# Lung Opacity Detection Using Transfer Learning

**M.N.Satish Kumar[1],G.Nikhila Sai Vijaya Sri[2],D.Karthik[3],B.Bhuvana Sri[4],D. Swarupa Rani[5]**

## Abstract

In the present day medical scenario, lung related diseases are growing in tremendous scale. There is a large leap of death rates due to lung related diseases in recent years. Since covid and other viruses are seriously causing a large variety of lung diseases including pneumonia, lung opacity, and other nostalgic diseases. Due to large amount of people visiting the hospitals everyday it becomes quite tough to evaluate reports of every patient manually. Thus, it requires automation of report analysis.

With the advancement in artificial intelligence, it is now possible to analyse the scan of the patient and to produce appropriate result. This automation not only help the management but also helps the patient by reducing the costs. With the help of this project, we will introduce a platform where the user can submit their scan or lung X-ray and can get the appropriate result.

In this project we are concentrating on the lung related disease called lung opacity. It is a disease where the lung of patient is filled with pus leading to inconvenience in breathing and in few severe cases it even leads to death. In order to help the hospital managements in automation and patients in early identification of diseases we are proposing a brilliant model that can tackle this problem and provide best automation solution.

**KeyWords:**Opacity, Transfer, Learning.

**DOI Number: 10.48047/NQ.2022.20.16.NQ880301** **NeuroQuantology2022; 20(16):2917-2923**



## Introduction

The study of lung disease and its fragmentation has emerged as one of the most exciting fields of study in recent years. Due to the use of a range of medical imaging techniques in hospitals and diagnostic institutions, the size of medical imaging[9] databases is rapidly growing to track diseases in hospitals[15]. Despite the fact that this topic has been thoroughly investigated, the field is still challenging and confusing. According to the literature, there are numerous ways to categorise medical imaging. An important drawback of traditional methods [2] is the semantic gap between the high-quality semantic information that a person receives and the low-level visual information gathered by imaging instruments. In response, a novel technique known as the Deep convolutional neural network was created.[11].

In this paper, weare going to study the impact of transfer learning (VGG16) on detecting lung opacity over the traditional Convolutional neural network.

In this paper we will apply the traditional CNN as well as pre-determined weighted vgg16[3] algorithm on the lung opacity dataset and evaluate their performance in detection of lung opacity disease.

## Dataset

The dataset has been derived from multiple sources including from Kaggle as well as multiple other websites. The data contains 5808 disease-effected training images and 3872 disease-uneffected training images. The dataset also contains a good number of testing images as well that includes 194 disease-effected image x-ray scans and 144 non-effected scans. This dataset contains mainly 2 kinds of images that are lung opacity effected images and lung opacity-non-effected images.

**Corresponding author:** M.N.Satish Kumar
**Address:** [1]Assistant Professor,Department of CSE, SRGEC, Gudlavalleru,[2,3,4,5]Undergraduate Student, Department of CSE, SRGEC, Gudlavalleru

**Figure 1: Lung opacity effected image**

**Figure 2: Lung opacity not effected image**

## Training

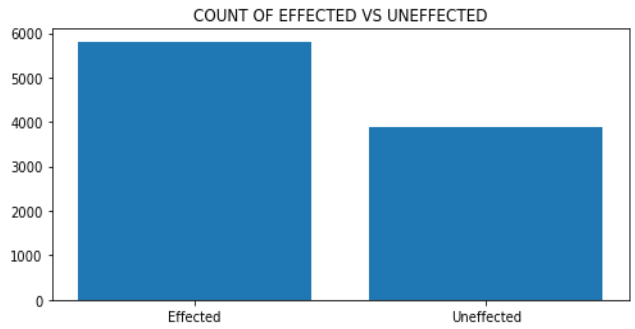



**Figure 3:Count of effected vs unaffected**

## Testing

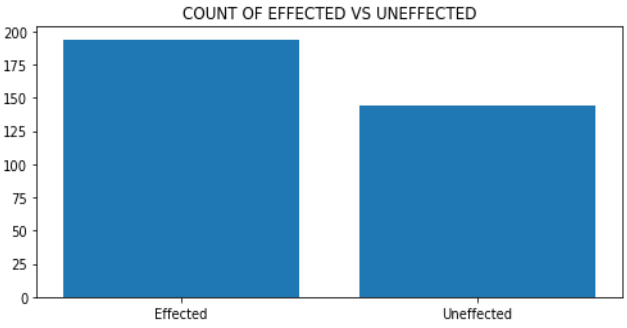



**Figure 4:Count of effected vs uneffected**

## Algorithms

Here we are mainly comparing the performance of 2 algorithms that is traditional CNN and VGG16. Although the working of these algorithms is somewhat the same from the core there is a large growth in accuracy for VGG16 compared to CNN.[6] At first, let us try to understand the internals of these algorithms that is

Convolution layer
Max-pooling layer
Flatten layer
Dense layer

## Convolution Layer

The convolution layer is the heart of the entire process. In this, we will apply several filters to extract features from an image which is also known as the feature extraction phase. In this phase, we will use a concept called filters[1].

## Filter

Before talking about filters, we first must understandthat ‘‘an image in deep learning is nothing but a matrix.’’ That is if the image size is 224X224 then there will be 224x224 values in a matrix that is in total there are224X224(50176) like below.

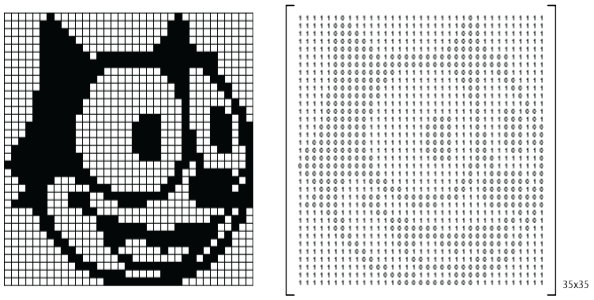



**Figure 5:Image to pixel conversion**

Now let’s see what is filter in simple terms ‘‘Filter is nothing but another small matrix which is used to manipulate the image to determine key features in the image.’’
There will be several filters for each convolution layer which will be applied to the original image to determine desired features.

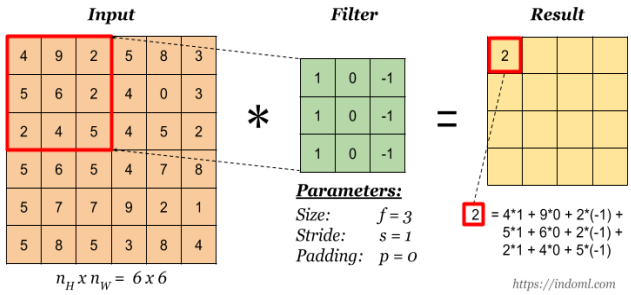



**Figure 6:Filter application**

This is how the filter works on the image. Here in the above example, the image is of size 6X6 and filter is of size 3X3 and the output is of size 4X4.
The examples we have seen are 2 Dimensional for

3d
The matrix would be like bellow

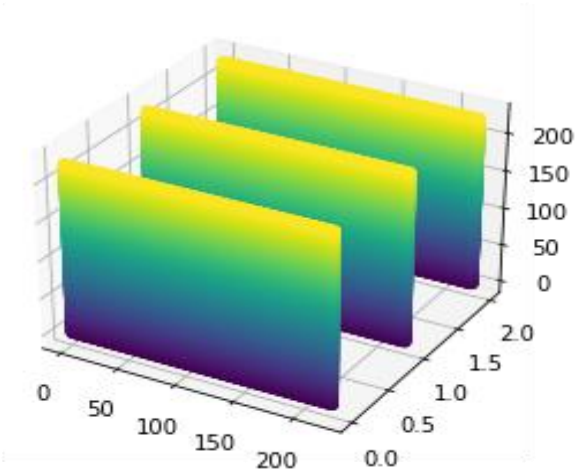



**Figure 7:3D array**

While talking about filters we must consider two things
Stride
padding

## Stride

By "stride," we refer to the distance the filtertravels between two positions.

If stride is 1

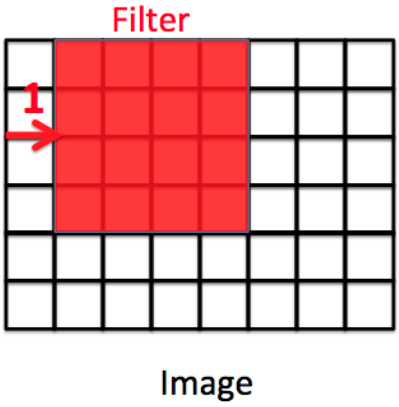



**Figure 8:Filter with stride 1**

If stride is 2

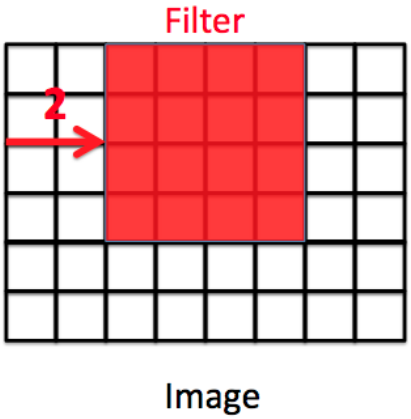



**Figure 9:Filter with stride 2**

## Padding

To prevent concerns with data loss, padding is merely the process of adding layers of zeros to our incoming photos.
Here data loss occurs due to stride we apply. When ever we apply filter on a image the corner values are used for evaluation[5] by the filter for only few times. So to avoid that we will apply padding.



## Maxpooling Layer

The term "max pooling" refers to a pooling process that chooses the largest element from the feature map area that the filter covers. The output of the max-pooling layer would therefore be a feature map that includes the most noticeable features from the prior feature map [16].

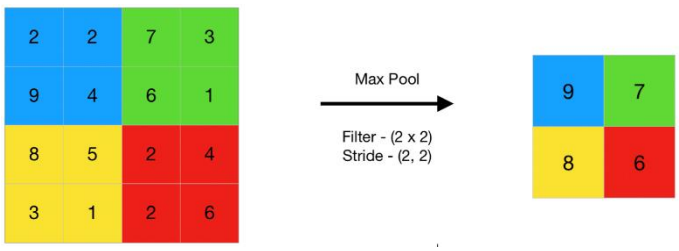



**Figure 10:Max pooling**

## Flatten Layer

The resultant 2-Dimensional arrays from pooled feature maps are all flattened into a single, lengthy continuous linear vector.

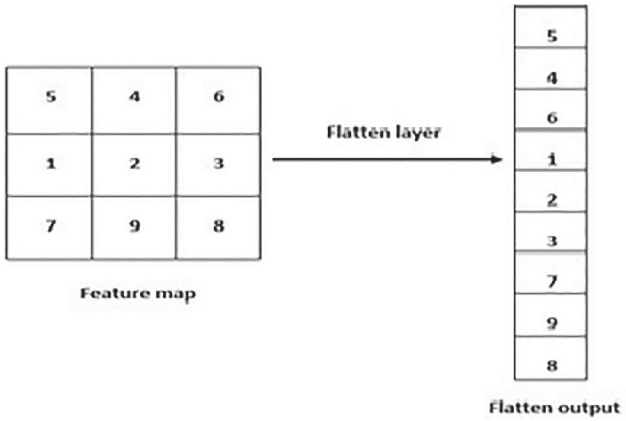



**Figure 11:Flattening**

## Dense Layer

A simple layer of neurons in which each neuron receives input from all neurons present in the previous layer, hence it is known as dense layer. Dense layer classifies images based on the output of convolutional layers[4].

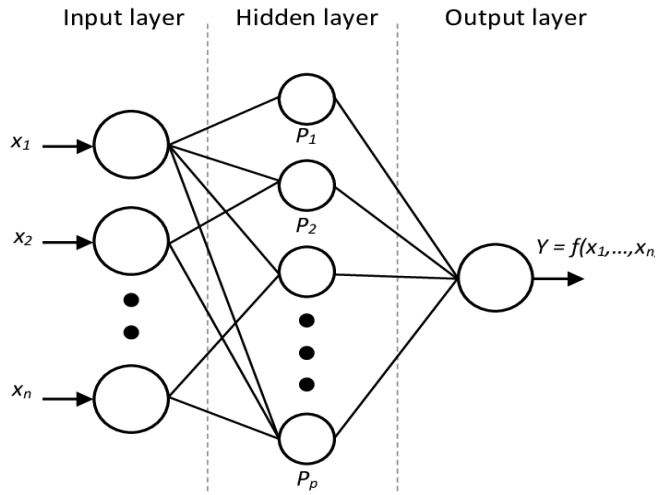



**Figure 12:Dense layer application**

## Architecture Of CnnFor Lung Opacity

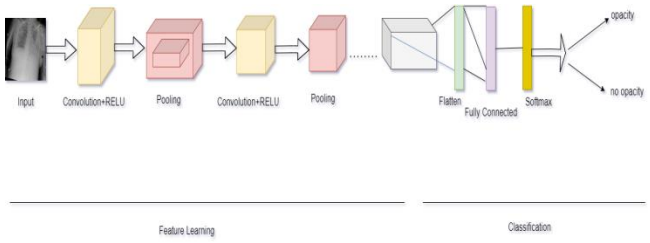



**Figure 13: Architecture of CNN**

Here at the initial stage we will give the image (image array) as input then it goes to several convolution layers, pooling layers, and finally flattening layer is applied and then the number of neuron layers is applied. Each dense layer contains some set of n neurons. However, the output dense layer contains two neurons, each neuron representing each classification label. Now, let's see how training using CNN works by using the below layers.

## Training Stage ForCnn

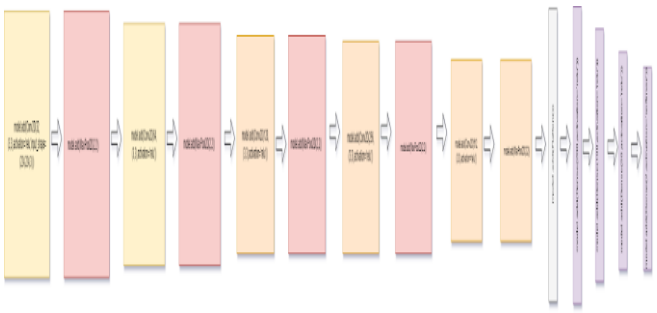


**Figure 14: Training phases of CNN**

Here in the training stage, we will take all the training images and every image will go to the convolution layer at the beginning then it goes to the max-pooling layer, which is used to reduce the size of the image, and then we will continue applying convolution layers and max-pooling layers. In this implementation, we are using a 10-layered convolution and max-pooling combination and then we are applying to flatten layer finally we are giving flatten layer as input to the dense input layer, and then we are using four dense layers where each layer consists of corresponding neurons including 200,100,50,2. Where last dense layer will have two neurons one for the lung opacity detected scenario and the other for no lung opacity case[12].

Here the main heart and soul of training are weights on the dense neuron connections. Which determines the entire classification process. Let's see how they work in brief.

Randomly initialize weights to each connection
For each training image try to find the output
If the output is correct, keep the weights the same else update weights
Repeat the above step for all training images
Finally, we will get good approximated weights for each connection

## Testing Stage ForCnn

In this, the input imageis passed to 9 layers of convolution and max-pooling, producing features for analysis, and these values or matrix obtained

after applying all layers is flattened and then passed to dense layers where the weights in these[7] layers are already predefined while training. And based on the data we will calculate the value and by using the softmax function we will classify the image as either lung opacity positive or negative.

Before going forward let's see about a few activation functions.

## Activation Functions

### Relu

ReLU, also known as the rectified linear activation function, is a non-linear or piecewise linear function that, if the input is positive[10], outputs the input directly; else, it produces zero.

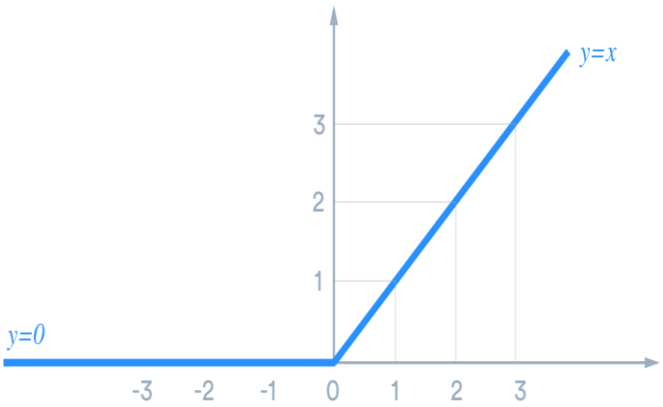



**Figure 15: Relu**

### Softmax

With the use of the mathematical function Softmax, one can turn a vector of integers into a vector of probabilities, where the probability of each value are inversely correlated with their relative scales.

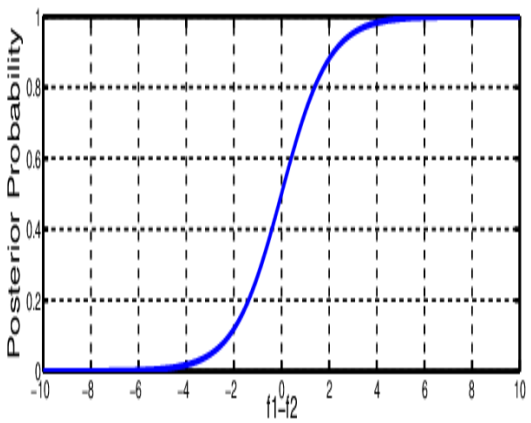



**Figure 16: Softmax**

In this training phase, we are applying 10 epochs where each epoch will take 75 images randomly from train and test data. And it will use them to train the weights on connections and filters in each epoch. For every stage, it will calculate the model's accuracy. Those accuracies can be seen in the below plot.

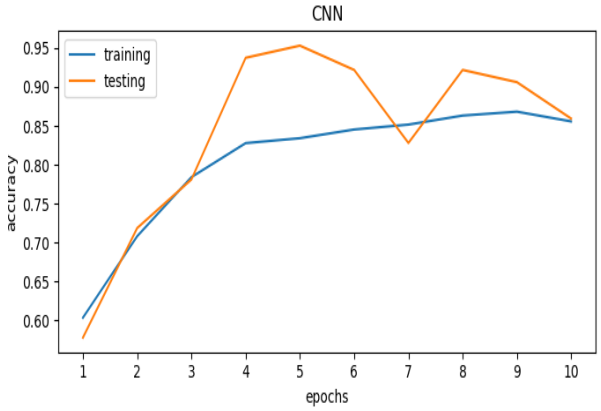



**Figure 17: Accuracy of CNN for different epochs**

From the graph, we can see that there is a study leap in training accuracy using CNN. The training accuracy hit the highest value of 85% and the testing accuracy of CNN is quite good, after the 3rd epoch, we can see the testaccuracy hitting 95%, and later on, it maintained an accuracy range of 90%.



## Working And Training Of Vgg16

Now to understand the effect of transfer learning on the dataset and to learn how the pre-weighted models trained by google are producing better accuracy, we now try to implement the vgg16 algorithm.

Before going to the training part let's now try to understand the internals of vgg16[8].

## VGG16

The vgg16 is a pre-trained convolution model that has 16 layers in total including 13 convolution layers and 3 fully connected layers. The model also contains 5 max-pooling layers which are used to reduce the size of the image.

The major difference between a normal convolution network and vgg16 is nothing but the weights on the traditional CNN are trained based on your training images. Whereas, in-case of VGG16 the weights on the fully connected layers are pre-trained using the Google imagenet dataset.

## Imagenet

The WordNet hierarchy is used to annotate 14,197,122 images in the ImageNet collection. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC), a benchmark in picture classification and object recognition, has been using the dataset since 2010. The dataset contains images of 1000 different categories and the weights on the neural network are trained with the usage of these 14 million images.

## Training Phase For Vgg16

Now first let's try to understand the architecture of VGG16.

## Architecture For Vgg16

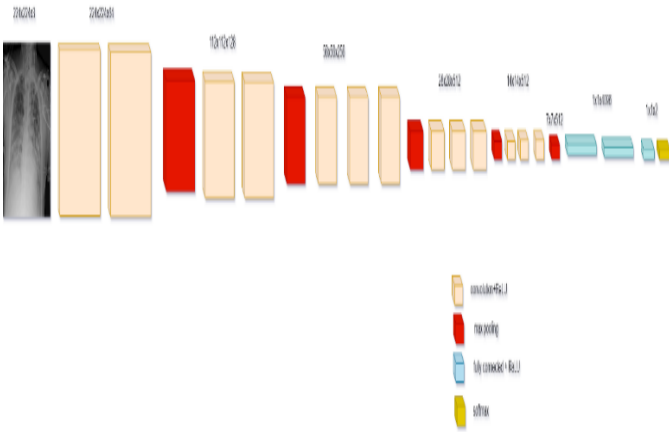


**Figure 17: Accuracy of VGG pre-trained for different epochs**

In this, we will implement vgg16 just bypassing all the training images to the entire feature selection architecture, and then we will determine the best filters, and thereby we will pass these flattened values as input to dense layers,
But here there is a catch we are not training hidden layers or the inside layers. Here we are just changing the input layer which is the feature map obtained by the image and we are also changing the output layer that is here we adding 2 neurons in the output layer one for each class. These hidden layers are trained by google'simagenet dataset.[13]
Now let's try two things one is just by keeping pre-trained weights unmoved and the other way is by modifying the weights according to our data. Now let's see the impact of the weights on the accuracies of these models.

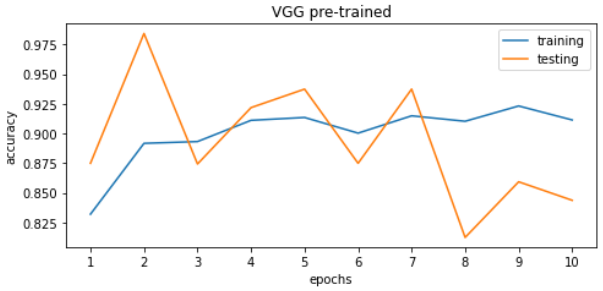
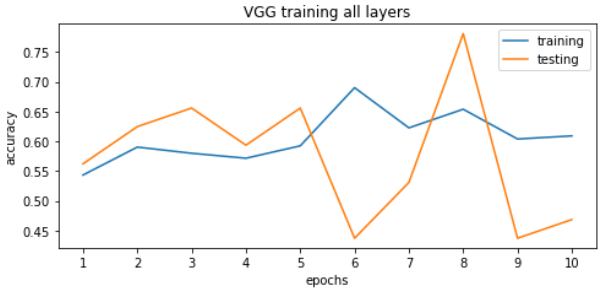



**Figure 18: Accuracy of VGG pre-trained for different epochs**

Here from this plot, we can see that the VGG16 algorithm is yielding a training accuracy of 82.5% at the end of the first epoch itself. Later on, we can see a study rise in accuracy for each epoch in the training stage. At the end of the 10th epoch, we can see the training accuracy is around 92%.
While coming testing or validation accuracy we can observe that the accuracy is around 97% in the initial stage but later on with often fluctuations it produced an average accuracy above 90%.






**Now let's see the algorithm result without pre-training.**

**Figure 19: Accuracy of VGG trained for different epochs**

From the above plot, we can see that there is a huge depression in the accuracy of both training and testing. The training accuracy at the beginning is just around 55% and even with training for 10 epochs, there is only a 5% improvement in the total training accuracy. While coming to the testing the peak accuracy was around 76% which is quite low. And the average testing accuracy is also around 55% which is not a great value.

## Results&Conclusion

Here we have seen the 3 major implementations. One is CNN which produced an average training accuracy of 75% and has average testing accuracy

of around 80%. While, coming to the accuracy of pre-trained VGG16 is around 90%[14] for the training phase, and it also produced an average accuracy of 86%.On the other hand, the trained VGG16 model with our dataset produced an average training accuracy of 58% and it has an average testing accuracy of 55%.

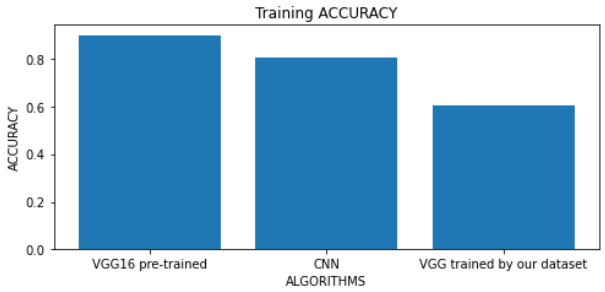



**Figure 20: Accuracy training comparison**

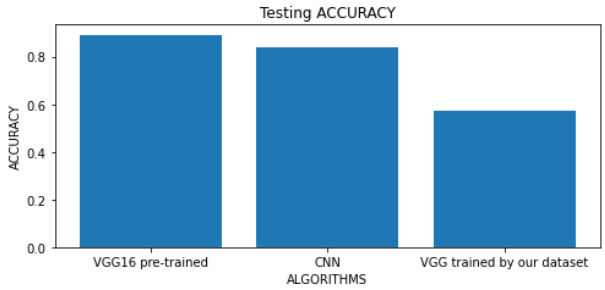



**Figure 21: Accuracy testing comparison**

From this, we can conclude that there is an impact of pre-trained weights on the overall accuracy of the model even though the dataset that we are considering is different from the imagenet dataset. For the application of lung opacity that we have considered in this paper, the VGG16 pre-trained algorithm produced the highest average training accuracy of 90% and produced an average testing accuracy of 86%.

## References

T Rahmat, A Ismail and S. Aliman, "Chest X-Ray Image Classification Using Faster R-Cnn", Malaysian J Comput., vol. 4, no. 1, pp. 225-36, 2019.

J Rubin, D Sanghavi, C Zhao, K Lee, A Qadir and M.+ Xu-Wilson,"Large scale automated reading of frontal and lateral chest x-rays using dual convolutional neural networks", 2018.

R Jain, P Nagrath, G Kataria, V Sirish Kaushik and D. Jude Hemanth,"Pneumonia detection in chest X-ray images using convolutional neural networks and transfer learning", Meas J Int Meas Confed., vol. 165, pp. 108046, 2020.

AK Jaiswal, P Tiwari, S Kumar, D Gupta, A Khanna and JJPC Rodrigues, "Identifying pneumonia in chest X-rays: A deep learning approach", Meas J Int Meas Confed, vol. 145, pp. 511-8, 2019.

T. Nagamani.,et.al., "Malaria Detection Using Convolutional Neural Network", Journal of Engineering Sciences, ISSN NO: 0377-9254,Vol 11, Issue 6 , June, 2020..

Butt et al., "Deep learning system to screen coronavirus disease2019 pneumonia", Applied Intelligence, vol. 1, 2020.

Zech et al., "Variable generalization performance of a deep learning model to detect pneumonia in chest radiographs: A cross-sectional study", PLoS medicine, vol. 15, no. 11, pp. e1002683, 2018

Asnaoui et al., "Automated methods for detection and classification pneumonia based on x-ray images using deep learning", arXiv preprint, 2020.

Xianghong Gu et al., "Classification of Bacterial and Viral ChildhoodPneumonia Using Deep Learning inChest Radiography", Proceedings of the 3rd International Conference on Multimedia and Image Processing (ICMIP 2018), pp. 88-93, 2018.

Ioannis D. Apostolopoulos and Tzani A. Mpesiana, "Covid-19:automatic detection from x-ray images utilizing transfer learning with convolutional neural networks", Physical and Engineering Sciences in Medicine, pp. 1, 2020.

Ashok Reddy Kandula, et al. "Machine LearningAlgorithmsforClassification of Genetic Mutations arose during the cell's rehabilitation to Cancer Tumor", Solid State Technology 64(2) 2752-2758, 2021.

BaburaoMarkapudi, Kavitha Chaduvula, D.N.V.S.L.S. Indira &Meduri V. N. S. S. R. K. Sai Somayajulu, "Content-based video recommendation system (CBVRS): a novel approach to predict videos using multilayer feed forward neural network and Monte Carlo sampling method", published in the journal of Multimedia Tools and Applications (Springer Nature), published on 11th August 2022. PP:1-27, (2021) .

D.N.V.S.L.S. Indira, Babu Rao Markapudi, Kavitha Chaduvula, Rathna Jyothi Chaduvula, "Visual and buying sequence features-based product image recommendation using optimization based deep residual network", published in the journal of Gene Expression Patterns, Volume 45, 2022, September 2022.

NeuroQuantology|November2022| Volume 20 | Issue 15 | PAGE 5932-5944| DOI: 10.48047/NQ.2022.20.15.NQ88597 G.Deep Aman / PERFORMANCE ANALYSIS OF VISUAL GEOMETRY GROUP ALGORITHMS ON PNEUMONIA CLASSIFICATION

NeuroQuantology|November2022| Volume 20 | Issue 15 | PAGE 5945-5951-5962| DOI: 10.48047/NQ.2022.20.15.NQ88599 Ashok Reddy Kandula / Machine Learning Algorithms for Predictive Analysis in the Identification of Chronic Liver Disease

Dr GVSNRV Prasad, "Adaptive Optimization-enabled Neural Networks to handle the imbalance churn data in churn prediction", International Journal of Computational Intelligence and Applications, Dec-2011.