A

DISSERTATION REPORT

On

Transfer Learning Based Classification of Medical Images

By

Chandra Prakash Soni

2016PEB5220

Under the supervision of

Dr. Lava Bhargava

Professor, ECE Department

MNIT, Jaipur

Submitted in partial fulfillment of requirement of degree of

MASTER OF TECHNOLOGY

To the

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

JUNE – 2018

Department of Electronics and Communication Engineering
Malaviya National Institute of Technology, Jaipur

Certificate

This is to certify that this Dissertation report entitled ***"Transfer Learning Based Classification of Medical Images"*** by **Chandra Prakash Soni** (2016PEB5220), is the work completed under my supervision and guidance, hence approved for submission in partial fulfillment for the award of degree of ***Master Of Technology*** in ***Embedded Systems*** to the Department of Electronics and Communication Engineering, Malaviya National Institute of Technology, Jaipur in the academic session 2017-2018 for full time post-graduation program of 2017-2018. The contents of this dissertation work, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Lava Bhargava
Professor, ECE Department
MNIT, Jaipur

Declaration

I, hereby declare that the work which is being presented in this project entitled " Transfer Learning Based Classification of Medical Images " in partial fulfillment of degree of Master of Technology in Embedded Systems is an authentic record of my own work carried out under the supervision and guidance of Prof. Dr. Lava Bhargava in Department of Electronics and Communication, Malaviya National Institute of Technology, Jaipur.

I am fully responsible for the matter embodied in this project in case of any discrepancy found in the project and the project has not been submitted for the award of any other degree. I also confirm that I have consulted the published work of others, the source is clearly attributed and I have acknowledged all main sources of help.

(Chandra Prakash Soni)

Acknowledgement

I am grateful to my supervisor Professor Dr. Lava Bhargava for his constant guidance and encouragement and support to carry out this work. His excellent cooperation and suggestion provided me with an impetus to work and made the completion of work possible. He has been great source of inspiration to me, all through. I am very grateful to him for guiding me how to conduct research and how to clearly & effectively present the work done.

I would like to express my deepest sense of gratitude and humble regards to our Head of Department Prof. Dr. D. Boolchandani for providing necessary facility in the Department. I am very thankful to all other faculty members of ECE, MNIT for their valuable assistance and advice.

I would also like to thank my friends for their support in discussions which proved valuable for me. I am indebted to my parents and family for their constant support, love, encouragement, and sacrifices made by them so that I could grow up in a learning environment.

Finally, I express my sincere thanks to all those who helped me directly or indirectly to complete this work successfully.

(Chandra Prakash Soni)

Abstract

With the advent of technology every day and night people are working to make human life easier so that the lesser amount of time is spent in doing trivial things and people can focus more on important things. Deep learning is being used trivially in this data driven ecosystem. It has made remarkable improvements in the field of computer vision. This advancement has led to new research areas like driverless cars, surveillance robotics and many others. However there is a need in the improvement in medical imaging as the work is carried out manually for diagnosis of diseases using X-Ray, CT Scans & MRI imaging techniques. The idea is to make use of image classification techniques using Deep learning and transfer learning to classify different lung diseases thereby reducing the need for a radiologists and clinicians to do manual diagnosis.

The second part of the thesis is dedicated to the work carried out at Intel Corporation India Pvt. Ltd. Bangalore. Being a part of Bluetooth communication group, the work is focused on validating different operations related to Bluetooth across different platforms and debugging the issues.

Table of Contents

All rights reserved i

Declaration iii

Acknowledgement iv

Abstract v

Table of Contents vi

Table of figures ix

List of Tables xi

Chapter 1 1

Introduction 1

1.1 Motivation 1

1.2 Problem statement 1

1.3 Objective 2

1.4 Thesis Outline 2

Chapter 2 3

2.1 Literature Survey 3

2.2 Theory 4

2.2.1 Artificial Neurons vs Biological Neurons 4

2.2.2 Basic Structure of a Convolution Neural Network 5

2.2.3 Activation Functions 8

2.2.4 Learning Rule 11

2.2.5 Backpropagation 11

Chapter 3 13

Proposed Work 13

3.1 Dataset 13

3.2 Pre-trained model-Inception V3 14
3.3 Dimensionality reduction 16
3.4 Transfer learning 16
3.4.1 Develop Model Approach 17
3.4.2 Pre-trained Model Approach 17
Chapter 4 19
Implementation 19
4.1 Preprocessing the images from data set 19
4.2 Calculating Bottleneck values 19
4.3 Hyperparameters Setting 20
4.3 Training 20
4.4 Visualizing the Retraining with Tensorboard 21
4.5 Improving the Retrained Model 21
Chapter 5 23
Results 23
1. Inception V3 23
2. MobileNet 26
3. Results Summary 30
Chapter 6 31
Conclusion 31
Future Scope 31
Bibliography 32
Part-B 34
Chapter 1 35
Introduction 35
1.1 Bluetooth Version History 36

1.2 Classic Bluetooth vs Low Energy 37
1.3 BR/EDR OPERATION 37
Chapter 2 41
Project profile at Intel Corporation India Pvt. Ltd. 41
1. Stability Regression for BR/EDR Controller 41
1.1 Inquiry / Inquiry Scan 41
1.2 Page / Page scan 41
1.3 ACL Data P2P 42
1.4 ACL Data Piconet/Scatternet + SCO 42
2. Manual Testing of Privacy Feature of Bluetooth LE controller 43
2.1 Setup 44
2.2 Commands used 45

Table of figures

Figure 1 (a) Artificial Neuron (b) Biological Neuron *4*

Figure 2 Basic CNN Structure *5*

Figure 3 Basic Convolution operation *6*

Figure 4 Pooling operation *7*

Figure 5 Fully connected layer *7*

Figure 6 DropOut and DropConnect *8*

Figure 7 Sigmoid Function *9*

Figure 8 Hyperbolic tangent function *9*

Figure 9 Rectified Linear Unit function *10*

Figure 10 Leaky ReLu function *10*

Figure 11 Error Minimization *12*

Figure 12 Chest X-Rays Pneumonia *14*

Figure 13 Inception Model Architecture (Google) *15*

Figure 14 Inception Module *16*

Figure 15 (a) Machine learning (b) Transfer learning *17*

Figure 16 Advantages of transfer learning *18*

Figure 17 workflow to be implemented using transfer learning *19*

Figure 18 Retrained model *21*

Figure 19 Execution result of Inception V3 model with steps=4000 ..... *24*

Figure 20 Execution result of Inception V3 model with steps=6000 ..... *25*

Figure 21 Execution result of Inception V3 model with steps=10000 ..... *26*

Figure 22 Execution result of MobileNet architecture with steps=4000 ..... *27*

Figure 23 Execution result of MobileNet architecture with steps=6000 ..... *28*

Figure 24 Execution result of MobileNet architecture with steps=10000 ..... *30*

Figure 25 (a) Piconets with a single slave, (b) a multi-slave and (c) a Scatternet 38

Figure 26 Transition states of a Bluetooth 39

Figure 27 setting random device address 44

Figure 28 Adding device to white list 45

Figure 29 Adding IRK to resolving list 45

Figure 30 Ellisys Bluetooth Explorer 47

List of Tables

Table 1 Distribution of Dataset *14*

Table 2 No. of parameters used in different model architectures..... *15*

Table 3 Results Summary..... *30*

Table 4 Single-Mode, Dual-Mode, and Classic Compatibility..... *35*

Table 5 Bluetooth version history..... *36*

Table 6 Classic Bluetooth vs Low Energy *37*

Chapter 1

Introduction

With the advent of machine learning and data science Technology has evolved exponentially in the current decade. The challenging problems in computer vision are being sold without the need of coding explicitly. Starting from object recognition to natural language processing all are becoming possible with the advent of AI and neural nets. However there is a need to improve in the field of Medical Sciences. This field has not been evolved with such a pace medical imaging is such an example which requires trained professionals or radiologist to diagnose the diseases based on the images. This area can be improved if a good model is developed to classify those is Cannes thereby reducing the effort of humans requiring experience person to detect the disease.

1.1 Motivation

The progress of deep learning has rolled out gigantic improvements in the field of computer vision and has created off the shelf prepared models especially convolution neural networks have been broadly used to construct image classification model which allows researchers to transfer the prepared learning model for a different arrangements.

Google's Inception V3 is such a model which has been trained to classify thousand different categories and also can be trained on a custom image data set for different classes. The idea is to use transfer learning to develop a modified model to classify new set of objects.

1.2 Problem statement

There are various challenges to build an accurate neural net model. In general to train a model there must be a good data set having versatility such that each and every scenario is covered. The second thing is resources for developing a good model need training the model for use data set. It requires tons of processing arithmetic thereby requiring good GPU support which are costly. For the above mentioned reasons the idea to classify new images having small data set is of very less use. Instead we can employ transfer learning to classify different sets of objects in minimal time and resources with good accuracy.

1.3 Objective

The objective of this thesis is to implement an image classifier which can classify different types of pneumonia like normal Viral and bacterial so that an automated test based on X-Ray images can be performed to classify different kinds of pneumonia.

We will use different models like Inception V3 from Google and mobile net to compare the overall results based on their classification abilities.

1.4 Thesis Outline

In Chapter 2 literature of the related work that has been done and background of CNN is given. In chapter 3, Proposed work is described in details and various aspect addressed in details as well. Chapter 4 describes the implementation of the proposed transfer learning based approach for medical images classification using Tensorflow is given. In chapter 5, Training & testing results are shown along with their discussion. Chapter 6 concludes the thesis parts 1 with scope of the future work.

Part B contains the internship work done at Intel Corporation India Pvt. Ltd. Bangalore. Chapter 1 provides the introduction to the Bluetooth Communication, controller and various aspects of it. Chapter 2 provides brief details about the tasks that were done during the course of 3 months of internship at Intel Corporation India Pvt. Ltd. Bangalore.

Chapter 2

2.1 Literature Survey

LeNet was the only pioneering work in CNN by LeCun *et al.* in 1990 and later improved on it. But due to lack of resources and training data it failed in complex situations. Later in the year 2012 Krizhevsky *et al.* had come up with a CNN model which was successful in 2012 on ILSVRC challenge. There was a significant reduction in error rate [1].
Over the years their work has been influential in the field of computer vision.
Alexnet was significantly successful to achieve remarkable results compared to its predecessors.

In [1] by Neena *et al.* has describe various architectures been proposed till date starting from the LeNet, based on their their variations from time to time. The paper serves as a beginners guide into this area. It clearly defines in brief the convolution layers, activation functions, CNN training and testing.
A brief summary of different architectures of CNN commonly used along with their implementation and the open issues associated with them is given in detail. The author discusses in brief about the frameworks available for deep learning like Tensorflow, Keras, Theano, Caffe and Torch. Open issues in the area of CNN are described.

In [2] by Jongwon Chang *et al* an implementation of transfer learning based on pre-trained model from Google is used to classify different breast cancers from their mammography reports. A technique for data augmentation is employed to compliment insufficient data for training. By rotating, mirroring the images the training data can be Virtually increased for better training.

In [3] Kermany *et al.* describe biomedical image interpretation and medical decision making. Diagnosis of paediatric pneumonia is challenging as it is single leading cause of child mortality. In developing countries fast radiologic interpretation and diagnosis of diseases is not always available. The different classes of pneumonia and the types of treatment necessary based on its class are suggested.

In [4] SJ Pan *et al.* Have reviewed several trends currently employed in the area of deep learning and machine learning. Transfer learning has been classified to three different settings as

: unsupervised, inductive and transductive transfer learning. Author generalize this transfer learning approach for classification, regression and clustering problems which are more closely attached to that data mining tasks. This study is done by the fact that people could apply their knowledge learned previously to the problem sets in new domain for faster growth and with better solutions. The author cited the main issues in transfer learning as what ? How ? and When to transfer? , So that it must be in an aid to reduce data mining and data relabeling effort.

In [5] Jason Yosinki *et al.* have described various experiments to show the efficacy of transferring features from one network to another. The experiments have demonstrated the features which transfer from one layer to another. Imagenet dataset comprising of thousand different classes is used and a network with variations are used to quantify the transferring ability of features from each layer of a convolutional neural network which specifies their generality or specificity. The author is cited the distinct issues like Optimisation difficulties and specialisation of high level features. Finally has found that initialising with transferred features can aid to improve generalisation which could be useful technique for improving deep neural networks performance.

2.2 Theory

To understand deep neural networks it's not necessary for introducing some key concepts.

2.2.1 Artificial Neurons vs Biological Neurons

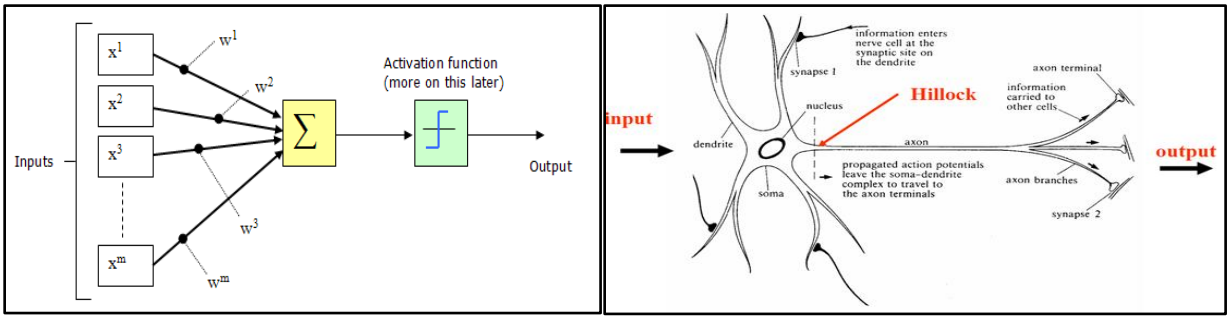


Figure 1 (a) Artificial Neuron (b) Biological Neuron

Nerve cells in the brain are called neurons. Neurons are the basic units inside brain which process information. A neuron contains a cell body having various extension from it *fig 1(a)*. Most of these are the branches known as dendrites. There is one extended process called Axon. The dashed line shows the axon Hill rock where transmission starts. The connection *between* a neuron and another are called synapses. Information only leaves a neuron through its axon (i.e. neuron fire's when input is greater than some threshold).

On the other hand the artificial neuron resembles a modeled neuron *fig1(b)*. The configuration of artificial neuron is actually called a perceptron. It takes the weighted sum of inputs and based on the activation functions which outputs 1 or 0 based on the adjustable threshold. The perceptron itself carries a weight, the summation processor, an activation function and all adjustable threshold processes called bias. A bias can be thought of as a tendency towards a particular way of behaving of a perceptron to fire independent of its input.

$$f = \sum_{i=1}^{m} bias + (w^i x^i)$$

The perceptron fires if the weighted sum is greater than zero.

2.2.2 Basic Structure of a Convolution Neural Network

Convolution nets can be very much inferred to as a bunch of neurons arranged in an acyclic manner. The deep architecture consists of input layer followed by hidden layers and then output layer as shown in figure below.

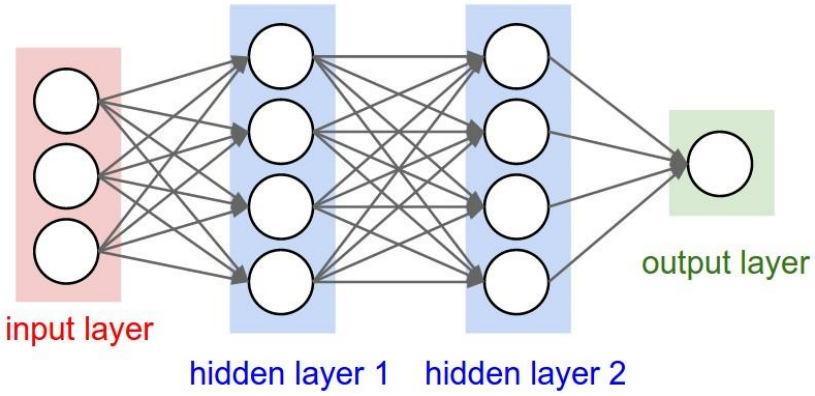


Figure 2 Basic CNN Structure

A deep architecture of convnet results in larger feature extraction like the first layer corresponds to extraction of features like edges or colour blobs, the second layer as shapes and the subsequent layers as some parts and the final layers can then estimate the object based on these extractions [6]. Following is a brief summary of different layers in a CNN.

2.2.2.1 Convolution Layer

The convolution layer is the basic building block of any convolution networks that does the heavy computational task. These layer parameter include learnable filters. Filters are spatially small but they extend to full depth of the input volume. It coontains a set of feature map with neurons arranged on it. The figure below shows the convolution operation in a neural network [7].

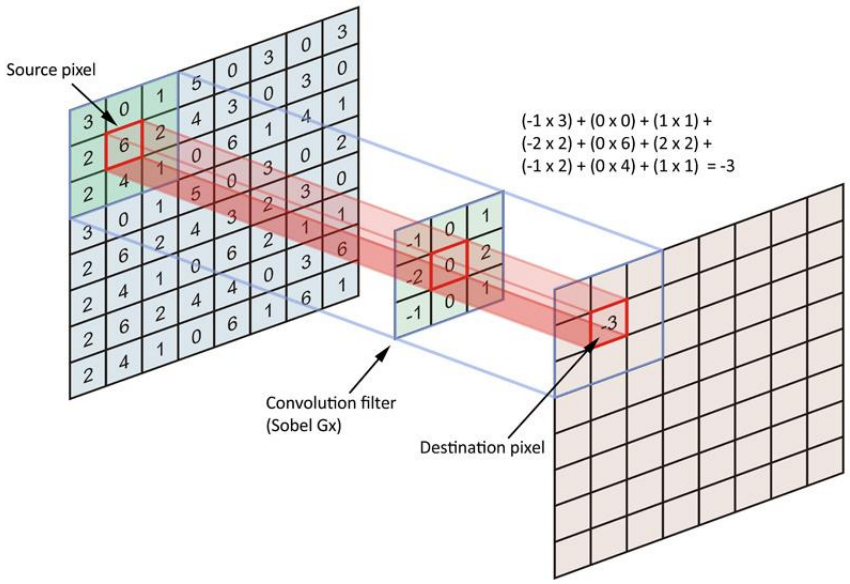


Figure 3 Basic Convolution operation

The parameters of the filter is a set of kernels or learnable filters which are convolved with feature maps to make a two dimesional activation map by combining the depth making an output volume. The hyperparameters that control the size of the produced volume are zero padding, depth, and stride. The CNN are trained with forward pass and the backward pass using backpropagation and involve convolution operation with geometrically flipped filters. Convolutional net which are made for image classification are being adopted for predicting problems like semantic segmentation which differ significantly from image classification.

2.2.2.2 Pooling Layer

A usual CNN architecture is alternating convolution layers and pooling layers. Pooling layers are used for reducing the dimension of the activation maps without being worrying about the loss of any information & the used number of parameters [8]. Thereby reducing the computational complexities. The figure below shows the pooling operation.

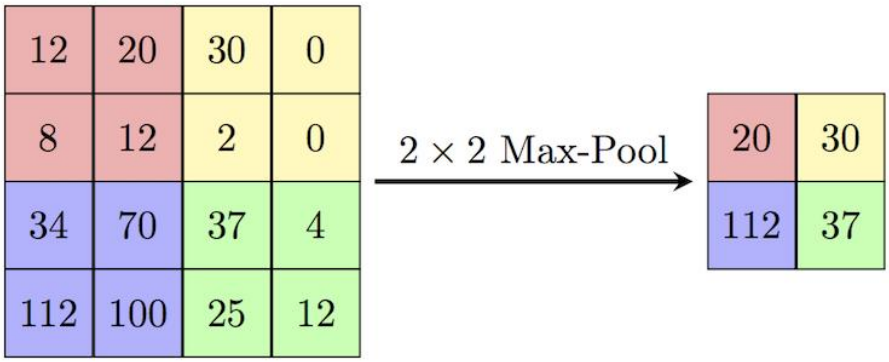


Figure 4 Pooling operation

This solves the problem of over fitting. Some pooling operations are average pooling, max pooling, stochstic pooling, special pyramidal pooling and multiscale order less pooling. Some new researchers have founded that the pooling layers can be exchanged with convolution layers having a stride size of two thereby simplifying the complex architecture.

2.2.2.3 Fully Connected Layer

Neurons present in this layer are fully connected to all the neurons in the previous layer. Fig below depicts the functionality of a fully connected layer.

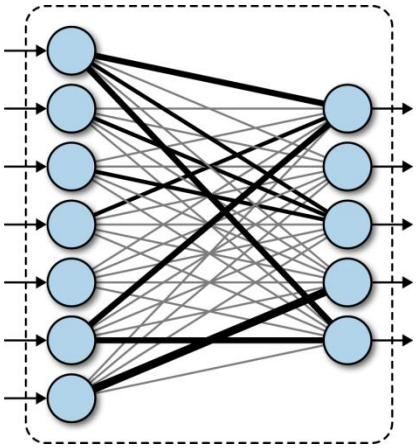


Figure 5 Fully connected layer

A high level reasoning is performed in this layer. There can't be any subsequent layer following a fully connected layer as neurons are not spatially arranged.

2.2.2.4 ReLu Layer

This layer applies a non saturating activation function. It increases the non-linear properties of the decision function & the overall network have an effect on the convolution layers.

2.2.2.5 Loss Layer

The final fully connected layer works as a loss layer which eastimates the loss or error between the actual and predicted values of outputs. There are various kinds of loss functions like softmax, large margin classifier, SVM, Euclidean loss out of which Softmax loss function is commonly used. It gives the probability distribution over the classes.

2.2.2.6 Regularization

One of the main problems associated with a CNN is overfitting. Regularization is a method used to overcome the results arising due to overheating. There are two techniques which are employed commonly. Dropout and Drop connect. The figure depicts a common neural network (a). Applying dropout to this network is eaual to sampling a neural net (b). The activations for some of the neuron are set to zero during backpropaagation passes in training phase. While the Drop connect that's the link weights to 0 (c).

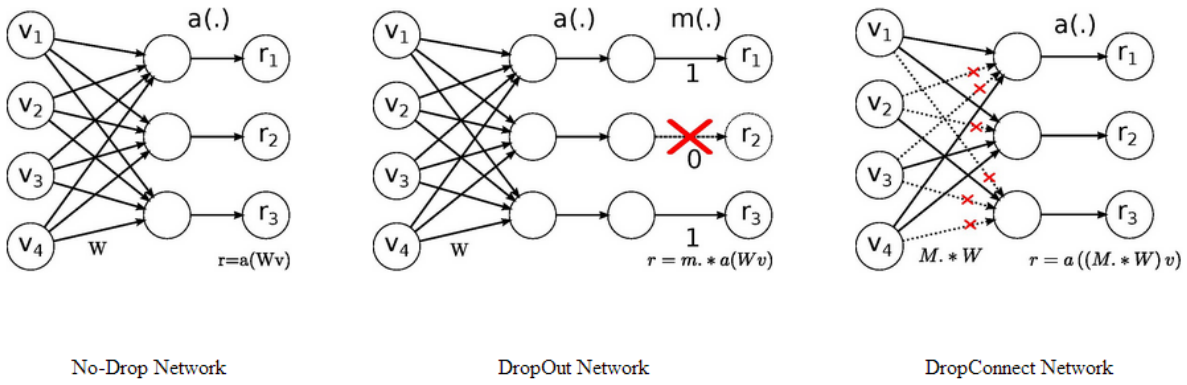


Figure 6 DropOut and DropConnect

This technique ensures extraction of only the general features without over fitting the training data.

2.2.3 Activation Functions

They are mainly used to introduce non linearity in any neural network. The main purpose of an activation function is to transfer an input signal of a node in a neural network to an output signal which can be used as an input to the next layer in the domain. Most popular activation functions are Sigmoid funcrion, Tan*h* or hyperbolic tangent function, and ReLu- Rectified linear unit function.

2.2.3.1 Sigmoid function

Range of f of X is between 0 and 1.Major problems with sigmoid function are:

- Vanishing gradient problem
- Its output is not zero centered *i.e.* it makes the Optimization harder.
- This saturate and kill gradients
- It is slow to converge
- $f(x) = \frac{1}{1+e^{-x}}$

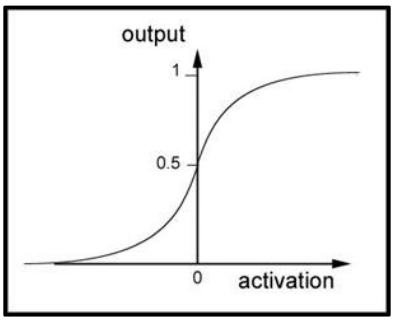


Figure 7 Sigmoid Function

2.2.3.2 Hyperbolic tangent function- $\tan h$

The problems of sigmoid can be solved using tan*h* as its output is zero centered.

- It ranges between -1 to 1.
- Optimization is easier
- But still suffers from gradient vanishing problem

- $f(x) = 1 - \frac{e^{-2x}}{1+e^{-2x}}$

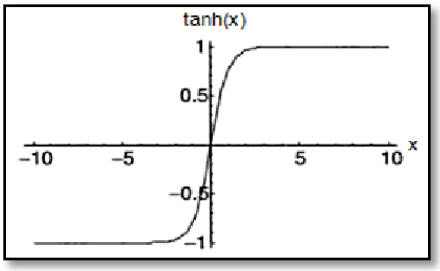


Figure 8 Hyperbolic tangent function

2.2.3.3 ReLu Function

The problems associated with the above mentioned functions can be greatly improved using ReLu function.

- It has 6x convergence improvement from tan*h* function.
- It avoids and rectifies vanishing problem of gradient.
- Almost all the deep learning models used relu now.
- Only limitation is that it should be used only with the hidden layers.
- $f(x) = \max(0, x)$

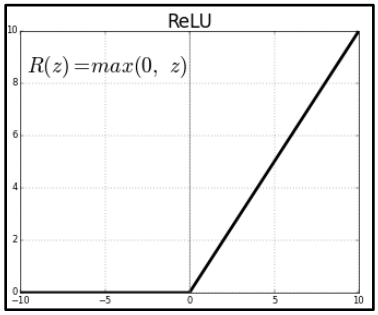


Figure 9 Rectified Linear Unit function

2.2.3.4 Softmax Function

It is used for classification problems to which output the probability for the classes and for a regression problem it uses simple linear function.

2.2.3.5 Leaky ReLu

The problem with relu is that some of the gradients can be fragile during training process and can ultimately die. That can cause a weight update which will make it to never activate on any data points again *i.e.* it can make neurons dead. By making a slight change in a relu function the problem associated like dead neurones can be solved.

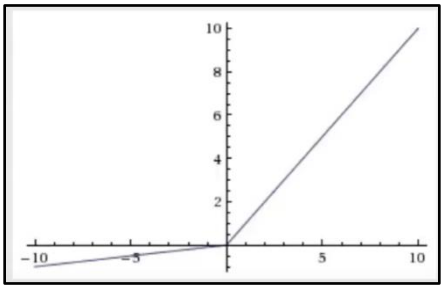


Figure 10 Leaky ReLu function

2.2.4 Learning Rule

A neuron is trained to make a response to every input vector with a corresponding output of zero or one. The learning rule is proven various times to converge to a solution if it exists. It can be given as:

$$b_i = b + [T - A]$$

For inputs *i:*

$$w[i] = w[i-1] + [T - A] * P[i]$$

Where,

w= vector of weights

P=input vector presented to the network

T=Correct result that neuron must show

A=Actual output of neuron

B=bias

2.2.5 Backpropagation

The way a neural network is able to update its kernel values called as weights is using a training method known as Backpropagation. In order for a neural network to work it needs to optimize your updated weights along the different layers so that these different layers corresponds to different feature extractions like edges, ridges, shapes, texture *etc*. These updating are done during the training phase where the images are feeded into the neural network. The training process is called Backpropagation [9].

Backpropagation can be separated into four different processes as the forward pass followed by a loss function calculation which proceeds the backward pass and finally the updation of the filter weight.

During the forward pass a batch of training images which is width x height x depth array of numbers and passed through the complete network. In the first pass the output would probably be justified for all the classes like [0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1] that is the output is completely uniform. This goes to the loss function part of the backpropagation. As we are feeding the training data the data has a label associated with it. The label will be of kind [0 1 0 0 0 0 0 0 0 0]. A loss function has many ways of defining but most commonly mean squared error MSE is used which is half Times actual minus predictable squared value *i.e.*

$$E_{Total} = \sum 0.5(target - output)^2$$

Loss would tend to be higher in the 1st set of training steps. The idea is to search for a point where the loss is minimum thereby predicting the right label for the given image.

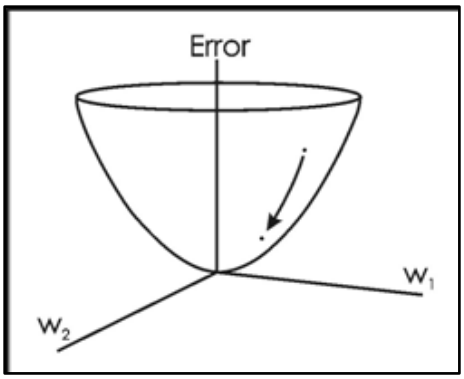


Figure 11 Error Minimization

As can be seen from the 3D figure the error or loss needs to be minimised. This can be done by adjusting the weights thereby reaching to a minimum Point. This is mathematical equal to slope dL/ dW, where W represents the weights of a previous layer.

This leads to performing a backward pass along the network and determining which weight contributes for the loss and finding ways to tweak them so that the loss can be minimized.

When this slope is computed then the weight update which is the last step is performed. This is the stage where all weights are taken and updating them so that they change in the direction opposite to the gradient.

$$w = w_i - \eta \frac{d\boldsymbol{L}}{d\boldsymbol{W}}$$

Where,

w= weight (updated)

w_i *=previous weight*

η*= learning rate*

The hyper parameter learning rate is to be set by the user. A higher learning rate mean larger steps are taken for updating the weights. It may take lesser time to converge on a given set of weights, but a high learning could jump much higher and not precisely to reach the valley Point.

During a training iteration all the tasks such as forward passing, loss function calculation, backward propagation and parameter updating is performed. The program repeats this process for a defined number of iterations called epoch.

Chapter 3

Proposed Work

In this section the proposed work inspired from Jongwon Chang *et al.* [2] in which they performed a pilot study based on the histopathology of breast cancer is presented for the application part classification of different lung diseases is to be implemented using transfer learning on different architectures and thereby evaluating their performance over the same.

In this thesis work an implementation of same task is performed across different architectures [10] using transfer learning and compares to verify the reliability and accuracy.

The primary motivation behind this implementation is to use transfer learning to make a neural network classify general things in common without being needed to train it explicitly for a given set of problem [11].

3.1 Dataset

For the proposed work we have used dataset of chest X-ray images (pneumonia) from Kaggle. There are 5863 X-ray images in JPG format comprising categories like Normal, Viral and Bacterial pneumonia.

Chest X-ray scan images are selected from subsequent quotes of paediatric patients in the age ranging 0 to 5 years from "*Guangzhou women and children Medical Centre, Guangzhou*". They are all taken as a part of patient's routine clinical care.

For the analysis of X-ray images chest radiographs is screened initially for Quality assessment by removing all the unreadable and poor quality scans. The diagnoses were then graded by two expert clinicians before training the neural network.

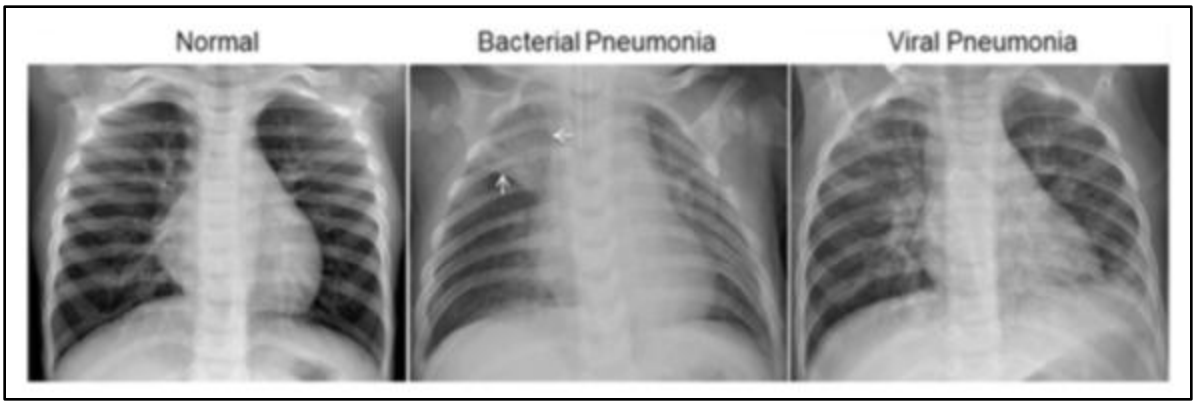


Figure 12 Chest X-Rays Pneumonia

The above figure is an example of chest X-ray in patients having pneumonia. As depicted from the figure X-Ray of normal person it shows clear lungs without having any areas of abnormal opafication in the given X-Ray scan. Bacterial pneumonia typically depicts a focal lobar consolidation where is in viral pneumonia it reveals with more diffused intersticial like pattern in both the lungs [12]. The organization of dataset is given in the table.

Type	Training	Validation	Testing
Normal	1342	8	234
Viral	1345	6	148
Bacterial	2530	8	242

Table 1 Distribution of Dataset

3.2 Pre-trained model-Inception V3

Since the 2012 image net competition to classify 1000 different classes there has been a creation of history in the field of computer vision [13] [14]. Alexnet by krizhevsky *et al.* Had been successful applied to a larger variety of image recognition task. These successes promoted to a larger research base focused on finding a higher performing CNN.

Starting in 2014 the quality of deep architectures improved significantly by the networks such as VGGNet and GoogleNet. Although there was a significant increase in the performance but also had a trade with its computational costs. As Alexnet used 60 million parameters, the VGGNet employed about three times larger amount of parameters than Alexnet. However the model

introduced by GoogleNet called Inception had parameters 12x less than the above mentioned architectures.

Model architecture	Parameters
Alexnet	60 Million
VGGNet	180 Million
GoogleNet (Inception)	5 Million

Table 2 No. of parameters used in different model architectures

This led to reduce in an overall computational costs compared to its predecessors. This is the primary reason for the selection of inception V3 model architecture for our application.

The name Inception V3 is because of the small parallel modules it uses which are basically Mini models embedded in a large model.

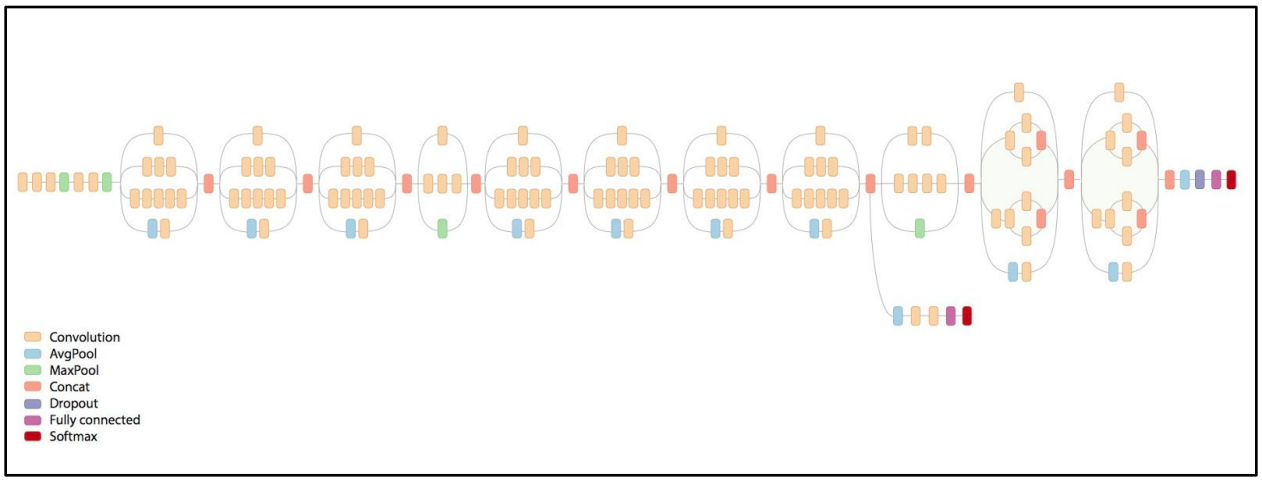


Figure 13 Inception Model Architecture (Google)

For a deep neural network we need to make a decision regarding which filter size should be used for convolution at each layer like a 3 x 3? or a 5 x 5? And this can generate different results. The idea is to use all sort of filters in parallel and let the model decide which to use. The convolutions are performed in parallel and are concatenated resulting feature maps before progressing to the subsequent layer. Additionally, this architecture allows the model to extract all local features by using filters with small sizes for calculating convolutions and high extracted features with larger size filters for calculating convolutions. Figure below describes the architecture of one inception module.

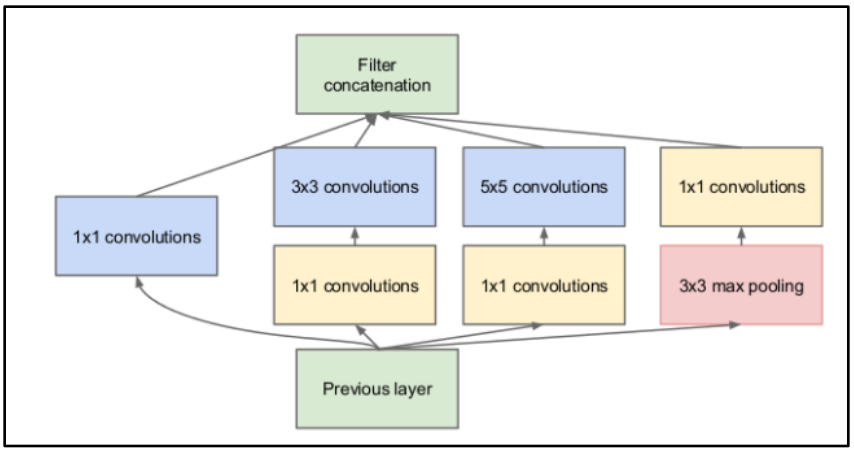


Figure 14 Inception Module

From the figure it can be observed that we have plenty of choices for convolution operation like 1 x 1, 3 x 3 & 5 x 5 along with a 3 x 3 pooling layer. Pooling is added to down sample the feature maps. The larger convolutions tends to be more expensive requiring more computations , so [13] suggested first performing 1 x 1 convolution for reducing the spatial dimension of its activation map then passing the results through relu and then performing the larger convolution (5 x 5, 3 x 3). The 1 x 1 convolution is the key to reduce the dimensionality of its feature map.

3.3 Dimensionality reduction

The [13] says we can use 1 x 1 convolution to reduce the dimensionality thereby reducing the computations instead of using a pooling layer.

3.4 Transfer learning

It is a technique where a model is trained model on one dataset and is reused for a different class of problem dataset. Model checkpoints are used for tuning the new network. Thus it is an optisation technique generally used for saving computational time and resources [15]. Figure below shows the process used in traditional machine learning and transfer machine learning approach.

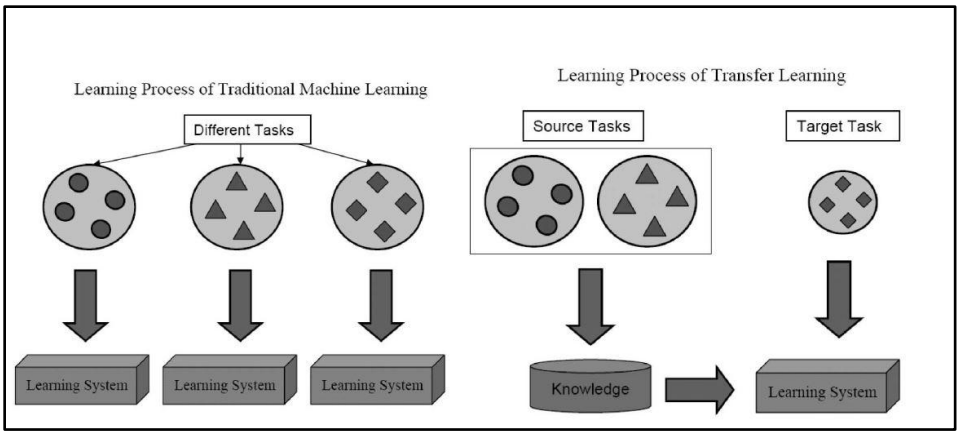


Figure 15 (a) Machine learning (b) Transfer learning

Two common approaches are used in transfer learning [16].

- Develop Model Approach
- Pre-trained Model Approach

3.4.1 Develop Model Approach

Selecting a source Task – Select a dataset which is similar to the model to be developed. This model should have abundance of data which is related to the new set of data. The data selected must have some sort of relationship between its input data , output data and the features learned during training phase.

Develop Source Model- Develop a model which is better than the original model developed such that the new features are learned during the training phase.

Reuse Model –Reuse the model by using the checkpoints of the originally developed model.This involves using some or all the parts of an originally developed model for devising the new set of task.

Tune Model – The redeveloped model may need a tuning to get adapted or refined on the input and output data pair for the task of interest.

3.4.2 Pre-trained Model Approach

Select Source Model- A model which is already trained on a huge set of data like imagenet is chosen from available models like AlexNet, Inception. Many organizations and research teams release models on vast variation datasets that can be used for constructing a new classifier.

Reuse Model- This selected model can then be retrained using a new classifier layer on top of the existing large model. The model checkpoints are used for the new set of training data. This

involves using some or all the parts of the existing model which truly depends on the modeling technique used for training a new classifier.

Tuning Model- The model may need a refinement for the new classification problem. It must get adapted to new task of interest. Different parameters can be tweaked for improving the performance.

The Pre-trained approach is used commonly in Deep learning.

Generally it is not a surety that the new model will perform better for the new class of problem until it is developed and evaluated . The possible benefits when employing transfer learning which can be looked upon are:

High start-The initial learning performance when employing transfer learning is higher as compared to the traditional machine learning training process.

High slope- The rate of learning during the training phase is steeper as compared to the traditional machine learning training process.

High asymptote- The converging skill of the retrained model is better as compared to the traditional machine learning process.

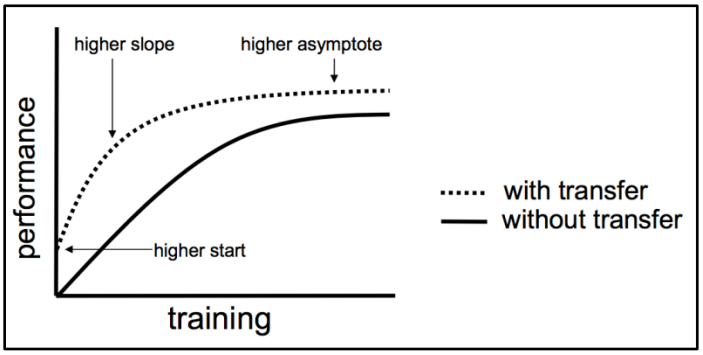


Figure 16 Advantages of transfer learning

Ideally, all the above benefits must be observed when employing transfer learning.

Chapter 4

Implementation

The block diagram of the proposed model is shown in Fig 17. As depicted from the block diagram a pre-trained model Inception V3 from Google is taken and is to be trained for a new set of training dataset *i.e.* classification of different types of Pneumonia.

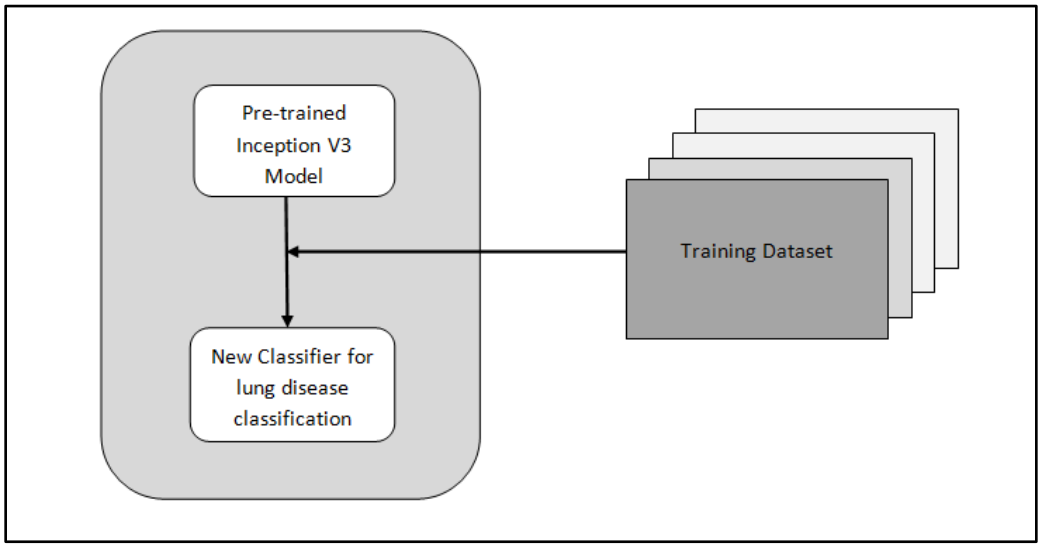


Figure 17 workflow to be implemented using transfer learning

This is implemented using the API provided by google's Tensorflow [17].

4.1 Preprocessing the images from data set

The image net model takes the images of size 299 x 299. Whereas MobileNet model uses images of size 224 x 224. The data set available does not have same sizes of images. Some images are smaller and some are larger in size. The whole data set images are resized according to above mentioned two sizes for feeding into our model.

4.2 Calculating Bottleneck values

The first process takes all the images and analyses them to calculate their bottleneck values. Bottleneck values are used for the output layer to aid the classifier classify different object categories. This also caches these values. This can be also called as "image feature vector". This layer is trained to output set of values that are accurate enough for distinguishing between the classes. It provides meaningful values and compact summary to the classifier to make a good choice among the set of classes.

Also these images are used multiple times during the training so it stores these bottleneck values to the cache so that they can be reused without having to be calculated again.

4.3 Hyperparameters Setting

There are various hyperparameters like the learning rate, training step size, training batch size. The learning rate is used to determine the step size which controls values of the weight updates for the final layer. Larger learning rate can aid in higher steps and lesser time to converge. But can jump off too frequently from the minima to achieve. A smaller learning rate however gives smaller step sizes and larger time to converge. This needs to be found out using experimentation. The training batch size determines how many images are feed into the network at each training step for estimating the weight update at each training steps for the final layer.

4.3 Training

The bottlenecks are calculated then the actual training of the topmost layer begins. The output counting steps shows training accuracy, validation accuracy and the cross entropy. The training accuracy visualizes the number of training images that are successfully classified in the current training step. The validation accuracy determines the percentage of images that are successfully classified taken from a different set. Validation accuracy is used for determining the learning of the network. This is necessary to know so that the network is not overfitting the data. The only difference between the two is that the network is trained using the training data and validation accuracy is used for checking between the training steps for checking the learning of the network. If the training accuracy is high and the validation accuracy is low that means the network is memorizing the training data. This is generally not helpful. Cross entropy gives a glimpse into how well the network is learning. This is a loss function whose objective is to make the losses as minimum as possible. The loss curve must keep trending downwards through ignoring the short time noises indicating an efficient training process.

We run the training for different number of training steps *i.e.* 4000, 6000, 8000 steps using the parameter --how_many_training_steps. Each step will choose ten images which is the size we have chosen for our batch by randomizing the training set, will calculate their bottleneck values and store them in cache for retrieving it fast and then feed them to the final layer to estimate the prediction. These estimations are then compared against the actual data labels for calculating the

loss function and updating the weights using backpropagation algorithm. As the process grow we are able to see the reported training , validation accuracy and the cross entropy. After completion of all the training steps a final accuracy test is done from the data which is different from the training data and the validation data This estimates the general performance of the model and how well the model has been able to classify the input images. After completion of the test the new model is saved along with the text file containing the labels of all the classes with a new layer on top of the model.

Cross entropy is the cost function which is calculated as follows:

$$H(x) = H(p) = -\sum_{i} p(x_i) \log(p(x_i))$$

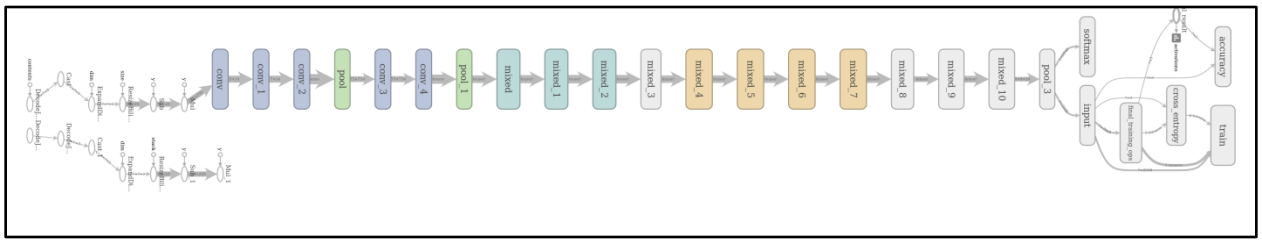


Figure 18 Retrained model

4.4 Visualizing the Retraining with Tensorboard

For visualizing all the flow of parameters Tensorboard is used. Tensorboard is used to draw visual TF graphs, plot value metrics about the graph execution, and additionally the data like images passing through various layers. During training the summary logs are save in a separate directory.

Tensorboard is started by giving the following command:

tensorboard --logdir /tmp/retrain_logs

By logging into localhost:6006 the summaries start visualizing the training process.

Fig tensorboard visual

4.5 Improving the Retrained Model

For enhancing the results of training the training data can be deformed, cropped or the brightness can be improved in random ways. This aids in increasing the data size virtually. This also ensures training the classifier for all the distortions which could be met in real world scenarios. The only disadvantage of using these ways is that the bottleneck values are no longer useful and

the training can take bigger time. To enable these distortions random cropping, random scaling and random brightness parameters are passed to the script. These are the percentage values which determine the amount of distortion added to the images.

Chapter 5

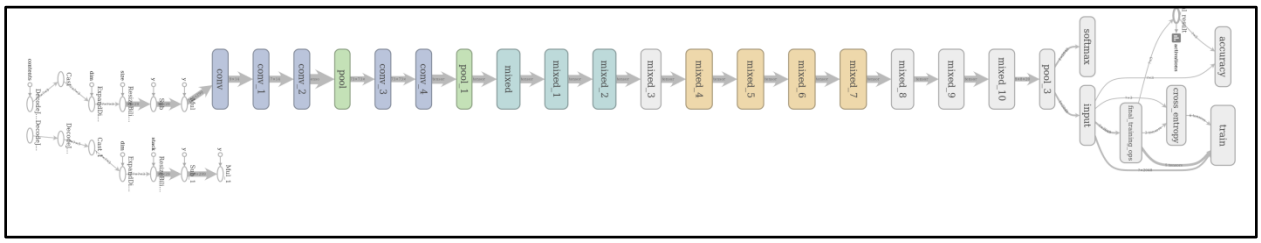
Results

The results of training of different architectures are given in this chapter.

1. Inception V3

Trained with Steps=4000

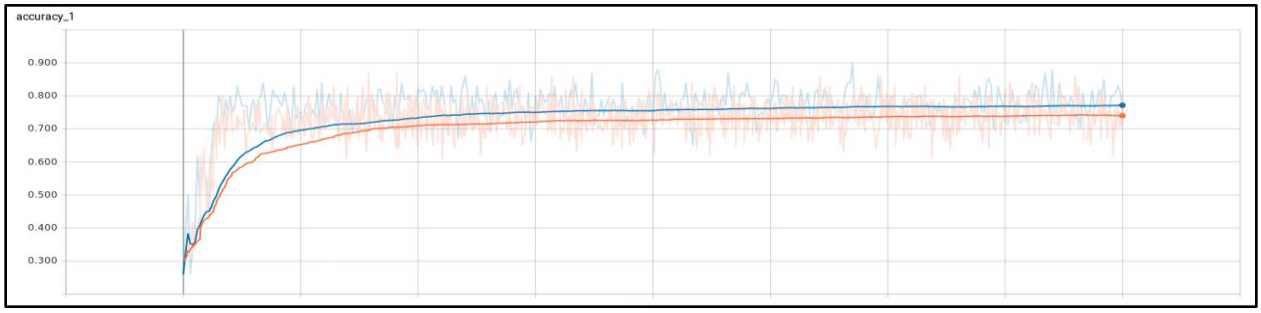
Learning Rate=0.0001



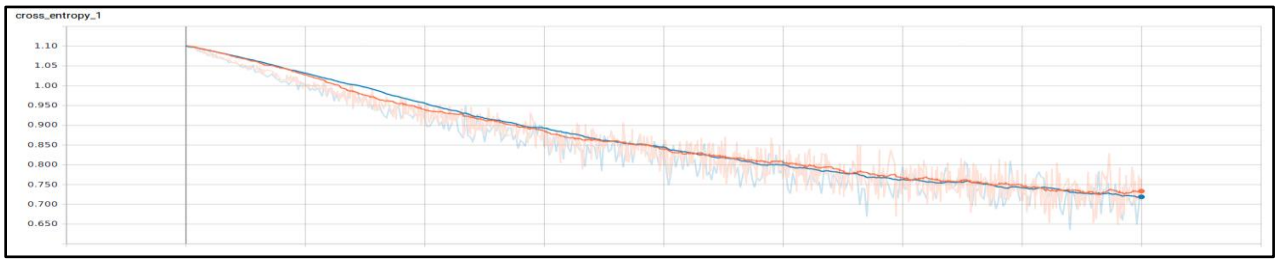
(a) – Retrained Graph

```
INFO:tensorflow:2018-07-06 02:47:32.365960: Step 3999: Train accuracy = 78.0%
INFO:tensorflow:2018-07-06 02:47:32.366198: Step 3999: Cross entropy = 0.708614
INFO:tensorflow:2018-07-06 02:47:32.463457: Step 3999: Validation accuracy = 77.
0% (N=100)
INFO:tensorflow:Final test accuracy = 71.2% (N=525)
INFO:tensorflow:Froze 2 variables.
Converted 2 variables to const ops.
caesor@caesor1:~/workspace/Inception V3_4000$
```

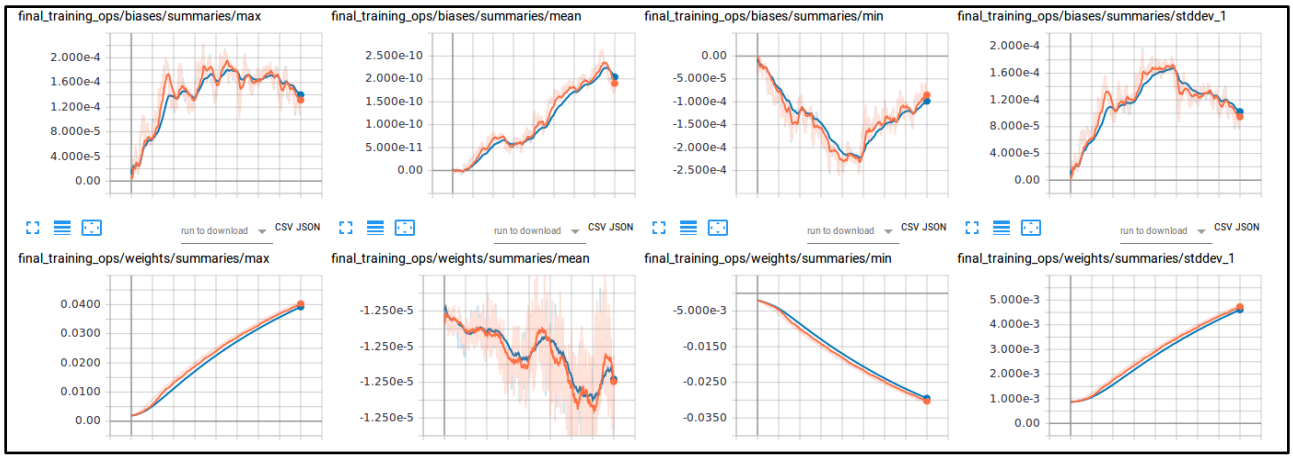
(b) – Test Accuracy



(c) – Training Accuracy vs Validation Accuracy



(d) – Cross Entropy



(e) – Final Training Outputs

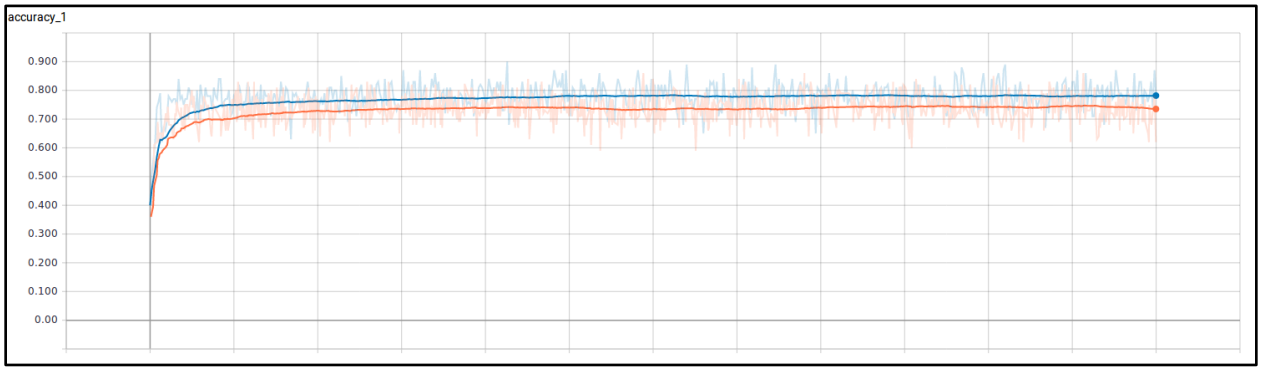
Figure 19 Execution result of Inception V3 model with steps=4000

Training with steps=6000

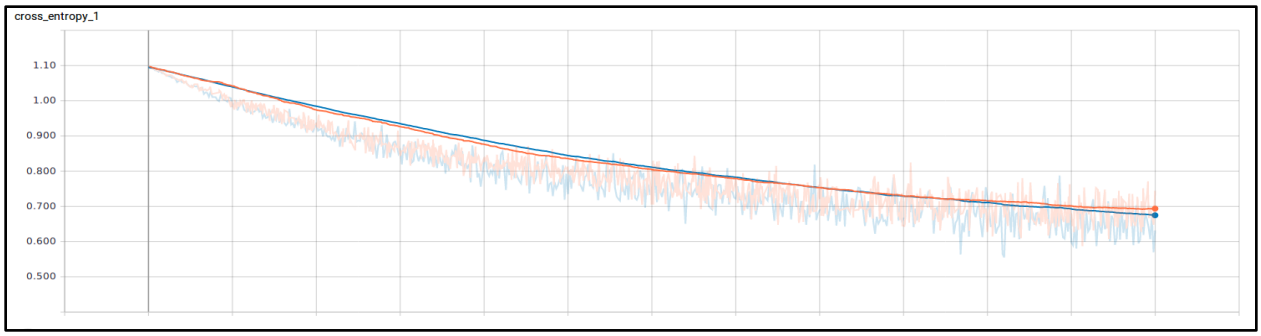
Learning Rate=0.0001

```
INFO:tensorflow:2018-07-06 03:35:00.215321: Step 5999: Train accuracy = 62.0%
INFO:tensorflow:2018-07-06 03:35:00.215550: Step 5999: Cross entropy = 0.744038
INFO:tensorflow:2018-07-06 03:35:00.343359: Step 5999: Validation accuracy = 79.
0% (N=100)
INFO:tensorflow:Final test accuracy = 71.4% (N=525)
INFO:tensorflow:Froze 2 variables.
Converted 2 variables to const ops.
```

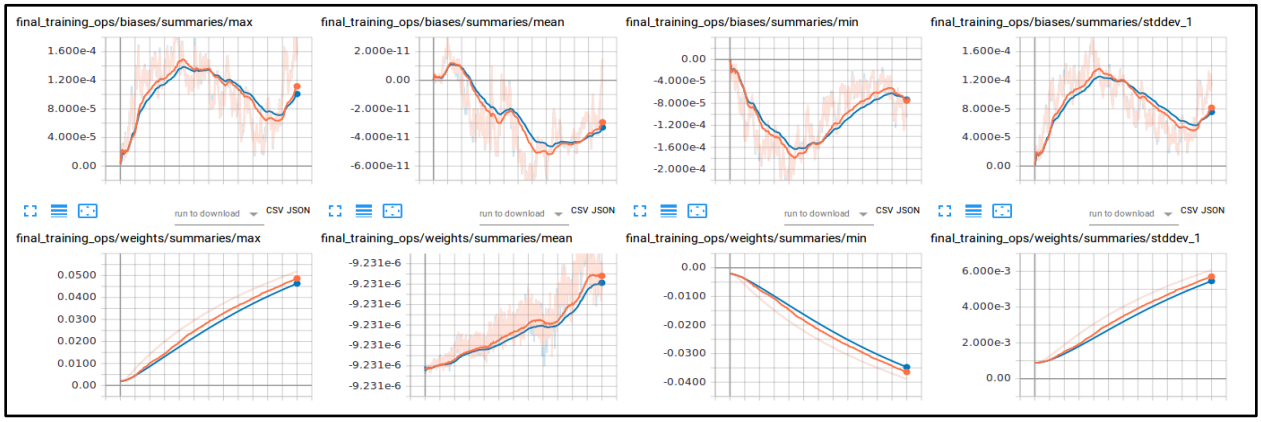
(a) – Test Accuracy



(b) - Training Accuracy vs Validation Accuracy



(c) – Cross Entropy



(d) – Final Training Outputs

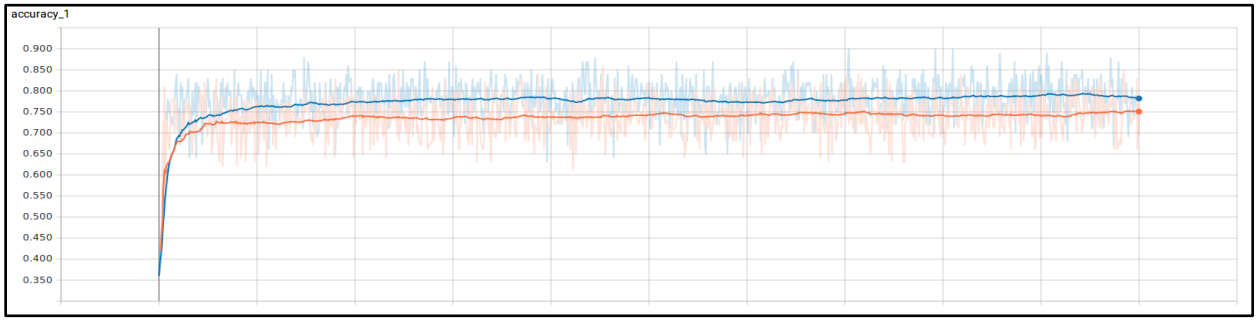
Figure 20 Execution result of Inception V3 model with steps=6000

Training with steps=10000

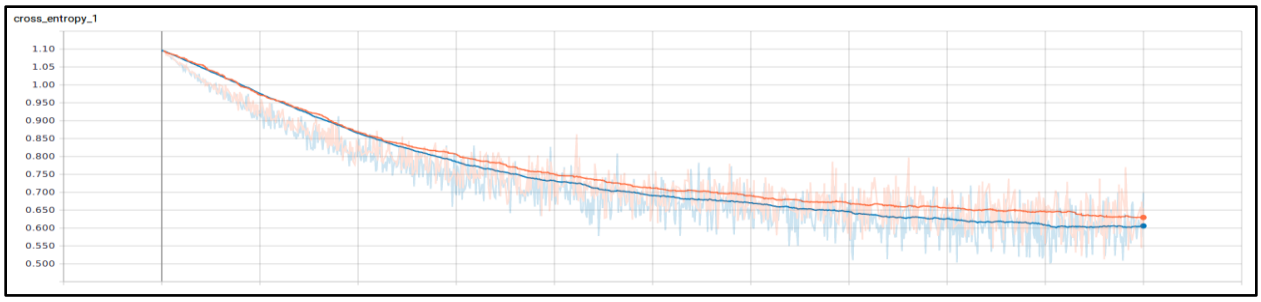
Learning Rate=0.0001

```
INFO:tensorflow:2018-07-06 04:47:10.367107: Step 9999: Train accuracy = 77.0%
INFO:tensorflow:2018-07-06 04:47:10.367345: Step 9999: Cross entropy = 0.612419
INFO:tensorflow:2018-07-06 04:47:10.461926: Step 9999: Validation accuracy = 71.0% (N=100)
INFO:tensorflow:Final test accuracy = 72.0% (N=525)
INFO:tensorflow:Froze 2 variables.
Converted 2 variables to const ops.
caesor@caesor1:~/workspace/Inception_V3_10000$
```

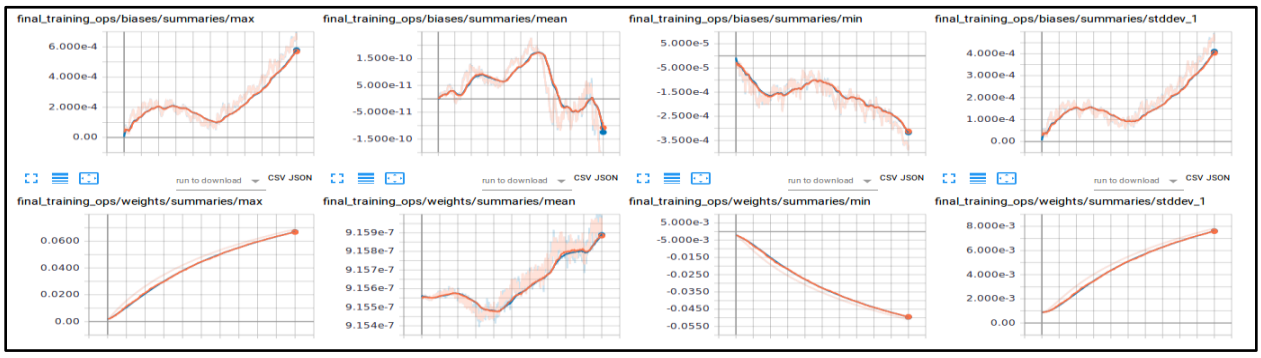
(a) – Test Accuracy



(b) - Training Accuracy vs Validation Accuracy



(c) – Cross Entropy



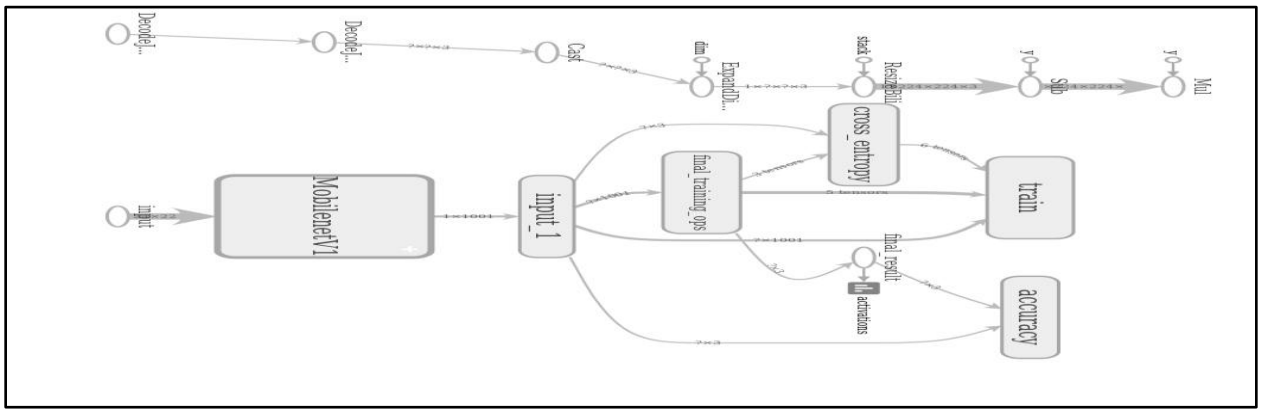
(d) – Final Training Outputs

Figure 21 Execution result of Inception V3 model with steps=10000

2. MobileNet

Training with steps=4000

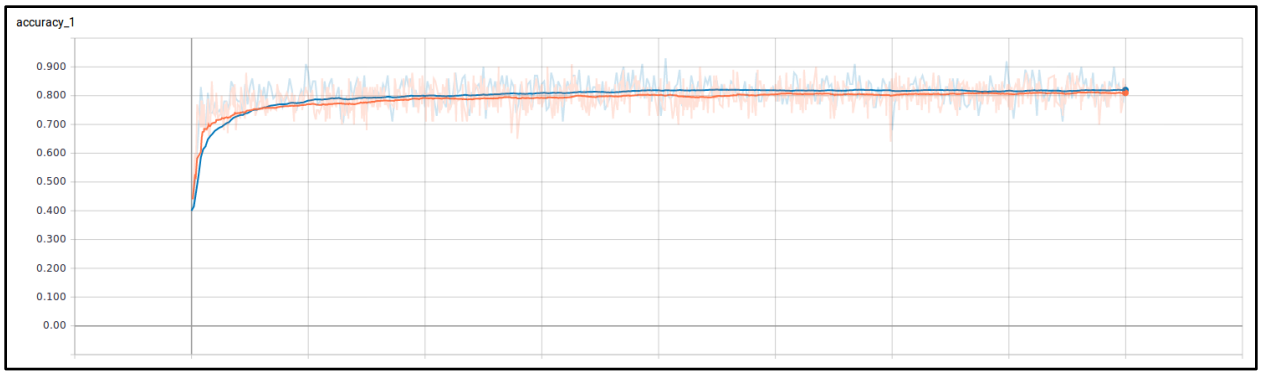
Learning Rate=0.0001



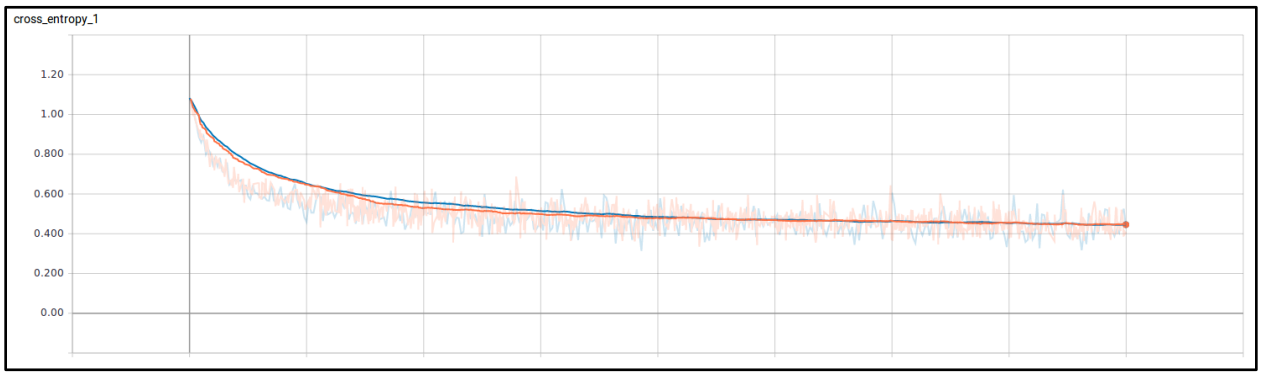
(a) – Retrained Graph

```
INFO:tensorflow:2018-07-06 01:25:20.998034: Step 3999: Train accuracy = 81.0%
INFO:tensorflow:2018-07-06 01:25:20.998283: Step 3999: Cross entropy = 0.407289
INFO:tensorflow:2018-07-06 01:25:21.057335: Step 3999: Validation accuracy = 84.0% (N=100)
INFO:tensorflow:Final test accuracy = 77.9% (N=525)
INFO:tensorflow:Froze 2 variables.
Converted 2 variables to const ops.
caesor@caesor1:~/workspace/mobilenet_4000_0.0001$
```

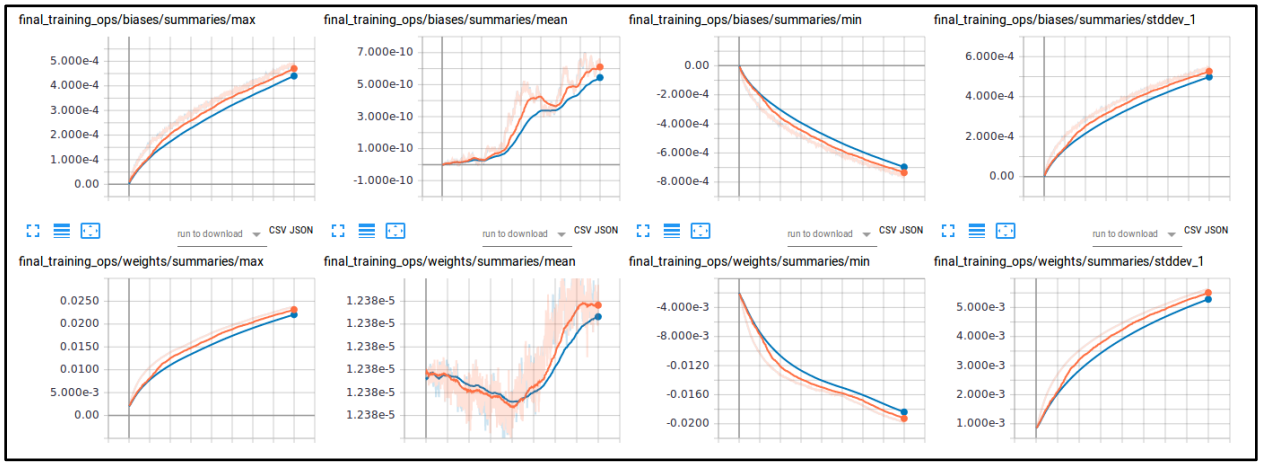
(b)- Test Accuracy



(c)- Training Accuracy vs Validation Accuracy



(d) – Cross Entropy



(e) – Final Training Outputs

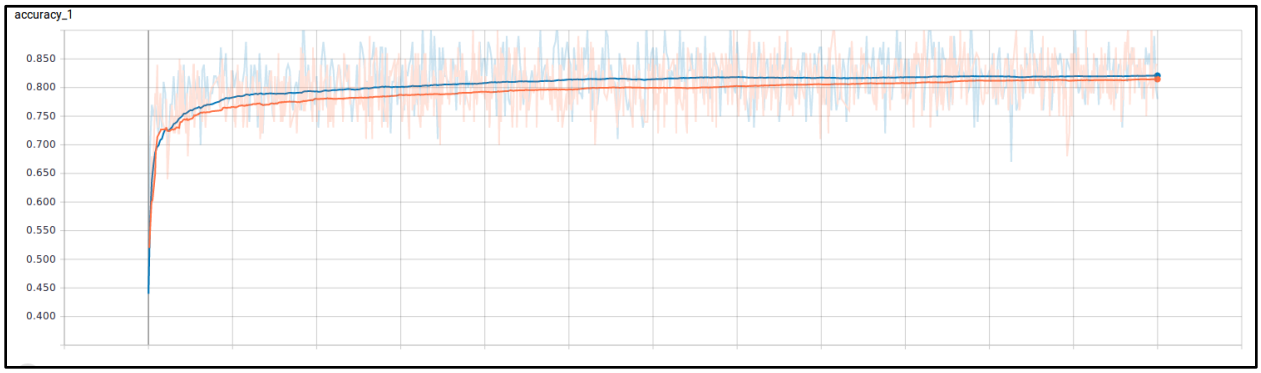
Figure 22 Execution result of MobileNet architecture with steps=4000

Training with steps=6000

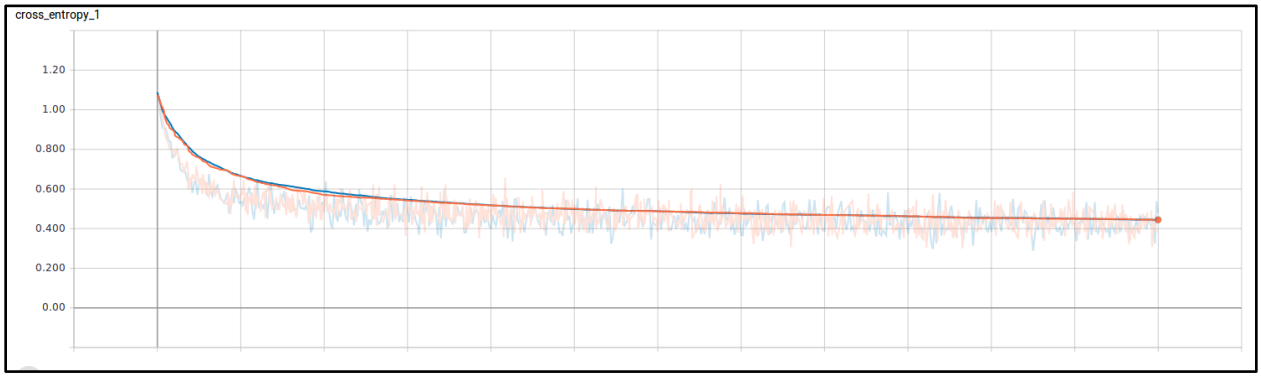
Learning Rate=0.0001

```
INFO:tensorflow:2018-07-06 01:59:36.633841: Step 5999: Train accuracy = 87.0%
INFO:tensorflow:2018-07-06 01:59:36.634059: Step 5999: Cross entropy = 0.382678
INFO:tensorflow:2018-07-06 01:59:36.684058: Step 5999: Validation accuracy = 78.0% (N=100)
INFO:tensorflow:Final test accuracy = 77.5% (N=525)
INFO:tensorflow:Froze 2 variables.
Converted 2 variables to const ops.
caesor@caesor1:~/workspace/mobilenet_6000_0.0001$
```

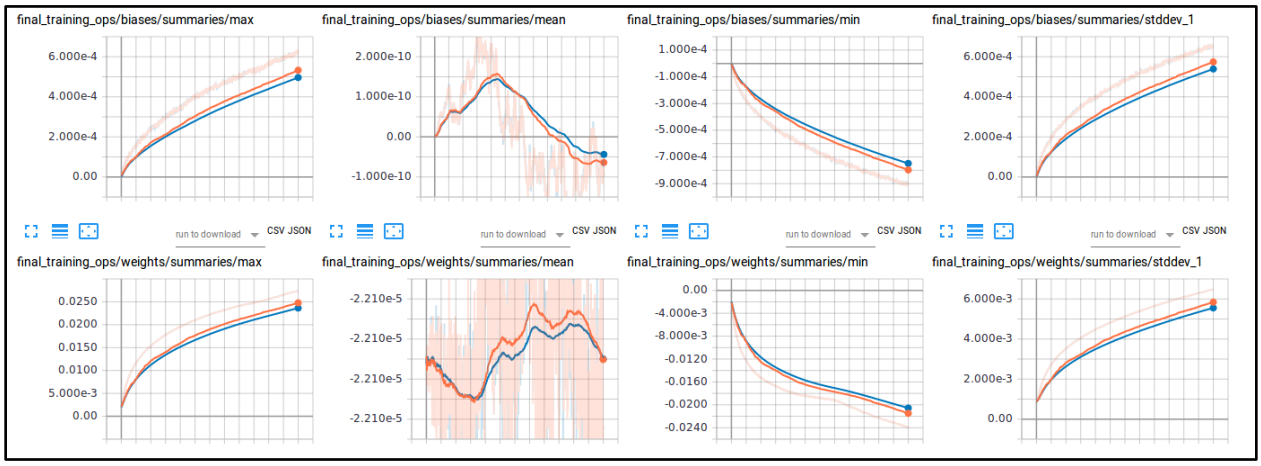
(a)- Test Accuracy



(b)-Training Accuracy vs Validation Accuracy



(c)-Cross Entropy



(d) - Final Training Outputs

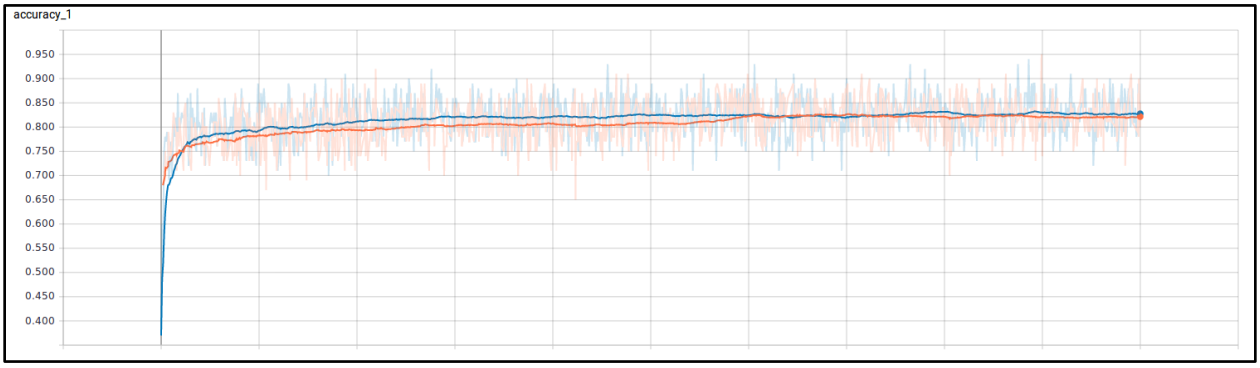
Figure 23 Execution result of MobileNet architecture with steps=6000

Training with steps=10000

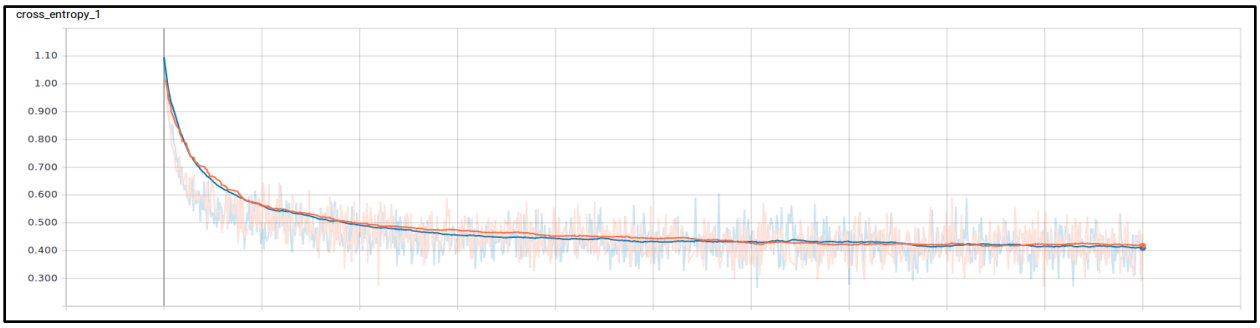
Learning Rate=0.0001

```
INFO:tensorflow:2018-07-06 04:00:15.014934: Step 9999: Train accuracy = 86.0%
INFO:tensorflow:2018-07-06 04:00:15.015134: Step 9999: Cross entropy = 0.387802
INFO:tensorflow:2018-07-06 04:00:15.066838: Step 9999: Validation accuracy = 82.
0% (N=100)
INFO:tensorflow:Final test accuracy = 77.9% (N=525)
INFO:tensorflow:Froze 2 variables.
Converted 2 variables to const ops.
caesor@caesor1:~/workspace/mobilenet_10000_0.0001$
```

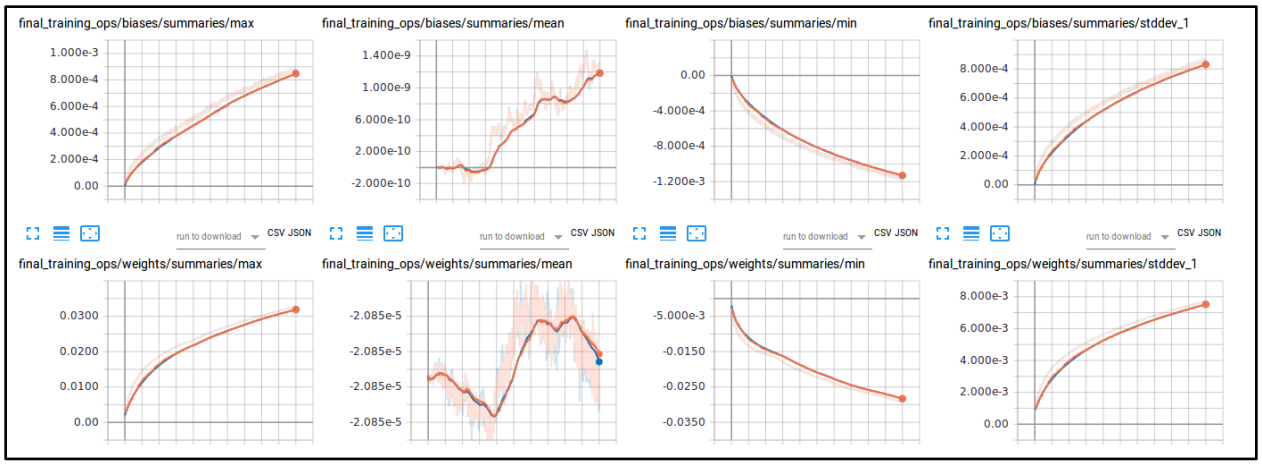
(a)- Test Accuracy



(b)-Training Accuracy vs Validation Accuracy



(c)-Cross Entropy



(d) - Final Training Outputs

Figure 24 Execution result of MobileNet architecture with steps=10000

3. Results Summary

Table 3 shows a summary. As can be seen Mobilenet architecture has performs well as compared to Inception V3 architecture. Accuracy of Inception model is not very tempting and the cost metric *i.e.* cross entropy is also high. Ideally it tends toward zero as the neuron gets better at computing the desired output for all training inputs. Cross entropy of MobileNet architecture is quite lesser than the inception model.

Model Architecture	Training Steps	Training Accuracy	Validation Accuracy	Cross Entropy	Final Test Accuracy
MobileNet	4000	81%	84%	0.40	78%
	6000	87%	78%	0.38	77.5%
	10000	86%	82%	0.38	77.9%
Inception V3	4000	78%	77%	0.70	71.2%
	6000	62%	79%	0.74	71.4%
	10000	77%	71%	0.61	72%

Table 3 Results Summary

The Final test accuracy is also comparable. MobileNet has been able to classify almost 78% of the input images representing different kinds of Pneumonia while Inception model had passed for only 71.5% on test images.

Chapter 6

Conclusion

The MobileNet architecture is computationally cheaper than inception v3 module on the same ground but outperforms in classification of medical image classification. This also infers that small model can be trained in order to achieve higher accuracy for small classification problems having small dataset. Training a classifier using transfer learning approach reduces the computational efficiency requires for training any neural network.

The inception module does not generalize well in extracting the features for a new classifier. Though we get good results but there is a need for improvement in following transfer learning approach.

Future Scope

Although there is a breakthrough in the performance in the domain of computer vision there is a need for improvement in generalizing a model so that it can be retrained explicitly for new class of objects as training any model from scratch requires high computations and time.

Bibliography

[1] Neena, and M. Geetha Aloysius, "A review on deep convolutional neural networks," in *In Communication and Signal Processing (ICCSP) , IEEE*, 2017, pp. 0588-0592.

[2] Jisang Yu, Taehwa Han Jongwon Chang, "A Method for Classifying Medical Images using Transfer Learning: A Pilot Study on Histopathology of Breast Cancer," in *e-Health Networking, Applications and Services (Healthcom),IEEE*, Dalian, China, 2017.

[3] D.S., Goldbaum, M., Cai, W., Valentim, C.C., Liang, H., Baxter, S.L., McKeown, A., Yang, G., Wu, X., Yan, F. and Dong Kermany, "Identifying medical diagnoses and treatable diseases by image-based deep learning," *Cell*, vol. 5, pp. 1122-1131, 2018.

[4] Qiang Yang Sinno Jialin Pan, "A Survey on Transfer Learning," in *IEEE Transactions on knowledge and data engineering*, 2010, pp. 1345-1359.

[5] Jeff Clune, Yoshua Bengio, Hod Lipson Jason Yosinski, "How transferable are features in deep neural networks," in *Advances in neural information processing systems*, 2014, pp. 3320-3328.

[6] Adit Deshpande. (2016, July) A Beginner's Guide To Understanding Convolutional Neural Networks. [Online]. https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

[7] Matthew D., and Rob Fergus. Zeiler, "Visualizing and understanding convolutional networks.," *European conference on computer vision Springer, Cham*, pp. 818-833, 2014.

[8] Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed Christian Szegedy, "Going Deeper with Convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, Boston, MA, USA, 2015, pp. 1-9.

[9] Nirmala, and Sachchidanand Singh. Singh, "Object classification to analyze medical imaging data using deep learning," in *Innovations in Information, Embedded and Communication Systems (ICIIECS) International Conference, IEEE*, 2017.

[10] Andrew G., Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Howard, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[11] Pratik. Devikar, "Transfer Learning for Image Classification of various dog breeds.," *International Journal of Advanced Research in Computer Engineering and Technology*, vol. 5, no. 12, pp. 2707-2715, 2016.

[12] Pranav, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Ding et al. Rajpurkar, "Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning," *arXiv preprint arXiv:1711.05225*, 2017.

[13] Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens Christian Szegedy, "Rethinking the Inception Architecture for Computer Vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 2818-2826.

[14] Alex, Ilya Sutskever, and Geoffrey E. Hinton Krizhevsky, "Imagenet classification with deep convolutional neural networks.," *Advances in neural information processing systems*, pp. 1097-1105, 2012.

[15] Xiaoling, Cui Xu, and Bing Nan Xia, "Inception-v3 for flower classification," in *Image, Vision and Computing (ICIVC), 2nd International Conference, IEEE*, pp. 783-787.

[16] Jason Brownlee. (2017, December) A Gentle Introduction to Transfer Learning for Deep Learning. [Online]. https://machinelearningmastery.com/transfer-learning-for-deep-learning/

[17] Faizan Shaikh. (2016, October) An Introduction to Implementing Neural Networks using TensorFlow. [Online]. https://www.analyticsvidhya.com/blog/2016/10/an-introduction-to-implementing-neural-networks-using-tensorflow/

[18] Bluetooth SIG. (2016, December) Core Specifications. [Online]. https://www.bluetooth.com/specifications/bluetooth-core-specification

Part-B

Internship Work at Intel Corporation India Pvt. Ltd.

Chapter 1

Introduction

The Bluetooth is a universal radio system that enables electronic devices to establish short-range wireless connections. This wireless technology adopts packet-based communication scheme which is similar to other standards of data communication systems such as Wi-Fi and ZigBee. It was originally developed as a cable replacement to connect devices such as mobile phone handsets, headsets, and portable computers. The key features of Bluetooth wireless technology are robustness, low power consumption, and low cost. Its major use case was an audio link from the cell phone to a headset placed on or around the ear. With advancements in technology, more and more use cases were added, including stereo music streaming, wireless printing, data (meeting schedules, phone numbers), audio, graphic images and video from one device to the other etc.

There are two forms of Bluetooth wireless technology systems [18] : 1.Basic Rate (BR) and 2. Low Energy (LE). Both systems include device discovery, connection establishment and connection mechanisms.

The Basic Rate system may include Enhanced Data Rate (EDR), Alternate Media Access Control (MAC) and Physical (PHY) layer extensions. The Basic Rate system offers synchronous and asynchronous connections with data rates of 721.2 kb/s for Basic Rate, 2.1 Mb/s for Enhanced Data Rate and high speed operation up to 54 Mb/s with the 802.11 AMP.

The LE system includes features designed for products that require lower current consumption, lower complexity and lower cost than BR/EDR. The LE system is designed for use cases and applications with lower data rates and has lower duty cycles. Depending on the use case or application, one system including any optional parts may be more optimal than the other.

Bluetooth low energy makes it possible to build two types of devices: dual-mode and single-mode devices.

	Single-Mode	Dual-Mode	Classic
Single-Mode	LE	LE	None
Dual-Mode	LE	Classic	Classic
Classic	None	Classic	Classic

Table 4 Single-Mode, Dual-Mode, and Classic Compatibility

The Bluetooth core system consists of a Host and one or more Controllers. A Host is a logical entity defined as all of the layers below the non-core profiles and above the Host Controller Interface (HCI). A Controller is a logical entity defined as all of the layers below HCI. An implementation of the Host and Controller may contain the respective parts of the HCI. Two types of Controllers are defined in this version of the Core Specification: Primary Controllers and Secondary

- An implementation of the Bluetooth Core has only one Primary Controller which may be one of the following configurations:
- A BR/EDR Controller including the Radio, Baseband, Link Manager and optionally HCI.
- An LE Controller including the LE PHY, Link Layer and optionally HCI.
- A combined BR/EDR Controller portion and LE Controller portion into a single Controller.

1.1 Bluetooth Version History

1.2	Adaptive frequency hopping & Extended SCO link.
2.0	Enhanced Data Rate (EDR). EDR provides a set of additional packet types that use the new 2 Mb/s and 3 Mb/s modes.
2.1	Secure Simple Pairing
3.0	Added IEEE 802.11 as alternate physical channel for high speed transfer
4.0	Included Bluetooth Low energy for IOT applications
4.1	• 32-bit UUID Support in LE • LE Dual Mode Topology
4.2	• LE Data Packet Length Extension • LE Secure Connections • Link Layer Privacy
5.0	• Slot Availability Mask (SAM) • 2 Msym/s PHY for LE • LE Long Range • High Duty Cycle Non-Connectable Advertising • LE Advertising Extensions

Table 5 Bluetooth version history

1.2 Classic Bluetooth vs Low Energy

Bluetooth Classic	Bluetooth Low Energy
79 channels with raw symbol rate of 1 Msymbols/s	40 channels with raw symbol rate of 1 Msymbols/s
1 MHz. channel bandwidth	2 MHz. channel bandwidth
GFSK, QPSK modulation schemes is used	GFSK is used
No advertising of channel is there BT device directly inquires for the devices and creates connection	3 advertising and 37 data channels are there, No inquiry is required. Devices keep on advertising through advertising packets, interested device initiates the connection and becomes master
Connection is slow in conventional Bluetooth	Quickly connects two devices

Table 6 Classic Bluetooth vs Low Energy

1.3 BR/EDR OPERATION

The Basic Rate / Enhanced Data Rate (BR/EDR) radio (physical layer or PHY) operates in the unlicensed ISM (Industrial, Scientific and Medical radio) band at 2.4 GHz frequency. The system employs a frequency hopping transceiver to combat interference and fading and provides many FHSS carriers. The symbol rate is 1 mega symbol per second (Msym/s) supporting the data rate of 721 Kb/s. Enhanced Data Rate has a primary modulation mode that provides a gross air data rate of 2 Mb/s, and a secondary modulation mode that provides a gross air data rate of 3 Mb/s and an effective data rate of 2.1 Mb/s.

Bluetooth operates at 2.4 GHz and employs frequency hopping techniques with the carrier modulated using Gaussian Frequency Shift Keying (GFSK). Bluetooth frequencies are located within the 2.4 GHz ISM band. The ISM band ranges from 2400 MHz to 2483.5 MHz. The Bluetooth channels are spaced 1 MHz apart, starting at 2402 MHz and finishing at 2480 MHz. This can be calculated as 2401 + n, where n varies from 1 to 79.It has a guard band of 2 MHz at the bottom end of the band and 3.5 MHz at the top.

The Bluetooth standard is based upon a master/slave operational mode. The Bluetooth system provides a point-to-point connection or a point-to multipoint connection. In a point-to-point connection the physical channel is shared between two Bluetooth devices. In a point-to-

multipoint connection, the physical channel is shared among several Bluetooth devices. Bluetooth network consists of two topologies Piconet and Scatternet.

Piconet: Two or more devices sharing the same physical channel form a Piconet. One Bluetooth device acts as the master of the Piconet, whereas the other device(s) act as slave(s). The channel access is controlled by the master it controls the Frequency hopping, Timing and synchronization. Up to seven slaves can be active in the piconet limiting the maximum number of devices in a network to 8(7 slaves and 1 master).

Scatternet: Piconets that have common devices are called a Scatternet, each Piconet has only a single master but slaves can participate in multiple different Piconets using time-division multiplexing, a master in one Piconet can be a slave in other Piconets. Piconets in a Scatternet shall not be frequency synchronized and every Piconet has its own hopping sequence.

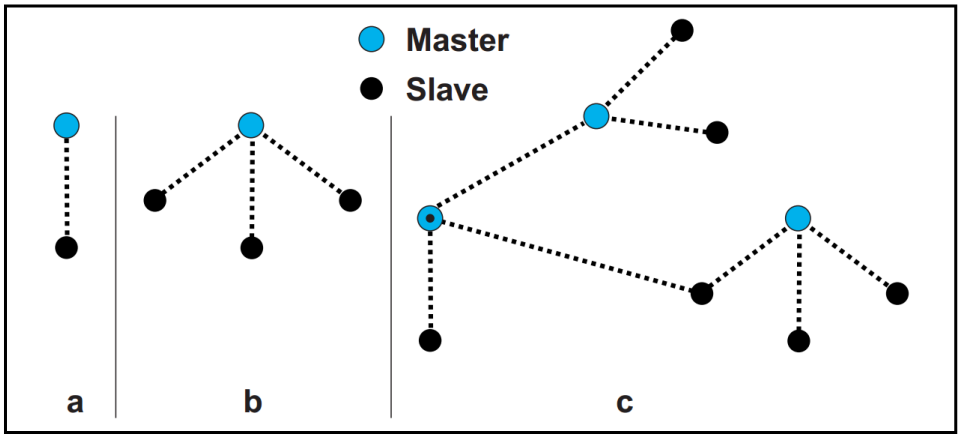


Figure 25 (a) Piconets with a single slave, (b) a multi-slave and (c) a Scatternet

There are two main types of Bluetooth link that are available and can be set up:

- **ACL** Asynchronous Connectionless communications Link
- **SCO** Synchronous Connection Orientated communications link

The ACL (Asynchronous Connectionless Communications Link) is mostly used Bluetooth link. The ACL Bluetooth link is generally used for carrying framed data like, data submitted from an application to logical link control and adaptation protocol channel. The channel may support either unidirectional or bidirectional Bluetooth data transfer. The ACL provides connections for most applications in Bluetooth. Data transfers are normally supported by profiles which allow the data to be incorporated into frames and transferred over Bluetooth link and is extracted from the frames and passed to the relevant application at the receiving device.

The SCO (Synchronous Connection Orientated communications link) is used where data is to be streamed rather than transferred in a framed format. A Bluetooth master node can support up to three simultaneous SCL channels and these can be split between up to three slave nodes.The SCO was developed to ensure that audio data can be streamed without suffering delays waiting for frames or packet slots to become available. The SCO links are assigned guaranteed time slots so that they will transport data within the required time with a known maximum latency.

The establishment of a Bluetooth network goes through four stages. Initially, all devices are in the **standby** mode, and they do not know about each other. To know about its neighbors, the master device initiates an **inquiry** as a request for information of other devices in its vicinity. The inquired devices respond by sending an inquiry response to the inquiring devices. The inquiring device (Master) becomes aware of other devices in its range, but no connection is established at this state. To start a connection, the device sends a **page request** to the intended device. In the paging states both the master and slave synchronize their local clocks and setup a frequency hopping sequence. The paged device will respond and starts a connection procedure and the two get **connected,** a device in the connected state can also inquiry and page, this is used to establish scatternets.

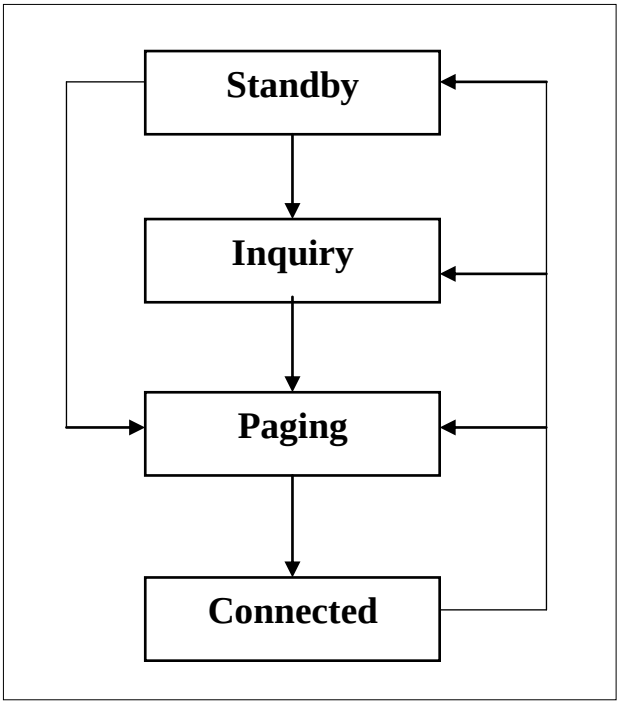


Figure 26 Transition states of a Bluetooth

Connection state can be in any of the modes as described below:

Active Mode:

In this mode, the Bluetooth device actively participates in the Piconet. The master schedules the transmission based on requirements to and from the different slaves. It supports frequent transmissions to keep slaves synchronized to the channel.

Sniff Mode:

In this mode, a slave device listens to the Piconet at slower rate, reducing its power consumption. The interval time is programmable and depends on the application.

***Hold Mode*:**

The master device can put slave device into HOLD mode, in this mode only an internal timer is running. Slave units can also demand to be put into HOLD mode. Data transfer restarts instantly when units transition out of HOLD mode.

Park Mode:

In this mode, the Bluetooth device is still synchronized to the Piconet but does not participate in the traffic. A Piconet can have up to 255 devices when in parked mode.

Chapter 2

Project profile at Intel Corporation India Pvt. Ltd.

1. Stability Regression for BR/EDR Controller

In this task test cases for the Bluetooth controller are to be performed and debug if any issue arises. Regression testing is done to check the functioning of Bluetooth operations due to some hardware or firmware change. There are two kinds of testing *i.e.* Functional and Stability. Functional testing is done to test the proper functionality of a Bluetooth device. Stability testing is performed to test the efficiency of the Bluetooth controller under stress tests. The types of tests that are performed are as follows:

1.1 Inquiry / Inquiry Scan

Objective: To perform interlaced inquiry scan in Piconet and repeating the same for 1000 cycles to check for the stability. Changing device role and repeating the same.

- Multiple devices with Inquiry access codes in Scatternet + ACL connection=> Master/slave performing inquiry for multiple IAC + Duplex data transfer using all packet types. Repeating for 1 hour.
- Changing master and slave role and repeating the same.

Expected Result

within 1000 loops the module Inquiry for checking R1 Scan Repetition mightsometimes fail.

Possible Reasons for a fail:

Expected Devices not found

CMD Status - CMD disallowed

Other reason for fail

Any of the above described failure have to be counted and displayed accordingly.

1.2 Page / Page scan

Objective: Defining page scan activity (interval and window) & Performing Interlaced page scan in a Piconet.

- Create an encrypted ACL connection between two devices and repeating same for 1000 loops.

- Switching the devices (device previously under page test -> device under page scan test) and repeating the same.

Expected Result

Within 1000 loops the module encrypted ACL-Connection Setup might sometimes fail.

Possible Reasons for a fail

Connection Complete - Page Tout

Connection Complete - Connection Timeout

Other reason for fail

Any of the above described failure has to be counted and displayed accordingly.

1.3 ACL Data P2P

Objective: To perform 1 hour Stable Full Duplex Data Transmission using different packet types and encryption in a Piconet/Scatternet.

- Setting up host flow config EDR. Encrypted ACL connection between two devices. 1 hour of duplex data transmission between devices. (Packet types DM1, DM3, DM5, 2-DH1, 2-DH3, 2-DH5, 3-DH1, 3-DH3, 3-DH5).
- Setting small buffer size (40), large packet size (1021). Establishing ACL connection and performing full duplex data transmission. Repeat for 1 hour.
- Setting large buffer size (512), large packet size (1021). Establishing ACL connection and performing full duplex data transmission. Repeat for 1 hour.

Expected Result

No link failures

1.4 ACL Data Piconet/Scatternet + SCO

Objective: To perform 1 hour/3MB Stable Full Duplex Data Transmission on one link & voice transmission on another in a Piconet/Scatternet using different packet types.

- Two different connection setup – Encrypted ACL + SCO (Piconet) - 3MB duplex data transmission on ACL link (DM1 & DH1) and voice transmission on other link.
- Two different connection setup – Encrypted ACL + SCO (Piconet) – One hour duplex data transmission on ACL link (DM1 & DH1) and voice transmission on other link.
- Link exchange for the same cases.

- Two different connection setup – Encrypted ACL + SCO (Scatternet) - 3MB duplex data transmission on ACL link (DM1 & DH1) and voice transmission on other link.
- Two different connection setup – Encrypted ACL + SCO (Scatternet) – One hour duplex data transmission on ACL link (DM1 & DH1) and voice transmission on other link.
- Two Encrypted ACL link + SCO link in Piconet performing 3MB duplex data transmission on both link and voice also.
- Two Encrypted ACL link + SCO link in Piconet/Scatternet performing one hour duplex data transmission on both link and voice also.

Expected Result

No data failures, Good voice quality, SCO link stable

2. Manual Testing of Privacy Feature of Bluetooth LE controller

From Bluetooth 4.0, Bluetooth with low energy supports a feature that reduces the ability to track a LE device over a period of time by changing the Bluetooth device address on a frequent basis. The frequently changing address is called the private address and the trusted devices can resolve it. In order to use this feature, the devices involved in the communication need to be previously paired. The private address is generated using the devices IRK exchanged during the previous pairing/bonding procedure. There are two variants of the privacy feature. In the first variant, private addresses are resolved and generated by the Host. This is used in the pre-4.2 Bluetooth stacks. In the second variant, private addresses are resolved and generated by the Controller without involving the Host after the Host provides the Controller device identity information. Bluetooth 4.2 compliant devices use this design.

Bluetooth Smart devices, such as activity trackers, announce their presence to other devices through a process known as advertising. Since most of the Bluetooth with low energy advertisement and data packets have the source addresses of the devices that are sending the data, third-party devices could associate these addresses to the identity of a user and track the user by that address. This can be protected by frequently changing private addresses so only the trusted parties could resolve them. This feature causes the MAC address within the advertising packets to be changed with a random value that changes at timing intervals determined by the manufacturer. Any malicious device(s), placed at intervals along your travel route, would not be

able to determine that the series of different, randomly generated MAC addresses received from your device actually relates to the same physical device. It actually looks like a series of different devices, thus, it will not be possible to track you using the advertised MAC address.

Without privacy any other device can sniff the packets and collect information as it gets to know the identity of the device, it is a serious concern in the IOT age where everything is connected. If the Bluetooth has a fixed constant address a man in the middle can read all the packets and gather a lot of information about it, causing information theft. For example in a smart home environment where all the Bluetooth beacons send information about the sensed parameters in the rooms to a central BLE device , any other device listening to the broadcasting packets can get to know the details of the home as they can find out the identity of the device sending the packet. With privacy and changing RPA the listening device cannot identify the device from which it has received the packet unless it is paired and can resolve the address.

2.1 Setup

2.1.1 Setting random device address

A device may use a random device address, but this address has to be configured before being used during advertising, scanning or initiating.

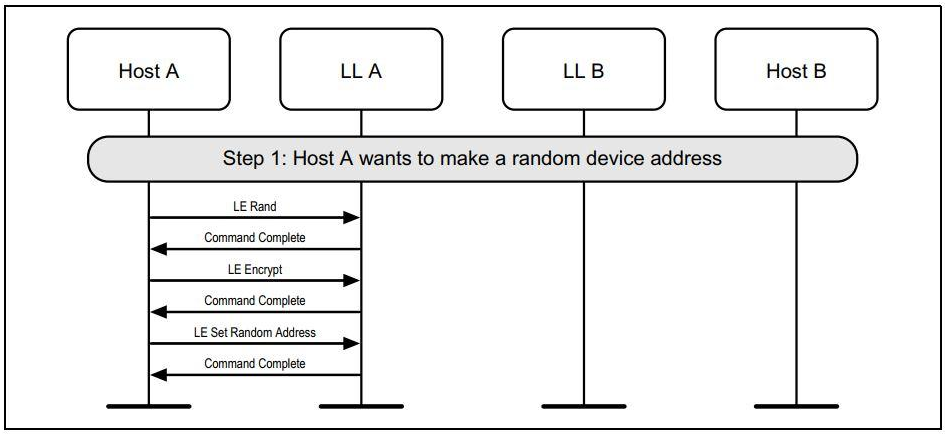


Figure 27 setting random device address

2.1.2 Adding device to white list

Before advertising, scanning or initiating can use White Lists, the White List may be cleared and devices added in as required.

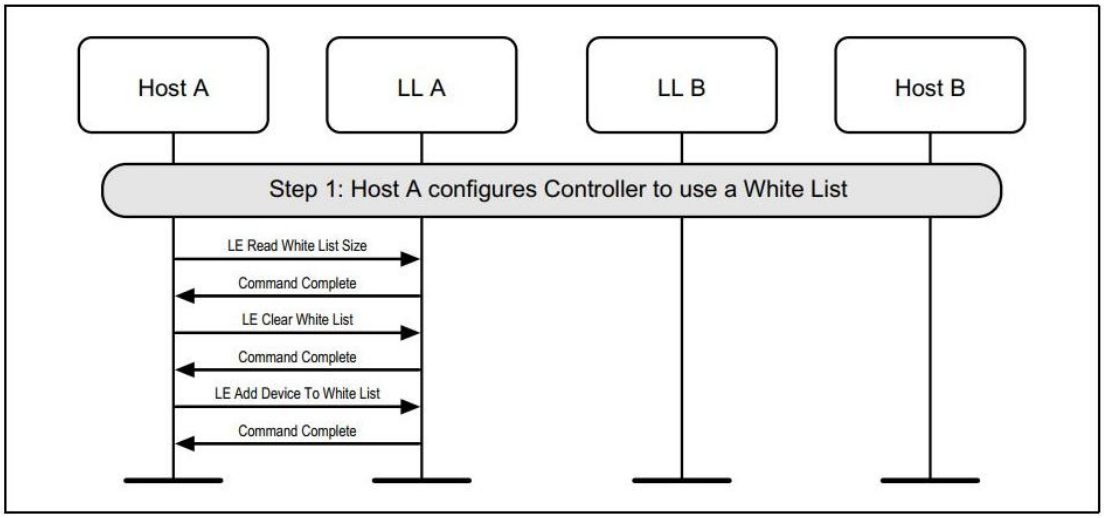


Figure 28 Adding device to white list

2.1.3 Adding IRK to resolving list

Before advertising, scanning or initiating can use resolving lists, the resolving list may be cleared and devices added in as required.

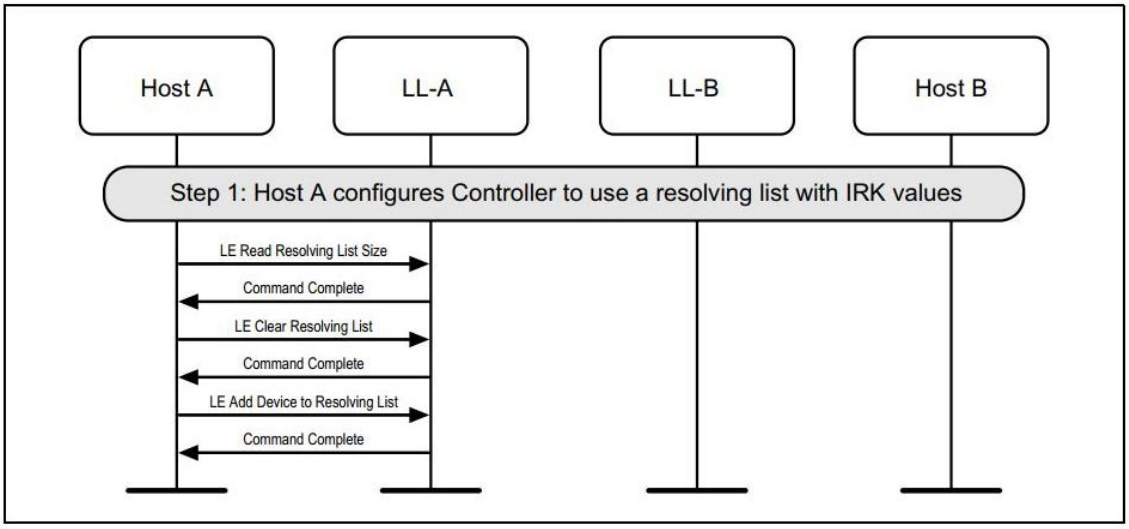


Figure 29 Adding IRK to resolving list

2.2 Commands used

2.2.1 Set Event Mask Command (S&A)

The Set_Event_Mask command is used to control which events are generated by the HCI for the Host.

2.2.2 LE set random address (S)

The LE_Set_Random_Address command is used by the Host to set the LE Random Device Address in the Controller.

2.2.3 LE Advertising set random address (A)

The LE_Set_Advertising_Set_Random_Address command is used by the Host to set the random device address specified by the Random_Address parameter.

2.2.4 LE add device to Resolving List (S&A)

The LE_Add_Device_To_Resolving_List command is used to add one device to the list of address translations used to resolve Resolvable Private Addresses in the Controller.

2.2.5 Set Resolvable private address timeout (A)

The LE_Set_Resolvable_Private_Address_Timeout command set the length of time the controller uses a Resolvable Private Address before a new resolvable private address is generated and starts being used. This timeout applies to all addresses generated by the Controller.

2.2.6 Set Address Resolution enable (S & A)

The LE_Set_Address_Resolution_Enable command is used to enable resolution of Resolvable private Addresses in the Controller. This causes the Controller to use the resolving list whenever the Controller receives a local or peer Resolvable Private Address.

2.2.7 LE set Extended scan parameters (S)

The LE_Set_Extended_Scan_Parameters command is used to set the extended scan parameters to be used on the advertising channels.

2.2.8 LE set ExtendedAdv dat a(A)

The LE_Set_Extended_Advertising_Data command is used to set the data used in advertising PDUs that have a data field.

2.2.9 LE set Extended scan enable(S Disable)

The LE_Set_Extended_Scan_Enable command is used to enable or disable scanning. The Enable parameter determines whether scanning is enabled or disabled. If it is disabled, the remaining parameters are ignored.

The manual commands opcodes are written in a text file. This is then executed on the host and using Ellisys Bluetooth sniffer air-logs are taken.

The ellisys captures all the packets in the air. Based on these captured packets it can determine if the address can be decoded. The figure below shows the window of an ellisys explorer.

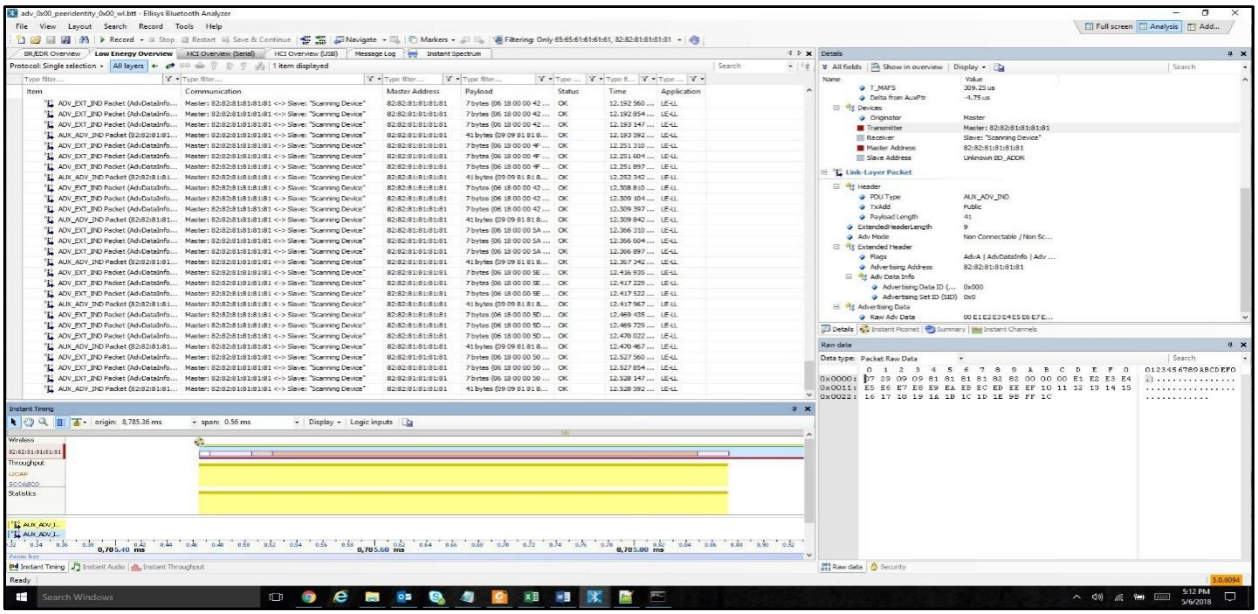


Figure 30 Ellisys Bluetooth Explorer