimizer': 'adam'}

Accuracy on test data: 97.05%

2.

epochs = [25,35]

batches = [50,104,150]

optimizers = ['adam']

Cross validation = 10

Best parameter combination

{'batch_size': 150, 'epochs': 25, 'optimizer': 'adam'}

Accuracy on test data: 97.29%

3.

epochs = [20,25,40]

batches = [200,150]

optimizers = ['adam']

Cross validation = 10

Best parameter combination

```
{'batch_size': 200, 'epochs': 40, 'optimizer': 'adam'}
```

Accuracy on test data: 96.38%

4.

```
epochs = [20,25,40]
```

```
batches = [200,150]
```

```
optimizers = ['adam']
```

Cross validation = 3

Best parameter combination

```
{'batch_size': 200, 'epochs': 40, 'optimizer': 'adam'}
```

Accuracy on test data: 97.51%

5.

```
epochs = [40,45]
```

```
batches = [200,250]
```

```
optimizers = ['adam']
```

Cross validation = 3

Best parameter combination

```
{'batch_size': 250, 'epochs': 40, 'optimizer': 'adam'}
```

Accuracy on test data: 96.83%

6.

```
epochs = [40,45]
```

```
batches = [150,250,300]
```

```
optimizers = ['adam']
```

Cross validation = 3

Best parameter combination

{'batch_size': 150, 'epochs': 45, 'optimizer': 'adam'}

Accuracy on test data: 97.29%

7.

epochs = [40,45]

batches = [150,250,300]

optimizers = ['adam']

Cross validation = 3

Best parameter combination

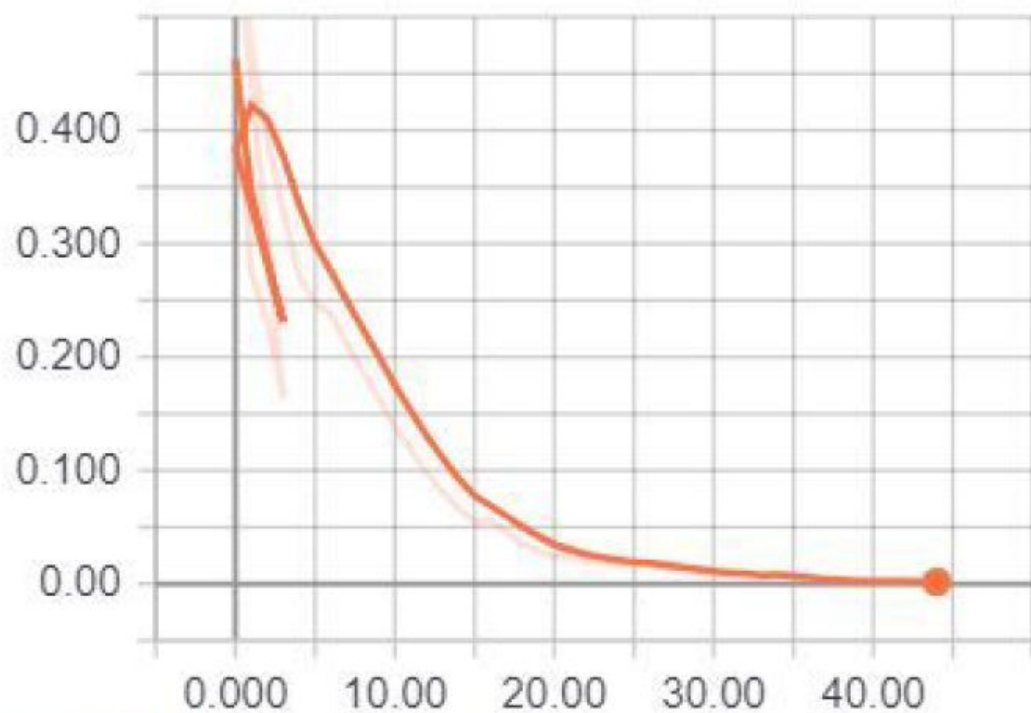
{'batch_size': 300, 'epochs': 45, 'optimizer': 'adam'}

Accuracy on test data: 97.51%

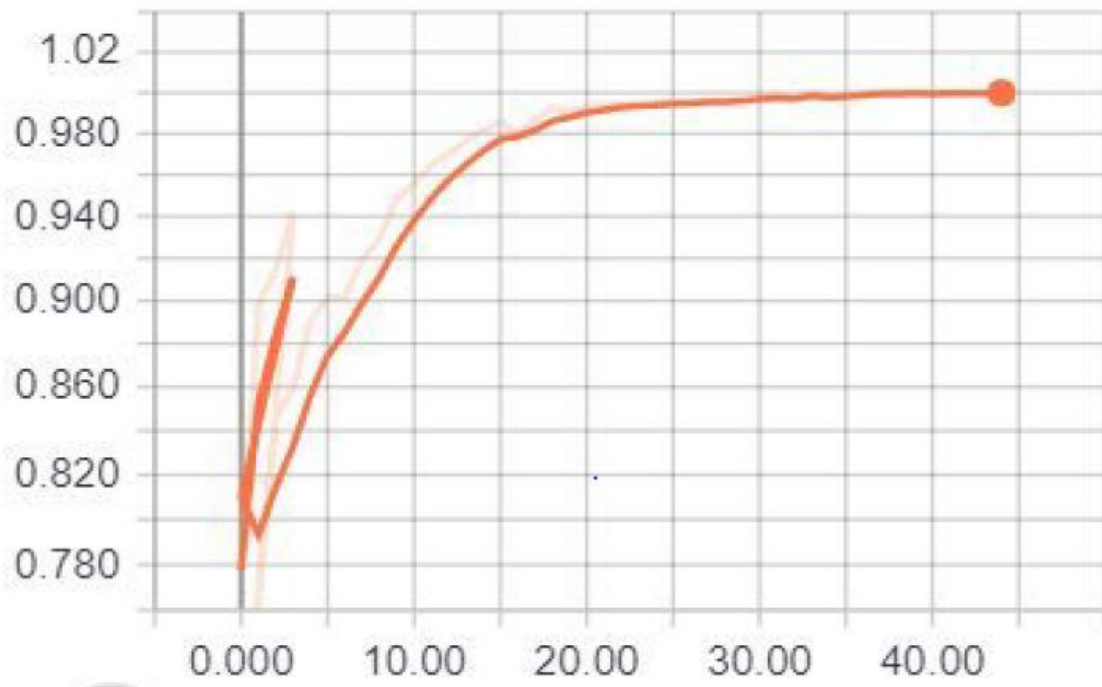
Results and Discussion

The best combination of parameters calculated from grid search i.e parameters of 7th experiment was used to train the model. The model achieved an accuracy of 97.51%. The final loss and accuracy graphs for training and validation have been plotted using tensorboard as below:

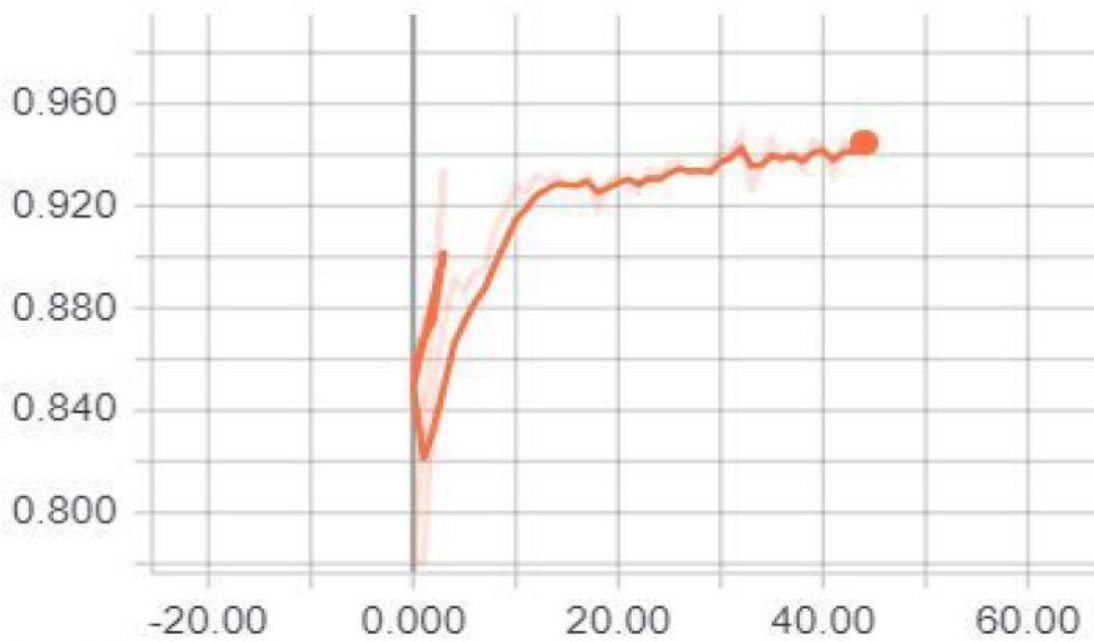
loss



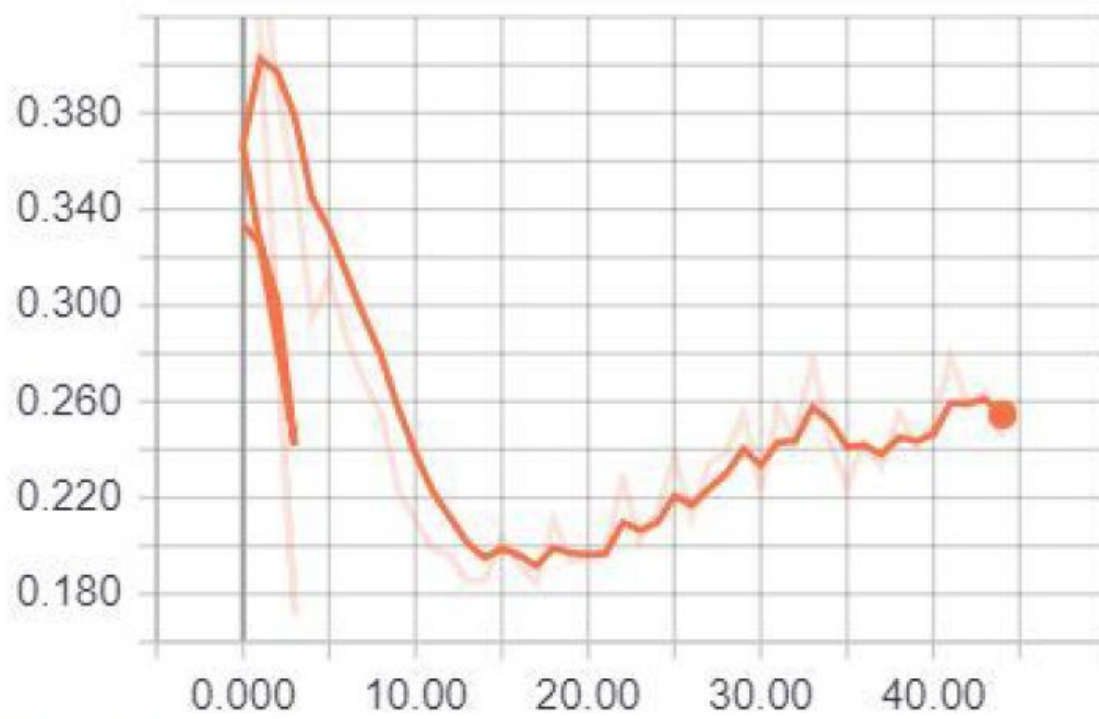
acc



val_acc



val_loss



Confusion Matrix

After finding the best combination of hyperparameters for our model, we ran a further analysis to find misclassified images of the final test results and calculated confusion matrix to determine why it was not close to 100%. Our model has classified more examples as negative when they should be positive than vice versa which might be because of the nature of some of the positive examples. It is likely that it would be just as difficult for a human to classify those images as a doctor. Also, there is always a possibility of the data being wrongly classified.

True Positive: 208

False Positive: 4

True Negative: 222

False Negative: 8

Confusion matrix

	Predicted Positive Class	Predicted Negative Class
Actual Positive Class	208	8
Actual Negative Class	4	222

Conclusion and Future Scope

In this project, we explored the very powerful Convolution Networks and applied it to predict the presence of lung cancer in a patient using an image snippet of a scan. Experimentation showed that CNN was highly efficient in performing the task and the accuracy was further improved by optimization of hyperparameters. The images used specify a predetermined section of the lung and the model is also rather simple but still we were able to get ground-breaking results.

We plan to explore Deconvolution, Transpose-Convolution for Upscaling and other possible extensions of CNN. For the Cancer model, we would like to implement one or more of the named CNN networks like Inception or Alexnet using transfer learning and compare the workings and result with the previous models. The current model works on images of 40 x 40 dimensions, increasing which might be plausible to perform testing on larger scans, possibly of the entire lung instead of a predetermined section.

References

1. <https://apollack11.github.io/machine-learning.html>
2. <https://keras.io/>
3. <https://www.tensorflow.org/>
4. <https://www.healthline.com/health/lung-cancer#stages>
5. <https://towardsdatascience.com/building-a-convolutional-neural-network-cnn-in-keras-329fbbadc5f5>
6. <https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/>