Table of Contents

Abstract

Image classification has become one of the main tasks in the field of computer vision technologies. State-of-the-art approaches in automated classification use deep convolutional neural networks (CNNs). However, these networks require a large number of training samples to generalize well. This project investigates the use of capsule networks (CapsNets) as an alternative to CNNs. We show that CapsNets significantly outperforms CNNs when the number of training samples is small. To increase the computational efficiency, we introduce a consistent dynamic routing mechanism that results in a speedup of CapsNet. Finally, we show that the original CapNets performs poorly on volumetric data. We created an efficient alternative, a transfer learning-based approach that yields lower reconstruction error and higher classification accuracy.

Acknowledgements

Primarily we would like to thank God for being able to complete this project with success. Then we would like to thank our supervisor Dr Nadeem Akhtar, whose valuable guidance has been the one that helped me patch this project and make it full proof success. His suggestions and his instructions had served as the major contribution in the project.

We would also like to thank our parents and our friends who have helped us with their valuable suggestions and guidance have been helpful in various phases of the project.

Mohiuddeen Khan

18COB035

Mohammad Anas

18COB104

01/12/2020

List of Figures

List of Tables

Chapter 1 : Introduction

This chapter includes the motivation and objectives of the project. It includes the default in the current traditional methods of classification and includes the solution to tackle it.

1.1 Motivation

Deep learning is no doubt the future of technology. Many scientists and researchers are investing huge amounts of money in the research and development in deep learning. The models generated from the technology can be used in various fields to make the current systems more intelligent and robust. In this project we are using a recently born technology in deep learning known as “Capsule Networks” which are considerably more improved than the standard deep learning algorithms. We made this project in order to contribute to medical science in order to improve accuracy in recognizing various lung diseases. We believe this project not only will help doctors but will open a new field for research and development.

1.2 Objectives and Scope

The objective of the project is to recognize pneumonia from chest x-rays using capsule networks and compare them with the traditional convolutional neural networks in terms of performance. The scope of the project is quite wide, traditional methods use CNN’s and transfer learning utmost for classification to cover up the deficiency of data in medical analysis. To make the models more accurate with less data, Geoffrey Hinton introduced the idea of capsule networks. This architecture performs way well compared to CNN’s when the training data provided is very less. Also, CNN has so many drawbacks from which they lose so much of essential information. Our project implements capsule networks in order to cover those drawbacks.

Chapter 2 : Literature Review

2.1 Convolutional Neural Networks

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

2.2 Transfer Learning

Many deep neural networks trained on images have a curious phenomenon in common: in early layers of the network, a deep learning model tries to learn a low level of features, like detecting edges, colours, variations of intensities, etc. Such kinds of features appear not to be specific to a particular dataset or a task because of no matter what type of image we are processing either for detecting a lion or cars. In both cases, we have to detect these low-level features. All these features occur regardless of the exact cost function or image dataset. Thus learning these features in one task of detecting lions can be used in other tasks like detecting humans. This is what transfer learning is. Nowadays, it is very hard to see people training whole convolutional neural network from scratch, and it is common to use a pre-trained model trained on a variety of images in a similar task, e.g models trained on ImageNet (1.2 million images with 1000 categories), and use features from them to solve a new task.[2]

2.2.1 VGG 16

VGG16 is a convolutional neural network[1] model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper “Very Deep Convolutional Networks for Large-Scale Image Recognition". The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. It was one of the famous model submitted to ILSVRC-2014. It makes the improvement over AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another. VGG16 was trained for weeks and was using NVIDIA Titan Black GPU.[3]
The architecture of VGG 16 is shown below:-

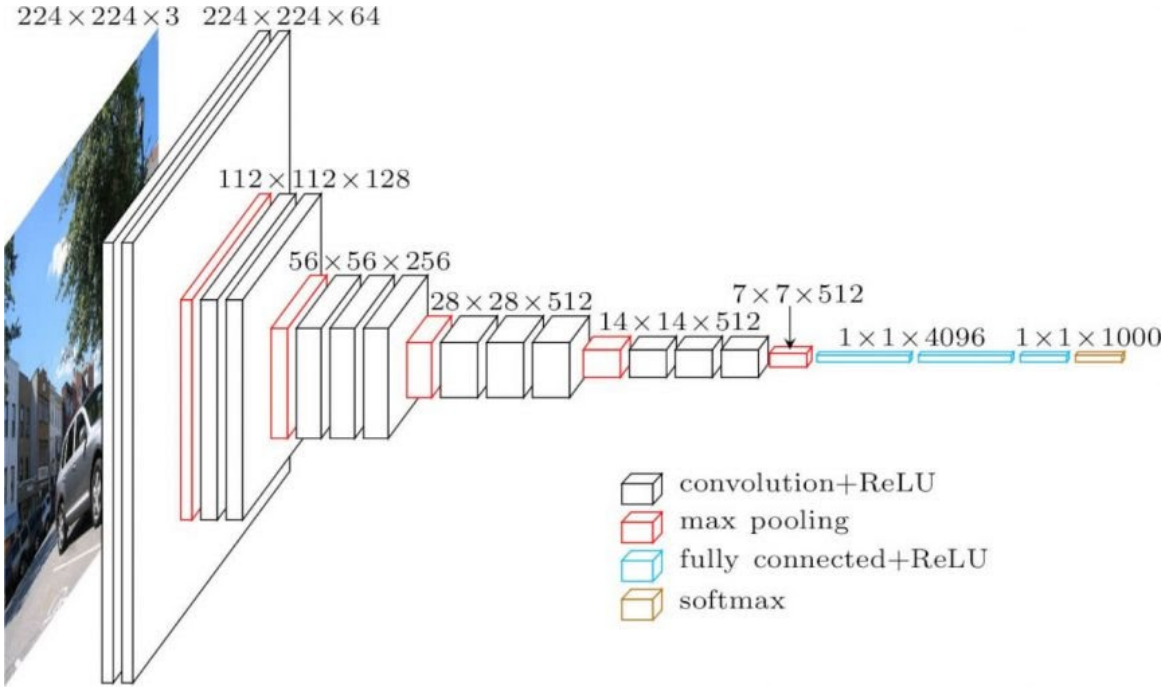

fig 2.1 VGG 16 Architecture

2.3 Capsule Networks

Capsule Neural Networks (Capsnets) are a type of ANN (Artificial Neural Network) whose major objective is to better replicate the biological neural network for better segmentation and recognition. The word capsule here represents a nested layer within a layer of capsule networks. Capsules determine the parameters of features in an object. During the process of identification of a face, capsules not only determine the presence or absence of facial features but also take into account the respective parameters in which facial features are organized. This means the system will only detect a face if the features detected by capsules are present in correct order. The job of capsules is to do inverse image rendering which means that we obtain the instantiation parameters such as object angle, scale, and position by analyzing the object according to the given object samples in its training set.[4]

The working of capsules are defined as:-
(i) First capsules proceed with the matrix multiplication of the input vectors with weight matrices which actually tells us briefly about the spatial relationship of some low-level features with high-level features.

(ii) Then the capsules decide their parent capsule. The selection of the parent capsule is done by using dynamic routing.

(iii) After making the decision about their parent capsules they proceed doing the sum of all the vectors that ultimately squashed between 0 and 1 while retaining their direction. Squashing is done by using the cosine distance as the measure of agreement and norm of the coordinate frame as the existence probability

Dynamic Routing

During the process of dynamic routing, the lower capsules send their data to the most suitable capsule. This capsule that receives the output of the lower level capsules is called the parent capsule. The parent capsules proceed with routing by

following the agreement and assignment mechanism i.e based on the dot product, expectation-maximization and using mixture models. The capsule having the largest dot product is chosen as the parent capsule. This dot product takes place between the prediction vector computed by the lower capsule layers and the weight matrix.

Architecture of Capsule Network:

Encoder – It takes the image input and displays the image as a vector that contains all the instantiation parameters needed to render the image. Encoder further encapsulates the :
(i) Convolutional layer – It detects basic features in the image.
(ii) PrimaryCaps layer – They produce combinations based on the basic features detected by the convolutional layer.
(iii) DigitCaps layer – This is the highest level capsule layer that contains all the instantiation parameters.

Decoder – Its job is to decode the 16-dimensional vector from DigitCap into an image. It recreates the output image without the loss of pixels. They force capsules to learn the features that are useful for reconstructing the image. The decoders further have three fully connected (dense) layers.

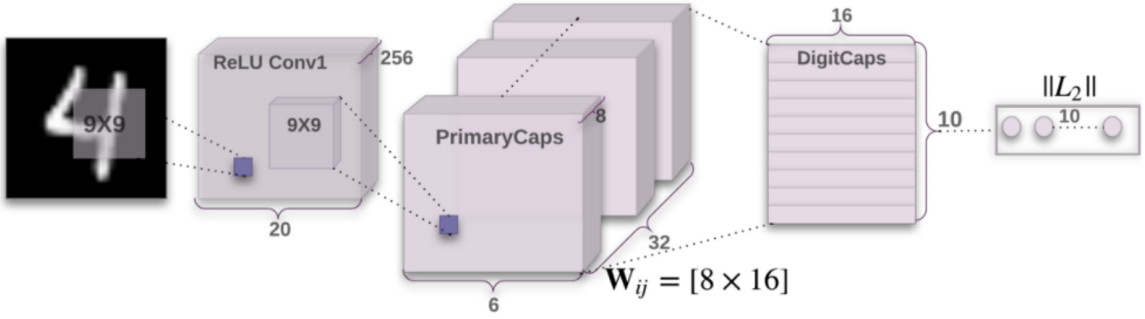

fig 2.2 Architecture of Capsule Network

2.4 CNN vs Capsule Networks

The main idea behind the introduction of Caps Nets was to reduce the training set size i, e usually very large in case of CNN (Convolutional Neural Network). CNN is also a type of neural network but in this network, output depends on the volume of the training set. In the case of CNN, the training and testing set size can be from 60M to 10M.CNN has a major drawback that they are not able to adjust to the viewpoint. If a particular image is inverted, CNN may not be able to identify the picture. Capsnet here exploits the fact that viewpoint changes have a nonlinear effect at the pixel level and linear impact at the object level. Caps Nets are able to adjust to viewpoint changes as they learn the linear manifold between an object and it's posed as a matrix of weights. Here, linear manifold refers to a linear relation between various object vectors in Euclidean space having n dimensions. CNNs use the max pool system while capsules maintain the weighted sum of features of the previous layer which is more suitable for detecting overlapping features. These features offered by CapsNets are really helpful in identifying overlapping digits in the handwriting. CNN identifies objects using too many layering systems that slow down the recognition process, however, Capsnets do not believe in too many layers instead they use nesting of layers in one layer. CapsNets are currently only tested for MNIST, (large database of handwritten digits that are used commonly for training various image processing systems) and they struggle on more complex data found in Imagenet. Also, capsules take a longer training time. Despite having such drawbacks, they possibly have a long way to go in the future.

Chapter 3 : Proposed Work

3.1 Preparing dataset

The dataset was taken from kaggle[5] for our experiment. The dataset is organized into 3 folders (train, test, val) and contains subfolders for each image category (Pneumonia/Normal). There are 5,863 X-Ray images (JPEG) and 2 categories (Pneumonia/Normal).

Chest X-ray images (anterior-posterior) were selected from retrospective cohorts of pediatric patients of one to five years old from Guangzhou Women and Children's Medical Center, Guangzhou. All chest X-ray imaging was performed as part of patients' routine clinical care.

For the analysis of chest x-ray images, all chest radiographs were initially screened for quality control by removing all low quality or unreadable scans. The diagnoses for the images were then graded by two expert physicians before being cleared for training the AI system. In order to account for any grading errors, the evaluation set was also checked by a third expert.
Following are 2 images of dataset:-

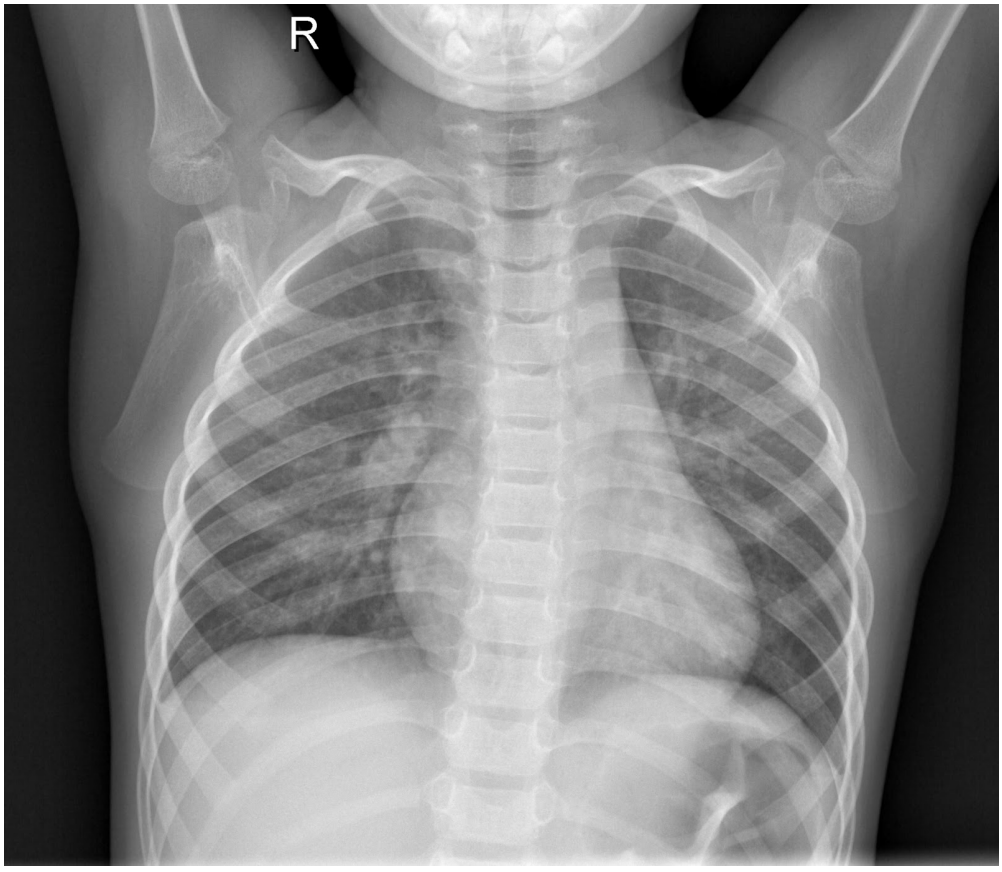

fig 3.1 Normal X Ray

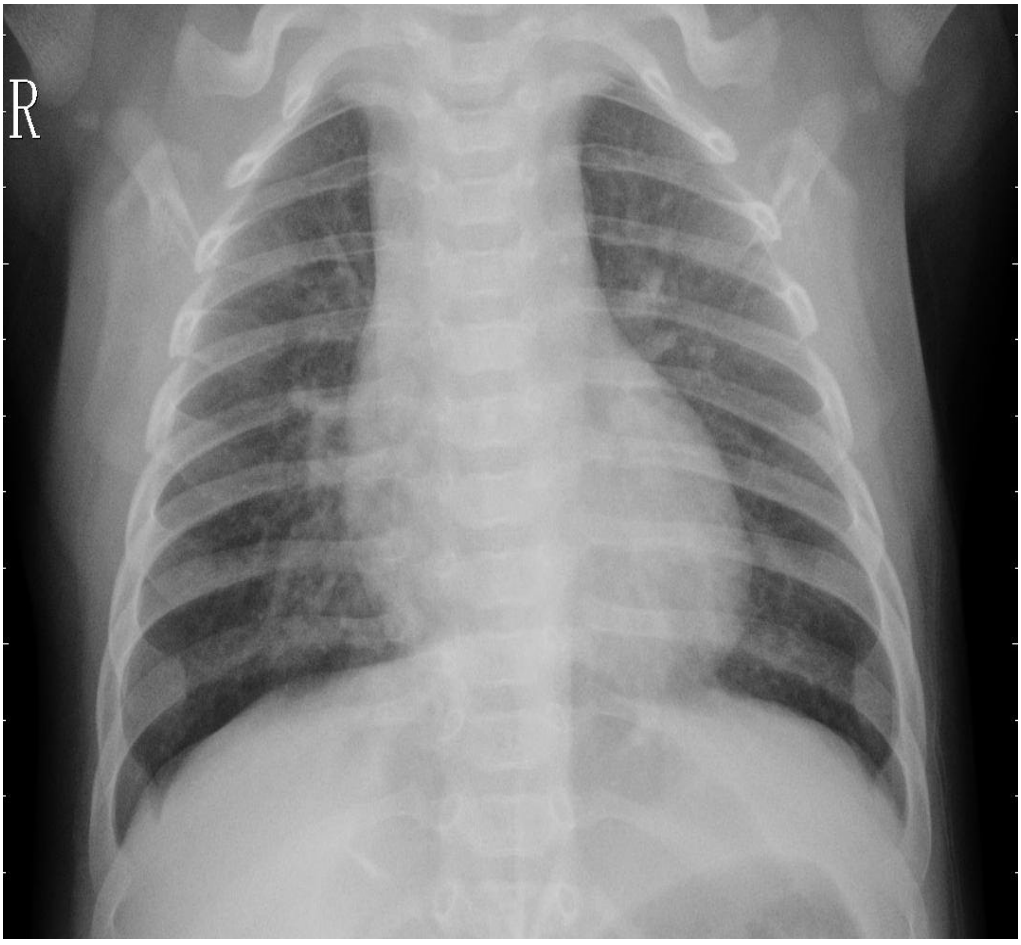

fig3.2. Pneumonia patient

The dataset is downloaded from kaggle in realtime on google colab to use it.

3.2 Creating VGG 16 CNN model

We implemented the VGG CNN model with two optimizers. The first one was computed using RMSprop as an optimizer and the second model used SGD(stochastic gradient descent) as an optimizer. Both the models used the following architecture:-

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 299, 299, 3)]	0
block1_conv1 (Conv2D)	(None, 299, 299, 64)	1792
block1_conv2 (Conv2D)	(None, 299, 299, 64)	36928
block1_pool (MaxPooling2D)	(None, 149, 149, 64)	0
block2_conv1 (Conv2D)	(None, 149, 149, 128)	73856

```
block2_conv2 (Conv2D)         (None, 149, 149, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)    (None, 74, 74, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)         (None, 74, 74, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)         (None, 74, 74, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)         (None, 74, 74, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)    (None, 37, 37, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)         (None, 37, 37, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)         (None, 37, 37, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)         (None, 37, 37, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)    (None, 18, 18, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)         (None, 18, 18, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)         (None, 18, 18, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)         (None, 18, 18, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)    (None, 9, 9, 512)         0
_________________________________________________________________
global_average_pooling2d (Gl  (None, 512)               0
_________________________________________________________________
dense (Dense)                 (None, 4096)              2101248
_________________________________________________________________
dropout (Dropout)             (None, 4096)              0
_________________________________________________________________
dense_1 (Dense)               (None, 2)                 8194
=================================================================
Total params: 16,824,130
Trainable params: 2,109,442
Non-trainable params: 14,714,688
_________________________________________________________________
```

3.3 Creating Capsule Network Model

Following architecture was used for capsule network with VGG16 with SGD optimizer:-

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 299, 299, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 299, 299, 64)      1792
_________________________________________________________________
```

```
block1_conv2 (Conv2D)         (None, 299, 299, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)    (None, 149, 149, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)         (None, 149, 149, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)         (None, 149, 149, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)    (None, 74, 74, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)         (None, 74, 74, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)         (None, 74, 74, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)         (None, 74, 74, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)    (None, 37, 37, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)         (None, 37, 37, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)         (None, 37, 37, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)         (None, 37, 37, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)    (None, 18, 18, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)         (None, 18, 18, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)         (None, 18, 18, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)         (None, 18, 18, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)    (None, 9, 9, 512)         0
_________________________________________________________________
conv2d_4 (Conv2D)             (None, 1, 1, 256)         10617088
_________________________________________________________________
reshape_4 (Reshape)           (None, 1, 256)            0
_________________________________________________________________
capsule_4 (Capsule)           (None, 2, 16)             8192
_________________________________________________________________
lambda_1 (Lambda)             (None, 2)                 0
=================================================================
Total params: 25,339,968
Trainable params: 17,704,704
Non-trainable params: 7,635,264
```

The first layer is the VGG 16 layer, followed by the max-pooling layer. The second layer is also a convolutional one, followed by average pooling. The last layer is reshaped to form the first Capsule layer. Consequently, two Capsule layers are embedded in our network to perform the routing by agreement process. The last Capsule layer contains the instantiation parameters of the two classes of positive and negative PNEUMONIA. The length of these two Capsules represents the probability of each class being present.

Chapter 4 : Implementation

4.1 Implementing VGG 16 CNN Model

In the first model we used the VGG 16 with its layers frozen and RMSprop as the optimizer. This was done to prevent the weights from being updated during training. This allows feature extraction from the pre-trained model.The intuition behind this is that if a model is trained on a large and general enough dataset, this model will effectively serve as a generic model of the visual world. You can then take advantage of these learned feature maps without having to start from scratch by training a large model on a large dataset.

In our second network, the architecture is almost the same. We have fine tuned our VGG 16 net and changed the optimizer to SGD. In the feature extraction, we were only training a few layers on top of an VGG 16 base model. The weights of the pre-trained network were not updated during training. One way to increase performance even further is to train (or "fine-tune") the weights of the top layers of the pre-trained model alongside the training of the classifier we added. The training process will force the weights to be tuned from generic feature maps to features associated specifically with the dataset.

4.2 Implementing Capsule Network Model

Capsule network model was implemented with SGD optimizer and ReLU activation with the architecture described above. The model was created with an early stopping mechanism having 10 epochs patience. It was on google colab with GPU runtime.
We used keras and tensorflow to realize all our functions and the Capsule Layer.

Chapter 5 : Results

The following is the models summary- after training:-

Model	Accuracy(%)	Loss
VGG16 + CNN + (optimizer=RMSprop)	74.84	0.461508
VGG16 + CNN + (optimizer=SGD)	84.13	0.357592
VGG16 + Capsule Network	88.94	0.08783

Table 1 Models Summary

Chapter 6 : Conclusions and Future Work

6.1 Conclusions

With the capsule network outperforming standard CNN in terms of accuracy and robustness we concluded that capsule networks which are recently devised can be used as an alternative for CNN's in deep learning tasks including recognition and classification. It is great to see such architecture doing great work in classification of pneumonia from x rays as compared to the CNN.

6.2 Future Work

Potential future projects include improving the Capsule Network architecture and thereby making the model more robust and accurate. The capsule network is a great invention which can replace the traditional methods of deep learning in future. The above approach can also be used in different medical fields including brain tumor detection, covid-19 detection[6] and many other fields where researchers cannot build accurate models due to deficiency of data.

References

[1] Simonyan, Karen & Zisserman, Andrew. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv 1409.1556.

[2] https://www.geeksforgeeks.org/ml-introduction-to-transfer-learning/

[3] https://neurohive.io/en/

[4] https://www.geeksforgeeks.org/capsule-neural-networks-ml/

[5] Kermany, Daniel; Zhang, Kang; Goldbaum, Michael (2018), "Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images for Classification", Mendeley Data, V2, doi: 10.17632/rscbjbr9sj.2

[6] Islam, Md Zabirul, Md Milon Islam, and Amanullah Asraf. "A combined deep CNN-LSTM network for the detection of novel coronavirus (COVID-19) using X-ray images." Informatics in Medicine Unlocked 20 (2020): 100412