An Approach to Classify Pneumonia Infection Data Using Deep Learning with GoogLeNet

Submitted By:

Tanjina Nasrin Mim
Student ID: 2015-3-60-017

Shahriar Zaman Sunve
Student ID: 2015-3-60-028

Maksuda Akhter Meem
Student ID: 2015-3-60-047

Supervised By:

Dr. Taskeed Jabid
Chairperson
Associate Professor
Department of Computer Science and Engineering
East West University

This research paper is presented in partial fulfillment as a requirement for the degree of Bachelor of Science in Computer Science and Engineering

Department of Computer Science and Engineering
East West University
Spring 2020

DECLARATION

We hereby declare that this research work **"An Approach to classify pneumonia infection data using Deep learning with GoogLeNet"** delivered in this report is a result of our own hard work with long hours of research work within the given time for this thesis under the supervision of Dr. Taskeed Jabid, Chairperson & Associate Professor, Department of Computer Science and Engineering, East West University. Also to note, no other organization or source has any contribution to this paper work other than the ones we have referred to within this paper. We also guarantee that no part of this work has been submitted elsewhere for the award of any degree or diploma from another institute.

……

Dr. Taskeed Jabid
Chairperson & Associate Professor,
Department of Computer Science and Engineering,
East West University

……

Tanjina Nasrin Mim
Student ID: 2015-3-60-017

……

Shahriar Zaman Sunve
Student ID: 2015-3-60-028

……

Maksuda Akhter Meem
Student ID: 2015-3-60-047

ACKNOWLEDGEMENT

First of all, we want to thank to Almighty Allah, the supreme power of the universe, for His immeasurable grace and profound kindness to enable us in successfully completing our research work with such ease because He gave us the ability, chance, and such cooperative supervisor.

We express our heartfelt gratitude to our thesis supervisor Dr. Taskeed Jabid, who had given us his time and patience throughout the entirety of the project, guiding us into the right direction for making this a successful work, while also being a friend to whom we could openly share our problems with. This research work was complete due to the hard work and time our supervisor has spent on each and every one of us to grow us into the people we are today.

Finally, we are also grateful to our family members who has shown extreme support throughout the entirety of our undergrad program, and our friends who had always supported our cause and helped us through every ups and downs during this whole program.

ABSTRACT

Medical sectors are today greatly relying on AI sector, and as for these, this paper represents an important machine learning implementation of pneumonia dataset. There are a lot of works nowadays about it, and in this paper, our task is to upgrade and create more accurate machine learning implementation. Foe this, we've implemented transfer learning using Google net to classify more accurately an image set of pneumonia infection. We have created a model of transfer learning, where we used inception V3 model to create a pre-trained model, and adding custom layers with it to prepare for our dataset classification to achieve highest accuracy for this pneumonia classification database. As it was an imbalance database, our goal was to achieve more precision and recall rather as well as its accuracy.

Table of Contents

1. Introduction

In image Classification, learning models of visual category from a small portion of training example is always been considering a great challenge in computer vision. Generally, it's possible for people to learn from few amounts of example from surroundings; while, for a machine, it might take some available prior knowledge about visual category and method to learn and perform similar, and also to build a structure for identify unlabeled data as well.

For unlabeled data, semi-supervised learning method is available to identify structure I unlabeled data and later which can be implemented to improve performance on a supervised task. But it doesn't exploit data from the previous supervised task. In transfer learning method, a representation from previous task is built to reuse further on a future related task.

In this paper, we represent a visual category learning method, which use transfer learning method in which, we trained a model on purpose to run on a second related task in order to achieve maximum accuracy. We use pneumonia X-ray image dataset in our problem, and our goal is to classify whether an image is infected or non-infected image. To achieve this, we trained our model and tuned for its best hypermeter and reduction learning rate. And later, use inceptionV3 model to tune entirely to achieve maximum accuracy.

2. Related Works

In this section we are going to be looking into some past works done on CNN to broaden our perspective on how to approach this research paper.

2.1 Classification of Image of Childhood Pneumonia Using Neural Network

Pneumonia is one of the most common disease and causes of children death worldwide. Convolutional Neural Network is used for the detection and classification of images for detection of pneumonia from the chest X-Ray of patient Dataset used for training and validation. 5863 XRay images are used in two categories: normal and pneumonia. For training and validation three steps are followed. The first one is the division and normalization of the images. The second one the training of the images is realized. For validation, cross validation of K-Fold is used. The average accuracy of the classification model is 95.3% in the tests against 92.8% of the work.[1]

2.2 Pneumonia Classification Using Deep Learning in Healthcare

The deep learning techniques are mainly part of the artificial neural networks capable of learning unsupervised from data that is unstructured and unlabeled. Pneumonia classification problem mainly consists of chest x-Rays dataset and classify the images with the help of various data augmentation techniques.

For testing the effectiveness of the model chest X-Ray image dataset is used. Keras is an open source neural network library in deep learning is used for this model. First raw dataset is collected then preprocess the raw data for analysis. After that Data Augmentation is used. To train the neural network cropping data, shifting, rotating, padding flipping are so on techniques are applied. At last CNN model is used-convolutional layer, pooling layer, fully connected layer. Sigmoid activation function is used as the classifier. The final result is training loss as 0.1378, training accuracy is 0.9436. Validation loss 0.1988 and accuracy 0.9289. [2]

2.3 Transfer Learning With Deep Convolutional Neural Network (CNN) For Pneumonia Detection Using Chest X-Ray

Pneumonia occurs in the lungs caused by bacterial or viral infection. Using digital X-Ray images convolutional neural network automatically detect bacterial and viral pneumonia. For transfer learning four different pre trained deep convolutional neural network: Alexnet[9], Resnet18[10], DenseNet201[11], SqueezeNet[12] were used. Three type of classifications:

Normal vs. pneumonia, bacterial vs. viral pneumonia and normal, bacterial and viral pneumonia have reported. The kaggle chest X-ray pneumonia database was used. For train evaluate and test different algorithms MATLAB used. The classification accuracy of normal and pneumonia images, bacterial and viral pneumonia images and normal, bacterial and viral pneumonia were

98%, 97% and 99%. The precision were 95%, 95%, 96% and the recall were 93.3%, 93.7% and 93.2%. [3]

2.4 Transfer Learning For Image Classification with Sparse Prototype Representations

A visual category algorithm was developed which explicitly takes advantage of unlabeled data. This method uses unlabeled data to define a prototype which based on computing kernel distances to a large set of unlabeled points. The experiment is on a news topic prediction task. The goal of the experiment is to predict whether an image belongs to a particular topic. The experiment show that when the training sets are small the sparse prototype improves the performance of news topic image classifiers. A dataset of 10382 images was created for the experiment. The mean equal error and standard error of mean was used. [4]

2.5 Diagnosing Covid-19 pneumonia from X-Ray and CT images using Deep Learning and Transfer Learning Approach

Novel Covid-19 was first reported in Wuhan city, china November 2019. The world health Organization (WHO) on January 2020 announced this pandemic as public health emergency. There are several techniques for covid-19 detection including the Nucleic Acid Test (NAT) and computed Tomography (CT) scan. To test, images of 5 different sources to from a dataset which contains 170 x-ray images and 361 CT images of covid-19. Two main algorithms: CNN architecture and Transfer learning algorithm, AlexNet used for this method. CNN model consists only one convolutional layer that constitute 16 filters followed by batch normalization, rectified linear unit (ReLU), fully connected layer, softmax and output. Modifying pre- trained Alex net is used because it transfer the learned weights, bias and features. When X-Ray is used 100% sensitivity achieved and 90% sensitivity achieved when CT images used to the suspected patients of covid-19. Pretrained Alexnet transfer learning algorithm accuracy is around 98%.[5]

2.6 A Novel Transfer Learning Based Approach for Pneumonia Detection in Chest X-Ray Images

The author use deep learning framework for detection of pneumonia using the concept of transfer learning. The methodology includes: chest x-ray image preprocessing, data augmentation, transfer learning using Alexnet, Densenet121, inception V3, resnet18, and GoogleNet. The accuracy is 96.4% with a recall of 99.62%. [6]

2.7 Optimized CNN-Based Diagnosis System to Detect the Pneumonia from Chest Radiographs

In [7] paper, the authors have implemented three models for pneumonia detection: VGG16, ResNet-50 and InceptionV3. The results were observed as InceptionV3 model test accuracy is 53% while Resnet-50 had an accuracy of 58% and the best performing architecture was VGG16 with the accuracy of 75%.

3. Background study

To develop this model, CNN is our base method, and for deep learning, we use transfer learning with inceptionv3 which also known as GoogLeNet.

3.1 Convolution Neural Network

Convolution neural networks, or ConvNet, is a part of deep neural network, were inspired by biological processes of vision. The visual data from eye reassemble the connectivity pattern between neurons in the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field.

A Convolution Neural Network generally consists an input layer, multiple hidden layers and an output layer. Hidden layers in CNN is used as multilayer perceptron.

In CNN, the input is a tensor with shape number of images multiply by image height, image width and image depth. Then after passing through a convolutional layer, the image extracted and mapping as a set of features with shape multiplied by number of images, feature map height, feature map width, feature map channels. A convolutional layer within a neural network has this following attribute:

- Convolutional kernels defined by a width and height, which is also known as hypermeters.
- The number of input channels and output channels of hypermeters and
- The depth of the input channels or Convolution filter should be equal to the depth or number of channels of the input feature map.

Convolutional layers receive the input and pass its result to another layer. This works similar to the response of a neuron in animals' visual cortex. Each convolutional neuron processes data only for its involved field. Though fully connected FFN can be used to acquainted features as well as classify information, it is not practical to use this structure to images. The necessary number of neurons would be very high, due to the massive input sizes connected with images, where each and every pixel is variable. For instance, a fully connected layer for a small image consisting size 200 x 200 has 40,000 weights for each neuron in the next layer. The convolution operation brings a solution to this problem by reducing the number of free parameters and allowing the network to go deeper with less parameters. For instance, with the image size, tiling regions of size consisting 5 x 5, each with the same shared weights, only requires 25 parameters. By using this resized weight over lesser parameters, the disappearing gradient and exploding gradient problems experienced during backpropagation in traditional neural networks are being avoided.

For training and testing in Deep learning CNN models, every input image will pass it through a series of convolution layers with filters, pooling, fully connected layers (FC), SoftMax function to classify an object with probabilistic values between 0 and 1.

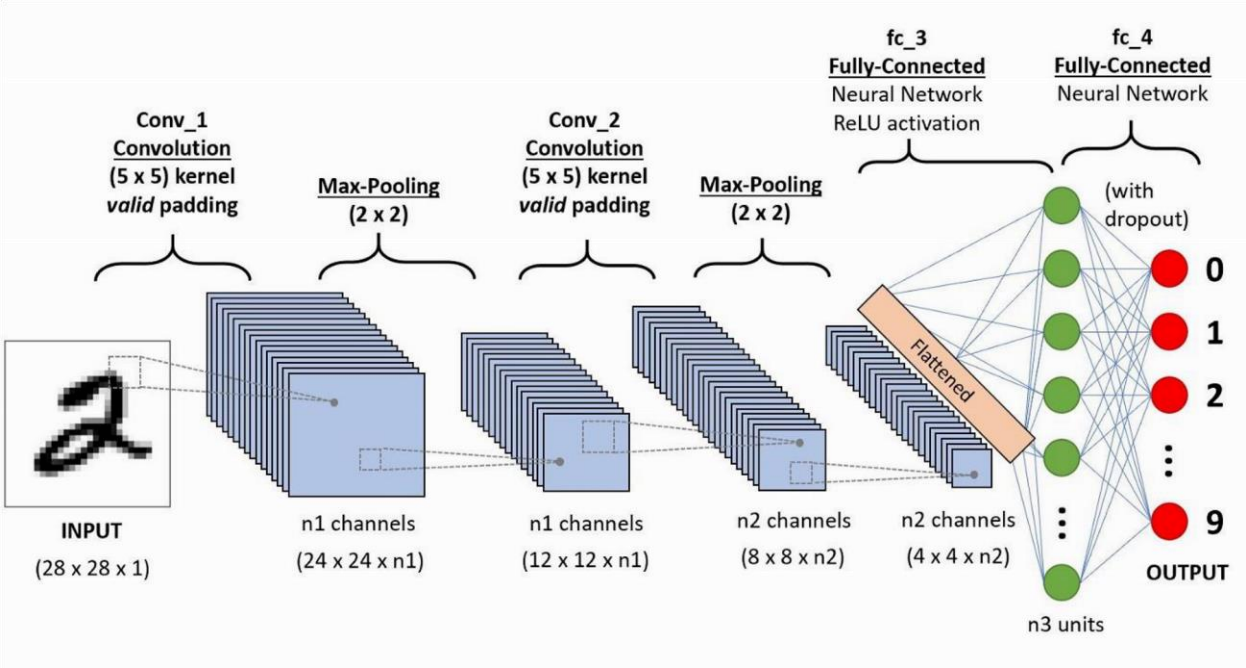

Figure 1 Convolution Neural Network

3.1.1 Convolution Layer

The core or kernel building block of CNN is convolutional layer, the layer's parameters included a set of learnable kernels or filters, which have a small receptive field, but it can be extended through the full depth of the input. On the forward pass, each filter is convolute across the height and weight of the input volume, calculating the dot product between the inputs of the filter and that creates a 2D activation map of that filter which makes the network learned about the activation of the filters when it detects some specific type of features at some defined position in the input.

Piling the activation maps for every filter along the depth dimension creates the full output volume of the convolution layer. Every input in the output volume can also be illustrate as an output of a neuron that grab a small area in the input and shares parameters with neurons in the same activation map.

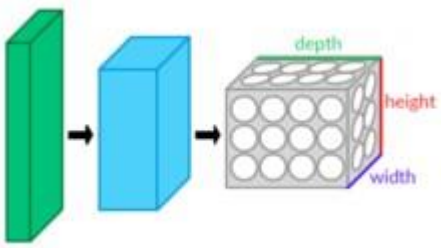

Figure 2: CNN layer in three dimensions

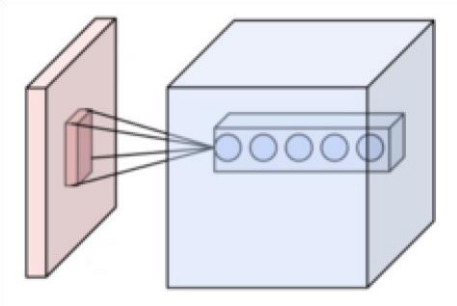

Figure 3: Convolution layer neuron arrangement

Convolution is the topmost layer to extract features from an input image. There are three hyper parameters control the size of the output volume: the depth, padding and stride. The output size of a convolutional layer is:

$$\frac{n+2p-f}{s} + 1 * \frac{n+2p-f}{s} + 1 \; []$$

Where p = Padding, s =Stride,

f= Number of filters, n = Image

width= Image height.

3.1.2 Stride

Stride monitor how depth columns around the spatial dimensions are allocated in convolution or the steps we are moving in every steps. When the stride set as 1, we transfer or slide the filters one pixel at a time. This makes heavily over-lapping retentive fields between the columns, and also to massive output volumes. When the stride set as 2, the filters jump 2 pixels at a time as they transfer or slide around. Similarly, for any integer, a stride of "S" causes the filter to be sliding by "S" units at a time per output. In practice, stride lengths of are rare. The receptive fields overlap less and the resulting output volume has smaller spatial dimensions when stride length is increased.

3.1.3 Pooling Layer

Pooling is used to down-sampling the image matrix to reduce the size and make the calculation more efficient. It's another stage of CNN, and indeed a very important one. A form of non-linear function is progressively downsizing the spatial size of the original representation and computation in the network. Pooling layers works on each feature map independently.

To implement pooling into CNN, there are various non-linear functions, among which *max pooling* is the most common. There are also several pooling functions like average pooling, sum pooling etc. Basically, to differentiate any object, we don't need the whole image of this object, rather than, a small portion can trigger our neurons to identify the object. The exact idea is use to build up pooling layers. Some feature is more important relatively rather than the whole feature. This is the idea behind the use of pooling in convolutional neural networks. The pooling layer use to progressively reduce the spatial size of the represented feature, to reduce the number of parameters and amount of computation in the network, and also help to control overfitting of data. Nowadays, it is common to periodically insert a pooling layer between successive convolutional layers in a CNN architecture. The pooling operation provides another form of translation invariance.

The pooling layer operates independently on every depth slice of the input and resizes it spatially. The most common form is a pooling layer with filters of size 2×2 applied with a stride of 2 down samples at every depth slice in the input by 2 along both width and height, discarding 75% of the activations:

Here are some popular pooling layers described:

3.1.4 Max Pooling

Max pooling is a sample-based discretization process. It reduces the computational cost by reducing the number of parameters. The output size of a max pooling layer is:

$$[\frac{nh-f}{s}+1 * \frac{nw-f}{s}+1]$$

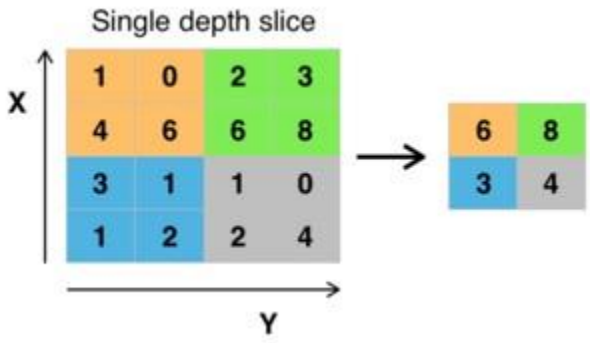

Figure 4 Max Pooling

3.1.5 Average pooling Layer

Average pooling layer reduce the variance and complexity in the data. The output size of an average pooling layer is :

$$[\frac{nh-f}{s}+1 * \frac{nw-f}{s}+1]$$

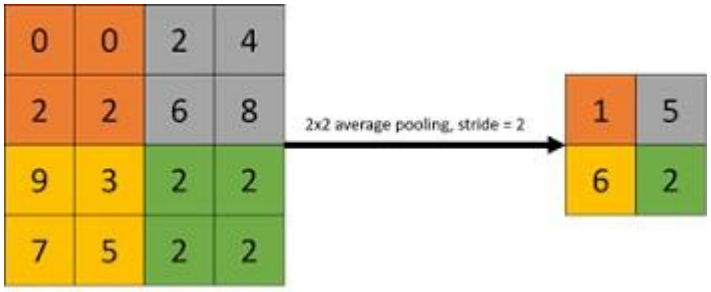

Figure 5 Average Pooling

3.1.6 Sum Pooling

Sum pooling works in a similar manner - by taking the sum of inputs instead of its maximum. Sum pooling (which is proportional to mean pooling) measures the mean value of existence of a pattern in a given region.

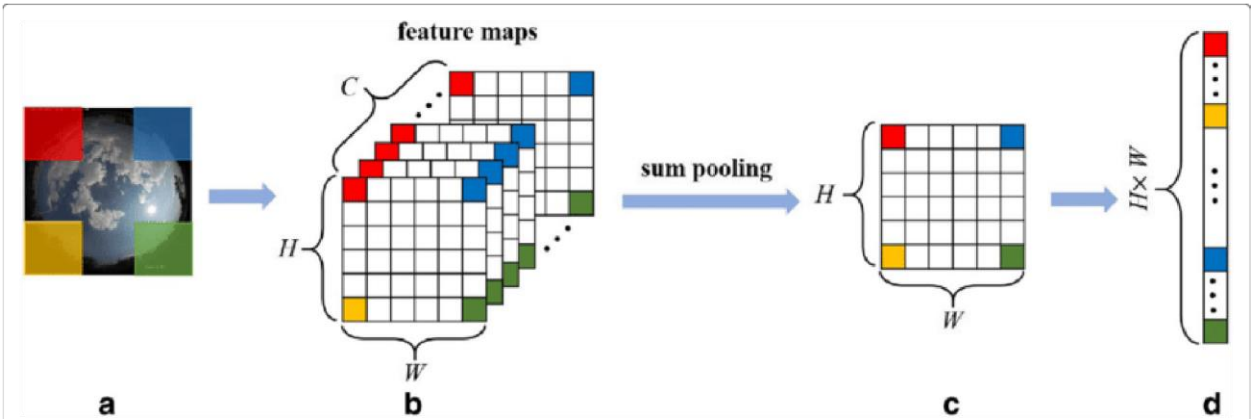

Figure 6 Sum Pooling

3.1.7 ReLu Layer

ReLu is the abbreviation of rectified linear unit, which applies the non-saturating activation function. It effectively removes negative values from an activation map by setting them to zero It increases the nonlinear property of the whole network without affecting the associated fields of the convolution layer. Generally, it represents as

$$(x) = \max(\ 0, x)$$

3.1.8 SoftMax Layer

SoftMax assigns decimal probabilities to each class in a multiclass problem. Those decimal probabilities must add up to 1.0.

3.1.9 Fully Connected Layer

The fully connected layer in the CNN represent the features vector for the input. It takes the results of the convolution process and use them to classify the image. Takes the output of the previous layers, “flattens” them and turns them into a single vector that can be an input for the next stage.

3.2 Transfer Learning

Transfer learning is a machine learning technique where a model trained on one task is repurposed on a second related task. It is a research problem that focus on storing knowledge gained while solving one problem and applying it to a different but related problem.

Transfer learning is related to problems such as multi-task learning and concept drift and is not exclusively an area of study of deep learning.

Transfer learning only works in deep learning if the model features learned from the first task are general.

In transfer learning, we first train a base network on a base dataset and task, and then we repurpose the learned features, or transfer them, to a second target network to be trained on a target dataset and task. This process will tend to work if the features are general, meaning suitable to both base and target tasks, instead of specific to the base task.

This form of transfer learning used in deep learning is called inductive transfer. This is where the scope of possible models is narrowed in a beneficial way by using fit on a different but related task.

There are two common approaches are as follows:

- Develop Model Approach
- Pre- trained Model Approach

3.2.1 Develop Model Approach

Select source Task: you must select a related predictive modeling problem with an abundance of data where there is some relationship in the input data, output data and concepts learned during the mapping from input to output data.

Develop source Model: Next, you must develop a skillful model for this first task. The model must be better than a naive model to ensure that some feature learning has been performed.

Reuse Model: The model fit on the source task can then be used as the starting point for a model on the second task of interest. This may involve using all of parts of the model.

Tune Model: Optionally, the model may need to be adapted or refined on the input-output pair data available for the task of interest.

3.2.2 Pre-trained Model Approach

Select source Model: A pre- trained source model is chosen from available models. Many research institutions release models on large and challenging datasets that may be included in the pool of candidate models from which to choose from.

Reuse Model: The pre-trained model can then be used as the starting point for a model on the second task of interest. This may involve using all parts of the model, depending on the modeling technique used.

Tune Model: Optionally, the model may need to be adapted or refined on the input-output pair data available for the task of interest.

3.2.1 Inception Model

There are many types of pre-trained models which developed over the time. AlexNet, Vgg16, ResNet, MobiteNet, Inception are some instance of this developed model, and among this, use preferred to use inception model for our classification[1]. It starts as a module of google net. The Inception network module was an important achievement in the development of CNN classifiers. Prior to its inception (pun intended), most popular CNNs just stacked convolution layers extended to much deeper and deeper, hoping to get better performance. The Inception network on the other hand, was complex and engineered deeply. It used a lot of calculative approach to push performance; both in terms of speed and accuracy. Its constant evolution lead to the creation of several versions of the network. The popular versions are as follows:

1. **Inception V1:**

It's all started to solve the large scale ImageNet challenge 2014.In the described challenge, the images used in there were very large variation and For instance, an image with a dog can be either of the following, as shown below. The area occupied by the dog is different in each image. Because of this extreme variation in the location of the information, choosing the right kernel size for the convolution operation becomes tough. A larger kernel is preferred for information that is distributed more globally, and a smaller kernel is preferred for information that is distributed more locally**.** Very deep networks are prone to overfitting. It also hard to pass gradient updates through the entire network.

Figure 7: From left: A dog occupying most of the image, a dog occupying a part of it, and a dog

Occupying very little space

To solve this, Why not have filters with multiple sizes operate on the same level? The network essentially would get a bit "wider" rather than "deeper". The authors designed the inception module to reflect the same.

The below image is the "naive" inception module. It performs convolution on an input, with 3 different sizes of filters (1x1, 3x3, 5x5). Additionally, max pooling is also performed. The outputs are concatenated and sent to the next inception module.

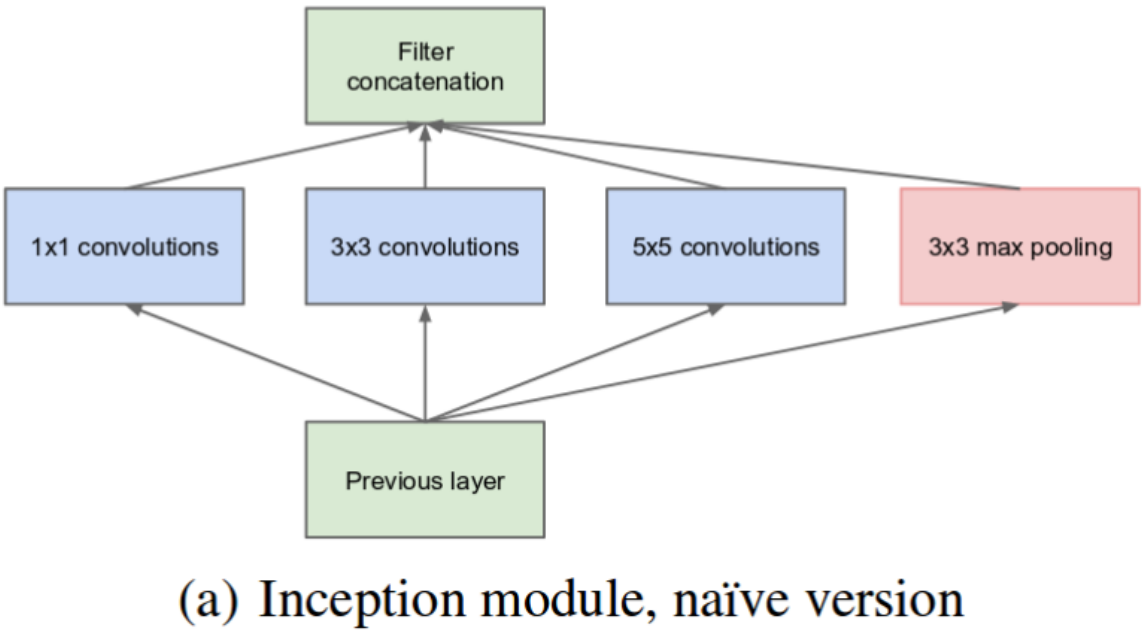

Figure 8: Inception Model v1

As stated before, deep neural networks are **computationally expensive**. To make it cheaper, the authors **limit** the number of **input channels** by adding an **extra 1x1 convolution** before the 3x3 and 5x5 convolutions. Though adding an extra operation may seem counterintuitive, 1x1 convolutions are far cheaper than 5x5 convolutions, and the reduced number of input channels also

help. Do note that however, the 1x1 convolution is introduced after the max pooling layer, rather than before.

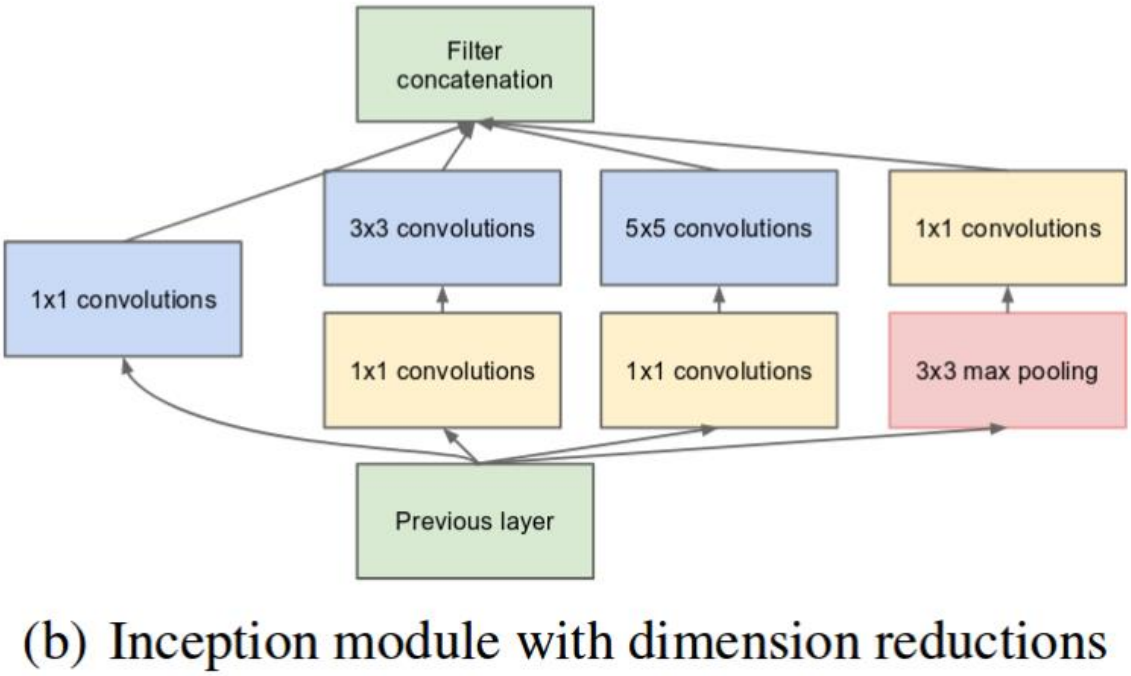

Figure 9: Inception module with dimension reduction

Using the dimension reduced inception module, a neural network architecture was built. This was popularly known as GoogLeNet (Inception v1). The architecture is shown below:

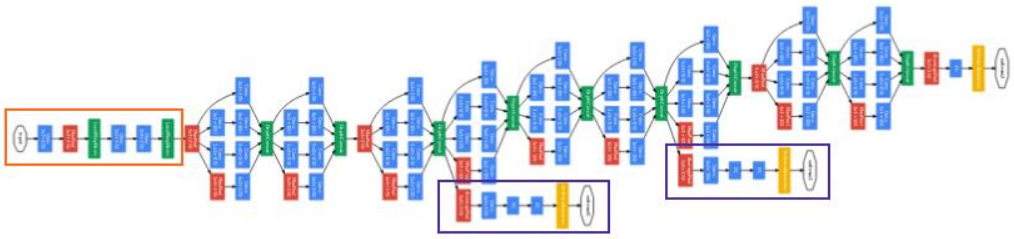

*Figure 10: GoogLeNet. The orange box is the **stem**, which has some preliminary convolutions. The purple boxes are **auxiliary classifiers**. The wide parts are the inception modules.*

GoogLeNet has 9 such inception modules stacked linearly. It is 22 layers deep (27, including the pooling layers). It uses global average pooling at the end of the last inception module.

Needless to say, it is a pretty deep classifier. As with any very deep network, it is subject to the vanishing gradient problem.

To prevent the middle part of the network from "dying out", the authors introduced two auxiliary classifiers (The purple boxes in the image). They essentially applied softmax to the outputs of two of the inception modules, and computed an auxiliary loss over the same labels. The total loss function is a weighted sum of the auxiliary loss and the real loss. Weight value used in the paper was 0.3 for each auxiliary loss.

2. Inception V2:

Inception v2 and Inception v3 were presented in the same paper. The authors proposed a number of upgrades which increased the accuracy and reduced the computational complexity.

Reduce representational bottleneck. The intuition was that, neural networks perform better when convolutions didn't alter the dimensions of the input drastically. Reducing the dimensions too much may cause loss of information, known as a "representational bottleneck". Using smart factorization methods, convolutions can be made more efficient in terms of computational complexity.

To solve this, Factorize 5x5 convolution to two 3x3 convolution operations to improve computational speed. Although this may seem counterintuitive, a 5x5 convolution is 2.78 times more expensive than a 3x3 convolution. So stacking two 3x3 convolutions infact leads to a boost in performance. This is illustrated in the below image.

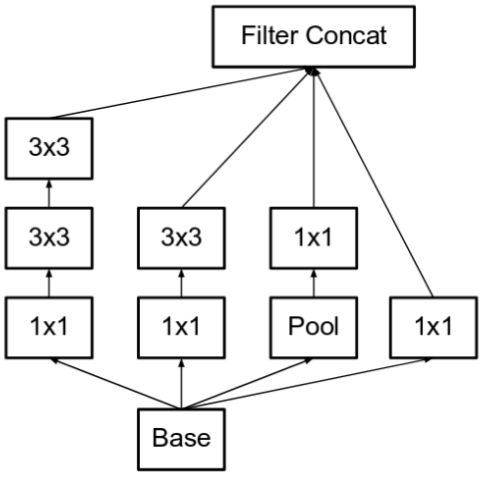

Figure 11: The left-most 5x5 convolution of the old inception module, is now represented as two 3x3 convolutions.

Moreover, they factorize convolutions of filter size n x n to a combination of 1 x n and nx1 convolutions. For example, a 3x3 convolution is equivalent to first performing a 1x3 convolution, and then performing a 3x1 convolution on its output. They found this method to be 33% more cheaply than the single 3x3 convolution. This is illustrated in the below image

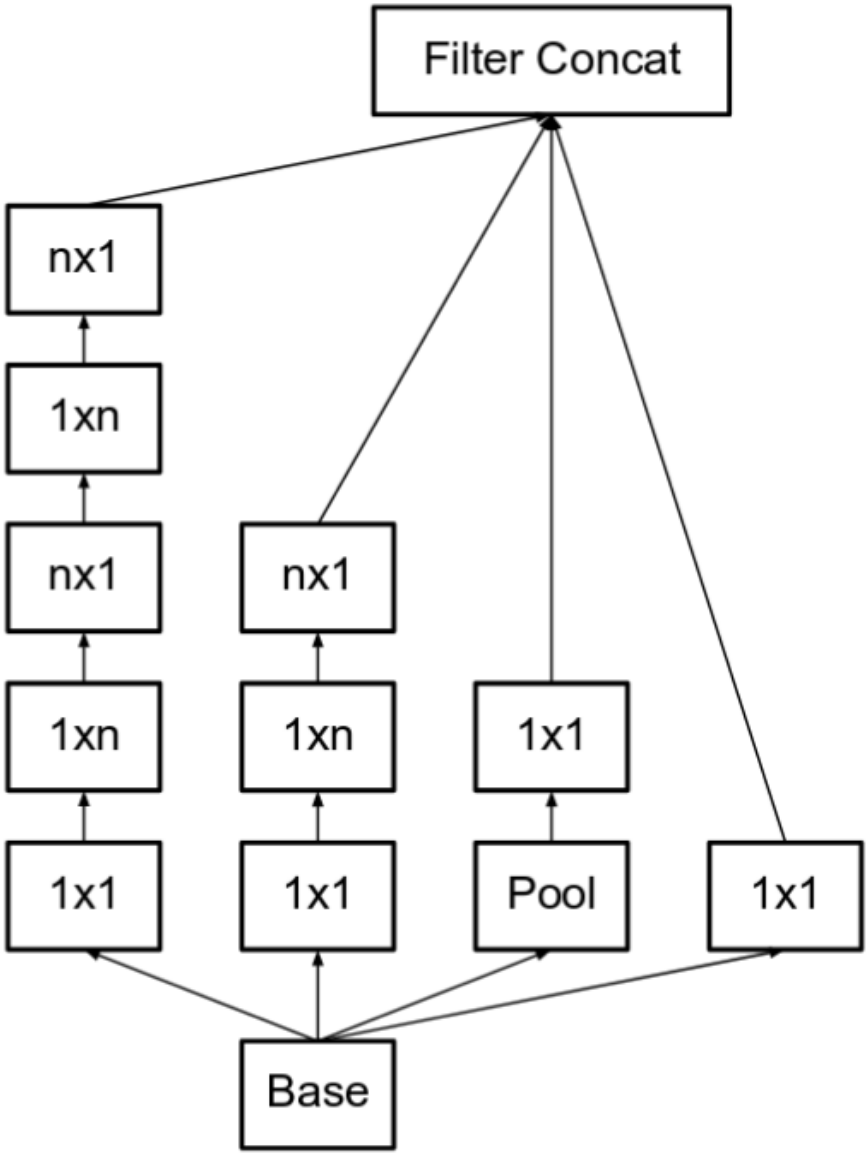

*Figure 12: Here, put **n=3** to obtain the equivalent of the previous image. The left-most 5x5 convolution can be represented as two 3x3 convolutions, which in turn are represented as 1x3 and 3x1 in series.*

2. Inception V3:

The authors noted that the auxiliary classifiers didn't contribute much until near the end of the training process, when accuracies were nearing saturation. They argued that they function as regularizes, especially if they have Batch Normalization or Dropout operations and possibilities to improve on the Inception v2 without drastically changing the modules were to be investigated.

The upgradation of inception V3 from V2, add some point and features like RMSProp Optimizer, Factorized 7x7 convolutions. BatchNorm in the Auxillary Classifiers and abel Smoothing (A type of regularizing component added to the loss formula that prevents the network from becoming too confident about a class. Prevents over fitting).

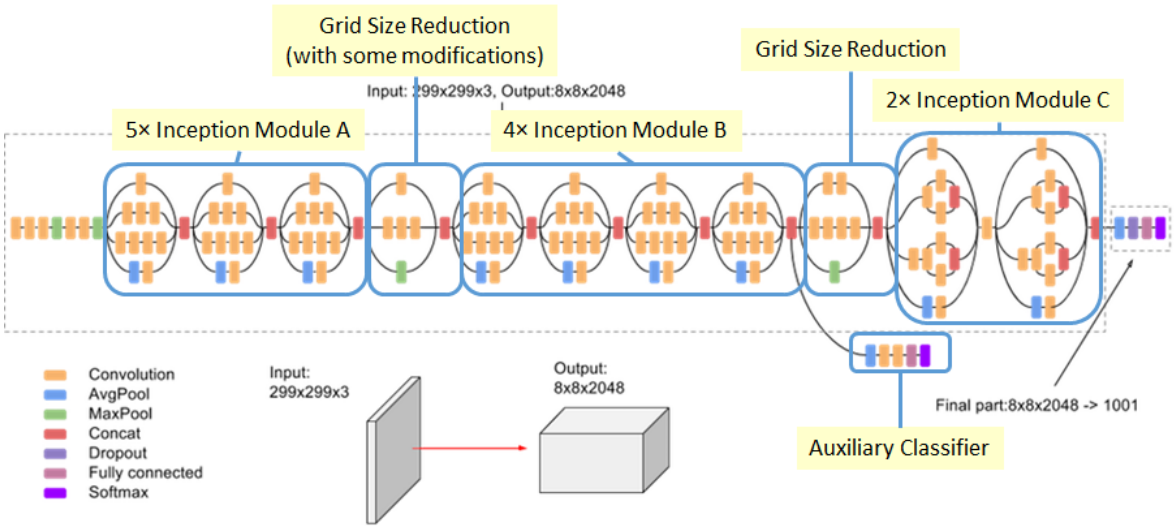

Figure 13: Structure of inception V3

4. Methodology

Here, we use Pneumonia dataset to classify normal and pneumonia infected image class. We developed a Convolution Neural Network to classify our image data. We then use transfer learning method pre-trained with ImageNet data to reuse in our model. Among various pre-trained model like VGGNet, ALEX Net, GoogLeNet, ResNet, MobileNet, Xception etc. we have use inceptionV3(GoogLeNet) in our model as pre-trained data.

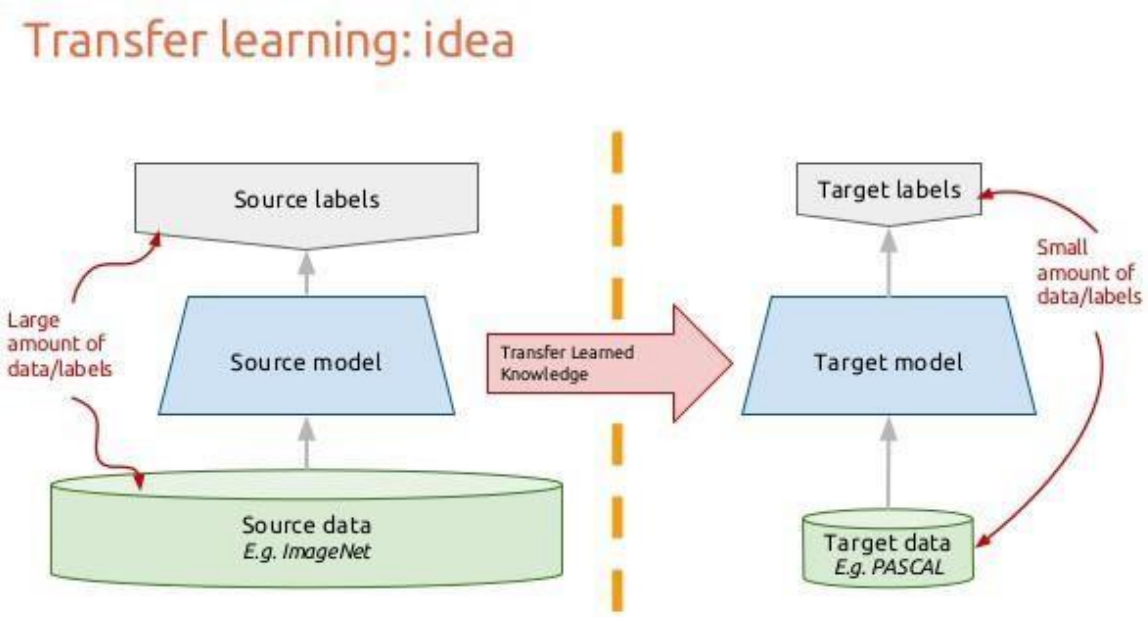

Figure 14: Representing Methodology

4.1 Dataset

We use Pneumonia dataset to classify our image data. This dataset[] have 5939 image of chest X-Ray image divided into two class; Train class and Test class, and each class have Normal image class dataset and Pneumonia infected image class dataset.

4.2 Pre-processing

We have pre-processed our dataset by taking them through some pre-processing steps to enhance the performance of classification. We have tried various pre-processing techniques over the time, but only stuck with the following methods as we find the highest accuracy only using them.

- Classify into categorical order: First mapping each into Train and Test class can categorized them into normal and pneumonia dataset

- Resize Image: Using skimage ,use resize the image into 150 x 150 x 3

- Greyscale: using OpenCV, we grayscale image as a step of preprocessing

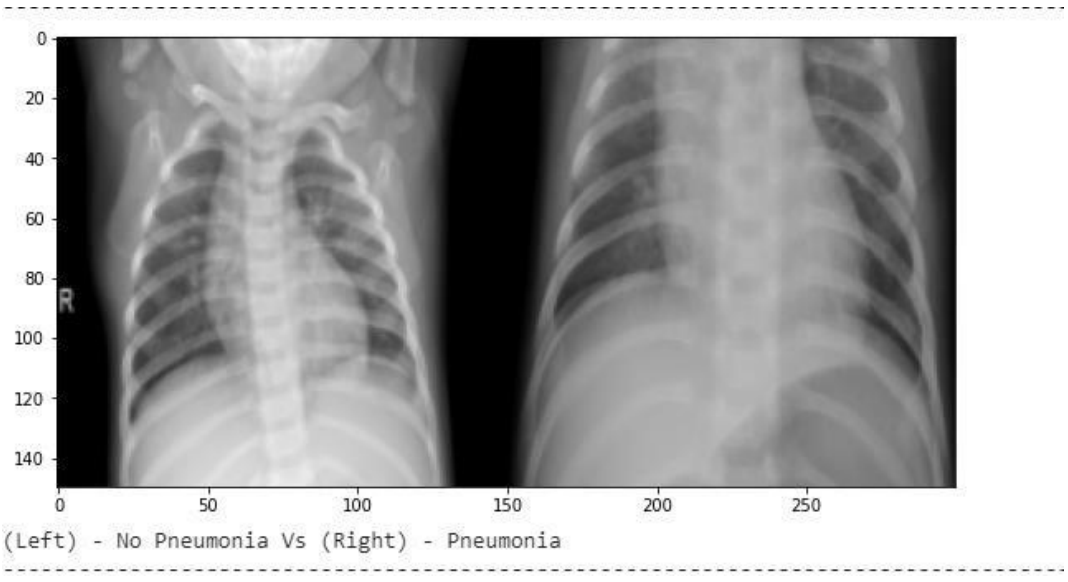

Figure 15: Data Set Sample (1)

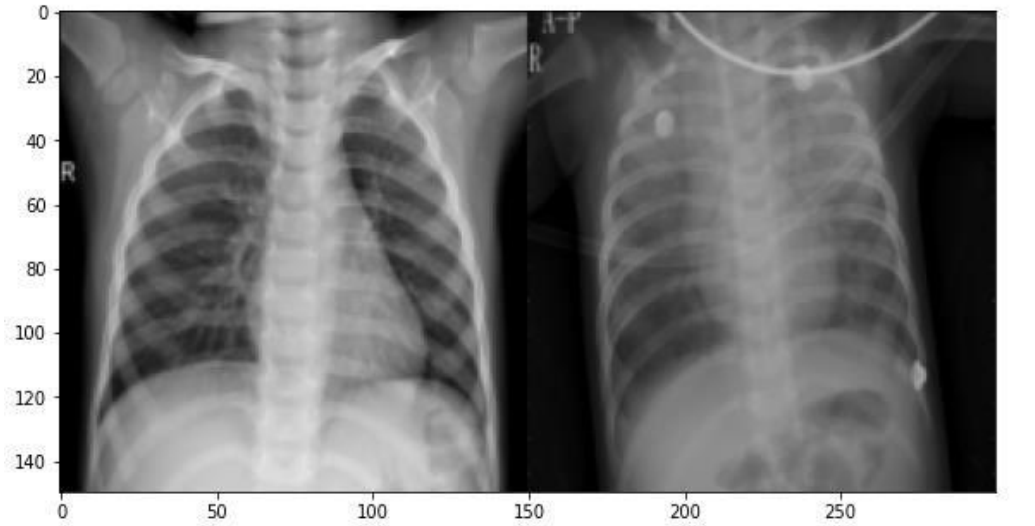

Figure 16: Data Set Sample (2)

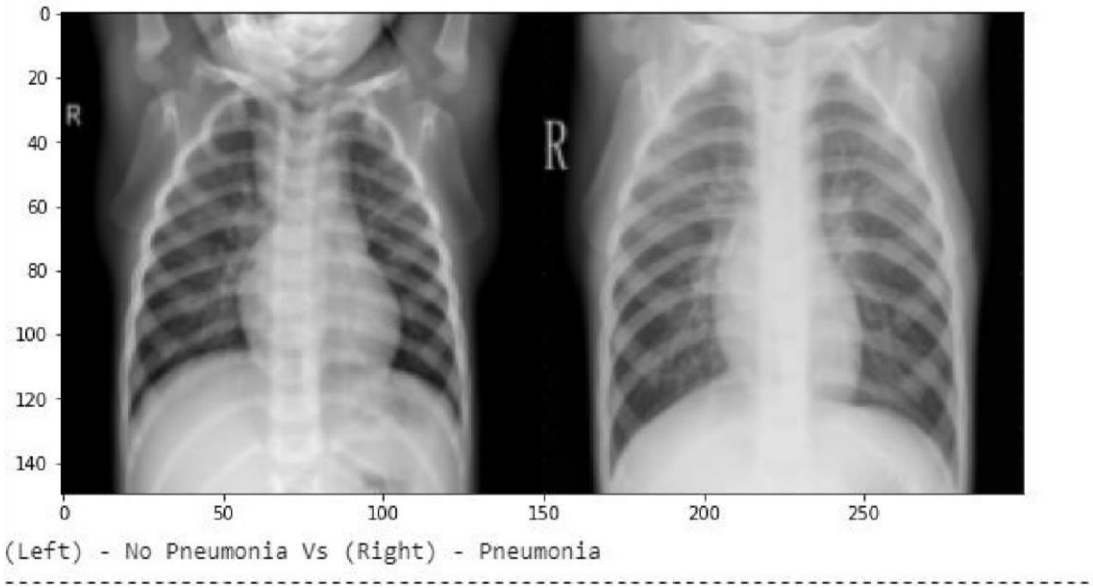

Figure 17: Data Set Sample (3)

4.3 Training Model and Implementation

After pre-processing and loading the dataset, first we create a pre-trained model.

Here we introduce inceptionV3, the successor of inceptionV1 and inceptionV2,a model created by is inceptionV3, with basic attributes set like weight set as ImageNet , as we are going to implement it on a different kind of dataset, we set include_top as 'False' and set input shape as (150 x 150 x 3). Now we run it through several fine tuning function and compile and run our dataset through this obtained model with a batch size of 64 and for 20 epoch.

The ImageNet dataset have 1000 class, with a large number of datasets. As we create a pe-trained model named as base model, we discard the last layer of inception v3 model, as, we have to train our own dataset and it is binary classification, we add some extra custom layers in it.

```
from keras.applications.inception_v3 import InceptionV3
base_model = InceptionV3(weights=None, include_top=False , input_shape=(150,150,3))
```

Figure 18: creating pre-trained model

In this layer, we add dropout and set it to 0.5, we've added average pooling 2D layers, a dense layer of 512 with relu activation, a batch normalization layer to normalized the values between 0 and 1 and set the prediction to dense 2 and use sigmoid as our activation function.

```
x = base_model.output
x = Dropout(0.5)(x)
x = GlobalAveragePooling2D()(x)
x = Dense(512, activation='relu')(x)
x = BatchNormalization()(x)
predictions = Dense(2, activation='sigmoid')(x)
```

Figure 19: Custom neural network

We do some fine tuning by freezing the top layers and set trainable as false, as the first layers are meant for the ImageNet class, we trained our data based on the last layers, running some epochs to view the result.

Then we compile the model using "Adam" optimizer, calculating loss function using "categorical cross entropy" and measure the data using accuracy metrics. We fit the data of our test set using train set for 20 epochs and in every epoch, we save the best result in a .h5 format file named mod.h5 to gain the highest accuracy.

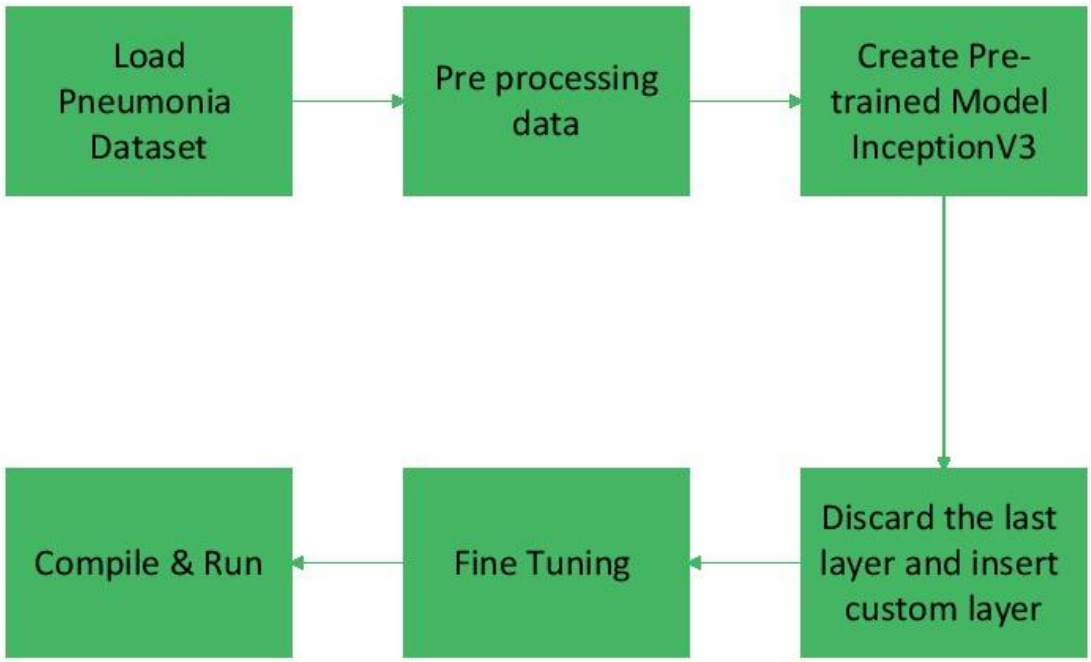

Figure 20: Basic model diagram

5. Experiment Results

5.1 Evaluation of Model

Our dataset was divided into two parts: training and testing and each part was containing both normal and pneumonia dataset. We used training dataset to train our model, based on which, we have tested our dataset. We mapped our training data labeled as normal and pneumonia. We preprocessed both training and testing dataset to preparing for evaluation and resized the image

size into 150 x 150 x 3. We evaluated the accuracy of the model by true positive (TP), true negative (TN), false positive (FP) and false negative (FN) after classification.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + TN}$$

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

5.2 Obtaining Result

The result obtaining from our developed model is given below:

Accuracy	90.1%
Precision	90.2%
Recall	94.7%
F1 Score	93.15%

Table: Obtained result

From this, we can clearly see that, the result from our model is very good, compared to the proposed models. Specially, the recall is about 95%, and in terms of medical sectors, as recall represents the ration of true positive vs the addition of true positive and false negative, the much greater will be the recall value, the more false negative result will be obtained, which is much better, as if it indicates more false negative result, the more it will be reliable to detects pneumonia infection better.

We can also obtain results as precision, recall, f1 score form the graph obtained from the program or algorithm from figure **5.1**, **5.2** and **5.3**

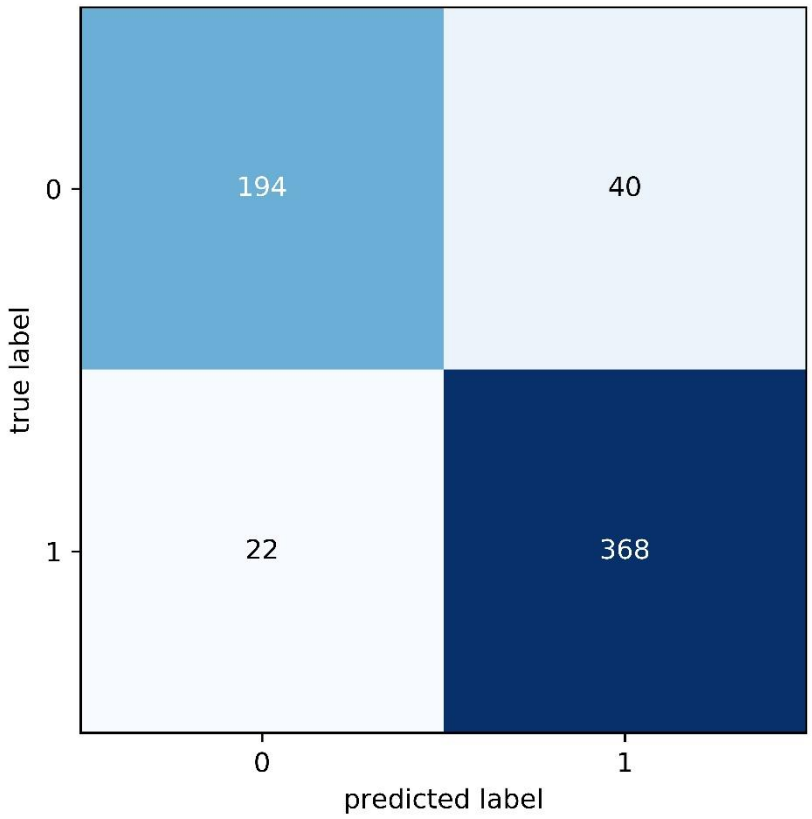

Figure 21: Confusion Matrix

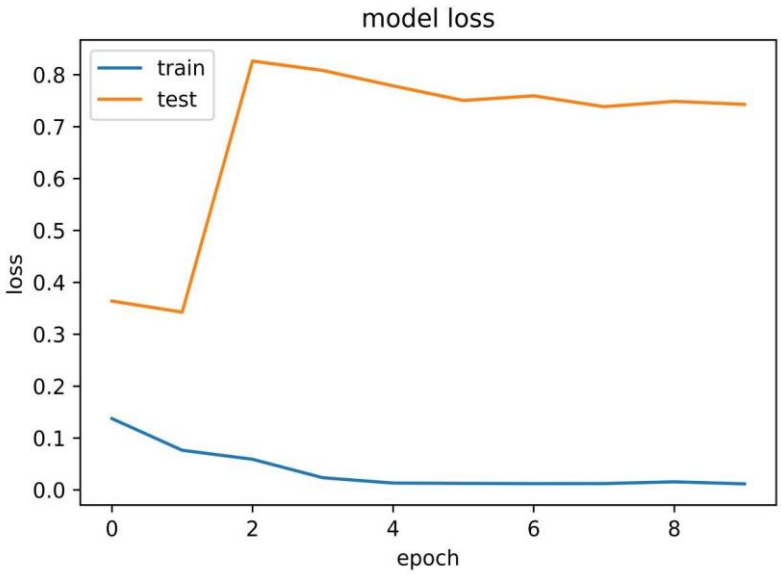

Figure 22: Loss Model

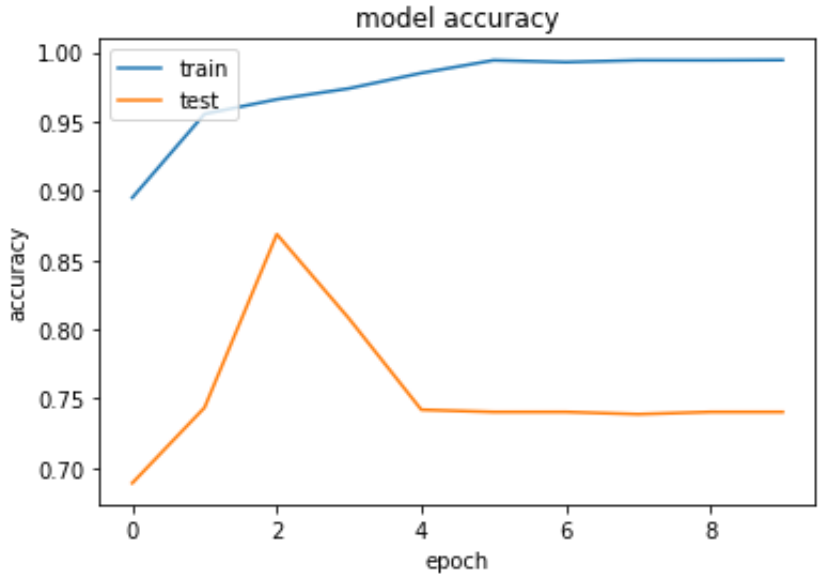

Figure 23: Accuracy model

6. Discussion

We have successfully implemented the model and get far better result from implementation only by CNN. The medical sector can utilize the best of machine learning and AI to improve primary accuracy to detect infection if everyone keep improving and upgrading the development era of machine learning

6.1 Future Works & Conclusion

Pneumonia can be caused by various types of infection, like bacterial infection, virus infection, fungus infection etc., our future work will be to make multiple classification about pneumonia infection with higher accuracy in different manner. Due to lag of resource, we were unable to create more variation and testing more to improve accuracy. Hope we are will pass all these limitations and push the limits as far as we could.

Bibliography

[1] A. A. Saraiva *et al.*, "Classification of images of childhood pneumonia using convolutional neural networks," *BIOIMAGING 2019 - 6th Int. Conf. Bioimaging, Proceedings; Part 12th Int. Jt. Conf. Biomed. Eng. Syst. Technol. BIOSTEC 2019*, pp. 112–119, 2019, doi: 10.5220/0007404301120119.

[2] G. Verma and S. Prakash, "Pneumonia Classification using Deep Learning in Healthcare," *Int. J. Innov. Technol. Explor. Eng.*, vol. 9, no. 4, pp. 1715–1723, 2020, doi: 10.35940/ijitee.d1599.029420.

[3] C. X-ray, T. Rahman, M. E. H. Chowdhury, and A. Khandakar, "applied sciences Transfer Learning with Deep Convolutional Neural Network (CNN) for Pneumonia Detection Using."

[4] O. Stephen, M. Sain, U. J. Maduh, and D. U. Jeong, "An Efficient Deep Learning Approach to Pneumonia Classification in Healthcare," *J. Healthc. Eng.*, vol. 2019, 2019, doi: 10.1155/2019/4180949.

[5] H. S. Maghdid, A. T. Asaad, K. Z. Ghafoor, A. S. Sadiq, and M. K. Khan, "Diagnosing COVID-19 Pneumonia from X-Ray and CT Images using Deep Learning and Transfer Learning Algorithms," pp. 1–8, 2020, [Online]. Available: http://arxiv.org/abs/2004.00038.

[6] V. Chouhan *et al.*, "A novel transfer learning based approach for pneumonia detection in chest X-ray images," *Appl. Sci.*, vol. 10, no. 2, 2020, doi: 10.3390/app10020559.

[7] S. Engineering, "Optimized CNN-based Diagnosis System to Detect the Pneumonia from Chest Radiographs," pp. 2405–2412, 2019.

[8] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z., 2020. *Rethinking The Inception Architecture For Computer Vision*. [Online] arXiv.org. doi:https://arxiv.org/abs/1512.00567

[9] Simonyan, K. and Zisserman, A., 2014. *Very Deep Convolutional Networks For Large-Scale Image Recognition*, doi: https://arxiv.org/abs/1409.1556.

[10] He, K., Zhang, X., Ren, S. and Sun, J., 2020. *Deep Residual Learning For Image Recognition*. , doi: https://arxiv.org/abs/1512.03385

[11] He, K., Zhang, X., Ren, S. and Sun, J., 2020. *Deep Residual Learning For Image Recognition*., doi: https://arxiv.org/abs/1512.03385

[12] He, K., Zhang, X., Ren, S. and Sun, J., 2020. *Deep Residual Learning For Image Recognition*. , doi: https://arxiv.org/abs/1512.03385