Pneumonia detection in chest X-ray by H-mask techniques

1. Abstract
2. Introduction
3. Data
4. Method (describe current methods vs our proposed method)
    a. Block method
    b. H-mask method
5. Experiment steps
    a. Generate H-shaped / Extract H-mask
    b. Model selection
    c. Parameter selection
    d. Resource optimization
6. Model comparison
    a. CNN
    b. VGG-16
    c. VGG-19
7. Outcomes / future goal
8. Conclusion
9. References

Abstract:

Pneumonia, a common respiratory disease, is a major worldwide health problem. Accurate and prompt diagnosis is critical for effective therapy and averting consequences. In this experiment, we particularly focus on pneumonia detention in chest X-ray, which has emerged as a critical diagnostic tool. However, the complexities of interpreting these pictures demand new approaches for reliable detection. This research introduces a unique strategy that uses the H-mask technique to improve the detection of pneumonia indications in chest X-ray images. The H-mask approach intelligently concentrates on regions of interest, dramatically enhancing accuracy. We establish the efficiency of this strategy and its potential influence on clinical practice through rigorous tests and comparative research.

Introduction:

Pneumonia is a serious worldwide health concern, impacting millions of persons annually. Prompt and correct diagnosis is critical for efficient treatment and avoiding serious consequences. Medical imaging, specifically chest X-ray analysis, is critical in the early detection of pneumonia. Interpreting these images, however, can be difficult because complex anatomical structures may obscure subtle indicators.
Traditional pneumonia diagnosis methods rely on manual interpretation or simple image processing techniques. While these approaches have yielded useful results, they may lack the sensitivity needed to detect complex patterns indicative of pneumonia. This highlights the need for more advanced and precise diagnostic tools.

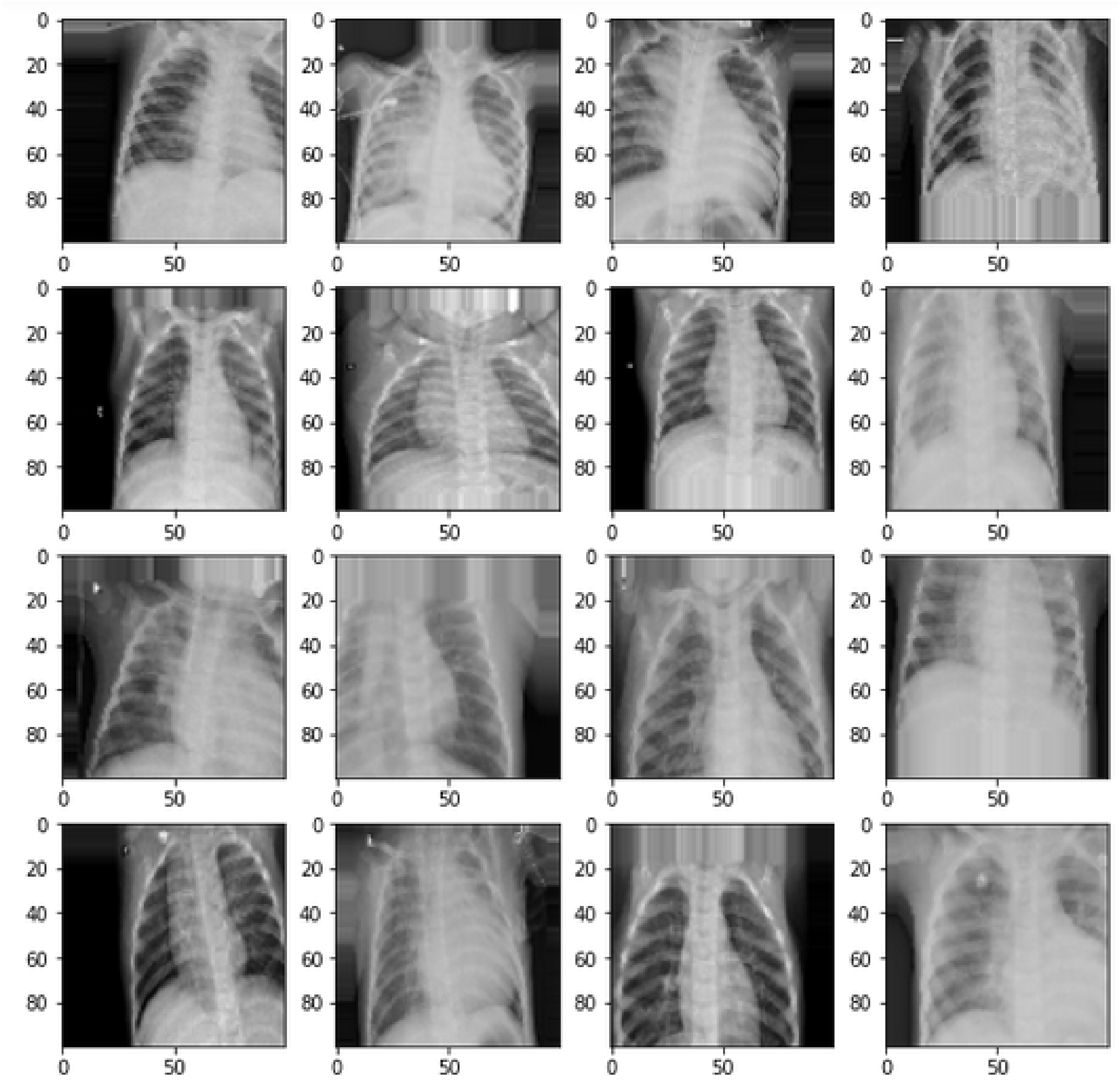
The suggested method uses the H-mask technique, a novel segmentation method inspired by the human visual system's natural ability to focus on specific areas of interest. We improve the ability of the model to detect vital data inside chest X-ray pictures by actively focusing its interest. This cutting-edge technique can greatly increase diagnostic precision as well as effectiveness.

We give an in-depth investigation of the H-mask approach for pneumonia detection in this article. We do systematic trials, evaluate the outcome of existing approaches, and give thorough insights into the method used. We additionally deal with critical issues such as parameter selection, resource optimization, and model selection to ensure practical viability for real-world clinical applications.
Our research will not only progress in the area of pneumonia detection but will also pave the road for more accurate and efficient diagnostic techniques, resulting in better patient outcomes and improved healthcare delivery.

Data:

Mainly data source[1] is divided into 3 categories as follows

1. Training (5232)
   a. Pneumonia (3883)
   b. Normal (1349)
2. Testing (640)
   a. Pneumonia (390)
   b. Normal (234)
3. Validation (16)
   a. Pneumonia (8)
   b. Normal (8)



Method: [Current methods vs Our Proposed method]

Current methods for pneumonia detection:

Present methods for detecting pneumonia rely mainly on standard image processing techniques and straightforward pattern recognition algorithms. These techniques frequently include pre-processing processes such as edge detection, thresholding, and morphological operations to separate possible infection sites. Furthermore, texture, shape, and intensity are collected from these divided sections for categorization. fortunately, these approaches may fail to detect minor signs of pneumonia, resulting in decreased sensitivity and specificity.

Block Method:

The block approach is an essential aspect of our stated methodology. It involves classifying the chest X-ray picture into distinct, non-overlapping chunks. We may examine potential indications of pneumonia more thoroughly by splitting the picture down into smaller parts. Each block is then independently evaluated, allowing for an overall evaluation of the image. This technique fulfills two functions. For the first time, it allows the model to detect specific features that may indicate pneumonia. Second, it provides a technique for completely evaluating the whole picture, to guarantee no potential information is ignored.

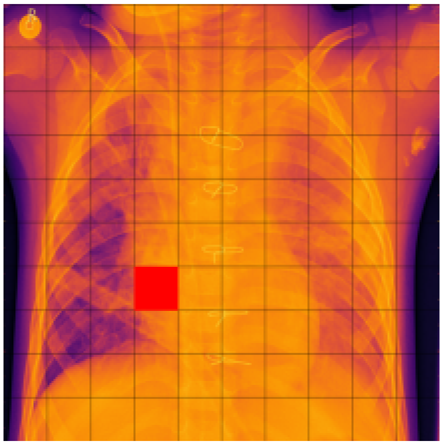


Fig.General idea of Block method.

Our Proposed method for Pneumonia Detection:

As opposed to standard techniques, the approach we suggest unveils an evolutionary alteration in pneumonia detection by using the H-mask technique. The proposed segmentation algorithm considerably enhances the detection of pneumonia indications in chest X-ray images. Unlike previous procedures, the H-mask methodology constantly shifts to focus on regions of interest, allowing for a more advanced study.

H-Mask method:

The H-mask approach offers a significant advancement in recognizing signs of pneumonia. This strategy simplifies the simulation's focus on key points inside the chest X-ray picture, based on the human visual system's essential ability to give higher priority to certain regions of significance. The H-mask approach dramatically enhances the model's ability to detect tiny patterns that indicate the presence of pneumonia by meticulously hiding insignificant regions.

A variety of complex calculations and methods are used to produce the H-mask. It recognizes and selects locations within an X-ray picture of the chest using a mix of intensity gradients, textural properties, and structural clues. As a consequence, the representation has been modified to highlight regions that are most likely to contain essential information for proper diagnosis.

In conclusion, combining the block method and the H-mask approach is a framework for our revolutionary approach to pneumonia detection. We want to change the accuracy and efficiency of pneumonia detection in chest X-ray pictures by systematically assessing localized characteristics and intelligently concentrating on regions of interest.

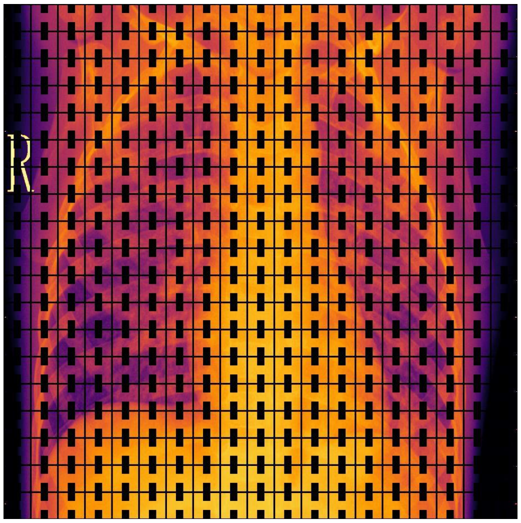


Fig General idea of the H-mask method

Experiment steps:

Standardize dimensionality of data:

Before the formation of an H-shaped mask, we have to ensure that all processed data have equal dimensions. Chest X-ray dimensionality ranges from 600*2400. So we generalized them by 1000*1000 as well as 912*912 by using Imgae.resize() from the PIL package.

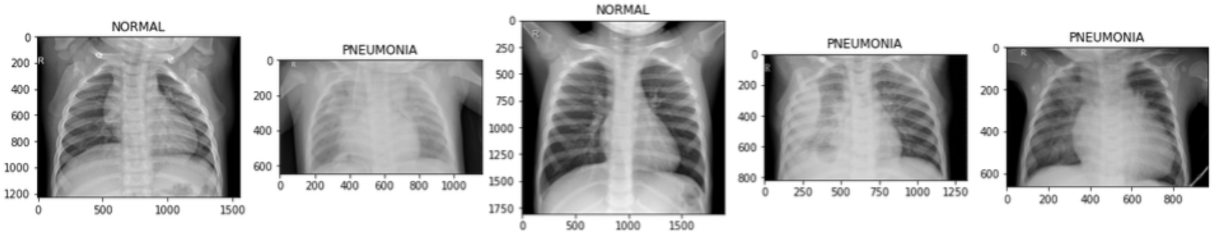


Fig. Chest X-ray before standardized size

After Standardizing dimensionality we can finally generate and extract H-shaped mask.

Generate H-shaped / Extract H-mask:

The process is divided into 3 phases,

1. Apply gird
2. Create rectangles
3. Combine 3 rectangles to generate H-shape mask

The formation of the **H** alphabet is a combination of 2 vertical rectangles and 1 horizontal rectangle

For 1000*1000 dimensionality,
grid size =40
bar_with=15
Spacing = 2 putting values in the following equation,

Bar_height = (grid_size - 2 * spacing) // 3

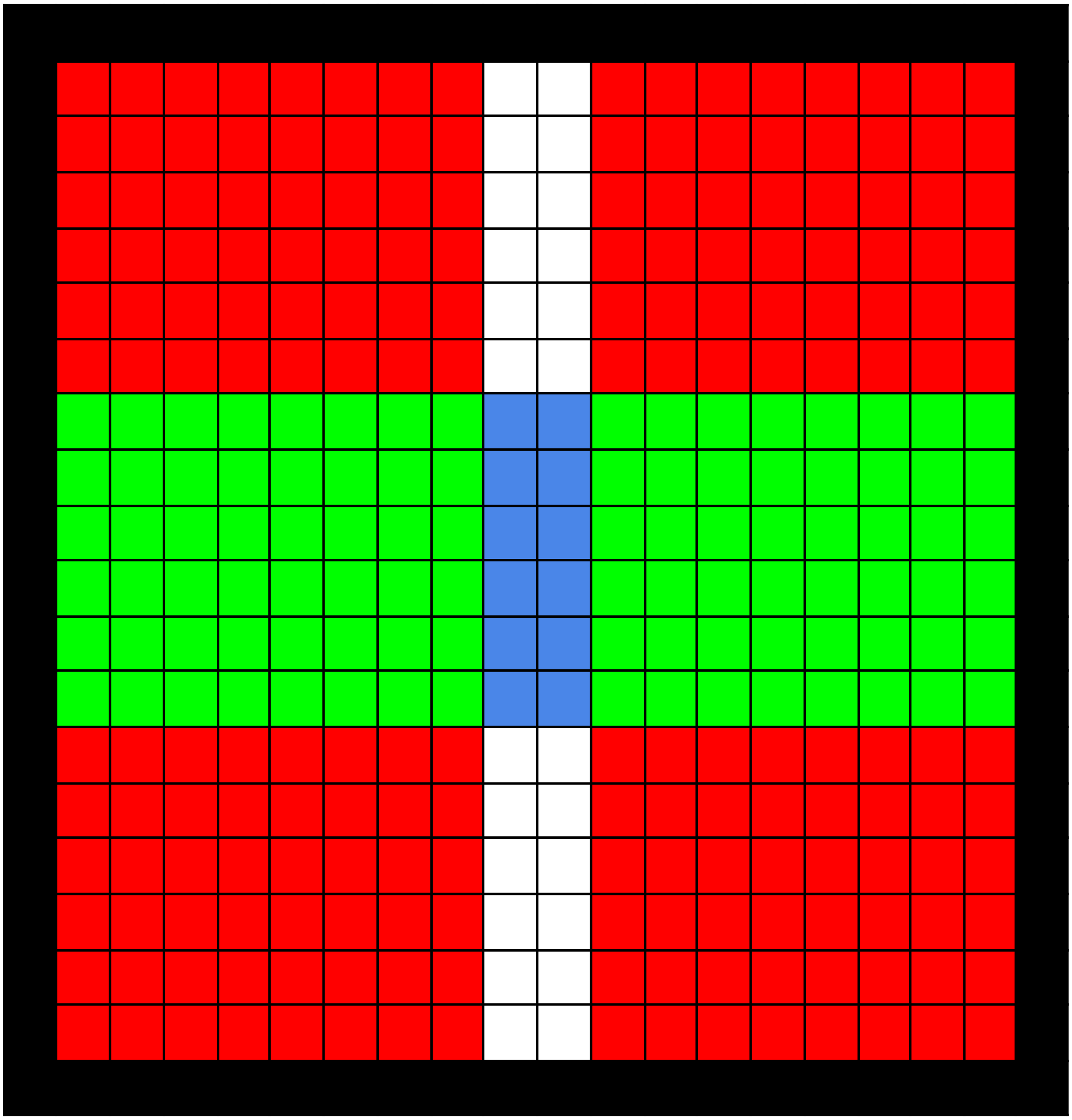
Hence we get Bar_height=12.
Now create rectangles in a grid chest x-ray image as per the following equations.

Rectangle ([(spacing, spacing), (spacing + bar_width, grid_size - spacing)]

Rectangle ([(grid_size - spacing - bar_width, spacing), (grid_size - spacing, grid_size - spacing)]

Rectangle ([(spacing, spacing + bar_height), (grid_size - spacing, spacing + bar_height + bar_width)]

Rectangle -1 = [(2,2), (17,38)]
Rectangle -2 = [(23,2), (38,38)]
Rectangle -3 = [(2,14), (38,29)]



Extract H-mask,
create 3 rectangle boxes.
We take spacing = 4

Each block of 🟥 ,🟩 and 🟥 is equal to 12+12+12=36
And the spacing on the border will be 2 on all sides.

Hence, if we have dimensionality 1000*1000 with grid size=40 we create 625 blocks. However, if we create 912*912 with grid size=48 we create 361 blocks. For this entire experiment, we are using 912*912 formation because it reduces the number of blocks compared to 1000*1000 which leads to less computation power and time.

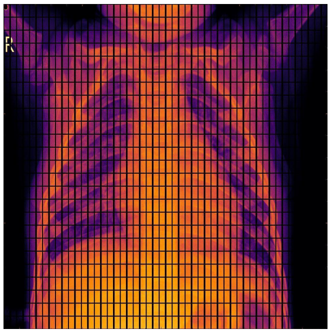
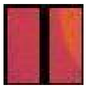


Image after creating 2 vertical rectangles



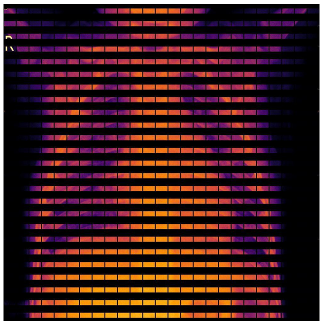
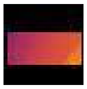


Image after creating 1 horizontal rectangle



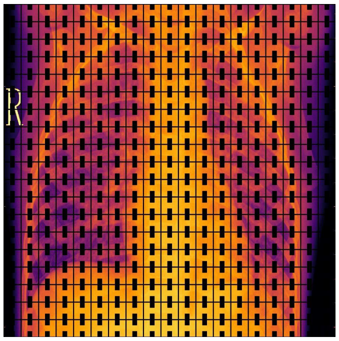
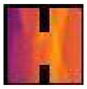


Image after creating 2 vertical rectangles & 1 horizontal rectangle



Hence by doing the same process, we can generate and extract H-mask for all images in train, test, and validation datasets.

Model selection:

The choice of model architecture is a critical factor in the success of our pneumonia detection framework. We consider three distinct models for evaluation:

- Convolutional Neural Network (CNN)
- VGG-16
- VGG-19

Each model brings unique strengths in feature extraction and pattern recognition. Comparative analyses are conducted to assess their performance in conjunction with the H-mask technique. Additionally, fine-tuning strategies, such as transfer learning and architecture modifications, are explored to further enhance the models' efficacy in pneumonia detection. In combination, these experimentation steps collectively form a comprehensive framework for evaluating the effectiveness of the H-mask technique in enhancing diagnostic accuracy for pneumonia detection in chest X-ray images.

CNN:

A Convolutional Neural Network (CNN) is a type of deep learning model specifically designed for processing grid-like data, such as images or video. CNN has shown significant effectiveness in image classification, object detection, face recognition, and the medical image analysis field.

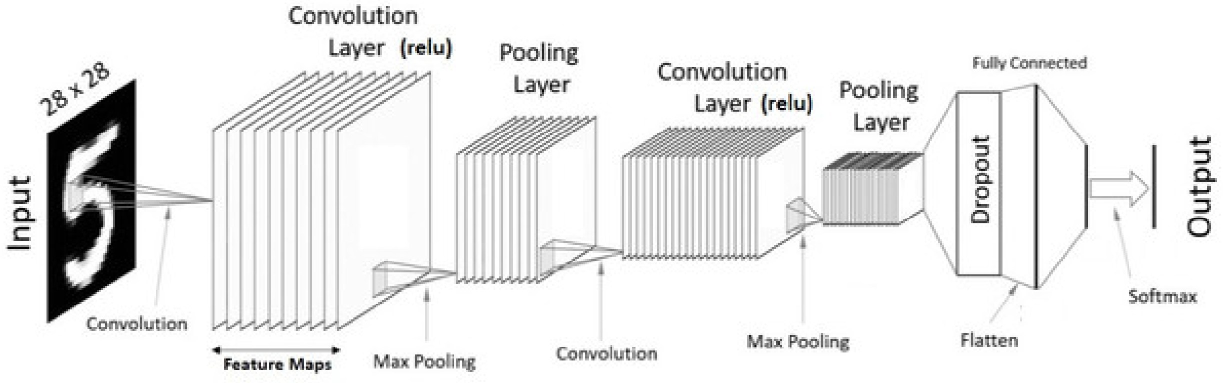


Fig. CNN model representation [2]

<u>Key components of CNN are as below:</u>

- ❖ Convolutional Layer:
    - ➢ The core building blocks of CNNs. They consist of a set of learnable filters that are convolved across the input data to detect patterns at various spatial hierarchies.
    - ➢ These filters learn to identify features like edges, textures, and more complex structures as the network progresses through layers[3].

- Pooling layer:
    - Often placed after convolutional layers, they serve to down-sample the spatial dimensions of the feature maps.
    - Common pooling operations include max pooling (retaining the maximum value in a region) or average pooling (taking the average value in a region)[5].
- Fully Connected layer:
    - These layers are akin to the traditional feedforward neural networks, where each neuron is connected to every neuron in the preceding and succeeding layers[3].
    - Fully connected layers are typically employed towards the end of the network to make predictions based on the learned features.

- Activation function:
    - After convolutional and fully connected layers, non-linear activation functions are employed to introduce non-linearity, allowing the network to learn more complicated correlations between features.

VGG-16:

VGG-16 is distinguished by its considerable length. It has 16 layers, including 13 convolutional layers and 3 fully connected layers. Its depth empowers it to learn complicated properties.

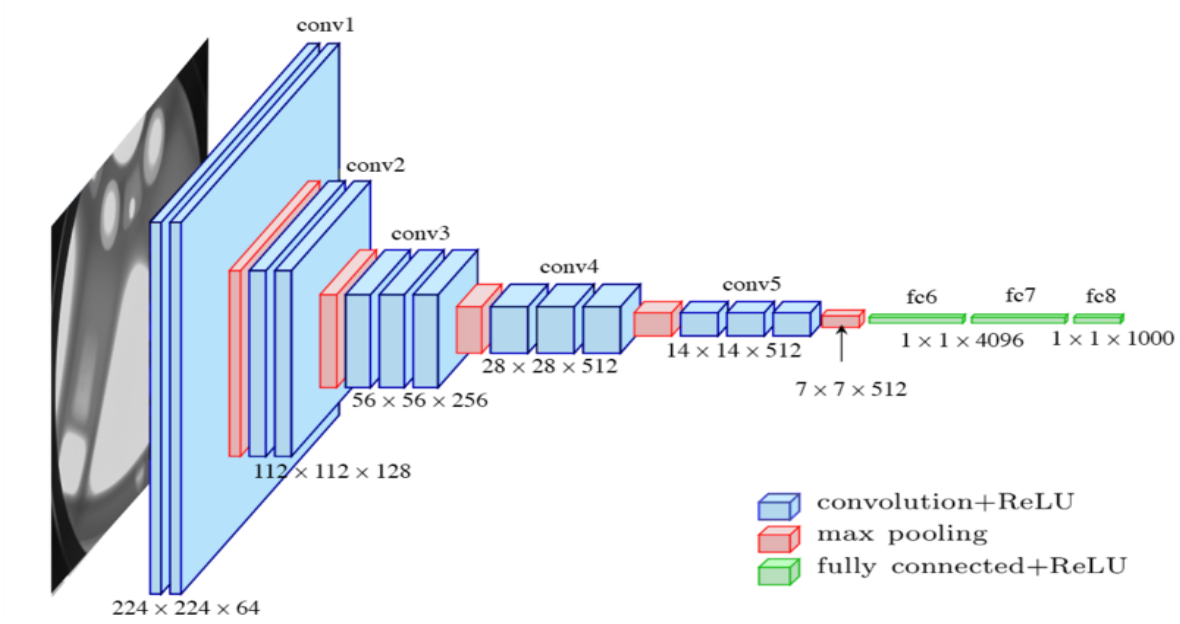


Fig.VGG-16 model representation [1]

- Input layer: It takes i/p of fixed size [224*224] with 3 channels (RGB).
- Convolution layer:
    - The main feature of this layer is to convert all pixels in their respective field as a single value.
    - Despite being placed on top of one another, the convolutional layers have tiny 3x3 pixel receptive fields. Because of its architecture, the network can pick up sophisticated characteristics.
- MAX Pooling layer:
    - The most significant information in the feature maps is retained even if their spatial dimensions are decreased by these layers.
    - Through data downsampling, max-pooling helps increase the network's resilience to changes in input location.

- Fully Connected layer:
    - Three fully linked layers make up the last three levels of the VGG-16 structure. These layers finally generate class probabilities after processing the flattened feature maps.
    - 1st and 2nd FC layer have 4096 neurons and 3rd layer has 1000 neurons match with ImageNet dataset.

Steps for import VGG-16:

```
from tensorflow.keras.applications.vgg16 import VGG16

vgg16 = VGG16(weights='imagenet', include_top=False)
```

VGG-19:

VGG-19, despite its increased depth of intricacy and complexity compared to VGG-16, has been found applicable in various computer vision applications, and tasks including such as image classification, object detection, and image generation.

The depth of VGG-19 is what makes it unique. There are 3 fully connected layers and 16 convolutional layers totaling 19 layers in this structure. The network can now learn even more intricate properties because of its deeper structure.

Steps for import VGG-19:

```
from tensorflow.keras.applications.vgg19 import VGG19

vgg19 = VGG19(weights='imagenet', include_top=False)
```

Parameter selection:

In a big data environment, the scalability of learning algorithms is crucial due to the rapid increase in observations, features, and classes. Deep learning algorithms aim to create models that can effectively map input variables to their corresponding class labels, addressing the challenges posed by large and complex datasets[6].

Model efficiency and performance depend on parameters used while training. Here are some factor that affects overall model performance.

1. Computational power
2. Model capacity and complexity
3. Memory and storage constrain
4. To avoid overfitting/underfitting

Specifically for this experiment, our major focus is on CNN and VGG-16/VGG-19 parameters only.

There are millions of combinations on which we can train the CNN model, however after doing a couple of performances with different parameters for the CNN model. we found following combination of parameters gives better performance. There may be a chance of optimal performance*.

CNN:

1) Image shape: 200*200
2) Batch size: 32, 64, 128
3) Kernels : (3x3), (5x5), (7x7)
4) Activation function: Softmax, ReLu
5) Pooling : Maxpool
6) Dense : 32, 128
7) Dropout: 0.2
8) Optimizer: adam
9) Loss function: categorical_crossentropy
10) Epochs: 5

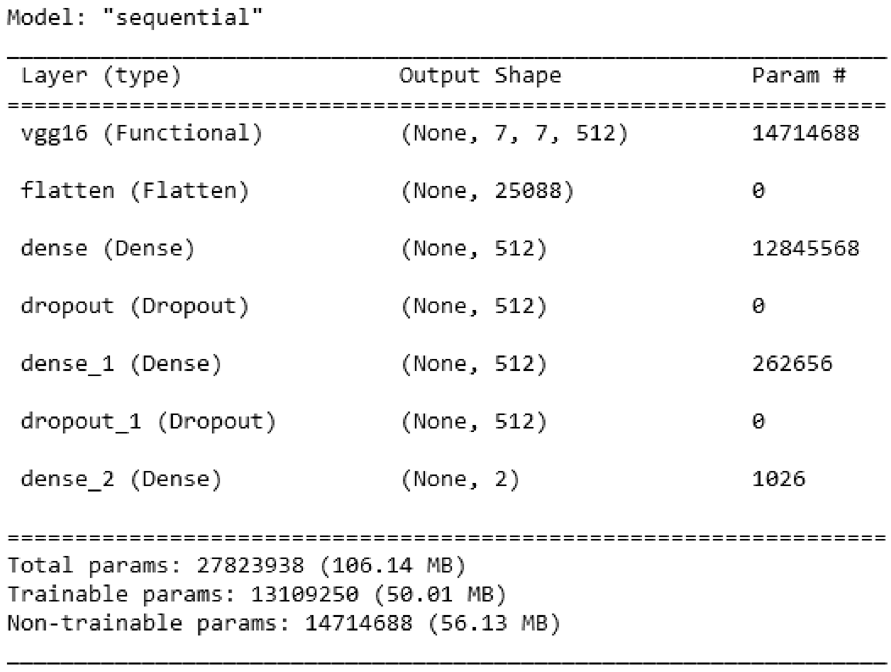
Moving towards VGG-16 and VGG-19, which includes 2 types of parameters

- Trainable [learnable]:
  - varying while training process for reducing a specified loss function.
  - Help in the prediction process
  - For instance, weights and biases in neural n/w
- Non-trainable:
  - they are fixed throughout the training process.

- ○ Help in structural components and guide the behavior of model.
- ○ For instance, number of filters.

VGG-16 / VGG-19:

1. Image shape: 224*224
2. Batch size : 32
3. Weights='imagenet'
4. Activation function: Softmax, ReLu
5. Dense : 512
6. Dropout: 0.2
7. Optimizer: adam
8. Loss function: categorical_crossentropy
9. Epochs: 5



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 7, 7, 512)         14714688

 flatten (Flatten)           (None, 25088)             0

 dense (Dense)               (None, 512)               12845568

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 512)               262656

 dropout_1 (Dropout)         (None, 512)               0

 dense_2 (Dense)             (None, 2)                 1026

=================================================================
Total params: 27823938 (106.14 MB)
Trainable params: 13109250 (50.01 MB)
Non-trainable params: 14714688 (56.13 MB)
_________________________________________________________________
```

Fig. VGG-16 summary for H-mask resource.

Besides tabular diagram shows no.of trainable and non-trainable param in the VGG-16 model.

Resource optimization:

After successfully acquiring knowledge of CNN, VGG-16, and VGG-19 we can compare models over the accuracy of training and testing. However, the chest X-ray data source has 5232 in the training set with 640 testing data and 16 for validation as mentioned above in the article.

As the medical field has limited resources for emerging diseases, our approach towards training models with a portion of H-mask resources and evaluating performance. Throughout this experiment, we check models with 6 distinct portions of the dataset.

Model comparison:

In this phase, we compare the accuracy of training and testing for CNN, VGG-16, and VGG-19 over a portion of a dataset with H-mask and without H-mask.

Comparison matrix of accuracy without H-mask [epoch=5,batch_size=32]

<table>
<tr><th>CNN</th><th>VGG-16</th><th>VGG-19</th><th>Resources (Datasets)</th></tr>
<tr><td><table><tr><th>kernel</th><th>Train</th><th>Test</th></tr><tr><td>3x3</td><td>96</td><td>68</td></tr><tr><td>5x5</td><td>87</td><td>79</td></tr><tr><td>7x7</td><td>89</td><td>70</td></tr></table></td><td>Train=98.67
Test=67</td><td>Train=98.42
Test=71</td><td>Train=600+600
Test=100+100
val=8+8</td></tr>
<tr><td><table><tr><th>kernel</th><th>Train</th><th>Test</th></tr><tr><td>3x3</td><td>98</td><td>71</td></tr><tr><td>5x5</td><td>95</td><td>83</td></tr><tr><td>7x7</td><td>92</td><td>75</td></tr></table></td><td>Train=98
Test=75.75</td><td>Train=98.79
Test=73</td><td>Train=700+700
Test=200+200
val=8+8</td></tr>
<tr><td><table><tr><th>kernel</th><th>Train</th><th>Test</th></tr><tr><td>3x3</td><td>98</td><td>76</td></tr><tr><td>5x5</td><td>95</td><td>70</td></tr><tr><td>7x7</td><td>93</td><td>75</td></tr></table></td><td>Train=99
Test=89.75</td><td>Train=97.31
Test=86</td><td>Train=800+800
Test=200+200
val=8+8</td></tr>
<tr><td><table><tr><th>kernel</th><th>Train</th><th>Test</th></tr><tr><td>3x3</td><td>96</td><td>68</td></tr><tr><td>5x5</td><td>87</td><td>79</td></tr><tr><td>7x7</td><td>89</td><td>70</td></tr></table></td><td>Train=99.35
Test=74.50</td><td>Train=98
Test=81</td><td>Train=1000+1000
Test=200+200
val=8+8</td></tr>
</table>

<table>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td><table><tr><th>kernel</th><th>Train</th><th>Test</th></tr><tr><td>3x3</td><td>94</td><td>73</td></tr><tr><td>5x5</td><td>80</td><td>70</td></tr><tr><td>7x7</td><td>92</td><td>72</td></tr></table></td><td>Train=98.30
Test=79</td><td>Train=98.60
Test=80.50</td><td>Train=500+500
Test=100+100
val=8+8</td></tr>
<tr><td><table><tr><th>kernel</th><th>Train</th><th>Test</th></tr><tr><td>3x3</td><td>92</td><td>61</td></tr><tr><td>5x5</td><td>69</td><td>73</td></tr><tr><td>7x7</td><td>53</td><td>50</td></tr></table></td><td>Train=98.50
Test=85.75</td><td>Train=99.62
Test=84</td><td>Train=400+400
Test=200+200
val=8+8</td></tr>
</table>

Comparison matrix of accuracy with H-mask [epoch=5,batch_size=32]

<table>
<tr><td>CNN</td><td>VGG-16</td><td>VGG-19</td><td>Resources (Datasets)</td></tr>
<tr><td><table><tr><th>kernel</th><th>Train</th><th>Test</th></tr><tr><td>3x3</td><td>95</td><td>65</td></tr><tr><td>5x5</td><td>80</td><td>63</td></tr><tr><td>7x7</td><td>89</td><td>68</td></tr></table></td><td>Train=96.67
Test=83</td><td>Train=96.92
Test=72</td><td>Train=600+600
Test=100+100
val=8+8</td></tr>
<tr><td><table><tr><th>kernel</th><th>Train</th><th>Test</th></tr><tr><td>3x3</td><td>95</td><td>72</td></tr><tr><td>5x5</td><td>76</td><td>71</td></tr><tr><td>7x7</td><td>97</td><td>79</td></tr></table></td><td>Train=97.29
Test=78</td><td>Train=97.36
Test=69</td><td>Train=700+700
Test=200+200
val=8+8</td></tr>
</table>

<table>
<tr>
<td>kernel|Train |Test<table><tr><td>3x3</td><td>97</td><td>70</td></tr><tr><td>5x5</td><td>89</td><td>72</td></tr><tr><td>7x7</td><td>90</td><td>68</td></tr></table></td>
<td>Train=96.19
Test=82.25</td>
<td>Train=96.75
Test=73.75</td>
<td>Train=800+800
Test=200+200
val=8+8</td>
</tr>
<tr>
<td>kernel|Train |Test<table><tr><td>3x3</td><td>97</td><td>75</td></tr><tr><td>5x5</td><td>84</td><td>65</td></tr><tr><td>7x7</td><td>95</td><td>71</td></tr></table></td>
<td>Train=97.70
Test=72</td>
<td>Train=96.90
Test=79</td>
<td>Train=1000+1000
Test=200+200
val=8+8</td>
</tr>
<tr>
<td>kernel|Train |Test<table><tr><td>3x3</td><td>97</td><td>71</td></tr><tr><td>5x5</td><td>78</td><td>65</td></tr><tr><td>7x7</td><td>92</td><td>68</td></tr></table></td>
<td>Train=98.20
Test=73.50</td>
<td>Train=97.20
Test=83.50</td>
<td>Train=500+500
Test=100+100
val=8+8</td>
</tr>
<tr>
<td>kernel|Train |Test<table><tr><td>3x3</td><td>95</td><td>65</td></tr><tr><td>5x5</td><td>90</td><td>61</td></tr><tr><td>7x7</td><td>93</td><td>69</td></tr></table></td>
<td>Train=98.12
Test=77.75</td>
<td>Train=97.25
Test=80</td>
<td>Train=400+400
Test=200+200
val=8+8</td>
</tr>
</table>

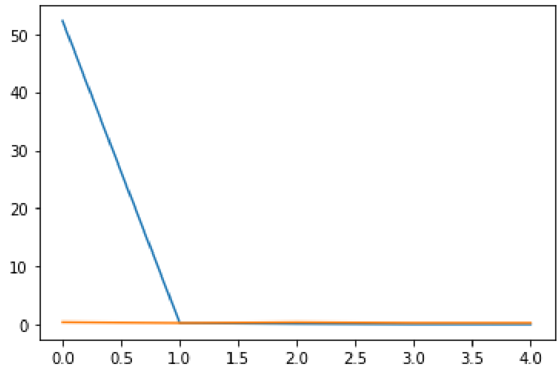


Fig. loss vs val_loss for CNN on H-mask resource.

Besides line graph illustrates a drastic drop in loss over epochs iteration which is from 52 to 0.07. hence, which denotes a successful training of the CNN model for (3x3) kernel size.

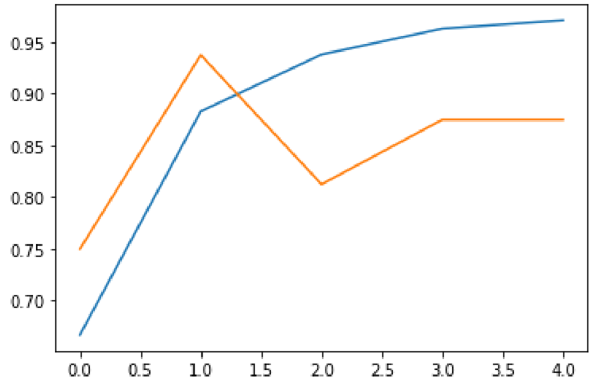


Fig. Accuracy vs val_accuracy for CNN on H-mask resource.

Besides line graph illustrates a boost in accuracy over epochs iteration which is from 0.66 to 0.97. hence, which denotes a successful training of the CNN model for (3x3) kernel size.

Interesting finding: While operating with VGG-16 and VGG-19 we found

when we set `model.trainable = true,` which unfreezes weights of pre-trained layers hence accuracy of training and testing was reduced by 53%, and 50% respectively.

However, when we set `model.trainable = false,` it means freezing the weights of a pre-trained model, whereas only custom layer weights could be updated.
This causes a remarkable rise in the accuracy of training and testing, due to all previously trained layers' features used to train the next layer. Hence overall performance increased. In addition, noticeable reduction in training time as well as overfitting risk.

Outcomes / Future goal:

Above model comparison with CNN, VGG-16, and VGG-19, we analyzed the proposed H-mask method gives a **higher testing accuracy** for *VGG-16* as **compared** to without H-mask for a dataset of 1200[600 normal, 600 pneumonia] training data, 200 [normal:100, pneumonia:100] testing as well 16 validation data. Moreover, it is also higher for individual model comparison with VGG-19 and CNN model.

Additionally, by using the proposed H-mask technique we not only save computational power but also get higher efficiency in pneumonia detection with limited resources.

For the upcoming process, we will introduce a method for the detection of pneumonia-affected regions in chest X-rays by using combined techniques of the H-mask and R-CNN model.

System requirement

The hardware used for all experiments was on a 10th generation Core i5 laptop PC with 12GB RAM, and a GPU of 4GB with the usual Windows OS.

Conclusion:

To sum up, this study presents a new method for identifying pneumonia in chest X-ray pictures by utilizing the cutting-edge H-mask method. The study tackles the urgent need for a precise and effective diagnosis of this common respiratory condition. It is clear from a thorough comparison of existing approaches and our suggested strategy that conventional methods might not be able to identify minor signs of pneumonia.

In addition, techniques for resource efficiency were put in place to guarantee the practicality of our approach, especially in medical settings with limited resources. This highlights the possibility of using our approach in environments with constrained processing resources and adds to the viability of real-world healthcare applications

All things considered, this work is a major step toward transforming the way the H-mask approach is used to identify pneumonia. By offering quicker and more accurate diagnoses, the suggested approach has the potential to completely transform clinical practice. This would eventually enhance patient outcomes and have a favorable effect on healthcare delivery globally.

Reference:

1. Mooney, P. (2018). Chest X-ray Images (Pneumonia). Kaggle. https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia
2. Kan, M. (n.d.). Convolutional Neural Network (CNN) Tutorial. Kaggle. https://www.kaggle.com/code/kanncaa1/convolutional-neural-network-cnn-tutorial
3. Victor Ikechukwu, A., Murali, S., Deepu, R., & Shivamurthy, R. (2021). ResNet-50 vs VGG-19 vs training from scratch: A comparative analysis of the segmentation and classification of Pneumonia from chest X-ray images. Global Transitions Proceedings, 2(2), 375–381. *https://doi.org/10.1016/j.gltp.2021.08.027*
4. Ruder, S. (n.d.). Convolutional Neural Networks Cheatsheet. Stanford University. https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks

5. Mandal,M. (2023, June 30). Introduction to Convolutional Neural Networks (CNN). Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/
6. Suthaharan, S. (2016). Ch-12 Deep Learning Models. In Machine Learning Models and Algorithms for Big Data Classification (p. 289). Springer. https://link.springer.com/book/10.1007/978-1-4899-7641-3

Special thanks to DR. SHAN SUTHAHARAN for their guidance, support as well as encouragement in the completion of this experiment.

Author

Pratik A Bhatt
Department of Computer Science
UNC - Greensboro
North Carolina, NC 27412
p_bhatt@uncg.edu

Mentor

Dr.Shan Suthaharan
Department of Computer Science
UNC - Greensboro
North Carolina, NC 27412
s_suthah@uncg.edu