Deep Convolutional Network for Thorax Disease Detection

Gitika Meher
Department of Electrical and Computer Engg.
University of California San Diego
9500 Gilman Dr, La Jolla, CA 92093
gkarumur@eng.ucsd.edu

Nasha Meoli
Department of Electrical and Computer Engg.
University of California San Diego
9500 Gilman Dr, La Jolla, CA 92093
nmeoli@eng.ucsd.edu

Ambareesh SJ
Department of Electrical and Computer Engg.
University of California San Diego
9500 Gilman Dr, La Jolla, CA 92093
asreekum@ucsd.edu

Farheen Ahluwalia
Computer Science Engg.
University of California San Diego
9500 Gilman Dr, La Jolla, CA 92093
fahluwal@ucsd.edu

Abstract

We use multiple deep CNN architectures to predict thoracic disease(s) present in chest X-rays. The dataset contains 112,120 images from 30,805 unique patients, where each image may be labeled with a single disease or multiple diseases. We approach the problem statement using 4 different architectures, different optimisations like Adams optimiser, Augmentations like rotation and random crop and compare different metrics, namely accuracy, precision, recall and Balance classification rate.

1 Introduction

We use multiple deep CNN architectures to predict thoracic disease(s) present in chest X-rays. The dataset contains 112,120 images from 30,805 unique patients, where each image may be labeled with a single disease or multiple diseases. We approach the problem statement using 4 different architectures and compare different metrics, namely accuracy, precision, recall and Balance classification rate.

1.1 Problem Statement

We aim to detect diseases from a frontal X-Ray image, with no additional information regarding gender, age or patient habits. The patient can have multiple diseases and the ideal solution would detect all possible diseases from an X-ray image. The statement can be formally called a multi-class and multi-label Medical image classification problem

1.2 Background and Importance

Deep Networks have been used in Computer Vision tasks successfully, and here we explore its capacity to identify diseases from X-Ray images. Unlike other tasks (like handwriting recognition, emotion detection) that we have explored in the past, the task is much more tasking, because only experts can read X-Ray images and infer diseases. Simpler computer vision tasks can be successfully completed by an average adult, but here is a task that only qualified medical professionals can do successfully. Another factor that increases the task complexity is the high stakes, a false negative (predicting that an image is not of category X, when it was actually of that category) or mis-classification (attributing an image to class X when it actually belonged to some other class) could be fatal. The third challenging aspect of the problem is the highly skewed distribution of data. A positive labeling is extremely rare in comparison to a negative label as only a small percentage of a population would be suffering from a particular disease. Hence given a sizable data set, it is quite possible that predicting "No disease" for all cases would lead to an extremely high accuracy and minimize a loss function that does not take the skewed-nature of the data set into account but this would miss the entire point of this task.

2 Related work

2.1 Work using the same dataset

The Chest X-Ray8 [1] data-set was much bigger than similar data sets (the closest being the OpenAI data set) in medical image classification. In the paper [1], where the data set was first introduced, and an attempt at classification made, the authors have demonstrated that thoracic diseases can be "detected and even spatially-located via a unified weakly-supervised multi-label image classification and disease localization framework". The authors claim that even though the initial quantitative results were promising, deep convolutional neural network based "reading chest X-rays remains a strenuous task". We aimed to compare our results (for the 8 of the 14 diseases in our expanded data-set) to the the state-of-the-art implementation of CNN's in the above paper.

2.2 Transfer Learning with a famous architecture

The pre-trained architecture selected for transfer learning was the Vgg-16 architecture [2] developed by the Visual Geometry Group at Oxford University. The model is trained on images from the ImageNet data set. The model is deeper than the other architectures presented here, which is advantageous due to the positive correlation between the depth of networks and their success. Vgg-16 was chosen over other deep architectures (such as ResNet and Vgg-19) because it offers faster and easier training than these due to its comparatively fewer layers [3]. The Vgg-16 architecture was shown to have high precision and recall on similar chest x-ray image classification tasks which inspired its usage in this work [4].

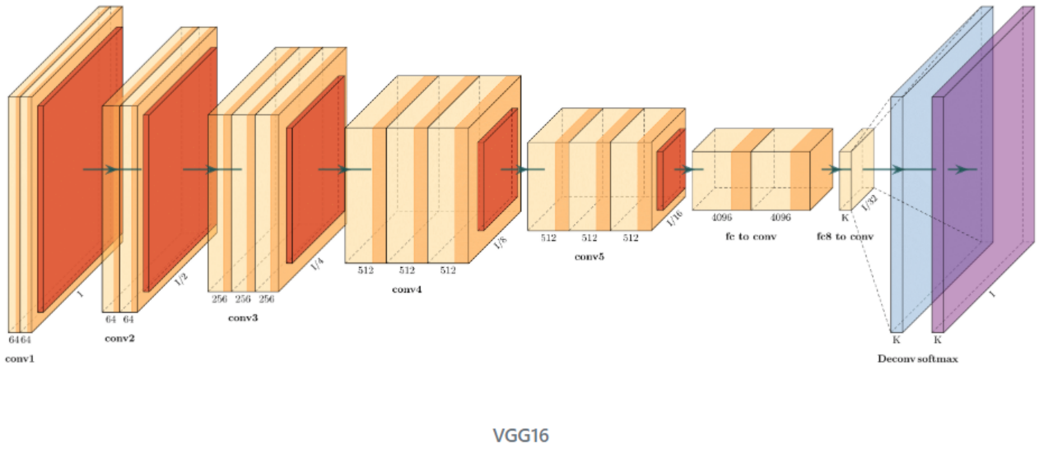


Figure 1: Figure showing VGG 16 Architecture

3 Methods

3.1 Baseline

Our baseline model is a neural network that consists of three convolution layers followed by a max pool layer, the output from the maxpool is then fed into a fully connected layer followed by fully connected output layer consisting of 14 output units. The first convolution layer has 12 channels and we use a filter of 8x8 to obtain these channels. Convolution layer 2 has 10 channels and uses 8x8 filters, convolution 3 uses 6x6 filters and has 8 channels. We then use maxpool for spatial pooling, the size of the filter used here is 3x3. Fully connected layer 1 and fully connected layer 2 have 128 units and 14 units respectively. The weights were initialized using Xavier initialization and batch normalization. Additionally, we used Adam optimizer for faster convergence.

We experiment with both the strategies- full connections as well as a dropout strategy with a dropout probability of 0.2. Further, we experimented with the baseline and added z-scoring, augmented the data with rotated images (20 degrees rotation) and randomly cropped images(see section 3.5 for details) to all the models discussed hereon.
inputs -> conv1 -> conv2 -> conv3 -> maxpool -> fc1 -> fc2 (outputs)

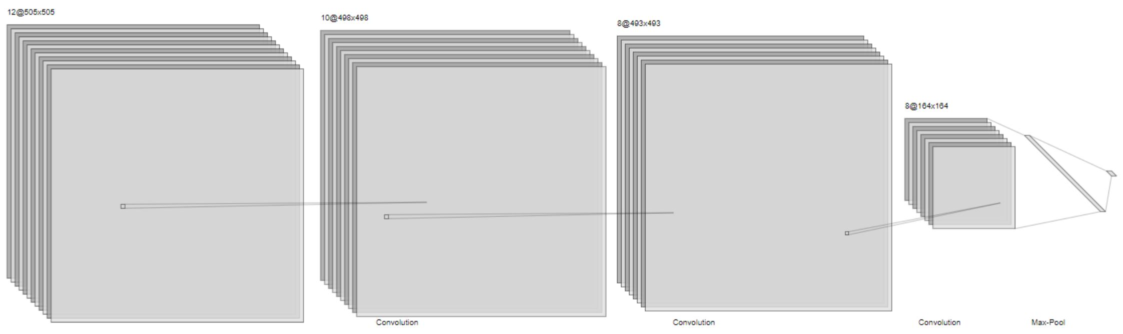


Figure 2: Figure showing the Baseline Architecture

3.2 Architecture I

We experimented over the baseline model to generate two models, the first architecture that we propose is as follows:
inputs->conv1->conv2->maxpool1->conv3->conv4->maxpool2->fc1->fc2->fc3(output)

Convolution Layer 1 has 4 channels and 8x8 filter, Convolution Layer 2 has 8 channels and 6x6 filter, Maxpool 1 has a 4x4 filter, Convolution Layer 3 has 12 channels and 8x8 filter, Convolution Layer 4 has 4 channels and 8x8 filters. Fully connected layer 1 and Fully connected layer 2 have 256 units, 128 units and 14 units respectively.

The idea is to increase the depth of the neural network to make it learn more features. Not only this we also experimented with the number of hidden units in each layer. The activation function is ReLU at each layer. We also experiment with different kernel sizes. A benefit of using a small kernel instead of a fully connected network is to benefit from weight sharing and reduction in computational costs[5]. To briefly explain this point, since we use the same kernel for different set of pixels in an image, the same weights are shared across these pixel sets as we convolve on them. And as the number of weights are less than those in a fully connected layer, we have fewer weights to back-propagate on.

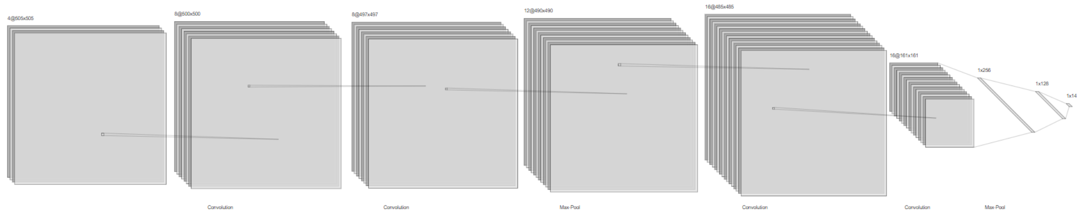


Figure 3: Figure showing Architecture I connection

3.3 Architecture II

This architecture was inspired by Deep Residual Learning for Image Recognition [6]. This architecture has three convolutional layers followed by a max-pooling layer followed by two fully connected layers. The first convolutional layer is residually connected to the last convolutional layer. To keep the dimensions compatible, the convolutional layers are symmetrically padded appropriately to improve the training speed and precision. Since complex networks are hard to train and easy to over-fit it may be very useful to explicitly add the skip connection as a linear regression term, when you know that your data has a strong linear component.
We also changed the activation functions in between the layers to P-ReLU because at a marginal computational cost, P-ReLU does not saturate the gradient as compared to ReLU [7].

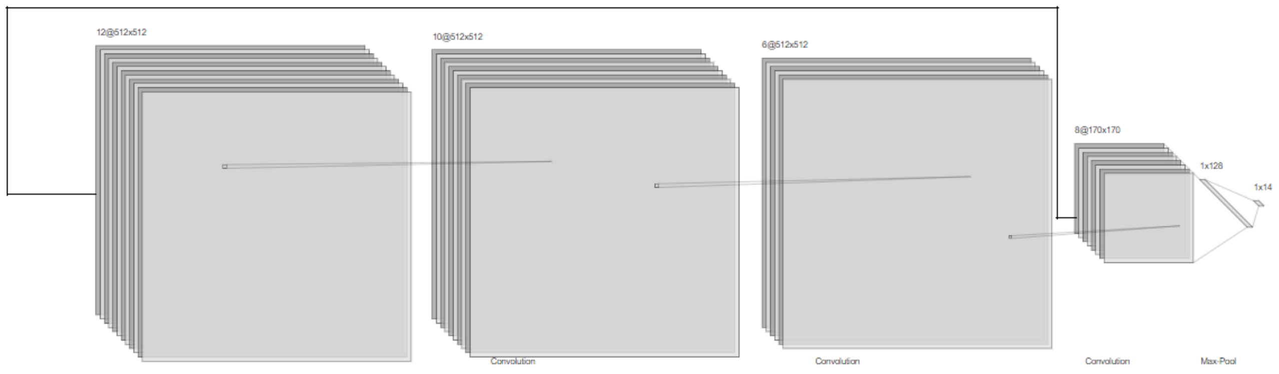


Figure 4: Figure showing Architecture II with the skip connection

3.4 Vgg-16 Transfer Learning

First, the linear classifier layer in the architecture was replaced with our own linear layer with an appropriate number of classes (14 classes). The output was then passed through a sigmoid for the multi-class classification.

Secondly, the images needed to be transformed according to how the PyTorch Vgg-16 model had been pre-trained. The pre-trained Vgg-16 model on PyTorch expected RGB images of dimensions of at least [224 x 224] and these images were required to be scaled using a mean of [0.485, 0.456, 0.406] and std of [0.229, 0.224, 0.225].

An Adam Optimizer was used and the learning rate used in training was no greater than 1e-04 as transfer learning generally requires lower learning rates. A weighted loss function was required to deal with the imbalanced nature of the data, the weights were tuned to achieve desirable precision and recall.

In the first phase, all the layers but the output layer were frozen and training was carried out only on the output layer weights.

In the second part of the transfer learning problem, all the weights from the pre-trained model were loaded and no layers in the architecture were frozen, this allowed for further fine-tuning of the weights in these layers for the given classification problem.

3.5 Adam's Optimiser, Activation Functions and Xavier Initialisation

The Adam optimiser is an optimization technique which combines two aspects that improve convergence speeds - Momentum and RMS Prop. The algorithm is an extension to stochastic gradient descent that has been adopted for deep learning applications in computer vision and natural language processing widely. Instead of changing the learning rate based on the average mean (RMSProp), Adam also uses the average of the second moments of the gradients (variance). The algorithm calculates an exponential moving average of the gradient and the squared gradient, and there are two parameter that control the decay rates of thes moving averages.

ReLU activation function has been seen to work empirically well in Computer Vision tasks. They are easy to implement, and the gradient calcluation is trivial, and perform well in Deep Networks. However, ReLU activation performs well when the weights are initialized by Xavier Initialization.

3.6 Z-score

Z-scoring inputs makes sure that the input has 0 mean and a standard deviation of 1. By making sure that the inputs are of the same scale, we ensure that weights for all features converge at a similar pace (rather than one converging too slowly as compared to the others). Z scoring thus ensures that the objective function converges faster, and the weight update is smoother.

3.7 Data Augmentation

We applied two different transformations to the data on the fly with an application probability of 50% for all training purposes. By adding these transformations, the bias for the model improves, decreasing the variance which in-turn helps the model generalize better.

3.7.1 Random rotate

Random Rotation transformation probabilistically rotates the image along the vertical axis at 20 degrees angle.

3.7.2 Centre crop

Centre Crop transformation probabilistically crops the image from the centre. All images are resized to the size of 512 x 512 and are converted to tensors before training, validation and testing.

3.8 Drop out

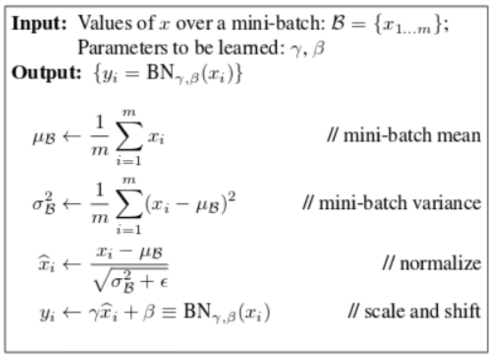
Drop out probabilistically turns off some nodes from firing making the network simpler and smaller than it is. Since, We cannot rely only one feature, during back propagation, there is more spread of weights due to drop-out. This can be viewed as L2 regularization weights where each weight is penalized differently. It also reduces the chances of overfitting on the training data. We apply a drop-out of 20% for the fully connected layers indicating to randomly ignore 20% of the nodes.

3.9 Ensembling

Ensemble methods use multiple learning algorithms (different architectures) to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone. In our implementation, We chose the baseline with z-scoring, data augmentation and drop out along with the two new architecture models for ensembling. We use the validation BCR results of the respective architectures as the corresponding ensembling weights for the model.

3.10 Batch Normalisation

Applying the same logic as was used when normalizing the input images, the natural extension was to z-score or mean-normalize activations from one layer to another. Batch normalization normalizes the output of an activation by subtracting the batch mean and dividing by the batch standard deviation. While providing Batch Normalization improves convergence by giving the network the ability to scale and resize the activations, there can be circumstances where possibly the set mean and standard deviation would not be the best for the network to learn at. Thus we provide two extra parameters which provides the network to de-normalize and learn, if it deems fit to do so. Batch normalization thus changes the amount by which the hidden unit activations shift around (covariance shift). Additionally Batch Normalization also introduces slight regularization effects as it adds some noise to each hidden layer's activations.



Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

```
$\mu_\mathcal{B} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$    // mini-batch mean
$\sigma_\mathcal{B}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_\mathcal{B})^2$    // mini-batch variance
$\widehat{x}_i \leftarrow \frac{x_i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B}^2 + \epsilon}}$    // normalize
$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i)$    // scale and shift
```

Figure 5: Batch Normalization Formulae from the original paper [8]

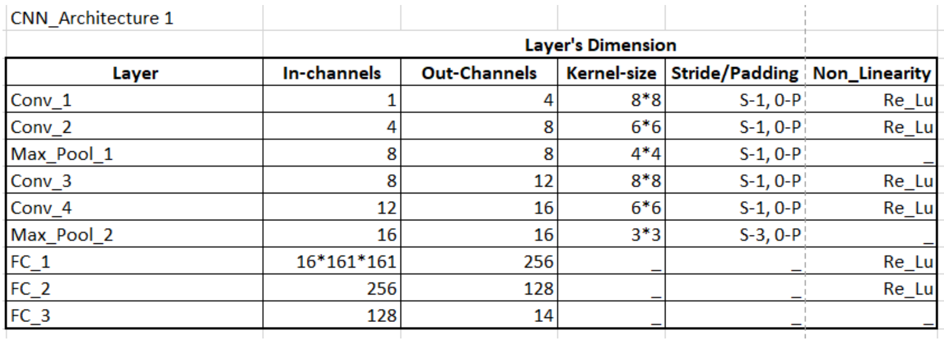
3.11 Loss Function

The loss function used is weighted Binary Cross Entropy Loss which is considered as a good choice for multi label classification problem. Since there are 14 different categories of diseases , and every X-RAY image can have multiple labels we choose BCE as the loss function. BCE considers a sigmoid loss at every output node and sums over all output nodes. However, the images with diseases are rare and thus there is a natural imbalance in the data set. To address this imbalance we use weighted sum of the sigmoid loss at every node. Because there are fewer positive labels in the data, we penalize the false negatives heavily as compared to the False Positives. So if for lung cancer there are 100 positive label images available and 3000 negative label images the factor for penalty for False negatives will be 30. This makes sure that the model does not classify all the images as negative to increase the accuracy and behave like a trivial model. Further we experimented with these factors to increase the recall and precision rather than accuracy of the models and empirically found that the following weights perform well to give good recall and precision. The Loss function, was inspired by [1], the original Chest X-Ray paper:

$$L_{W-CEL}(f(\vec{x}), \vec{y}) = \beta_P \sum_{y_c=1} -\ln(f(x_c)) + \beta_N \sum_{y_c=0} -\ln(1 - f(x_c)).$$

3.12 Experimentation

3.12.1 Architecture I

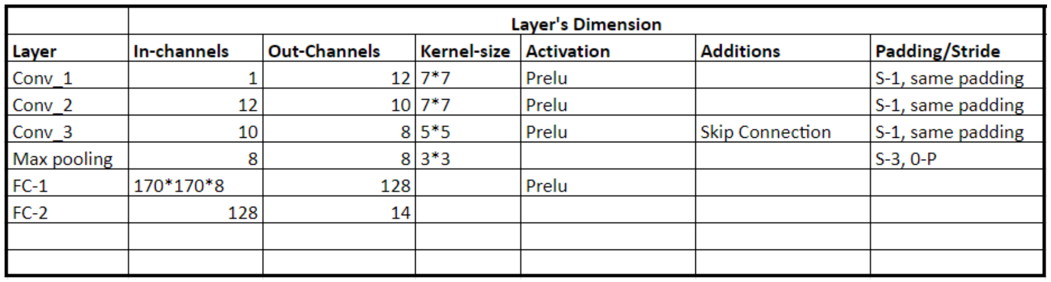


CNN_Architecture 1

	Layer's Dimension				
Layer	In-channels	Out-Channels	Kernel-size	Stride/Padding	Non_Linearity
Conv_1	1	4	8*8	S-1, 0-P	Re_Lu
Conv_2	4	8	6*6	S-1, 0-P	Re_Lu
Max_Pool_1	8	8	4*4	S-1, 0-P	_
Conv_3	8	12	8*8	S-1, 0-P	Re_Lu
Conv_4	12	16	6*6	S-1, 0-P	Re_Lu
Max_Pool_2	16	16	3*3	S-3, 0-P	_
FC_1	16*161*161	256	_	_	Re_Lu
FC_2	256	128	_	_	Re_Lu
FC_3	128	14	_	_	_

Table 1: Table detailing Architecture I layers and activation

3.12.2 Architecture II



	Layer's Dimension					
Layer	In-channels	Out-Channels	Kernel-size	Activation	Additions	Padding/Stride
Conv_1	1	12	7*7	Prelu		S-1, same padding
Conv_2	12	10	7*7	Prelu		S-1, same padding
Conv_3	10	8	5*5	Prelu	Skip Connection	S-1, same padding
Max pooling	8	8	3*3			S-3, 0-P
FC-1	170*170*8	128		Prelu		
FC-2	128	14				

Table 2: Table detailing Architecture II layers and activation

4 Results

4.1 Performance Metrics

Following are the performance metrics we report for all the models we used. We discuss why in our case precision, recall, and BCR to be stronger performance metrics than Accuracy alone.
True Positives are the number of diseases present that your model says are present, False Positives are the number of diseases you say are present that aren't, and False Negatives are misses. Precision (also called positive predictive value) is the fraction of relevant instances among the retrieved instances, while recall (also known as sensitivity) is the fraction of relevant instances that have been retrieved over the total amount of relevant instances. BCR is used as a measure of discriminating power of features in the classification of multi-class pattern recognition problems.

4.1.1 Accuracy

Accuracy is a good measure when the target variable classes in the data are nearly balanced. Accuracy should not be used as a measure when the target variable classes in the data are a majority of one class. Therefore we go with other performance metrics as discussed below in our case.
$Accuracy = \frac{totalcorrectpredictions}{totalnumberofsamples}$

4.1.2 Precision

Precision is a measure that tells us what proportion of patients that we diagnosed as having disease, actually had the disease. The predicted positives (People predicted with disease are TP and FP) and the people actually having the disease are TP.
$Precision = \frac{|TP|}{|TP|+|FP|}$

4.1.3 Recall

Recall is a measure that tells us what proportion of patients that actually had the disease were diagnosed by the algorithm with the disease. The actual positives (People having the disease are TP and FN) and the people diagnosed by the model having the disease are TP. (Note: FN is included because the Person actually had the disease even though the model predicted otherwise)
$Recall = \frac{|TP|}{|TP|+|FN|}$

4.1.4 Recall and Precision Trade-off

Recall gives us information about a classifier's performance with respect to false negatives (how many did we miss), while precision gives us information about its performance with respect to false positives(how many did we caught). Precision is about being precise. So even if we managed to capture only one cancer case, and we captured it correctly, then we are 100% precise. Recall is not so much about capturing cases correctly but more about capturing all cases that have "cancer" with the answer as "cancer". So if we simply always say every case as "cancer", we have 100% recall. So basically in our case we want to focus more on minimising False Negatives, and therefore we want our Recall to be as close to 100% as possible without precision being too bad.

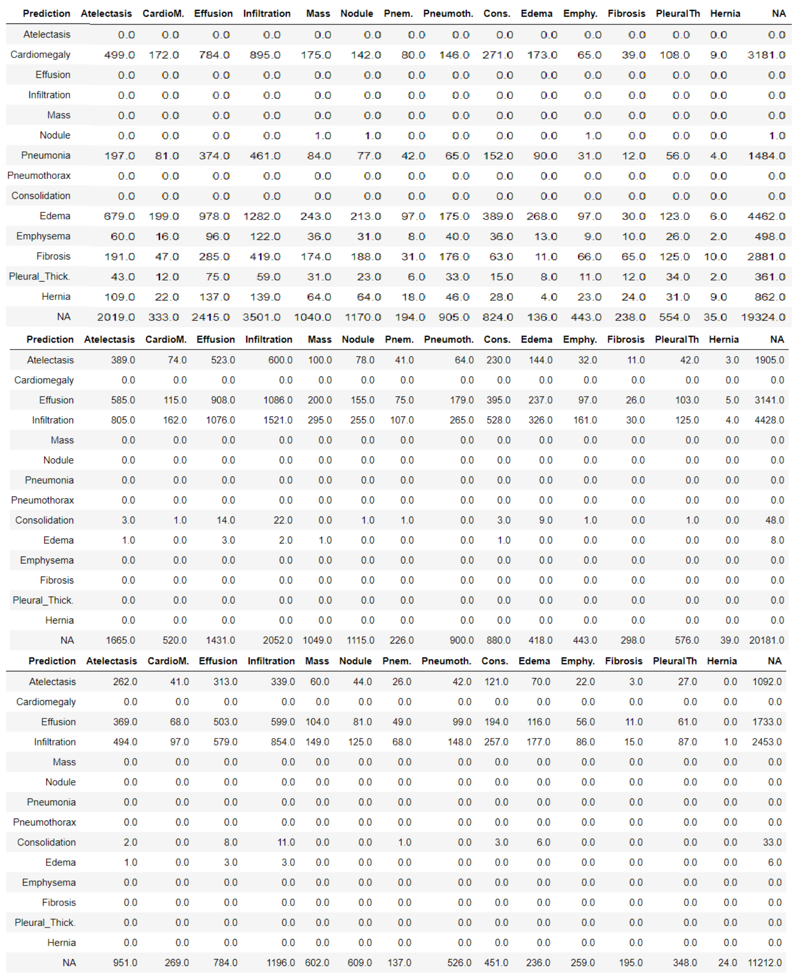
4.1.5 Balanced Error Rate(BCR)

BCR gives us a measure of measuring the average of precison and recall. In our model we want BCR to be high. From the discussion in the previous section it can be concluded that BCR is a good performance metric for this problem. And hence we expect the BCR for each disease to be as high as possible. $BCR = \frac{Precision+Recall}{2}$

4.1.6 Confusion Matrix

In classification problems, a confusion matrix (or an error matrix) is a table layout that allows visualization of the performance of an algorithm, typically a supervised learning one. In our problem, each row of the matrix are the predictions made for each disease and the columns represent targets. Ideal result would mean a diagonal matrix with all other instances being zero.

Confusion Matrices for Baseline, Architecture_1 and Architecture_2 in order:



Prediction	Atelectasis	CardioM.	Effusion	Infiltration	Mass	Nodule	Pnem.	Pneumoth.	Cons.	Edema	Emphy.	Fibrosis	PleuralTh	Hernia	NA
Atelectasis	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Cardiomegaly	499.0	172.0	784.0	895.0	175.0	142.0	80.0	146.0	271.0	173.0	65.0	39.0	108.0	9.0	3181.0
Effusion	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Infiltration	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Mass	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Nodule	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0
Pneumonia	197.0	81.0	374.0	461.0	84.0	77.0	42.0	65.0	152.0	90.0	31.0	12.0	56.0	4.0	1484.0
Pneumothorax	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Consolidation	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Edema	679.0	199.0	978.0	1282.0	243.0	213.0	97.0	175.0	389.0	268.0	97.0	30.0	123.0	6.0	4462.0
Emphysema	60.0	16.0	96.0	122.0	36.0	31.0	8.0	40.0	36.0	13.0	9.0	10.0	26.0	2.0	498.0
Fibrosis	191.0	47.0	285.0	419.0	174.0	188.0	31.0	176.0	63.0	11.0	66.0	65.0	125.0	10.0	2881.0
Pleural_Thick.	43.0	12.0	75.0	59.0	31.0	23.0	6.0	33.0	15.0	8.0	11.0	12.0	34.0	2.0	361.0
Hernia	109.0	22.0	137.0	139.0	64.0	64.0	18.0	46.0	28.0	4.0	23.0	24.0	31.0	9.0	862.0
NA	2019.0	333.0	2415.0	3501.0	1040.0	1170.0	194.0	905.0	824.0	136.0	443.0	238.0	554.0	35.0	19324.0

Prediction	Atelectasis	CardioM.	Effusion	Infiltration	Mass	Nodule	Pnem.	Pneumoth.	Cons.	Edema	Emphy.	Fibrosis	PleuralTh	Hernia	NA
Atelectasis	389.0	74.0	523.0	600.0	100.0	78.0	41.0	64.0	230.0	144.0	32.0	11.0	42.0	3.0	1905.0
Cardiomegaly	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Effusion	585.0	115.0	908.0	1086.0	200.0	155.0	75.0	179.0	395.0	237.0	97.0	26.0	103.0	5.0	3141.0
Infiltration	805.0	162.0	1076.0	1521.0	295.0	255.0	107.0	265.0	528.0	326.0	161.0	30.0	125.0	4.0	4428.0
Mass	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Nodule	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Pneumonia	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Pneumothorax	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Consolidation	3.0	1.0	14.0	22.0	0.0	1.0	1.0	0.0	3.0	9.0	1.0	0.0	1.0	0.0	48.0
Edema	1.0	0.0	3.0	2.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	8.0
Emphysema	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Fibrosis	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Pleural_Thick.	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Hernia	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
NA	1665.0	520.0	1431.0	2052.0	1049.0	1115.0	226.0	900.0	880.0	418.0	443.0	298.0	576.0	39.0	20181.0

Prediction	Atelectasis	CardioM.	Effusion	Infiltration	Mass	Nodule	Pnem.	Pneumoth.	Cons.	Edema	Emphy.	Fibrosis	PleuralTh	Hernia	NA
Atelectasis	262.0	41.0	313.0	339.0	60.0	44.0	26.0	42.0	121.0	70.0	22.0	3.0	27.0	0.0	1092.0
Cardiomegaly	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Effusion	369.0	68.0	503.0	599.0	104.0	81.0	49.0	99.0	194.0	116.0	56.0	11.0	61.0	0.0	1733.0
Infiltration	494.0	97.0	579.0	854.0	149.0	125.0	68.0	148.0	257.0	177.0	86.0	15.0	87.0	1.0	2453.0
Mass	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Nodule	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Pneumonia	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Pneumothorax	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Consolidation	2.0	0.0	8.0	11.0	0.0	0.0	1.0	0.0	3.0	6.0	0.0	0.0	0.0	0.0	33.0
Edema	1.0	0.0	3.0	3.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	6.0
Emphysema	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Fibrosis	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Pleural_Thick.	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Hernia	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
NA	951.0	269.0	784.0	1196.0	602.0	609.0	137.0	526.0	451.0	236.0	259.0	195.0	348.0	24.0	11212.0

Figure 6: Confusion Matrices for three different architectures

4.2 Baseline

The validation and training losses were plotted against 5 epochs for the baseline model. The plots and the results obtained for the test set have been summarised below.

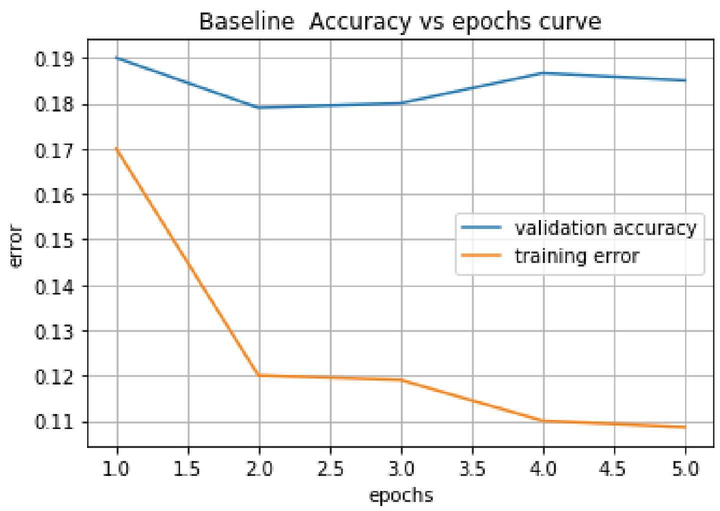
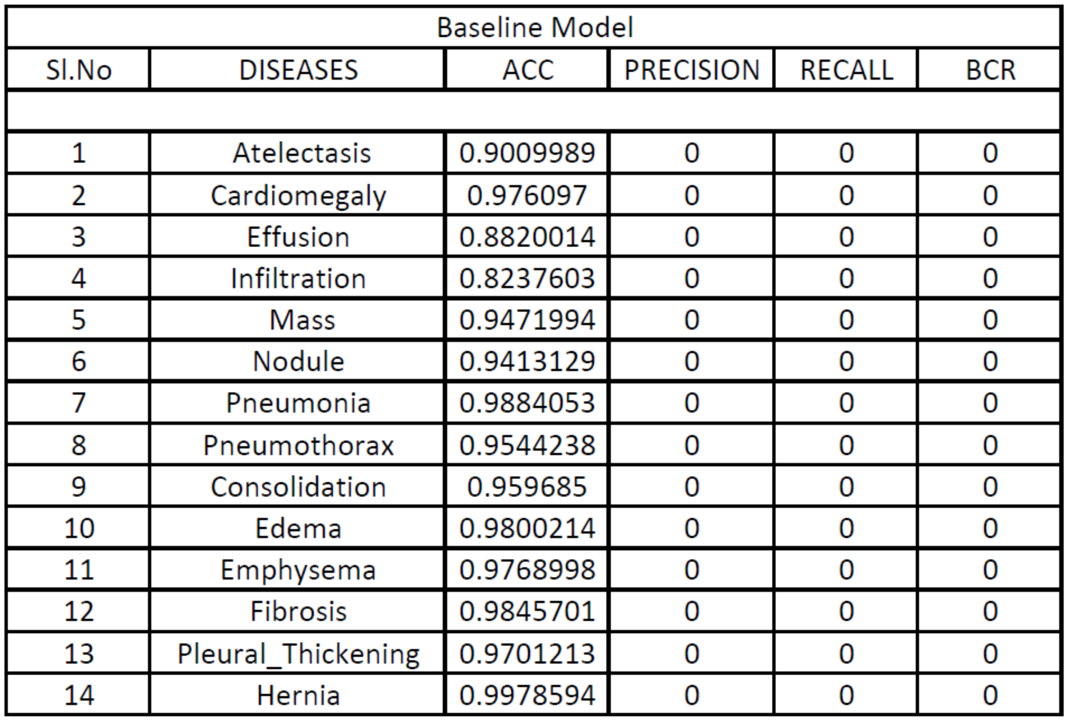


Figure 7: Figure showing the Baseline Architecture Loss

4.2.1 Performance



Baseline Model					
Sl.No	DISEASES	ACC	PRECISION	RECALL	BCR
1	Atelectasis	0.9009989	0	0	0
2	Cardiomegaly	0.976097	0	0	0
3	Effusion	0.8820014	0	0	0
4	Infiltration	0.8237603	0	0	0
5	Mass	0.9471994	0	0	0
6	Nodule	0.9413129	0	0	0
7	Pneumonia	0.9884053	0	0	0
8	Pneumothorax	0.9544238	0	0	0
9	Consolidation	0.959685	0	0	0
10	Edema	0.9800214	0	0	0
11	Emphysema	0.9768998	0	0	0
12	Fibrosis	0.9845701	0	0	0
13	Pleural_Thickening	0.9701213	0	0	0
14	Hernia	0.9978594	0	0	0

Table 3: Table showing Baseline Architecture performance

4.3 Baseline with Z scoring and Data Augmentation

The validation and training losses were plotted against 5 epochs for the baseline model with augmented data set and also a modified loss function. The plots and the results obtained for the test set have been summarised below.

4.3.1 Curves

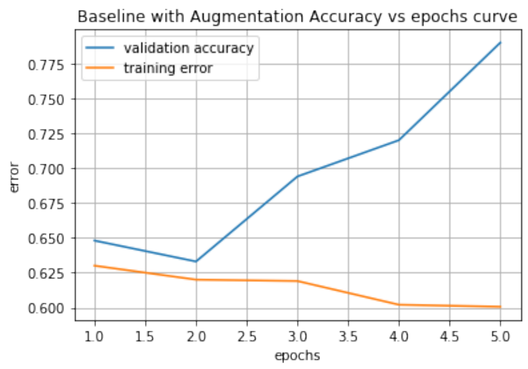
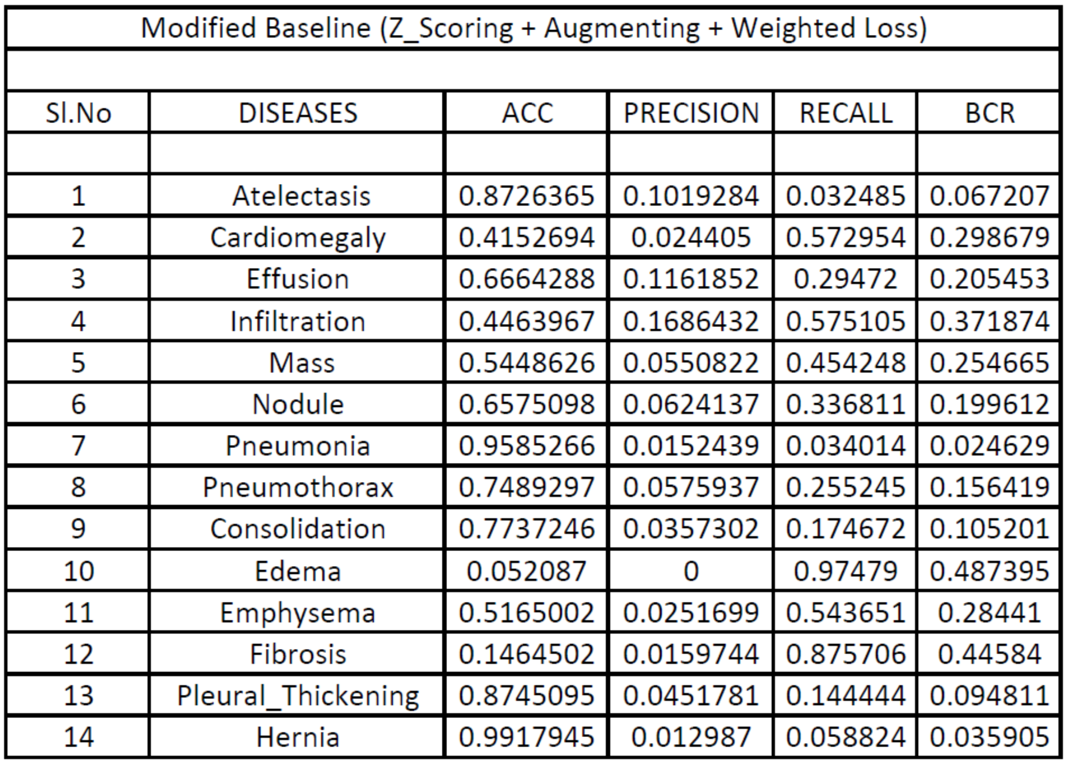


Figure 9: Fig showing the Augmented Baseline Architecture Loss

4.3.2 Performance



Modified Baseline (Z_Scoring + Augmenting + Weighted Loss)					
Sl.No	DISEASES	ACC	PRECISION	RECALL	BCR
1	Atelectasis	0.8726365	0.1019284	0.032485	0.067207
2	Cardiomegaly	0.4152694	0.024405	0.572954	0.298679
3	Effusion	0.6664288	0.1161852	0.29472	0.205453
4	Infiltration	0.4463967	0.1686432	0.575105	0.371874
5	Mass	0.5448626	0.0550822	0.454248	0.254665
6	Nodule	0.6575098	0.0624137	0.336811	0.199612
7	Pneumonia	0.9585266	0.0152439	0.034014	0.024629
8	Pneumothorax	0.7489297	0.0575937	0.255245	0.156419
9	Consolidation	0.7737246	0.0357302	0.174672	0.105201
10	Edema	0.052087	0	0.97479	0.487395
11	Emphysema	0.5165002	0.0251699	0.543651	0.28441
12	Fibrosis	0.1464502	0.0159744	0.875706	0.44584
13	Pleural_Thickening	0.8745095	0.0451781	0.144444	0.094811
14	Hernia	0.9917945	0.012987	0.058824	0.035905

Table 4: Table showing Architecture I performance

4.4 Architecture I

4.4.1 Curves

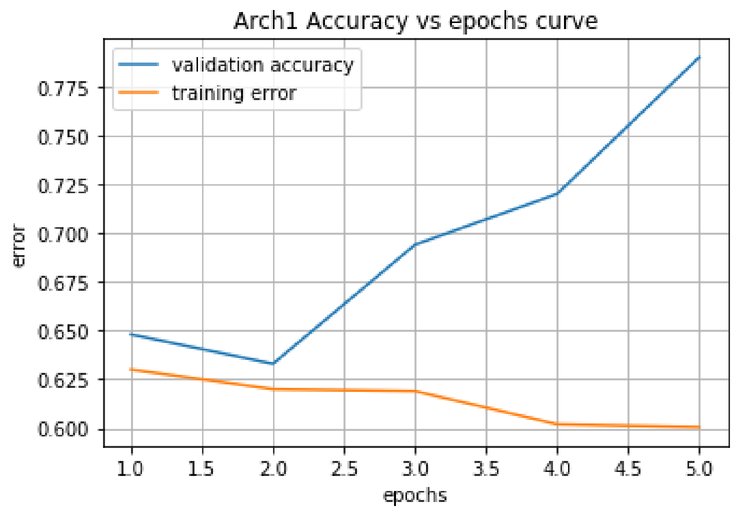
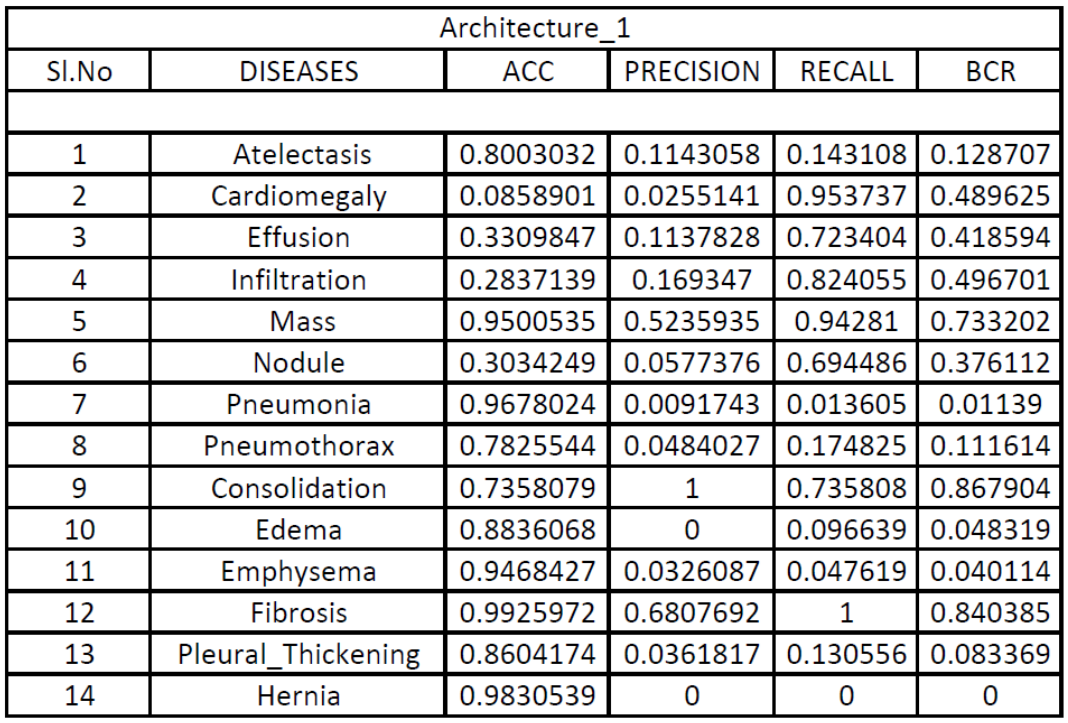


Figure 10 : Training and Validation Loss with Baseline

4.4.2 Performance



Architecture_1					
Sl.No	DISEASES	ACC	PRECISION	RECALL	BCR
1	Atelectasis	0.8003032	0.1143058	0.143108	0.128707
2	Cardiomegaly	0.0858901	0.0255141	0.953737	0.489625
3	Effusion	0.3309847	0.1137828	0.723404	0.418594
4	Infiltration	0.2837139	0.169347	0.824055	0.496701
5	Mass	0.9500535	0.5235935	0.94281	0.733202
6	Nodule	0.3034249	0.0577376	0.694486	0.376112
7	Pneumonia	0.9678024	0.0091743	0.013605	0.01139
8	Pneumothorax	0.7825544	0.0484027	0.174825	0.111614
9	Consolidation	0.7358079	1	0.735808	0.867904
10	Edema	0.8836068	0	0.096639	0.048319
11	Emphysema	0.9468427	0.0326087	0.047619	0.040114
12	Fibrosis	0.9925972	0.6807692	1	0.840385
13	Pleural_Thickening	0.8604174	0.0361817	0.130556	0.083369
14	Hernia	0.9830539	0	0	0

Table 5: Table showing Architecture I performance

4.5 Architecture II

4.5.1 Curves

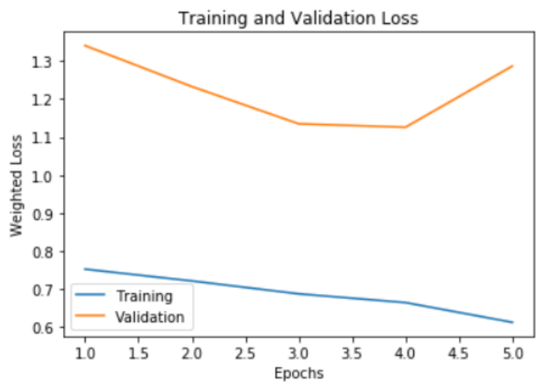
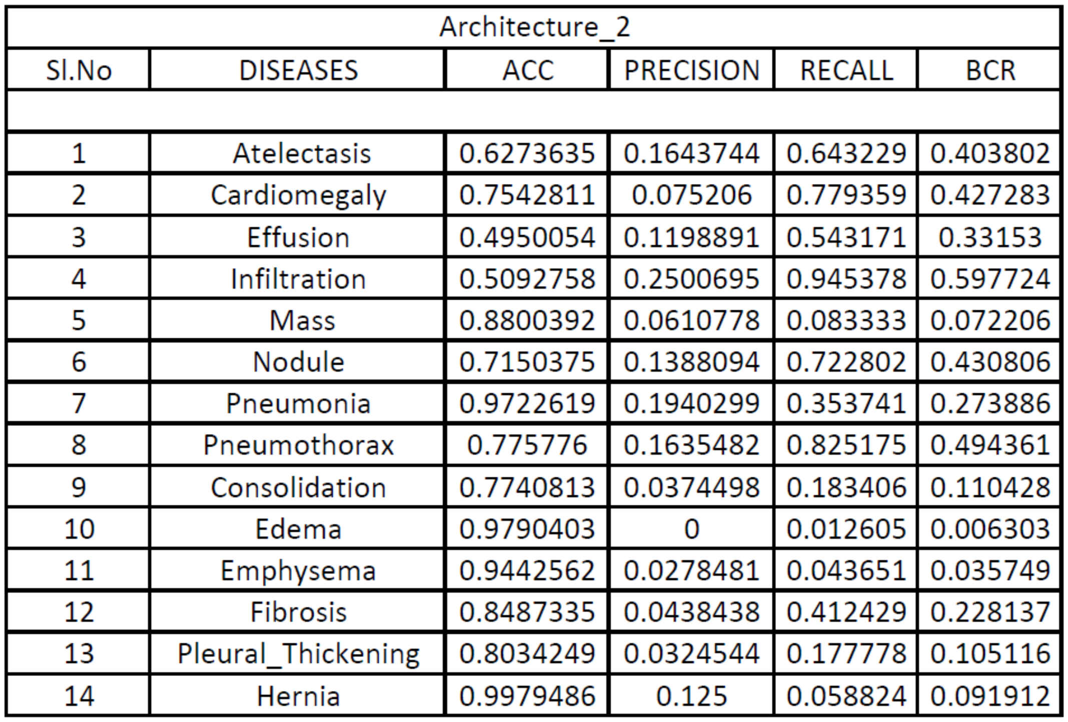


Figure 11 : Figure showing Training and Validation Loss with Architecture II

4.5.2 Performance



Architecture_2					
Sl.No	DISEASES	ACC	PRECISION	RECALL	BCR
1	Atelectasis	0.6273635	0.1643744	0.643229	0.403802
2	Cardiomegaly	0.7542811	0.075206	0.779359	0.427283
3	Effusion	0.4950054	0.1198891	0.543171	0.33153
4	Infiltration	0.5092758	0.2500695	0.945378	0.597724
5	Mass	0.8800392	0.0610778	0.083333	0.072206
6	Nodule	0.7150375	0.1388094	0.722802	0.430806
7	Pneumonia	0.9722619	0.1940299	0.353741	0.273886
8	Pneumothorax	0.775776	0.1635482	0.825175	0.494361
9	Consolidation	0.7740813	0.0374498	0.183406	0.110428
10	Edema	0.9790403	0	0.012605	0.006303
11	Emphysema	0.9442562	0.0278481	0.043651	0.035749
12	Fibrosis	0.8487335	0.0438438	0.412429	0.228137
13	Pleural_Thickening	0.8034249	0.0324544	0.177778	0.105116
14	Hernia	0.9979486	0.125	0.058824	0.091912

Table 6: Table showing the performance of Architecture II

4.5.3 Visualizations

Early layer:

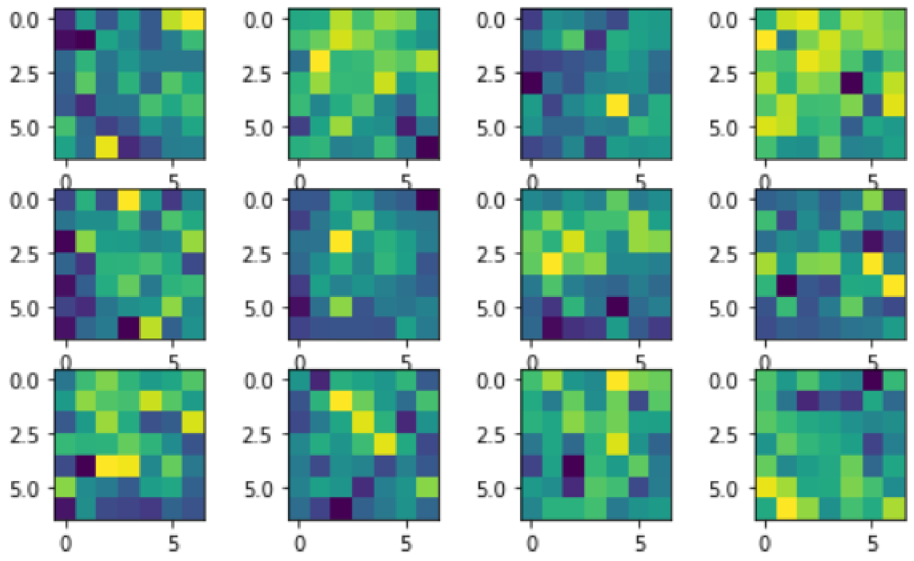


Figure 12: Figure showing the performance of Architecture II
Middle Layer:

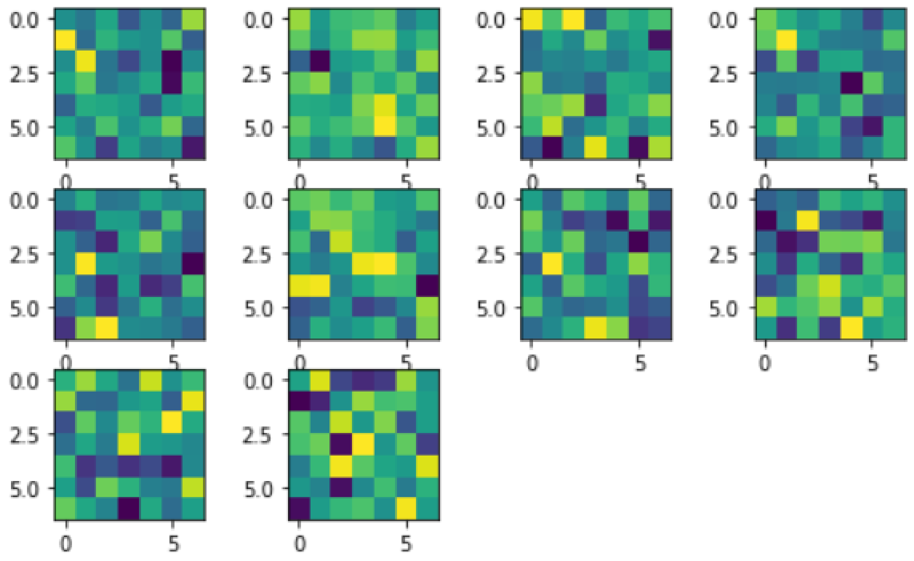


Figure 13: Figure showing the performance of Architecture II
Later Layer:

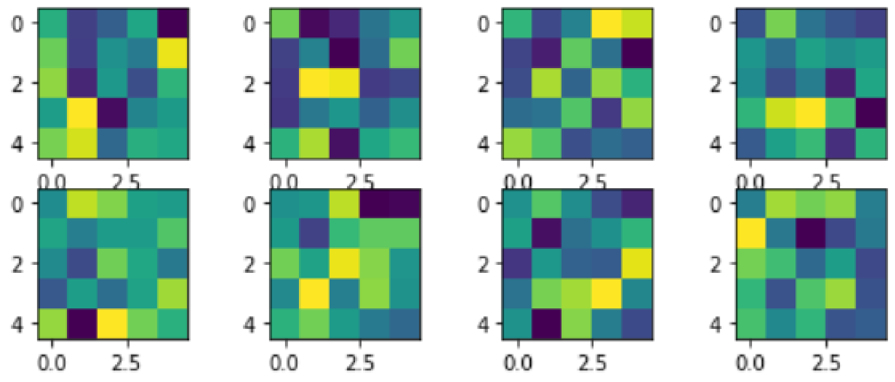


Figure 14: Figure showing the visualization of Architecture II

4.6 Transfer Learning

This section shows the results when none of the layers of the pre-trained network were frozen, these results were superior to those obtained when only the output layer was unfrozen.

4.6.1 Curves and Evaluation of Performance

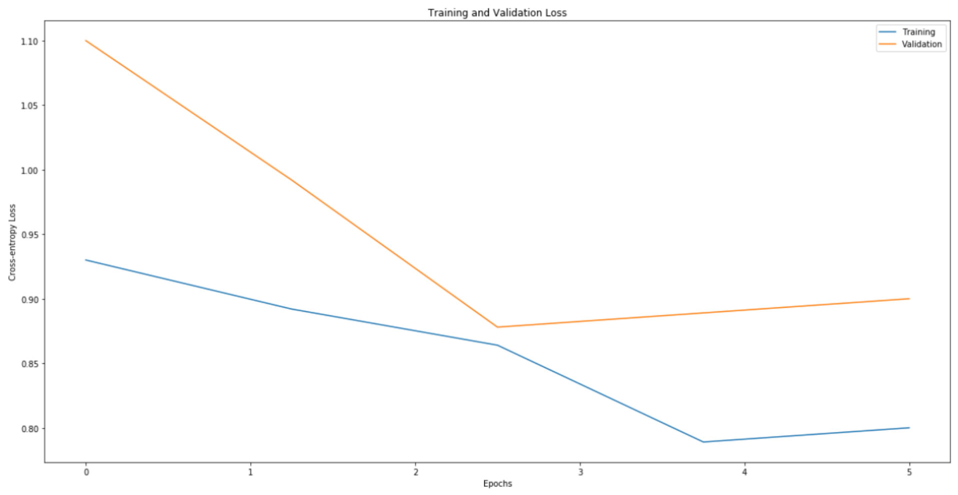
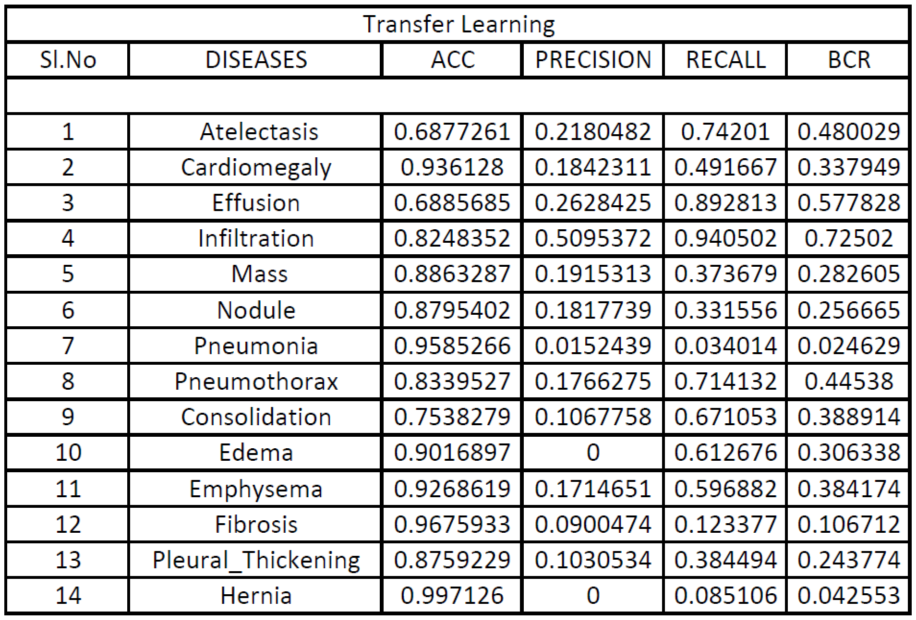


Figure 15: Figure showing Training and Validation Loss with Transfer Learning using vgg-16

Similar performance metrics to those described above were used in this section and the results are shown.

4.6.2 Performance

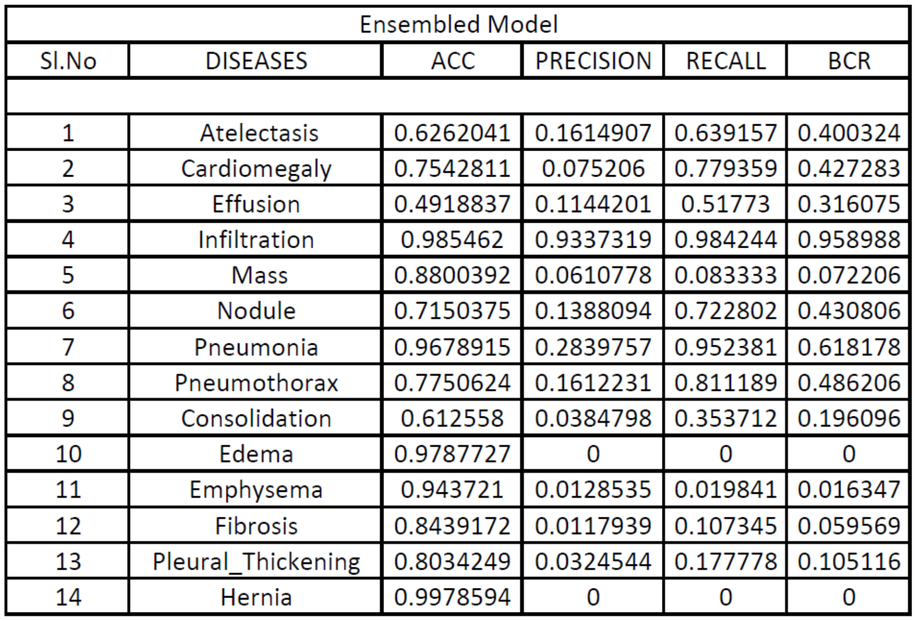
The performance of the Transfer Learning shows higher recall than precision, with the highest BCR achieved for Infiltration.



Transfer Learning					
Sl.No	DISEASES	ACC	PRECISION	RECALL	BCR
1	Atelectasis	0.6877261	0.2180482	0.74201	0.480029
2	Cardiomegaly	0.936128	0.1842311	0.491667	0.337949
3	Effusion	0.6885685	0.2628425	0.892813	0.577828
4	Infiltration	0.8248352	0.5095372	0.940502	0.72502
5	Mass	0.8863287	0.1915313	0.373679	0.282605
6	Nodule	0.8795402	0.1817739	0.331556	0.256665
7	Pneumonia	0.9585266	0.0152439	0.034014	0.024629
8	Pneumothorax	0.8339527	0.1766275	0.714132	0.44538
9	Consolidation	0.7538279	0.1067758	0.671053	0.388914
10	Edema	0.9016897	0	0.612676	0.306338
11	Emphysema	0.9268619	0.1714651	0.596882	0.384174
12	Fibrosis	0.9675933	0.0900474	0.123377	0.106712
13	Pleural_Thickening	0.8759229	0.1030534	0.384494	0.243774
14	Hernia	0.997126	0	0.085106	0.042553

Table 7 : Table showing Performance with Transfer Learning using Vgg-16

4.7 Ensembling



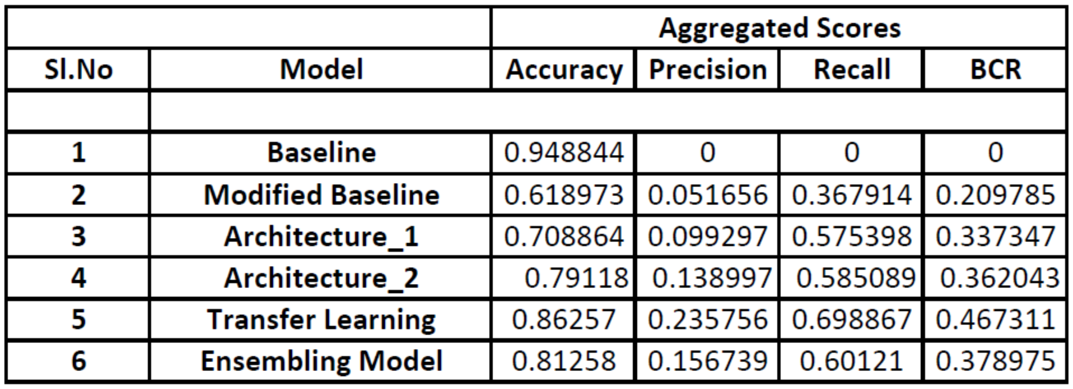
Ensembled Model					
Sl.No	DISEASES	ACC	PRECISION	RECALL	BCR
1	Atelectasis	0.6262041	0.1614907	0.639157	0.400324
2	Cardiomegaly	0.7542811	0.075206	0.779359	0.427283
3	Effusion	0.4918837	0.1144201	0.51773	0.316075
4	Infiltration	0.985462	0.9337319	0.984244	0.958988
5	Mass	0.8800392	0.0610778	0.083333	0.072206
6	Nodule	0.7150375	0.1388094	0.722802	0.430806
7	Pneumonia	0.9678915	0.2839757	0.952381	0.618178
8	Pneumothorax	0.7750624	0.1612231	0.811189	0.486206
9	Consolidation	0.612558	0.0384798	0.353712	0.196096
10	Edema	0.9787727	0	0	0
11	Emphysema	0.943721	0.0128535	0.019841	0.016347
12	Fibrosis	0.8439172	0.0117939	0.107345	0.059569
13	Pleural_Thickening	0.8034249	0.0324544	0.177778	0.105116
14	Hernia	0.9978594	0	0	0

Table 8: Table showing Performance with Ensemble Model (Voting Mechanism)

5 Discussion

5.1 Performance Analysis

The performance of different models implemented are summarized in the following table.



		Aggregated Scores			
Sl.No	Model	Accuracy	Precision	Recall	BCR
1	Baseline	0.948844	0	0	0
2	Modified Baseline	0.618973	0.051656	0.367914	0.209785
3	Architecture_1	0.708864	0.099297	0.575398	0.337347
4	Architecture_2	0.79118	0.138997	0.585089	0.362043
5	Transfer Learning	0.86257	0.235756	0.698867	0.467311
6	Ensembling Model	0.81258	0.156739	0.60121	0.378975

Table 9: Table showing Comparative Performance

The baseline model showed very high accuracy, but performed poorly in the metrics of Precision, Recall and subsequently BCR. The behavior was due to the heavy imbalance in positive and negative labels. For every disease, the number of samples with no disease hugely outweighed those with disease. Using a simple Binary cross Entropy Loss function, forces the network to behave in way to achieve as high accuracy as possible, but at the expense of predicting all Negatives. Accuracy is confirmed, to be a bad metric for this task.

The baseline was modified to optimize the weighted Binary Cross Entropy Loss function. The weighting was done in such a way that the False negatives were penalized more, and also that the rare diseases were given importance (in spite of this low occurrences in the data set). The modification largely reduced the accuracy, however at improved Recall. Rather than naively predicting that no disease was present in all images, the network started predicting (albeit overcompensating), diseases on the images. The precision was still low, and hypothetically it is because the baseline did not have enough depth or dexterity to perform the task. Also, the model was limited to running for 5 epochs (due to computational limitations and the size of the data set) More convolutions layers, and probably a better fully connected layer (or multiple layers) could help in improving the performance of the model. Following this line of thought we went on to design an architecture with more convolutions and fully connected layers.

Architecture_1 improved the performance on the task at hand, as evident in the improved Accuracy, Precision, Recall (and BCR) scores. By increasing the depth of the model, we successfully enabled it to learn more intricate features, and this increased the instances of True Positives. Addition of the fully connected layer, (though it helped in improving the performance) slowed the training process.

Architecture_2 was inspired partly by ResNet architecture. Skip connections surely helped the model to learn faster as the number of epochs required to train the model reduced. It also acts as a regularization constant. There was some information that was captured in the initial layers and was required for reconstruction during the up-sampling done using the FC layer. If we would not have used the skip architecture that information would have been lost (or should say would have turned too abstract for it to be used further). So the information that we had in the primary layers can be fed explicitly to the later layers using the skip architecture. This could explain the improvement seen in the precision values for some diseases and overall recall effectively.

Experiments with Transfer learning involved modifying the last layer of the Vgg-16 model. Vgg-16 was chosen because of its inherent simplicity and its good performance in image data sets. The tuning of the model by freezing all layers except the last worked favorably, but the one involving unfreezing all layers and training them to our task did not function as our expectation. Hence the former was chosen for the task and those are the results that have been reported. Having been pre-trained on the huge ImageNet database (and for a considerable amount of time), Vgg-16 out performed our architectures, but this was to be expected.

For the ensembling part, we chose a simple voting mechanism, by choosing the best two of three predictions from the modified Baseline, Architecture_1 and Architeture_2. The ensembled model provided slightly better results than the models individually. We hypothesise that the approach worked well for dealing with weak points of a particular model (say Model A is not sensitive to occurrences of disease X, but Model B and Model C are. Taking a majority vote, thus made sure that this weakness is dealt with, while keeping the strengths of Model A)

5.2 Common Confusions

As shown by the confusion matrix, the models seem to have a lot of confusion between cardiomegaly and infiltration. Other disease combinations that showed confusions, in decreasing order were : Pneumonia and infiltration, Edema and effusion, Edema and infiltration, Effusion and cardiomegaly, Effusion and Atelectasis, Infiltration and Atelectasis Infiltration, notably is a difficult disease to correctly classify, as it has been misclassified by almost all the models we tested.

5.3 Analysis of different metrics and models

Accuracy was a metric which did not perform well on this task. As evident from the baseline performance, (the model had the highest accuracy of all the models, but abysmal precision and recall rates). The naive prediction of "no disease" could ensure that one is accurate most of the time, but such a mechanism is literally useless for all practical purposes.

Recall metric is a very good metric because it deals with how many of the diseased instances were picked up by the model, which is what essentially the biggest challenge. Baseline model performed poorly on this metric, and on modification of the loss function the model performed much better. The Recall metric went on to improve steadily as the architectures were improved on. Precision was also shown to improve as the models were made increasingly intricate, and the average of precision and Recall, given by BCR just went up consequently.

5.4 Visualization of weights

Architecture_2 was the best performing model, and the visualization for the weights was chosen from different layer depths from this model. The weights were coarser and more structured for the early and the middle layer as opposed to the layer. As compared to neural networks implemented for detecting objects (like the ones on CIFAR or ImageNets), the weights were not easy to interpret or clear in terms of what they were encapsulating. In fact this is a clear indication of the difficulty of the task at hand, since the diseases are not clear cut in terms of their occurrence in X-Ray images (for e.g, one cannot predict that a patient is suffering from say, Pneumonia because there is a square shaped deviation on the frontal X-Ray). The task is more nuanced than simpler object detection where the shapes are more defined. The later layers are more adept at detecting broader features of an image.

6 Individual Contribution

Ambareesh implemented the baseline model with the appropriate modifications and experimented on the baseline. He also implemented the weighted BCE loss function which were used for the later models.
Farheen was responsible for completing Architecture 1, and tweaking the weights necessary for the optimal performance. She also contributed to implementing the Confusion Matrix.
Gitika was responsible for implementing Architecture 2 and visualization of weights, in addition to contributing to confusion matrix implementation.
Transfer Learning with VGG16 was completed by Nasha. The ensembling was also implemented by Nasha
All the group members were responsible for debugging the code, and reviewing each other's work, reporting and discussing the results of their respective CNNs. All the ideas were discussed and

brainstormed together and then implemented individually and then the results were studied and cross examined.

References

[1] Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2097–2106, 2017.

[2] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[3] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. *CoRR*, abs/1507.06228, 2015.

[4] Mohammad Tariqul Islam, Md Abdul Aowal, Ahmed Tahseen Minhaz, and Khalid Ashraf. Abnormality detection and localization in chest x-rays using deep convolutional neural networks. *CoRR*, abs/1705.09850, 2017.

[5] Sabyasachi Sahoo. Deciding optimal kernel size for cnn. `https://towardsdatascience.com/deciding-optimal-filter-size-for-cnns-d6f7b56f936`. Accessed: 2019-02-18.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[7] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. *CoRR*, abs/1710.05941, 2017.

[8] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.