

**TITLE**

**“Detection of Covid-19 patient using chest X-ray**

Presented To

The Academic Faculty

in partial fulfilment

of the requirement for the M.Tech Degree

of

Mechatronics

by

**SURAJ SINGH PATWAL**

**1911MT12**

Under the supervision of

**Dr. SRIPARNA SAHA**



**DEPARTMENT OF MECHANICAL ENGINEERING**

**INDIAN INSTITUTE OF TECHNOLOGY**

**PATNA**

**(JUNE, 2021)**

Copyright © SURAJ SINGH PATWAL 2021

All Rights Reserved

# ACKNOWLEDGEMENTS

I would like to thank Dr. SRIPARAN SAHA, my supervisor, working as Associate Professor at Computer Science Department, IIT Patna for her valuable guidance and support. She kept me in the right direction throughout my thesis and help me in every step whenever I was facing problem while doing anything new.

I would also like to thank Dr. PRATIK DUTTA to help me in the initial phase of the season and taught me the way of doing research.

I would also like to thank Computer Centre to provide me the GPU access which helps me a lot while doing practical implementation of my project.

# **CERTIFICATE**

This is to certify that the thesis entitled “DETECTION OF COVID-19 PATIENT USING CHEST X-RAYS”, submitted by SURAJ SINGH PATWAL to Indian Institute of Technology Patna, is a record of bonafide research work under my (our) supervision and I (we) consider it worthy of consideration for the degree of Master of Technology of this Institute. This work or a part has not been submitted to any university/institution for the award of degree/diploma. The thesis is free from plagiarized material.

---

---

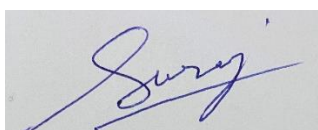
Supervisor

Date: 30/06/2021

# **DECLARATION**

I certify that

- a. The work contained in this thesis is original and has been done by myself under the general supervision of my supervisor/s.
- b. The work has not been submitted to any other Institute for degree or diploma.
- c. I have followed the Institute norms and guidelines and abide by the regulation as given in the Ethical Code of Conduct of the Institute.
- d. Whenever I have used materials (data, theory and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the reference section.
- e. The thesis document has been thoroughly checked to exclude plagiarism.

A handwritten signature in blue ink, appearing to read 'Suraj', is shown on a light gray background.

Signature of the Student

Roll No: 1911MT12

# TABLE OF CONTENT

CONTENT	PAGE
ACKNOWLEDGEMENTS	ii
LIST OF TABLES	vii
LIST OF FIGURES	viii
LIST OF ABBREVIATIONS	x
LIST OF SYMBOLS	xi
ABSTRACT	xii
<u>CHAPTERS</u>	
1. INTRODUCTION	1
1.1 Motivation	2
1.2 Objective	2
1.3 Problem Statement	3
1.4 Solution approach	3
1.5 Scope	4
1.6 Structure of Thesis	4
2. Literature review	5
2.1 CNN based review	5
2.2 Capsule-network based review	6
2.3 Research gap	7
3. Experiment Setup	8
3.1 Data	8
3.2 Metrics	9
3.3 Models	11
3.4 Code	17

4. Results and Discussion	18
4.1 Pretrained DenseNet121	18
4.2 Capsule Network	22
4.3 Further research	26
5. Conclusion	27
6. Reference	28

# LIST OF TABLES

Table 1	Number of images in each folder (Train, Test, Validation)	8
Table 2	Number of images in each subfolder	9

# LIST OF FIGURES

1.1	Mechanism of virus entry into the cell	1
1.2	Simple block diagram of the pipeline	2
4.1	Three CXR corresponding to each class.	8
4.2	Confusion Matrix	10
4.3	Various layers involve in the DenseNet based model.	11
4.4	Figure shows inside layers of DenseNet121	12
4.5	Shows the different layer used in 3 parallel CNN layered Capsule network	15
5.1	Shows training and validation accuracy and loss for each epoch for simple DenseNet model	18
5.2	Graphical representation of the variation of accuracy for train and validation dataset	19
5.3	Graphical representation of the variation of loss for train and validation dataset	19
5.4	Shows training and validation accuracy and loss for each epoch for simple DenseNet model with dropout	20
5.5	Graphical variation of accuracy and loss for simple model with dropout.	20
5.6	Shows training and validation accuracy and loss for each epoch for simple DenseNet model with L2 regularization.	21



5.7	Graphical variation of accuracy and loss for simple model with L2 regularization.	21
5.8	Various layers in basic capsule network.	22
5.9	Shows training and validation accuracy and loss for each epoch for Basic Capsule Network model	23
5.10	Graphical variation of accuracy and loss for Basic capsule network model.	23
5.11	Various layers in capsule network with concatenated 3 CNN layers.	24
5.12	Shows training and validation accuracy and loss for each epoch for non-basic capsule network model.	25
5.13	Graphical variation of accuracy and loss for non-basic capsule network model.	25

# LIST OF ABBREVIATIONS

	MEANING
Dr.	Doctor
SARS COV2	Severe acute respiratory syndrome coronavirus 2
RT-PCR	Reverse transcription polymerase chain reaction
CNN	Convolutional Neural Network
CXR	Chest X-rays
DL	Deep- Learning
ML	Machine Learning
RGB	Red Blue Green
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative

# LIST OF SYMBOLS

$g_t$ , is a gradient of objective function

$m_t$ , is updated first moment estimate

$v_t$ , is updated second moment estimate

$\hat{m}_t$ , is compute bias-corrected first moment estimate.

$\hat{v}_t$ , is compute bias- corrected second moment estimate.

$\theta_t$ , updated weight.

$\beta_1, \beta_2$  are exponentially decay rate for the moment estimates

$\alpha$ , step size.

$M$  is number of classes,

$y_C$  is label value of that particular class,

$p_C$  is a predicted value by model for that class.

$c_{ij}$  is Coupling coefficient.

$\hat{u}_{j,i}$  predictive vector.

$S_j$  Weighted sum.

$v_j$  Class vector.

$b_{ij}$  it is any variable which is use to define  $c_{ij}$ .

# ABSTRACT

In this era of pandemic, our study is on using different deep learning technique to predict that a patient has Covid-19 using his chest X-rays. As we all know SARS COV2 is a kind of pneumonia in which mucus is heavily dense in chest so our classes of classification are COVID-19 patient, Pneumonia patient and Normal during the study.

In a simple term what we have done in this study is creating a deep-learning model such as DenseNet121 and Capsule Network and train them using chest X-rays and use the model to classify a new set of chest X-rays into 3 classes as mentioned above.

This process might be used in patient screen before RT-PCR results will come which takes 8-10 hr and, in some places, may takes 20-40 hrs and think about that a SARS COV2 patient wondering around here and there and spread this infection for 8-10 hrs without any isolation, this is the one of the reasons for large amount of spread of this infection in the society in the form of first waves and second waves of this infection.

This method using deep-learning, detect a patient with in a second if use optimistically, and isolates patients who comes positive and wait for there RT-PCR report then further action will be taking, help our health care system in preventing this infection to spread

# Chapter-1

## INTRODUCTION

SARS COV2 was originated at Wuhan city, China and because of it there are 2.96 crore India and 17.7 crore in world cases and around 37 lakh people died in this pandemic. SARS COV2 is a kind of virus which consists of mainly 5 types of protein as shown in fig1, which are

- Spike protein
- Envelope protein
- Membrane glycoprotein
- RNA
- Nucleocapsid protein

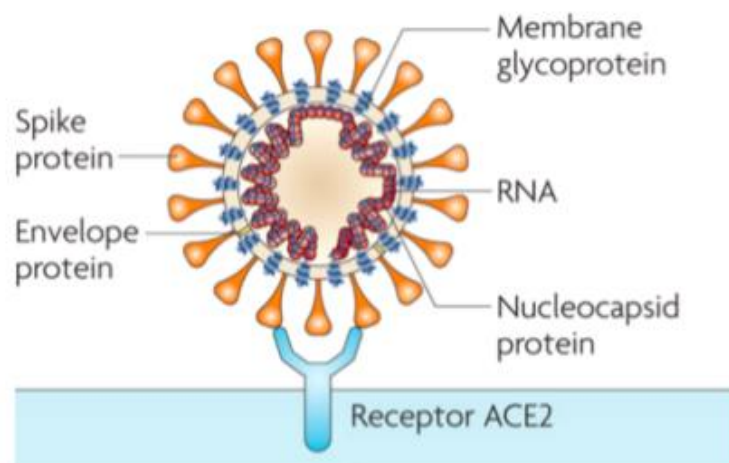


Fig 1.1: Mechanism of virus entry into the cell, Hoffmann et al. [1]

Mechanism: Spike protein on the virus stick with the receptor ACE2 which then broken down by TMPRESS, cell and virus membrane stick together and with the help of RNA, virus replicate in the lungs cell and kill it, these death cells are collected in the alveoli and cause difficulty in breathing.

## 1.1 Motivation

Due to day-to-day causalities, because of Covid-19 infection. Our motivation is to reduce the spread of this infection, which is not only be done by wearing mask, use sanitiser after touching any surface, social distancing like precaution but also by creating a screening test before conventional testing such as RT-PCR. As we all know mucous is generated in the lungs which cause difficulty in breathing and this mucous create some kind of pattern in the chest X-ray so we can use chest x-ray to predict that a patient has this infection or not. Chest X-ray either read by radiologist or Machine learning model, both have their own advantages and disadvantages, if it is predicted by radiologists then time and cost of prediction increases as a result in this case too infection spreading increase, so we go for Deep-learning algorithm which not only reduces the time but also the cost of the test reduces and we stop the spread by larger amount.

## 1.2 Objective

Main objective of the thesis is to create Deep-learning models which will take Chest X-rays and classify it into any of the 3 classes at which it is belong. We created DesnseNet121 and capsule network in to predict chest X-ray into covid-19 patient, pneumonia patient and normal classes.

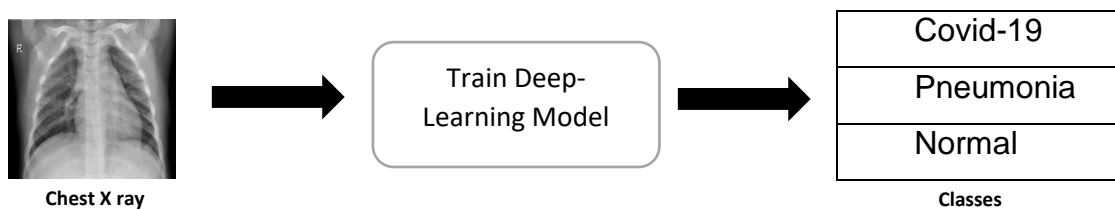


Fig 1.2 Simple block diagram of the pipeline.

### 1.3 Problem Statement

In this era of pandemic people are dying because of SARS COV2 as it is spreading infection and in order to diagnose it, RT-PCR is used as a conventional way of detecting the infection which takes around 8-10 h because of transportation and in some places this time will reach up to 20-24 hr as a result infection spread increases because covid patient is not isolated after the test, that is a reason a screening test is required after which those comes positive will be isolated and stop the spread of infection.

This is a reason our thesis is on “Detecting Covid-19 Patient Using Chest X Rays”, In which DL models are created and train them using a train dataset of the classes then this trained model is used to predict that a patient have that infection or not as shown in Fig1.2.

Fig 1.2 is a simple block diagram of which shows three blocks first is for CXR which is inserted into the Second block which represent Train DL model and then 3<sup>rd</sup> block is for 3 well known classes in which model will classify CXR into.

### 1.4 Solution Approach

First we collect all the three classes dataset then divide it into Train, Validation and Test directory each directory contain all the three class folder with corresponding image then a Deep-learning modal like Densenet121 and capsule network is train on the train set in simple way and weights of the model are updated using Adam optimiser algorithm or Dynamic routing by agreement algorithm,( for Densenet121 algorithm is gradient decent and for capsule network algorithm is both gradient descent and Dynamic routing) then this model is tested using test dataset, see the Fig 1.2.

## 1.5 Scope

There are various opportunities and possibility in that work, specially in pandemic, which are:

- Result will be provided by this test is economic and less time consuming.
- X-ray machine is portable, so by placing it in some kind of isolation help the health workers in the prevention of the infection.
- No transportation requires, it will provide you the results on the spot.
- If this test is taking before RT-PCR then it will provide some screening option to health care system and stop the further spread because of covid-19 positive patients.
- As X-ray machine is portable in less requirement of the personal protected equipment will be there.
- Model can use in detecting various type of pneumonia and in the similar way it can be use to diagnose lung cancer.
- It can be acting as a replacement of RT-PCR test which is costlier and pain full in nostrils and throat and sometime may cause bleeding too.

## 1.6 Structure of Thesis

- **Literature review:** This section includes the reviews of research papers and article which have been read during the study by us.
- **Technical chapters:** This section contains the theoretical knowledge about the CNN(DenseNet121) and Basic Capsule network.
- **Experiment Setup:** It deals about the kind of data we have chosen to train the models and metrics.
- **Results and Discussion:** It will show the results of the whole study and further research.
- **Conclusion:** To show the whole conclusion of the study.



# Chapter: 2

## LITERATURE REVIEW

On the bases of the models developed, research paper read by us can be classify into two categories.

1. CNN related research paper.
2. Capsule Network related papers.

### 2.1 CNN Based review

In order to predict a CXR is Covid-19 positive or not then if we are using CNN then basic approaches are:

1. Transfer learning.
2. Transfer learning with Bagging or stacking.

Narin et al.[12] pretrain models Densenet121 on ImageNet dataset having 3 different binary classes( Binary class1 = COVID-19 and Normal, Binary class 2 = COVID-19 and Viral pneumonia and Binary class 3 = COVID-19 and Bacterial pneumonia), after pretraining it is again trained by CXR which undergoes image segmentation of lungs and various image enhancement technique during image pre-processing technique which increases the accuracy up to certain level. Apart from this Kana et al.[6] also uses pretrained model ResetNet50 on Imagenet but uses differential learning rate to train the model for 15 epoch and make a web interface that will predict that a CXR instance is how much covid-19 positive and normal.

Mishra et al.[7] predict that a person is infected or not by using stacking technique and take output from all the pretrained model, and majority voted class is the output of the complete model. But in Pardamean et al.[4] DenseNet is used which is pretrained on CXR and then mammogram data set is inserted

to predict that a person has a breast cancer or not, in order to do that they uses 4 DenseNet121 because there are 4 mammogram present at one case. Decomposition and composition of classes is used in model DaTrac which is initially pretrained on ImageNet by abbas et al.[13].

CheXNet is the name of the model also used to classify covid-19 CXR uses DenseNet and pretrained on NIH chest X rays on 14 classes then after using 3 type of CXR of covid, pneumonia and normal to train fully connected layer. CovidAID by Mangal et al.[5] uses pretrained CheXNet and train fully connecting layer then this model weight is taking as a initial weight for further training of complete model (CheXNet and fully connected layer).

## **2.2 Capsule-Network Based Review**

As not much research has been done in the field of capsule network because its orientation is at 2017 by Hinton and Sabour, so they explain it by giving a paper Sabour et al.[15] which describes the algorithm Dynamic routing which governs the network and basic form of network used to predict numerical digits from MNIST database and reconstruct these digit too, so it give rise to two types of losses margin loss and constructive loss. Patrick et al.[16] explain the various deep learning techniques including capsule network and provide its various application in the field of computer vision.

Mukhometzianov et al.[17] does comparative study of basic capsule network using various datasets:

- Yale face database B: Capsule network accuracy 95.3%.
- MIT CBCL: Accuracy reached 99.87%.
- Belgium TS: Accuracy comes out to be 92%.
- CIFAR-100: Accuracy is 18%.

It is showing we can use capsule network for various application too.

Comparative study is to be done by Nair et al[21] using dataset:

- Handwriting digit MNIST Dataset.
- Fashion-MNIST.

- CIFAR 10
- SVHN

In all these datasets the network is able to give decent accuracy. Apart from this Nguyen et al[20] uses this network for building knowledge graph and search personalization.

Lin et al.[18] recognises two handwritten digit in one image and classify both of these using capsule network. Kim et al [23] uses the network for classification of text in various dataset.

DenseCapsNet proposed by Quan et al[22], which uses the combination of DenseNet and capsule network along with histogram equalisation technique during image pre-processing technique to predict that a CXR belong to various class such as Covid-19, pneumonia and normal. In a similar way Convolutional Capsnet also use capsule network to solve the same problem proposed by Toraman et al. [24].

## **2.3 Research Gap**

On seeing the literature review of both the DL techniques we see that there is a lot of scope left in using capsule network to classify a CXR into any of the 3 classes such as Covid-19, Pneumonia and Normal. By using various modified capsule networks, in addition with various image enhancement processes accuracy to predict that a patient has covid-19 might be increase up to a larger extent, as there is very less number of dataset available in terms of covid patients CXR, then using this network for such a small dataset is a good choice of model selection.

# Chapter: 3

## EXPERIMENTAL SETUP

### 4.1 Data

Data plays most important part in the study with it we couldn't build any model. Data uses during the study is in the form of chest X rays as shown in the Fig. 4.1. It represents 3 CXR of Covid-19 patient, Pneumonia patient and Normal person.

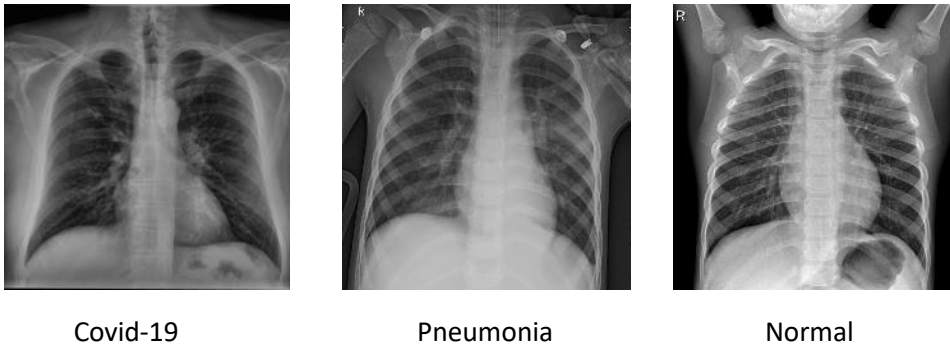


Fig 4.1 Three CXR corresponding to each class.

Data is organised into 3 folders (Train, Validation and Test) and each folder has 3 sub folders corresponding to each class (Covid, Pneumonia and Normal).

Train	Test	Validation
2123	700	705

Table 1: Number of images in each folder (Train, Test, Validation)

	Covid	Pneumonia	Normal
Train	263	930	930
Test	80	310	310
Validation	84	311	310

Table 2. Number of images in each subfolder.

As there are very a smaller number of images available for covid-19 patient as compare to other two classes and required lot of effort in order to extract them from their respective folders of dataset, Covid-19 images are extract from the following databases:

- Figure 1 Covid-19 Chest X Rays datasets [25].
- ActualMed Covid-19 image dataset [27].
- Covid-19 Chestxray image dataset master [28].
- Covid-19 radiography dataset [11].

Pneumonia and Normal images are collecting from:

- Chest X ray image (Pneumonia) [8].

## 4.2 Metrics

In this study, accuracy is considered to be the criteria of evaluation but there are other criteria too, these criteria of evaluation a model performance is called as metrics.

Accuracy is simply the ratio of total correct prediction and total prediction.

Other classification performance metrics of evaluation can be formulate as below:

		Predictive values	
		Positive	Negative
Actual Values	Positive	TP	FN
	Negative	FP	TN

Fig 4.2 Confusion Matrix

- $\text{Accuracy} = \frac{TP+TN}{TP+TN+FN+FP}$
- $\text{Precision} = \frac{TP}{TP+FP}$
- $\text{Recall} = \frac{TP}{TP+FN}$
- $\text{Specificity} = \frac{TN}{TN+FP}$
- $F1 = \frac{2(\text{Precision} \cdot \text{Recall})}{\text{Precision} + \text{Recall}}$

## 4.3 Model

Through out the year, there are two models are prepared, which are:

- DenseNet121
- Capsule network

**4.3.1 DenseNet121:** It is a residual kind of network which is divided into 5 blocks followed by convolutional and pooling operation alternatively.

DenseNet121 model consists of 5 blocks of DenseNet and one fully connected layer of 1024 neurons and followed by classification layer contains 3 neurons to classify the image into 3 classes.

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
=====	=====	=====
densenet121 (Functional)	(None, 7, 7, 1024)	7037504
flatten (Flatten)	(None, 50176)	0
dense (Dense)	(None, 1024)	51381248
dense_1 (Dense)	(None, 3)	3075
=====	=====	=====
Total params: 58,421,827		
Trainable params: 51,384,323		
Non-trainable params: 7,037,504		

Fig 4.3 Various layers involve in the DenseNet based model.

This model have Densenet121 layer has 7037504 weights all are pretrained in ImageNet dataset after that end layer of DenseNet get flattened into 50176 neuron involve no parameter to train, After flattening there is a dense layer of 1024 neurons involves 51381248 weights which get trained in second training process involving our dataset. At the end there is 3 layer of neuron connected to previous layer by 3075 weights which also get trained in our dataset.

Inside layers of DenseNet121 are shown in figure below:

Model: "densenet121"			
Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 224, 224, 3)]	0	
zero_padding2d (ZeroPadding2D)	(None, 230, 230, 3)	0	input_1[0][0]
conv1/conv (Conv2D)	(None, 112, 112, 64)	9408	zero_padding2d[0][0]
conv1/bn (BatchNormalization)	(None, 112, 112, 64)	256	conv1/conv[0][0]
conv1/relu (Activation)	(None, 112, 112, 64)	0	conv1/bn[0][0]
zero_padding2d_1 (ZeroPadding2D)	(None, 114, 114, 64)	0	conv1/relu[0][0]
pool1 (MaxPooling2D)	(None, 56, 56, 64)	0	zero_padding2d_1[0][0]
conv2_block1_0_bn (BatchNormalization)	(None, 56, 56, 64)	256	pool1[0][0]
conv2_block1_0_relu (Activation)	(None, 56, 56, 64)	0	conv2_block1_0_bn[0][0]
conv2_block1_1_conv (Conv2D)	(None, 56, 56, 128)	8192	conv2_block1_0_relu[0][0]
conv2_block1_1_bn (BatchNormalization)	(None, 56, 56, 128)	512	conv2_block1_1_conv[0][0]
conv2_block1_1_relu (Activation)	(None, 56, 56, 128)	0	conv2_block1_1_bn[0][0]
conv2_block1_2_conv (Conv2D)	(None, 56, 56, 32)	36864	conv2_block1_1_relu[0][0]
conv2_block1_concat (Concatenation)	(None, 56, 56, 96)	0	pool1[0][0] conv2_block1_2_conv[0][0]
conv2_block2_0_bn (BatchNormalization)	(None, 56, 56, 96)	384	conv2_block1_concat[0][0]
conv2_block2_0_relu (Activation)	(None, 56, 56, 96)	0	conv2_block2_0_bn[0][0]
.			
.			
conv5_block15_1_bn (BatchNormalization)	(None, 7, 7, 128)	512	conv5_block15_1_conv[0][0]
conv5_block15_1_relu (Activation)	(None, 7, 7, 128)	0	conv5_block15_1_bn[0][0]
conv5_block15_2_conv (Conv2D)	(None, 7, 7, 32)	36864	conv5_block15_1_relu[0][0]
conv5_block15_concat (Concatenation)	(None, 7, 7, 992)	0	conv5_block14_concat[0][0] conv5_block15_2_conv[0][0]
conv5_block16_0_bn (BatchNormalization)	(None, 7, 7, 992)	3968	conv5_block15_concat[0][0]
conv5_block16_0_relu (Activation)	(None, 7, 7, 992)	0	conv5_block16_0_bn[0][0]
conv5_block16_1_conv (Conv2D)	(None, 7, 7, 128)	126976	conv5_block16_0_relu[0][0]
conv5_block16_1_bn (BatchNormalization)	(None, 7, 7, 128)	512	conv5_block16_1_conv[0][0]
conv5_block16_1_relu (Activation)	(None, 7, 7, 128)	0	conv5_block16_1_bn[0][0]
conv5_block16_2_conv (Conv2D)	(None, 7, 7, 32)	36864	conv5_block16_1_relu[0][0]
conv5_block16_concat (Concatenation)	(None, 7, 7, 1024)	0	conv5_block15_concat[0][0] conv5_block16_2_conv[0][0]
bn (BatchNormalization)	(None, 7, 7, 1024)	4096	conv5_block16_concat[0][0]
relu (Activation)	(None, 7, 7, 1024)	0	bn[0][0]
Total params: 7,037,504			
Trainable params: 6,953,856			
Non-trainable params: 83,648			

Fig 4.4 Figure shows inside layers of DenseNet121.



Parameter used in model making:

- Learning rate = 1e-3
- Epochs = 20
- Steps per epoch = 100
- Validation steps = 35
- Batch size = 20

Optimizer = Adam.

Loss = Categorical cross entropy.

**Adam optimizer algorithm:**

$$g_t = \nabla_{\theta} f_t(\theta_{t-1})$$

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$$

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$$

$$\hat{m}_t = m_t / (1 - \beta_1^t)$$

$$\hat{v}_t = v_t / (1 - \beta_2^t)$$

$$\theta_t = \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} - \epsilon)$$

Where  $g_t$ , is a gradient of objective function

$m_t$ , is updated first moment estimate

$v_t$ , is updated second moment estimate

$\hat{m}_t$ , is compute bias-corrected first moment estimate.

$\hat{v}_t$ , is compute bias- corrected second moment estimate.

$\theta_t$ , updated weight.

$\beta_1, \beta_2$  are exponentially decay rate for the moment estimates

$\alpha$ , step size.

**Categorical Cross entropy loss:** this loss is also called as log loss.

$$= - \sum_{C=1}^M y_C \cdot \log(p_C)$$

Where, M is number of classes,  $y_C$  is label value of that particular class,

$p_C$  is a predicted value by model for that class.

### 4.3.2 Capsule Network:

Those networks are known as capsule networks in which primary capsules and class capsules are involves. Each primary capsule is a collection neuron which represent an entity of a class and each neuron is the attribute of that entity and class capsule also have neuron or dimensions which involves any attributes of that class equal in number as the dimension of that class.

In computer vision CNN plays a remarkable performance but it is failed to give right classification if some features are spatially oriented in some other way.

For example, A face has eye, nose and mouth and ear which have some spatial orientation and symmetrical in size, our CNN still classify as a face if spatial orientation of eye and nose is not proper or both the eye are not in same shape.

And in max pooling layer of CNN network, a large amount of information is lost, so these two are the big draw back of CNN that is why we move to the capsule network to solve that task. But it is a new network as it is originated in 2017 so it has more scope of improvement.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 28, 28, 3)	0	
conv1 (Conv2D)	(None, 20, 20, 256)	62464	input_1[0][0]
conv2 (Conv2D)	(None, 20, 20, 256)	62464	input_1[0][0]
conv3 (Conv2D)	(None, 20, 20, 256)	62464	input_1[0][0]
concatenate_1 (Concatenate)	(None, 20, 20, 768)	0	conv1[0][0] conv2[0][0] conv3[0][0]
primarycap_conv2d (Conv2D)	(None, 19, 19, 256)	786688	concatenate_1[0][0]
primarycap_reshape (Reshape)	(None, 11552, 8)	0	primarycap_conv2d[0][0]
primarycap_squash (Lambda)	(None, 11552, 8)	0	primarycap_reshape[0][0]
digitcaps (CapsuleLayer)	(None, 3, 16)	4435968	primarycap_squash[0][0]
capsnet (Length)	(None, 3)	0	digitcaps[0][0]
Total params: 5,410,048			
Trainable params: 5,410,048			
Non-trainable params: 0			

Fig 4.5 shows the different layer used in three parallel CNN layered Capsule network

Image is inserted of shape (28,28) of RGB type to the model and in the model, features are extracted by 3 convolutional layers parallelly by the filter of size 9\*9 and form 256 channels, as capsule network has wide more that is why channel size is taken to be large, it will contains 62464 weights to trains per layer, then after all the three layers get concatenated and form feature map of (20,20) but channels are larger in number the previous one which is 768. In the primary layer again, convolution is applied by using filter of size 9\*9 which make the feature map of (19,19) with 256 channels after this primary capsule layer start making capsule of size (1,1,8) and total of 11552 capsules are there which represent some entity of the CXR of lungs with 8 is the entities attributes. Then finally it is squash into the 3 class of 16 dimension shows the 16 properties of the class, this layer involves 4435968 weights to train during training and each of the vector magnitude is calculate corresponding to each class to classify.

It follows Dynamic routing by agreement algorithm and margin loss, as other parameters are not considered such as batch size learning rate because it has no effect on capsule network performance [16].

### Dynamic Routing by Agreement Algorithm:

$$b_{ij} = 0$$

$$c_{ij} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ik})} \quad \dots (1)$$

$$S_j = \sum_i c_{ij} \cdot \hat{u}_{j,i} \quad \dots (2)$$

$$v_j = \frac{\|S_j\|^2}{1 + \|S_j\|^2} \cdot \frac{S_j}{\|S_j\|} \quad \dots (3)$$

$$b_{ij} = b_{ij} + \hat{u}_{j,i} \cdot v_j \quad \dots (4)$$

Where,  $c_{ij}$  is Coupling coefficient.

$\hat{u}_{j,i}$  predictive vector.

$S_j$  Weighted sum.

$v_j$  Class vector.

$b_{ij}$  it is a any variable which is use to define  $c_{ij}$ .

### Margin Loss:

$$L_c = T_c \max(0, m_+ - \|v_c\|)^2 + \lambda (1 - T_c) \max(0, \|v_c\| - m_-)^2$$

Where,  $L_c$  is loss related to  $c$  classes

$T_c = 1$ , if predicted class is the  $c$  class.

$T_c = 0$ , if the class is not  $c$ .

$m_+ = 0.9$ ,  $m_- = 0.1$  and  $\lambda = 0.5$

#### 4.4. Code

In order to get results two codes are used for each model whose GitHub link is shown below:

- Pretrained DenseNet Model:  
<https://github.com/Surajxyz/Transfer-Learning>
- Capsule Network:  
<https://github.com/Surajxyz/Capsule-Network>

# Chapter: 5

## RESULTS AND DISCUSSION

### 5.1 Pretrained DenseNet121

In order to build a model, pretraining of DenseNet121 is to be performed using ImageNet data set, then various DL techniques are used to the basic model in order to see accuracy as metrics.

DL techniques are:

- Dropout
- Regularization

Data of Simple transfer learning DenseNet model:

1. **Simple Densenet121 model:** Batch size: 20 images, learning rate=1e-3, loss=Categorical cross entropy loss, Adam optimizer.

```
Epoch 1/20
100/100 [=====] - 830s 8s/step - loss: 6.5453 - acc: 0.8901 - val_loss: 1.1635 - val_acc: 0.9486
Epoch 2/20
100/100 [=====] - 41s 414ms/step - loss: 0.4116 - acc: 0.9743 - val_loss: 0.3903 - val_acc: 0.9800
Epoch 3/20
100/100 [=====] - 13s 126ms/step - loss: 0.5443 - acc: 0.9677 - val_loss: 0.7024 - val_acc: 0.9657
Epoch 4/20
100/100 [=====] - 10s 95ms/step - loss: 0.2288 - acc: 0.9844 - val_loss: 0.5428 - val_acc: 0.9714
Epoch 5/20
100/100 [=====] - 10s 95ms/step - loss: 0.0551 - acc: 0.9924 - val_loss: 0.8354 - val_acc: 0.9557
Epoch 6/20
100/100 [=====] - 10s 96ms/step - loss: 0.0802 - acc: 0.9929 - val_loss: 0.2775 - val_acc: 0.9857
Epoch 7/20
100/100 [=====] - 10s 96ms/step - loss: 0.1089 - acc: 0.9869 - val_loss: 0.5119 - val_acc: 0.9657
Epoch 8/20
100/100 [=====] - 10s 97ms/step - loss: 0.4239 - acc: 0.9788 - val_loss: 1.4993 - val_acc: 0.9343
Epoch 9/20
100/100 [=====] - 10s 97ms/step - loss: 0.1472 - acc: 0.9864 - val_loss: 0.3997 - val_acc: 0.9800
Epoch 10/20
100/100 [=====] - 10s 97ms/step - loss: 0.0571 - acc: 0.9939 - val_loss: 0.5394 - val_acc: 0.9714
Epoch 11/20
100/100 [=====] - 10s 96ms/step - loss: 0.0103 - acc: 0.9990 - val_loss: 0.4743 - val_acc: 0.9729
Epoch 12/20
100/100 [=====] - 10s 97ms/step - loss: 0.0968 - acc: 0.9895 - val_loss: 0.3933 - val_acc: 0.9814
Epoch 13/20
100/100 [=====] - 10s 97ms/step - loss: 0.0096 - acc: 0.9985 - val_loss: 0.4588 - val_acc: 0.9771
Epoch 14/20
100/100 [=====] - 10s 96ms/step - loss: 0.0277 - acc: 0.9960 - val_loss: 0.6855 - val_acc: 0.9614
Epoch 15/20
100/100 [=====] - 10s 95ms/step - loss: 1.4679e-04 - acc: 1.0000 - val_loss: 0.3959 - val_acc: 0.9814
Epoch 16/20
100/100 [=====] - 10s 95ms/step - loss: 6.7227e-04 - acc: 0.9995 - val_loss: 0.4144 - val_acc: 0.9814
Epoch 17/20
100/100 [=====] - 10s 97ms/step - loss: 0.0087 - acc: 0.9990 - val_loss: 0.3377 - val_acc: 0.9814
Epoch 18/20
100/100 [=====] - 10s 96ms/step - loss: 0.0044 - acc: 0.9985 - val_loss: 0.3985 - val_acc: 0.9800
```

Fig 5.1 Shows training and validation accuracy and loss for each epoch for simple DenseNet model

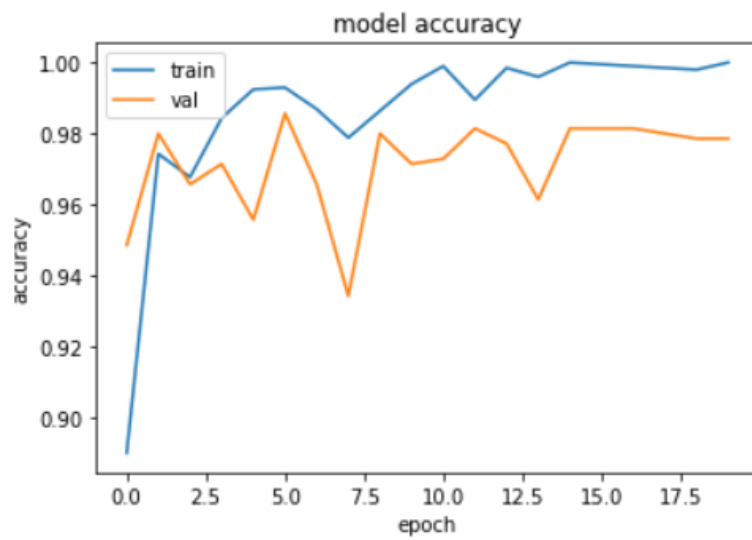


Fig 5.2 Graphical representation of the variation of accuracy for train and validation dataset.

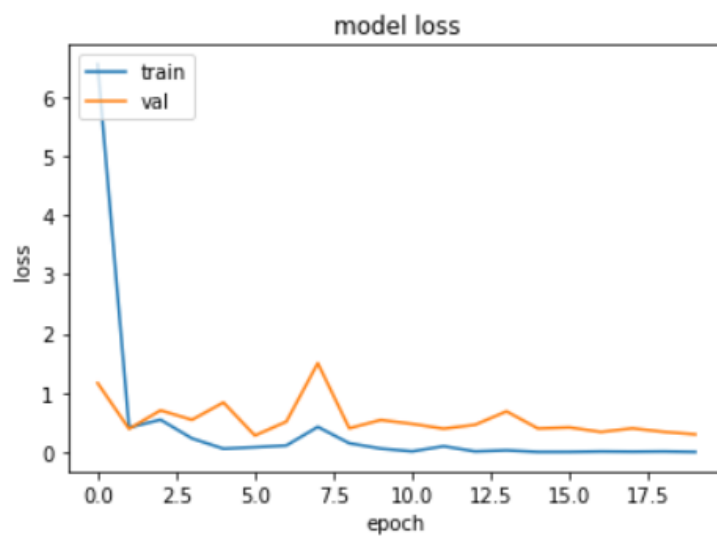


Fig 5.3 Graphical representation of the variation of loss for train and validation dataset

**Test accuracy comes out to be 84.57%**

**2.Simple DenseNet model with dropout at fully connected layer:** Batch size: 20 images, learning rate=1e-3, loss=Categorical cross entropy loss, Adam optimizer and dropout of 50% is applied.

```
Epoch 1/20
100/100 [=====] - 12s 119ms/step - loss: 2.1470 - acc: 0.8593 - val_loss: 0.6823 - val_acc: 0.9557
Epoch 2/20
100/100 [=====] - 10s 98ms/step - loss: 0.1768 - acc: 0.9470 - val_loss: 0.2241 - val_acc: 0.9643
Epoch 3/20
100/100 [=====] - 10s 99ms/step - loss: 0.0713 - acc: 0.9642 - val_loss: 0.1127 - val_acc: 0.9886
Epoch 4/20
100/100 [=====] - 10s 98ms/step - loss: 0.0354 - acc: 0.9733 - val_loss: 0.1320 - val_acc: 0.9657
Epoch 5/20
100/100 [=====] - 10s 97ms/step - loss: 0.0412 - acc: 0.9642 - val_loss: 0.0663 - val_acc: 0.9843
Epoch 6/20
100/100 [=====] - 10s 97ms/step - loss: 0.0437 - acc: 0.9697 - val_loss: 0.1642 - val_acc: 0.9743
Epoch 7/20
100/100 [=====] - 10s 96ms/step - loss: 0.0551 - acc: 0.9622 - val_loss: 0.0699 - val_acc: 0.9829
Epoch 8/20
100/100 [=====] - 10s 98ms/step - loss: 0.0273 - acc: 0.9798 - val_loss: 0.0555 - val_acc: 0.9857
Epoch 9/20
100/100 [=====] - 10s 97ms/step - loss: 0.0142 - acc: 0.9844 - val_loss: 0.0562 - val_acc: 0.9886
Epoch 10/20
100/100 [=====] - 10s 97ms/step - loss: 0.0085 - acc: 0.9904 - val_loss: 0.0584 - val_acc: 0.9886
Epoch 11/20
100/100 [=====] - 10s 97ms/step - loss: 0.0132 - acc: 0.9844 - val_loss: 0.0437 - val_acc: 0.9857
Epoch 12/20
100/100 [=====] - 10s 97ms/step - loss: 0.0160 - acc: 0.9813 - val_loss: 0.1102 - val_acc: 0.9743
Epoch 13/20
100/100 [=====] - 10s 97ms/step - loss: 0.0187 - acc: 0.9798 - val_loss: 0.0853 - val_acc: 0.9829
Epoch 14/20
100/100 [=====] - 10s 97ms/step - loss: 0.0319 - acc: 0.9773 - val_loss: 0.0944 - val_acc: 0.9829
Epoch 15/20
100/100 [=====] - 10s 98ms/step - loss: 0.1012 - acc: 0.9481 - val_loss: 0.5400 - val_acc: 0.8871
Epoch 16/20
100/100 [=====] - 10s 100ms/step - loss: 0.0595 - acc: 0.9526 - val_loss: 0.1436 - val_acc: 0.9600
Epoch 17/20
100/100 [=====] - 10s 97ms/step - loss: 0.0304 - acc: 0.9743 - val_loss: 0.2099 - val_acc: 0.9614
Epoch 18/20
100/100 [=====] - 10s 96ms/step - loss: 0.0346 - acc: 0.9738 - val_loss: 0.1113 - val_acc: 0.9729
Epoch 19/20
100/100 [=====] - 10s 97ms/step - loss: 0.0166 - acc: 0.9813 - val_loss: 0.0675 - val_acc: 0.9871
Epoch 20/20
100/100 [=====] - 10s 97ms/step - loss: 0.0156 - acc: 0.9859 - val_loss: 0.0990 - val_acc: 0.9786
```

Fig 5.4 Shows training and validation accuracy and loss for each epoch for simple DenseNet model with dropout.

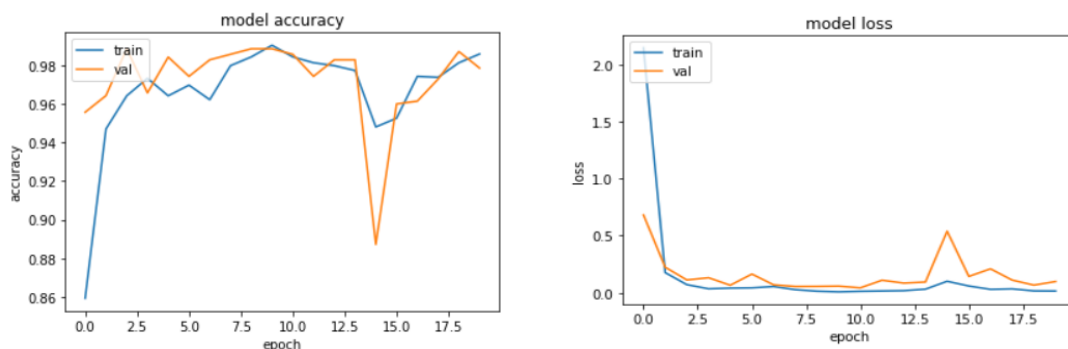


Fig 5.5 Graphical variation of accuracy and loss for simple model with dropout.

**Test accuracy =87.00%**



**3.Simple model with L2 regularization:** All the parameters are same as the simple model but L2 regularization is used with l value of 0.08.

```
Epoch 1/20
100/100 [=====] - 12s 119ms/step - loss: 18.3783 - acc: 0.8575 - val_loss: 4.7390 - val_acc: 0.8300
Epoch 2/20
100/100 [=====] - 11s 109ms/step - loss: 2.5958 - acc: 0.8709 - val_loss: 2.5314 - val_acc: 0.8586
Epoch 3/20
100/100 [=====] - 11s 105ms/step - loss: 0.7429 - acc: 0.9153 - val_loss: 0.6556 - val_acc: 0.8886
Epoch 4/20
100/100 [=====] - 10s 103ms/step - loss: 0.4989 - acc: 0.9112 - val_loss: 0.5408 - val_acc: 0.9414
Epoch 5/20
100/100 [=====] - 10s 103ms/step - loss: 0.3095 - acc: 0.9239 - val_loss: 0.5397 - val_acc: 0.9343
Epoch 6/20
100/100 [=====] - 10s 102ms/step - loss: 0.2756 - acc: 0.9299 - val_loss: 0.3596 - val_acc: 0.9271
Epoch 7/20
100/100 [=====] - 10s 103ms/step - loss: 0.2400 - acc: 0.9265 - val_loss: 0.3579 - val_acc: 0.9757
Epoch 8/20
100/100 [=====] - 10s 103ms/step - loss: 0.2512 - acc: 0.9244 - val_loss: 0.3466 - val_acc: 0.9543
Epoch 9/20
100/100 [=====] - 10s 103ms/step - loss: 0.2903 - acc: 0.9087 - val_loss: 0.2802 - val_acc: 0.9729
Epoch 10/20
100/100 [=====] - 10s 103ms/step - loss: 0.2089 - acc: 0.9193 - val_loss: 0.3005 - val_acc: 0.9357
Epoch 11/20
100/100 [=====] - 10s 103ms/step - loss: 0.1493 - acc: 0.9370 - val_loss: 0.1579 - val_acc: 0.9786
Epoch 12/20
100/100 [=====] - 10s 103ms/step - loss: 0.1481 - acc: 0.9360 - val_loss: 1.8544 - val_acc: 0.4086
Epoch 13/20
100/100 [=====] - 10s 102ms/step - loss: 0.2888 - acc: 0.9188 - val_loss: 0.2209 - val_acc: 0.9500
Epoch 14/20
100/100 [=====] - 10s 102ms/step - loss: 0.1657 - acc: 0.9380 - val_loss: 0.2511 - val_acc: 0.9500
Epoch 15/20
100/100 [=====] - 10s 103ms/step - loss: 0.1632 - acc: 0.9344 - val_loss: 0.2021 - val_acc: 0.9657
Epoch 16/20
100/100 [=====] - 10s 103ms/step - loss: 0.1199 - acc: 0.9506 - val_loss: 0.1836 - val_acc: 0.9729
Epoch 17/20
100/100 [=====] - 10s 102ms/step - loss: 0.1548 - acc: 0.9375 - val_loss: 0.2012 - val_acc: 0.9529
Epoch 18/20
100/100 [=====] - 10s 102ms/step - loss: 0.1573 - acc: 0.9334 - val_loss: 0.2161 - val_acc: 0.9571
Epoch 19/20
100/100 [=====] - 10s 102ms/step - loss: 0.1444 - acc: 0.9476 - val_loss: 0.2737 - val_acc: 0.9514
Epoch 20/20
100/100 [=====] - 10s 103ms/step - loss: 0.1490 - acc: 0.9360 - val_loss: 0.1615 - val_acc: 0.9714
```

Fig 5.6 Shows training and validation accuracy and loss for each epoch for simple DenseNet model with L2 regularization.

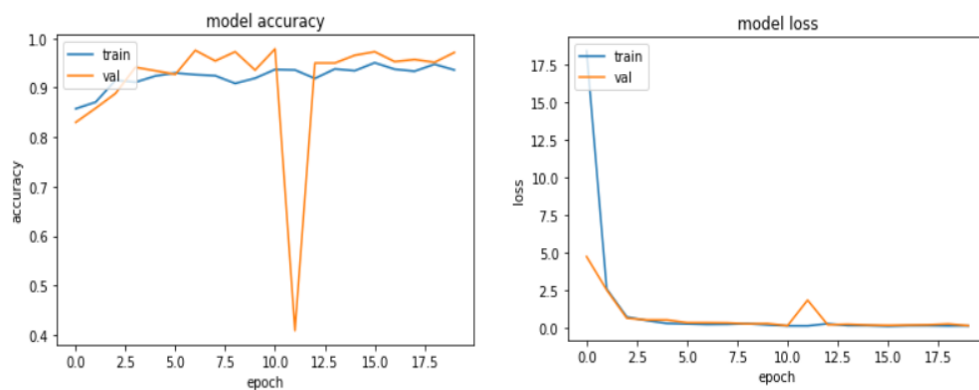


Fig 5.7 Graphical variation of accuracy and loss for simple model with L2 regularization.

**Test Accuracy = 90.28%**

On applying these DL techniques, on simple model of DenseNet121 one by one the accuracy is increases from 84.57 % to 90.28 %. Which is a good improvement, and on seeing the graphs of variations, the train accuracy is in general greater than validation accuracy and train loss is in general lesser than the validation losses of the models.

In order to increases the performance on the given datasets, our study was shifted from CNN based model to capsule network.

## 5.2 Capsule Network

In capsule network, we go from basic capsule structure to concatenated 3 branch based capsule network.

1. **Basic Capsule network:** learning rate is kept at 0.00001 and number of routing is 3, output capsule network dimension=16.

Layer (type)	Output Shape	Param #
input_12 (InputLayer)	(None, 28, 28, 3)	0
conv1 (Conv2D)	(None, 20, 20, 256)	62464
primarycap_conv2d (Conv2D)	(None, 6, 6, 256)	5308672
primarycap_reshape (Reshape)	(None, 1152, 8)	0
primarycap_squash (Lambda)	(None, 1152, 8)	0
digitcaps (CapsuleLayer)	(None, 3, 16)	442368
capsnet (Length)	(None, 3)	0
Total params: 5,813,504		
Trainable params: 5,813,504		
Non-trainable params: 0		

Fig 5.8 Various layers in basic capsule network.

```

Epoch 1/20
100/100 [=====] - 1130s - loss: 0.4254 - acc: 0.4335 - val_loss: 0.2784 - val_acc: 0.4414
Epoch 2/20
100/100 [=====] - 7s - loss: 0.2757 - acc: 0.4902 - val_loss: 0.2664 - val_acc: 0.7371
Epoch 3/20
100/100 [=====] - 7s - loss: 0.2544 - acc: 0.6754 - val_loss: 0.2383 - val_acc: 0.5014
Epoch 4/20
100/100 [=====] - 7s - loss: 0.2104 - acc: 0.7320 - val_loss: 0.1835 - val_acc: 0.7643
Epoch 5/20
100/100 [=====] - 7s - loss: 0.1626 - acc: 0.7922 - val_loss: 0.1573 - val_acc: 0.8171
Epoch 6/20
100/100 [=====] - 7s - loss: 0.1335 - acc: 0.8377 - val_loss: 0.1398 - val_acc: 0.8186
Epoch 7/20
100/100 [=====] - 7s - loss: 0.1244 - acc: 0.8520 - val_loss: 0.1315 - val_acc: 0.8286
Epoch 8/20
100/100 [=====] - 7s - loss: 0.1112 - acc: 0.8690 - val_loss: 0.1294 - val_acc: 0.8271
Epoch 9/20
100/100 [=====] - 7s - loss: 0.1042 - acc: 0.8840 - val_loss: 0.1332 - val_acc: 0.8029
Epoch 10/20
100/100 [=====] - 7s - loss: 0.1069 - acc: 0.8675 - val_loss: 0.1246 - val_acc: 0.8214
Epoch 11/20
100/100 [=====] - 7s - loss: 0.1008 - acc: 0.8762 - val_loss: 0.1271 - val_acc: 0.8200
Epoch 12/20
100/100 [=====] - 7s - loss: 0.0923 - acc: 0.8942 - val_loss: 0.1325 - val_acc: 0.8043
Epoch 13/20
100/100 [=====] - 7s - loss: 0.0959 - acc: 0.8900 - val_loss: 0.1254 - val_acc: 0.8257
Epoch 14/20
100/100 [=====] - 7s - loss: 0.0892 - acc: 0.8975 - val_loss: 0.1302 - val_acc: 0.8300
Epoch 15/20
100/100 [=====] - 7s - loss: 0.0871 - acc: 0.9065 - val_loss: 0.1270 - val_acc: 0.8071
Epoch 16/20
100/100 [=====] - 7s - loss: 0.0871 - acc: 0.9010 - val_loss: 0.1320 - val_acc: 0.8129
Epoch 17/20
100/100 [=====] - 7s - loss: 0.0857 - acc: 0.9015 - val_loss: 0.1415 - val_acc: 0.8129
Epoch 18/20
100/100 [=====] - 7s - loss: 0.0828 - acc: 0.9062 - val_loss: 0.1338 - val_acc: 0.8143
Epoch 19/20
100/100 [=====] - 7s - loss: 0.0785 - acc: 0.9160 - val_loss: 0.1314 - val_acc: 0.8114
Epoch 20/20
100/100 [=====] - 7s - loss: 0.0804 - acc: 0.9097 - val_loss: 0.1367 - val_acc: 0.8100

```

Fig 5.9 Shows training and validation accuracy and loss for each epoch for Basic Capsule Network model.

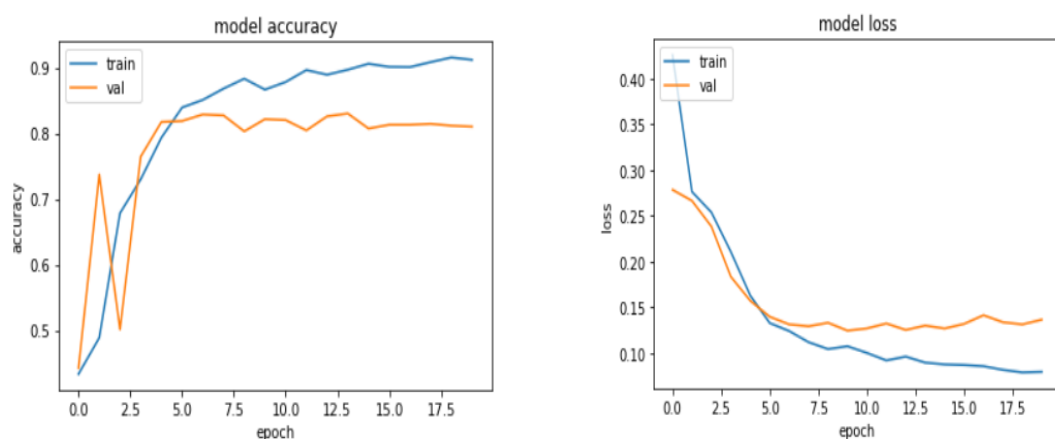


Fig 5.10 Graphical variation of accuracy and loss for Basic capsule network model.

In the above case validation accuracy is below the train accuracy and as far as validation losses is concern it is below till 5<sup>th</sup> epoch and then more than the train losses.

**Test accuracy =92.28 % and Test margin loss= 0.07379**

For a basic capsule network, it is good increment over the DenseNet model which show the accuracy of around 90 %.

**2. Capsule Network with 3 Concatenated CNN layers:** learning rate is kept at 0.00001 and number of routing is 3, output capsule network dimension=16 but it has 3 parallel convolutional layer concatenated at the end just before primary layer.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 28, 28, 3)	0	
conv1 (Conv2D)	(None, 20, 20, 256)	62464	input_1[0][0]
conv2 (Conv2D)	(None, 20, 20, 256)	62464	input_1[0][0]
conv3 (Conv2D)	(None, 20, 20, 256)	62464	input_1[0][0]
concatenate_1 (Concatenate)	(None, 20, 20, 768)	0	conv1[0][0] conv2[0][0] conv3[0][0]
primarycap_conv2d (Conv2D)	(None, 19, 19, 256)	786688	concatenate_1[0][0]
primarycap_reshape (Reshape)	(None, 11552, 8)	0	primarycap_conv2d[0][0]
primarycap_squash (Lambda)	(None, 11552, 8)	0	primarycap_reshape[0][0]
digitcaps (CapsuleLayer)	(None, 3, 16)	4435968	primarycap_squash[0][0]
capsnet (Length)	(None, 3)	0	digitcaps[0][0]
Total params: 5,410,048			
Trainable params: 5,410,048			
Non-trainable params: 0			

Fig 5.11 Various layers in capsule network with concatenated 3 CNN layers.

```

Epoch 1/25
140/140 [=====] - 1033s - loss: 1.2162 - acc: 0.5418 - val_loss: 0.2629 - val_acc: 0.6423
Epoch 2/25
140/140 [=====] - 23s - loss: 0.6705 - acc: 0.7651 - val_loss: 0.1192 - val_acc: 0.9014
Epoch 3/25
140/140 [=====] - 23s - loss: 0.4730 - acc: 0.8411 - val_loss: 0.0960 - val_acc: 0.8958
Epoch 4/25
140/140 [=====] - 24s - loss: 0.4110 - acc: 0.8594 - val_loss: 0.1115 - val_acc: 0.8648
Epoch 5/25
140/140 [=====] - 23s - loss: 0.3858 - acc: 0.8764 - val_loss: 0.0863 - val_acc: 0.8873
Epoch 6/25
140/140 [=====] - 24s - loss: 0.3692 - acc: 0.8803 - val_loss: 0.0889 - val_acc: 0.9155
Epoch 7/25
140/140 [=====] - 24s - loss: 0.3498 - acc: 0.8851 - val_loss: 0.0842 - val_acc: 0.8958
Epoch 8/25
140/140 [=====] - 23s - loss: 0.3319 - acc: 0.8998 - val_loss: 0.0815 - val_acc: 0.9014
Epoch 9/25
140/140 [=====] - 23s - loss: 0.3248 - acc: 0.8979 - val_loss: 0.0807 - val_acc: 0.9042
Epoch 10/25
140/140 [=====] - 24s - loss: 0.3137 - acc: 0.9016 - val_loss: 0.0806 - val_acc: 0.9155
Epoch 11/25
140/140 [=====] - 24s - loss: 0.3129 - acc: 0.8984 - val_loss: 0.0830 - val_acc: 0.9014
Epoch 12/25
140/140 [=====] - 23s - loss: 0.2903 - acc: 0.9112 - val_loss: 0.0710 - val_acc: 0.9127
Epoch 13/25
140/140 [=====] - 23s - loss: 0.2910 - acc: 0.9062 - val_loss: 0.0667 - val_acc: 0.9211
Epoch 14/25
140/140 [=====] - 24s - loss: 0.2836 - acc: 0.9137 - val_loss: 0.0724 - val_acc: 0.9211
Epoch 15/25
140/140 [=====] - 23s - loss: 0.2873 - acc: 0.9062 - val_loss: 0.0702 - val_acc: 0.9070
Epoch 16/25
140/140 [=====] - 23s - loss: 0.2726 - acc: 0.9098 - val_loss: 0.0747 - val_acc: 0.9239
Epoch 17/25
140/140 [=====] - 24s - loss: 0.2689 - acc: 0.9111 - val_loss: 0.0664 - val_acc: 0.9155
Epoch 18/25
140/140 [=====] - 24s - loss: 0.2638 - acc: 0.9159 - val_loss: 0.0566 - val_acc: 0.9352
Epoch 19/25
140/140 [=====] - 24s - loss: 0.2637 - acc: 0.9139 - val_loss: 0.0664 - val_acc: 0.9155
Epoch 20/25
140/140 [=====] - 24s - loss: 0.2559 - acc: 0.9182 - val_loss: 0.0746 - val_acc: 0.8986
Epoch 21/25
140/140 [=====] - 24s - loss: 0.2526 - acc: 0.9243 - val_loss: 0.0572 - val_acc: 0.9408
Epoch 22/25
140/140 [=====] - 23s - loss: 0.2447 - acc: 0.9282 - val_loss: 0.0611 - val_acc: 0.9239
Epoch 23/25
140/140 [=====] - 24s - loss: 0.2489 - acc: 0.9216 - val_loss: 0.0661 - val_acc: 0.9211
Epoch 24/25
140/140 [=====] - 24s - loss: 0.2429 - acc: 0.9252 - val_loss: 0.0562 - val_acc: 0.9408
Epoch 25/25
140/140 [=====] - 24s - loss: 0.2403 - acc: 0.9228 - val_loss: 0.0638 - val_acc: 0.9211

```

Fig 5.12 Shows training and validation accuracy and loss for each epoch for non-basic capsule network model.

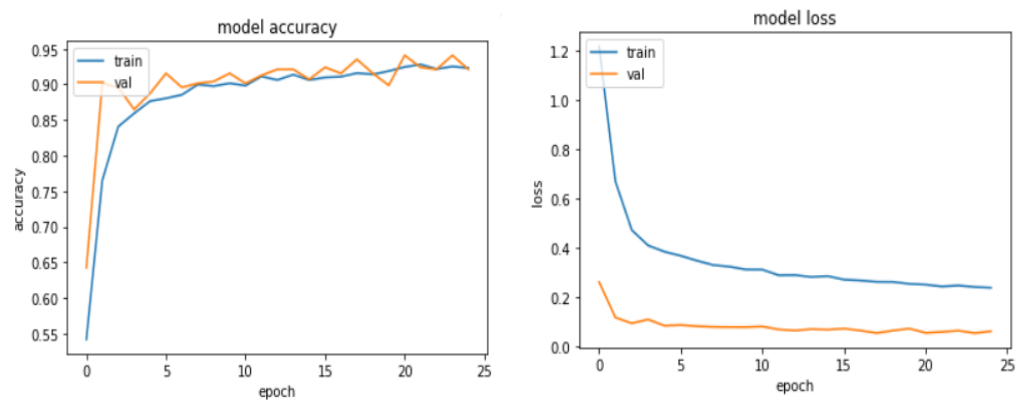


Fig 5.13 Graphical variation of accuracy and loss for non-basic capsule network model

In this Capsule network validation accuracy is more than train accuracy, and validation losses are lesser than train losses.

**Test accuracy = 94.13 % and Test margin loss = 0.0621.**

### **5.3 Future Research**

Capsule Network is a new network, it is originated at 2017 and it performs good in simple dataset but the complex dataset like ImageNet, CIFAR10 etc it is not been able to perform well so there are lot of scope is there in terms of improvement and medication in basic structure of the model.

As far as prediction of covid-19 patient using CXR are concern we have reached to the accuracy of around 94.13 % and by applying transfer learning to the current model we will get the more accuracy, transfer learning we means that pretrained the model on huge set of data and get some initial weights and then train its end layers on small data set then by doing so accuracy of the model definitely increases.

Not only this by applying image enhancement techniques such as histogram equalizer model will understand the images well as a result too accuracy increases.

## **Chapter: 6**

# **CONCLUSION**

- Capsule network of both the type showing more accuracy than pretrained DenseNet model.
- In 3 parallel CNN layered capsule network reaches up to the accuracy of 94.13 %, if we consider all the models, it is highest one.
- Number of trainable parameters in capsule network is less than as compared to pretrained CNN model, which reduces its chance of overfitting.
- In pretrained DenseNet model, if we use L2 regularization than accuracy is more as compared to dropout when applied to fully connected layers, means L2 regularization is more effective technique for this case than dropout.
- Capsule Network shows more accuracy, if transfer learning is applied and model will be pretrained on CXR with different number of classes.
- As RT-PCR is costly, time consuming and may cause infection or bleeding from mouth or nose then Deep-learning models consider to the best technique to detect Covid-19 patient using CXR.

# REFERENCE

- [1] Hoffmann, M., Kleine-Weber, H., Schroeder, S., Krüger, N., Herrler, T., Erichsen, S., ... & Müller, M. A. (2020). SARS-CoV-2 cell entry depends on ACE2 and TMPRSS2 and is blocked by a clinically proven protease inhibitor cell.
- [2] Sunita Nayak (2019) Image Classification using Transfer Learning in PyTorch, retrieved 20 March 2020. <https://www.learnopencv.com/imageclassification-using-transfer-learning-inpytorch/>
- [3] Wang, L., Wong, A.: Covid-net: A tailored deep convolutional neural network design for detection of covid-19 cases from chest radiography images (2020).
- [4] Pardamean, B., Cenggoro, T. W., Rahutomo, R., Budiarto, A., & Karuppiah, E. K. (2018). Transfer learning from chest X-ray pre-trained convolutional neural network for learning mammogram data. *Procedia Computer Science*, 135, 400-407.
- [5] Mangal, A., Kalia, S., Rajgopal, H., Rangarajan, K., Namboodiri, V., Banerjee, S., & Arora, C. (2020). CovidAID: COVID-19 Detection Using Chest X-Ray. *arXiv preprint arXiv:2004.09803*.
- [6] Kana, E. B. G., Kana, M. G. Z., Kana, A. F. D., & Kenfack, R. H. A. (2020). A web-based Diagnostic Tool for COVID-19 Using Machine Learning on Chest Radiographs (CXR). *medRxiv*.
- [7] Mishra, A. K., Das, S. K., Roy, P., & Bandyopadhyay, S. (2020). Identifying COVID19 from Chest CT Images: A Deep Convolutional Neural Networks Based Approach. *Journal of Healthcare Engineering*, 2020.
- [8] Chest X rays (pneumonia): <https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia>
- [9] Chowdhury, M. E., Rahman, T., Khandakar, A., Mazhar, R., Kadir, M. A., Mahbub, Z. B., ... & Islam, M. T. (2020). Can AI help in screening viral and COVID-19 pneumonia?. *IEEE Access*, 8, 132665-132676.
- [10] Rahman, T., Khandakar, A., Qiblawey, Y., Tahir, A., Kiranyaz, S., Kashem, S. B. A., ... & Chowdhury, M. E. (2021). Exploring the effect of image enhancement techniques on COVID-19 detection using chest X-ray images. *Computers in biology and medicine*, 132, 104319.



- [11] Covid-19 Radiography Database: <https://www.kaggle.com/tawsifurrahman/covid19-radiography-database>
- [12] Narin, A., Kaya, C., & Pamuk, Z. (2021). Automatic detection of coronavirus disease (covid- 19) using x-ray images and deep convolutional neural networks. *Pattern Analysis and Applications*, 1-14.
- [13] Abbas, A., Abdelsamea, M. M., & Gaber, M. M. (2021). Classification of COVID-19 in chest X-ray images using DeTraC deep convolutional neural network. *Applied Intelligence*, 51(2), 854-864.
- [14] Haghanifar, A., Majdabadi, M. M., Choi, Y., Deivalakshmi, S., & Ko, S. (2020). Covid-cxnet: Detecting covid-19 in frontal chest x-ray images using deep learning. *arXiv preprint arXiv:2006.13807*.
- [15] Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic routing between capsules. *arXiv preprint arXiv:1710.09829*.
- [16] Patrick, M. K., Adekoya, A. F., Mighty, A. A., & Edward, B. Y. (2019). Capsule networks—a survey. *Journal of King Saud University-Computer and Information Sciences*.
- [17] Mukhometzianov, R., & Carrillo, J. (2018). CapsNet comparative performance evaluation for image classification. *arXiv preprint arXiv:1805.11195*.
- [18] Lin, A., Li, J., & Ma, Z. (2018). On learning and learned representation with dynamic routing in capsule networks. *arXiv preprint arXiv:1810.04041*, 2(7).
- [19] Deliege, A., Cioppa, A., & Van Droogenbroeck, M. (2018). Hitnet: a neural network with capsules embedded in a hit-or-miss layer, extended with hybrid data augmentation and ghost capsules. *arXiv preprint arXiv:1806.06519*.
- [20] Vu, T., Nguyen, T. D., Nguyen, D. Q., & Phung, D. (2019, June). A capsule network-based embedding model for knowledge graph completion and search personalization. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (pp. 2180-2189).
- [21] Nair, P., Doshi, R., & Keselj, S. (2021). Pushing the limits of capsule networks. *arXiv preprint arXiv:2103.08074*.

- [22] Quan, H., Xu, X., Zheng, T., Li, Z., Zhao, M., & Cui, X. (2021). DenseCapsNet: Detection of COVID-19 from X-ray images using a capsule neural network. *Computers in Biology and Medicine*, 133, 104399.
- [23] Kim, J., Jang, S., Park, E., & Choi, S. (2020). Text classification using capsules. *Neurocomputing*, 376, 214-221.
- [24] Toraman, S., Alakus, T. B., & Turkoglu, I. (2020). Convolutional capsnet: A novel artificial neural network approach to detect COVID-19 disease from X-ray images using capsule networks. *Chaos, Solitons & Fractals*, 140, 110122.
- [25] Figure 1 covid-19 chest x rays: <https://github.com/agchung/Figure1-COVID-chestxray-dataset>
- [26] Covid chestxray dataset: <https://github.com/ieee8023/covid-chestxray-dataset>
- [27] ActualMed Covid-19 chest x rays dataset:  
<https://github.com/agchung/Actualmed-COVID-chestxray-dataset>