International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 11 | November - 2023 SJIF Rating: 8.176 ISSN: 2582-3930

Detection of Pneumonia Using Deep Learning

Mohammed Tajuddin ,Venkata Nikhil Amirisetty, Sourav Dutta, Dandu Shashank Arjun, Denepalli Sai Samhitha, V V Sai Aiswarya Gouri, Aditya Shah

ABSTRACT

Pneumonia stands out as one of the prevailing infections worldwide, typically stemming from bacterial, fungal, or viral sources. This inflammatory condition targets the lung tissue, causing swelling in either one or both lungs, with bacterial infection being a frequent culprit. Recognizable symptoms of pneumonia encompass a persistent cough, labored breathing, elevated heart rate, fever, profuse sweating, shivering, loss of appetite, and chest pain.

In the United States alone, pneumonia contributes to over one million hospitalizations and approximately fifty thousand deaths annually. The primary diagnostic method involves the use of chest X-rays, a widely adopted practice to identify and assess pneumonia symptoms. Within the human respiratory system, viral and bacterial pneumonia emerge as the most prevalent types. Medical professionals rely on chest X-ray examinations to differentiate between these two forms, aiding in accurate diagnosis and subsequent treatment.

OBJECTIVE

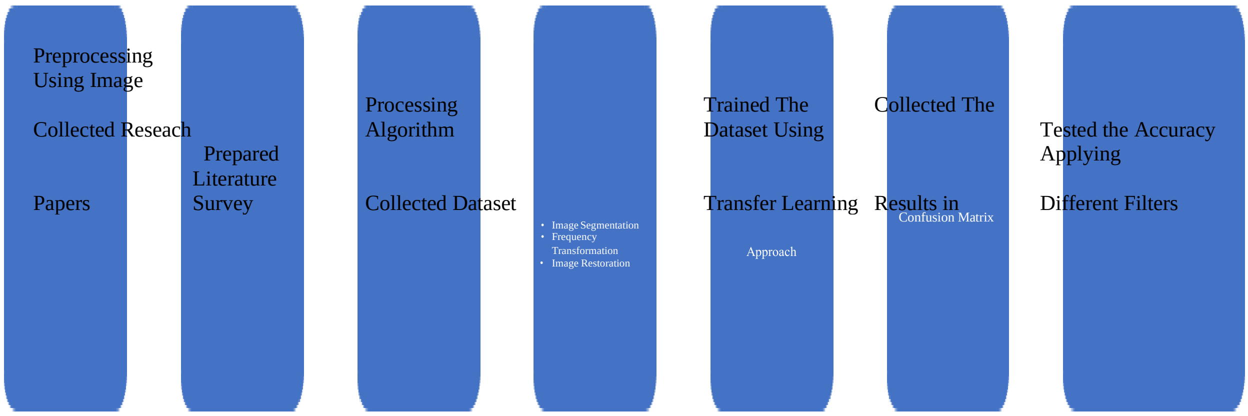
In this undertaking, our research revolves around the creation and visualization of a Convolutional Neural Network (CNN) with the primary goal of identifying symptoms indicative of pneumonia in patients. Furthermore, the CNN will be designed to discriminate between bacterial and viral pneumonia cases. An integral component of our research involves preprocessing the image dataset using a variety of image processing algorithms to enhance the accuracy and effectiveness of the CNN.

The CNN, a specialized deep learning model for image recognition tasks, will be instrumental in analyzing medical images to detect patterns associated with pneumonia symptoms. Leveraging its ability to learn hierarchical features from images, the CNN will contribute to a more nuanced understanding of the distinctions between bacterial and viral pneumonia.

To optimize the performance of our CNN, we will implement diverse image processing algorithms during the dataset preprocessing phase. These algorithms aim to enhance image quality, reduce noise, and extract relevant features that are crucial for accurate diagnosis. The integration of cutting-edge techniques in image processing will fortify the capabilities of our model, ensuring robust and reliable results in the detection and differentiation of pneumonia cases.

Ultimately, our research aspires to harness the power of advanced deep learning and image processing technologies to contribute meaningfully to the field of medical diagnostics, particularly in the identification and categorization of pneumonia, thereby facilitating more informed and targeted healthcare interventions.

RESEARCH WORK FLOW



DATASET USED

https://kaggle.com/paultimothymooney/chest-xray-pneumonia

In the second phase of our research, we are delving into Digital Image Processing (DIP), a method that involves the application of computer algorithms to enhance or extract valuable information from digital images using a digital computer. The process encompasses several key steps, commencing with Image Acquisition, where the images are imported into the system. Following this, the images undergo analysis and manipulation using a variety of algorithms, culminating in the generation of an output, which can either be an altered image or a comprehensive report based on the analysis of the input image.

Our focus in this stage is to implement a series of image manipulation techniques on the dataset previously collected during the initial review. These techniques encompass a spectrum of algorithms designed to improve the quality, extract features, and facilitate a more nuanced analysis of the images. The application of these techniques aims to refine the dataset, ensuring that it is optimized for subsequent stages of our research, particularly for the Convolutional Neural Network (CNN) development.

The manipulation techniques we plan to employ include, but are not limited to, methods for noise reduction, contrast enhancement, edge detection, and feature extraction. Each technique is carefully chosen to address specific aspects of the images and contribute to the overall improvement of the dataset's quality.

By integrating these Digital Image Processing techniques, we aim to create a robust and refined dataset that will serve as a foundation for the subsequent stages of our research, ultimately enhancing the accuracy and reliability of our Convolutional Neural Network in detecting and differentiating pneumonia symptoms in patients.

Basic Architecture of Inception-v3

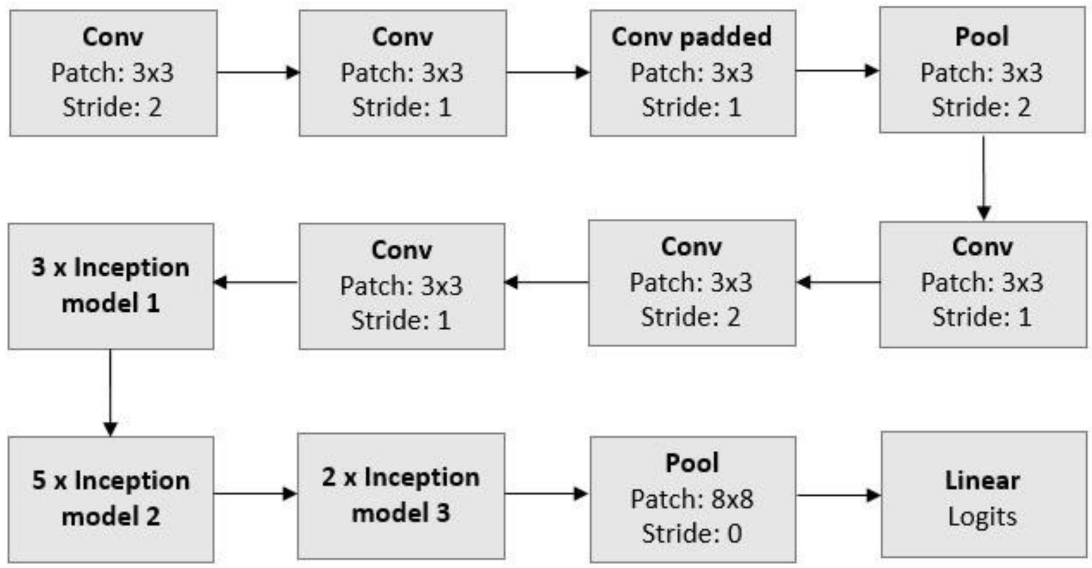


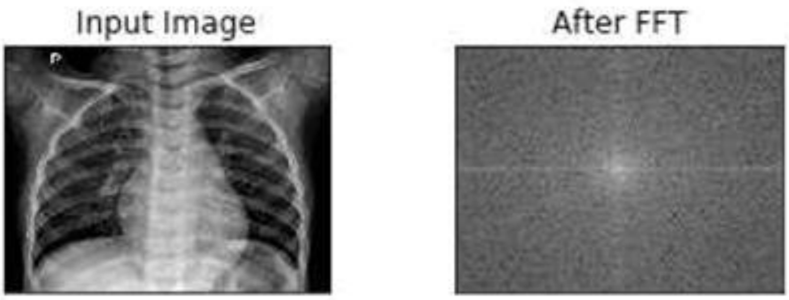
Image Enhancement in the Frequency Domain

Fourier Transform

The process you are describing is the transformation of an image from the spatial domain to the frequency domain, typically achieved through a mathematical operation known as the Fourier Transform. This transformation allows the visualization of an image as a combination of high and low frequencies, providing valuable insights into its frequency components.

In the frequency domain representation of an image, low frequencies are concentrated towards the center of the plot, while high frequencies are dispersed around the edges. This distribution is a result of the spatial arrangement of pixel intensities in the original image. Low-frequency components often correspond to gradual changes or smooth transitions in the image, such as broad color gradients or large objects. On the other hand, high-frequency components represent rapid changes, fine details, or edges in the image.

By converting the image to the frequency domain, this visualization helps in analyzing and understanding the dominant frequency components present in the image. This technique is widely used in image processing and computer vision applications, where the manipulation of frequency components can be beneficial for tasks such as image compression, filtering, and feature extraction. The Fourier Transform is a fundamental tool in signal and image processing for its ability to decompose signals/images into their constituent frequencies.



Absolutely, your understanding is spot on. Once an image has been transformed into the frequency domain through Fourier Transform, various filters can be applied to manipulate specific frequency components. This process is commonly known as frequency domain filtering.

1. High-Pass Filter for Edge Detection:

- High-pass filters allow high-frequency components to pass through while attenuating or eliminating low-frequency components. In the context of image processing, this can be employed for edge detection. Edges and fine details often correspond to high-frequency information in the frequency domain.

2. Low-Pass Filter for Noise Reduction and Blurring:

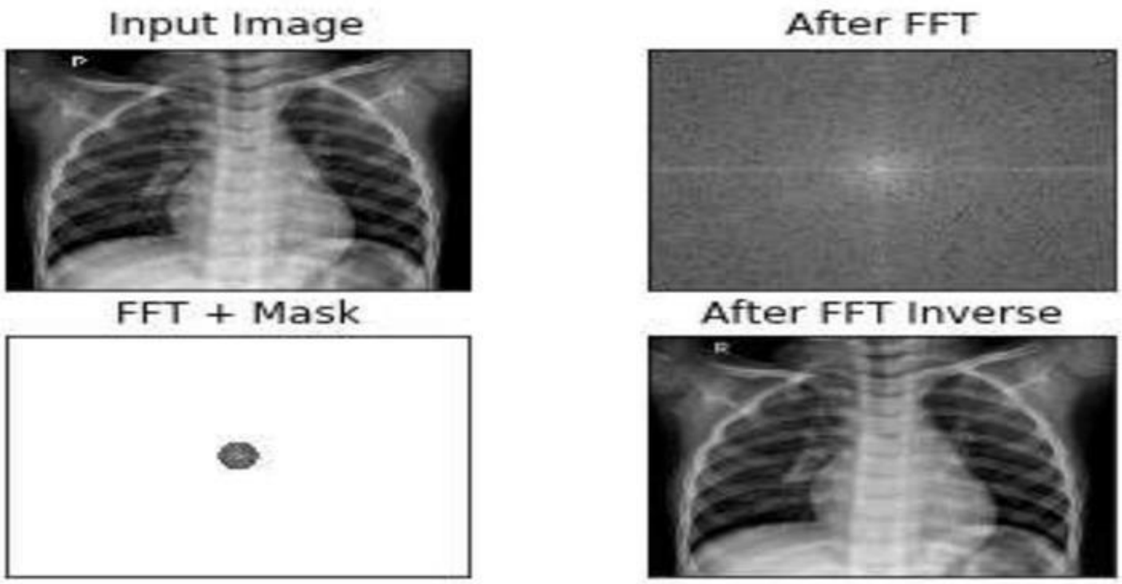
- Low-pass filters, conversely, permit low-frequency components and attenuate high-frequency ones. This property makes them useful for tasks like noise reduction and blurring. Noise, which is often high-frequency, can be diminished, and the overall image can be smoothed or blurred by eliminating high-frequency details.

3. Band-Pass Filter for Selective Frequency Range:

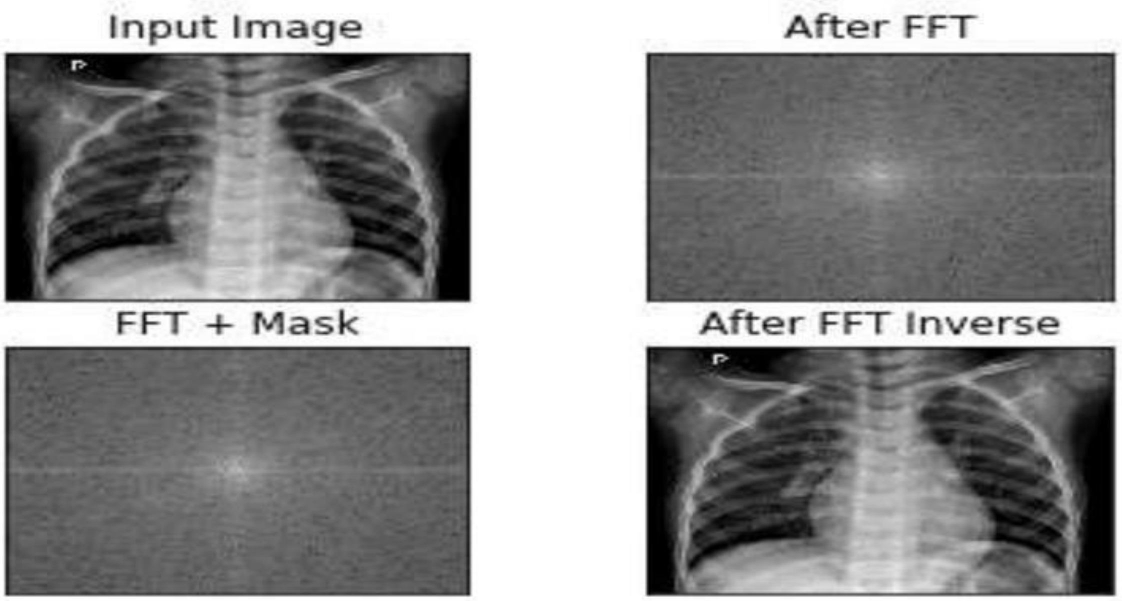
- Band-pass filters allow a certain range of frequencies to pass through while blocking others. This can be useful when you want to focus on a specific range of details within the image. It combines aspects of both low-pass and high-pass filtering.

By strategically applying these filters in the frequency domain and then performing the inverse Fourier Transform, you can achieve various image processing goals. Edge detection, noise reduction, and blurring are just a few examples of the applications made possible by manipulating the frequency components of an image. This approach is widely utilized in fields such as computer vision and medical image processing to enhance or extract specific features from images.

Applying High pass filter



Applying Low pass filter



Applying Band pass filter

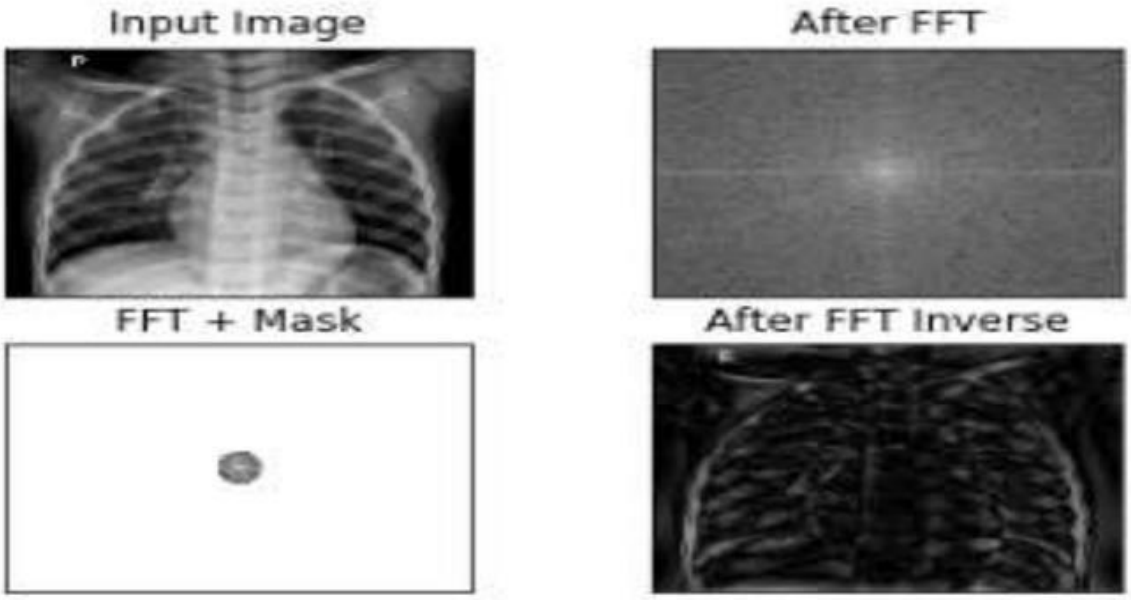
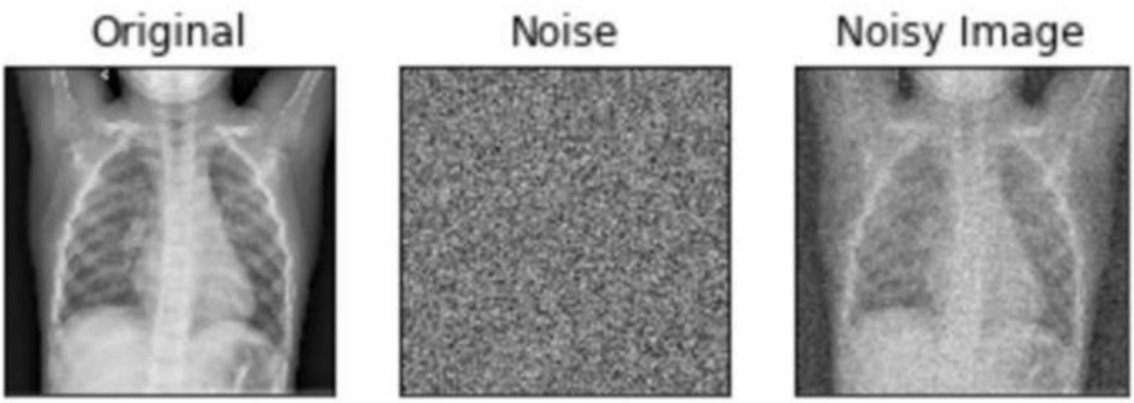


Image Restoration using morphological process

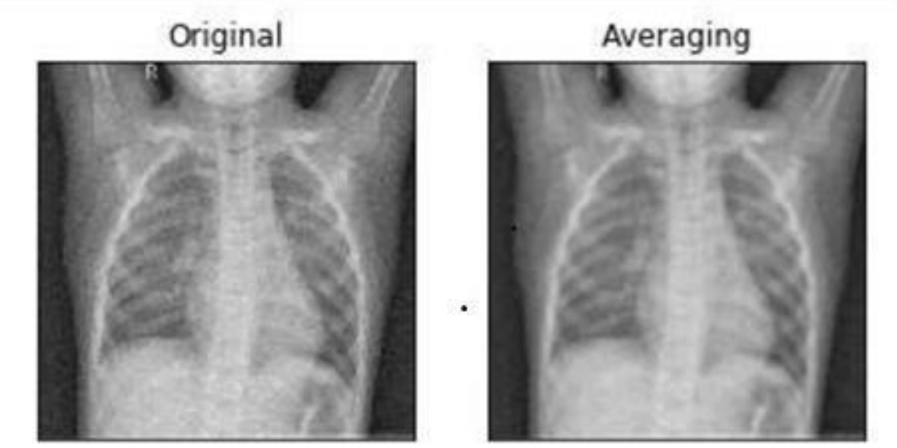
Introducing noise to image

Here we are adding gaussian noise to the input image

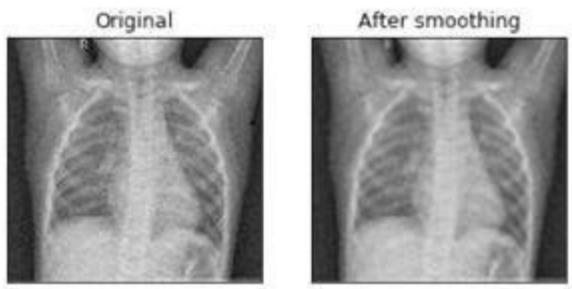


To reduce the noise introduced in the image various filters are tried.

2D Convolution



Averaging Filter

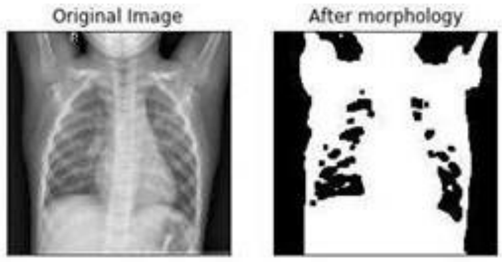


Median Filter

Image Segmentation

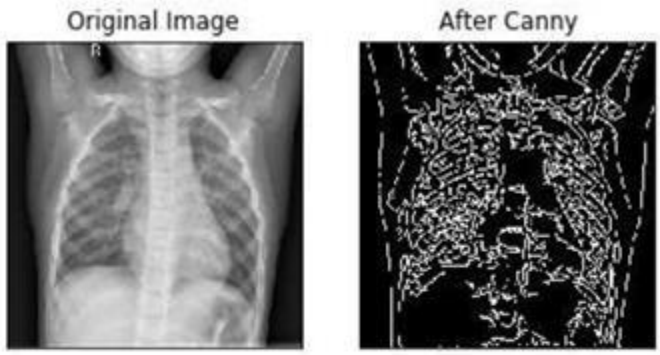
Image Thresholding

Morphological Process (for noisy and original)



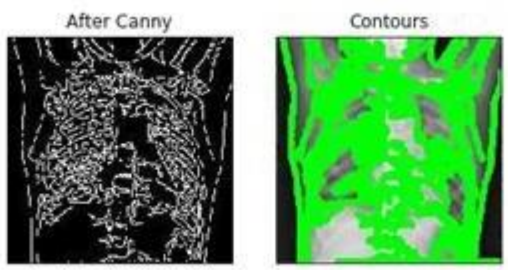
Canny algorithm

Here we have used canny algorithm of the OpenCV library in python for the detection of edges.



Contours

The next block of code is written to draw the image contours using drawContours function of the OpenCV library



Methodology

Your research is both important and timely, especially considering the increasing use of deep learning, specifically Convolutional Neural Networks (CNNs), in medical image analysis. The capability of CNNs to automatically learn hierarchical features from images makes them well-suited for tasks like pneumonia detection and classification.

It's noteworthy that the conventional approach to CNN architectures in medical image classification has often involved trial-and-error processes to design effective networks. Your research acknowledges this and suggests a need for innovation in the architecture to improve classification accuracy and efficiency.

The demonstration you provided, comparing chest X-rays of a normal patient, a patient with bacterial

International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 11 | November - 2023 SJIF Rating: 8.176 ISSN: 2582-3930

pneumonia, and a patient with viral pneumonia, illustrates the real-world application of your research. The visual contrast in the X-ray images makes it clear how critical accurate classification is for medical practitioners in diagnosing and treating respiratory conditions.

In your CNN development, you might consider exploring transfer learning, where a pre-trained CNN model on a large dataset (e.g., ImageNet) is fine-tuned on your medical image dataset. This approach often helps when working with limited medical image data.

Moreover, keep in mind the ethical implications, such as patient privacy and the importance of collaboration with medical professionals to ensure the CNN's outputs align with clinical expertise.

.

How is pneumonia diagnosed and evaluated?

Your primary healthcare provider will initiate the diagnostic process by inquiring about your medical history and symptoms. A comprehensive physical examination will follow, during which your doctor will auscultate your lungs, listening for unusual sounds like crackling, rumbling, or wheezing. In the assessment for pneumonia, the doctor will pay particular attention to identifying abnormal respiratory sounds.

If pneumonia is suspected, your doctor may recommend an imaging test to confirm the diagnosis. Several tests may be prescribed to assess for pneumonia, including:

1. Chest X-ray
 - This examination allows the doctor to visualize your lungs, heart, and blood vessels, aiding in the determination of pneumonia. The radiologist will analyze the X-ray for the presence of white spots in the lungs, known as infiltrates, which are indicative of infection. Additionally, the X-ray can reveal potential complications related to pneumonia, such as abscesses or pleural effusions (accumulation of fluid around the lungs).

Dataset used
https://kaggle.com/paultimothymooney/chest-xray-pneumonia

The dataset comprises 5,863 JPEG images of chest X-rays categorized into two groups: Pneumonia and Normal. These images were sourced from retrospective cohorts of pediatric patients aged one to five years at the Guangzhou Women and Children's Medical Center in Guangzhou. The chest X-rays were acquired as part of routine healthcare procedures for the pediatric population.

The dataset comprises 5,863 JPEG images of chest X-rays categorized into two groups: Pneumonia and Normal. These images were sourced from retrospective cohorts of pediatric patients aged one to five years at the Guangzhou Women and Children's Medical Center in Guangzhou. The chest X-rays were acquired as part of routine healthcare procedures for the pediatric population.In the chest X-ray image analysis, a meticulous quality control process was implemented initially. This involved the removal of all scans deemed to be of low quality or unreadable. Subsequently, the images underwent a grading process conducted by two expert physicians to assign diagnoses. Only after this rigorous grading by the experts were the images considered suitable for training the artificial intelligence (AI) system.

To ensure the accuracy of the grading process and account for any potential errors, a third expert conducted an evaluation of the image set. This additional step was taken to enhance the reliability of the diagnostic information and to minimize the risk of inaccuracies in the training data for the AI system..

Chest X-Ray Dataset

Type	Test	Train
Normal	234	1341
Pneumonia	390	3875

Table 1. Number of images under test and train files

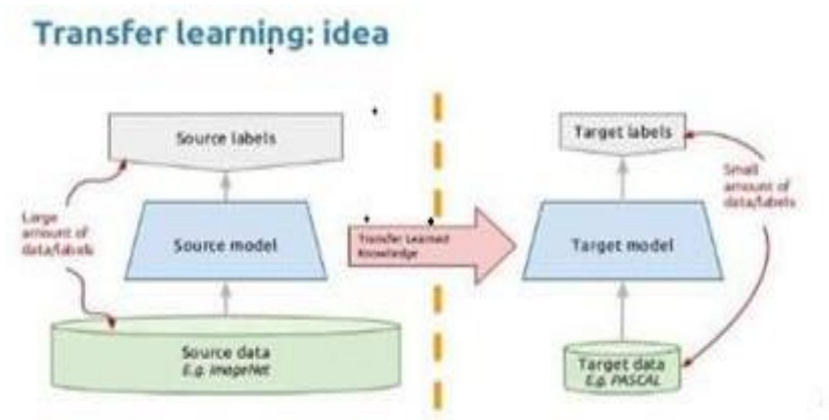
Expected Result

To find out how accurately the applied algorithm can distinguish between patients diagnosed with pneumonia or not on the basis of the chest x rays.

ALGORITHM USED

Transfer Learning

Transfer learning is a machine learning technique that involves leveraging a model developed for one task as the initial foundation for a model aimed at a different task. This method is particularly prominent in the realm of deep learning, where pre-existing models are employed as starting points for tasks in computer vision and natural language processing. This is especially beneficial due to the considerable computational and time resources typically required for the development of neural network models for these complex problems. Utilizing pre-trained models not only expedites the training process but also capitalizes on the significant advancements in performance and accuracy that these models have achieved on related problems.



Goals:
1. Image Classification without Custom Models:
- The primary objective is to perform image classification without developing a custom model, utilizing pre-existing models provided by Keras.

2. High Accuracy Across Diverse Images:
- Achieving a high level of accuracy in classifying a wide range of images is a key goal. This encompasses diverse image types and content.

Advantages of Transfer Learning:
1. Simplicity of Integration:
- Transfer learning is straightforward to incorporate into the research, streamlining the development process and allowing for quicker implementation.

2. Quick Attainment of High Performance:
- By leveraging pre-trained models, the research aims to achieve comparable or enhanced performance swiftly, capitalizing on the knowledge embedded in these models.

3. Reduced Dependency on Labeled Data:
- Transfer learning reduces the reliance on extensive labeled datasets, making it more feasible to achieve robust model performance with limited labeled data.

4. Versatility in Use Cases:
- The advantages extend to various applications, including transfer learning, prediction tasks, and feature extraction, highlighting the flexibility and broad utility of this approach.

Implementation

Importing libraries

```
In [4]: import pandas as pd
        import cv2
        import numpy as np
        import os
        from random import shuffle
        from tqdm import tqdm
        import scipy
        import skimage
        from skimage.transform import resize
```

A function to label the images in the dataset based upon whether the person is normal or diagnosed with pneumonia

Normal patient has been labeled 0

Pneumonia diagnosed patient has been labeled 1 Testing

Accuracy with different Filters

```
In [8]: def get_label(Dir):
            for nextdir in os.listdir(Dir):
                if not nextdir.startswith('.'):
                    if nextdir in ['NORMAL']:
                        label = 0
                    elif nextdir in ['PNEUMONIA']:
                        label = 1
                    else:
                        label = 2
            return nextdir, label
```

A function which takes the dataset directory as an argument written to perform preprocessing on the image dataset which includes

- Image grey-scaling
- Image resizing
- Storing the greyscale values and labels in x and y variables

Combining the above two functions to write the function get_data with directory as an argument returning the x and y variables containing arrays of image greyscales and labels respectively.

International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 11 | November - 2023 SJIF Rating: 8.176 ISSN: 2582-3930

```
In [10]:
def get_data(Dir):
    X = []
    y = []
    for nextDir in os.listdir(Dir):
        if not nextDir.startswith('.'):
            if nextDir in ['NORMAL']:
                label = 0
            elif nextDir in ['PNEUMONIA']:
                label = 1
            else:
                label = 2

            temp = Dir + nextDir

            for file in tqdm(os.listdir(temp)):
                img = cv2.imread(temp + '/' + file)
                if img is not None:
                    img = skimage.transform.resize(img, (150, 150, 3))
                    #img_file = scipy.misc.imresize(arr=img_file, size=(299, 299, 3))
                    img = np.asarray(img)
                    X.append(img)
                    y.append(label)

    X = np.asarray(X)
    y = np.asarray(y)
    return X,y
```

Applying the get_data function to train and test directories to get (X_train, y_train) and (X_test, y_test) pairs respectively.

```
In [11]: X_train, y_train = get_data(TRAIN_DIR)

100%|██████████| 3876/3876 [07:54<00:00,  8.18it/s]
100%|██████████| 1342/1342 [09:48<00:00,  2.85it/s]

In [12]: X_test , y_test = get_data(TEST_DIR)

100%|██████████| 390/390 [00:36<00:00, 12.23it/s]
100%|██████████| 234/234 [01:41<00:00,  2.08it/s]

In [37]: print(X_train.shape,'\n',X_test.shape)

(5216, 3, 150, 150)
 (624, 3, 150, 150)

In [57]: print(y_train.shape,'\n',y_test.shape)

(5216, 2, 2)
 (624, 2, 2)
```

Importing to_categorical function from keras library which returns a binary matrix representation of the input vector supplied. Applying the to_categorical function on the y_train and y_test arrays (label arrays)

```
In [15]: from keras.utils.np_utils import to_categorical

Using TensorFlow backend.
/anaconda3/lib/python3.7/site-packages/tensorflow/python/framework/dtypes.py:516: FutureWarning: Passing (type, 1) or '1type' a
s a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/anaconda3/lib/python3.7/site-packages/tensorflow/python/framework/dtypes.py:517: FutureWarning: Passing (type, 1) or '1type' a
s a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/anaconda3/lib/python3.7/site-packages/tensorflow/python/framework/dtypes.py:518: FutureWarning: Passing (type, 1) or '1type' a
s a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/anaconda3/lib/python3.7/site-packages/tensorflow/python/framework/dtypes.py:519: FutureWarning: Passing (type, 1) or '1type' a
s a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/anaconda3/lib/python3.7/site-packages/tensorflow/python/framework/dtypes.py:520: FutureWarning: Passing (type, 1) or '1type' a
s a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/anaconda3/lib/python3.7/site-packages/tensorflow/python/framework/dtypes.py:525: FutureWarning: Passing (type, 1) or '1type' a
s a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
/anaconda3/lib/python3.7/site-packages/tensorboard/compat/tensorflow_stub/dtypes.py:541: FutureWarning: Passing (type, 1) or '1
type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/anaconda3/lib/python3.7/site-packages/tensorboard/compat/tensorflow_stub/dtypes.py:542: FutureWarning: Passing (type, 1) or '1
type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/anaconda3/lib/python3.7/site-packages/tensorboard/compat/tensorflow_stub/dtypes.py:543: FutureWarning: Passing (type, 1) or '1
type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/anaconda3/lib/python3.7/site-packages/tensorboard/compat/tensorflow_stub/dtypes.py:544: FutureWarning: Passing (type, 1) or '1
type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
```

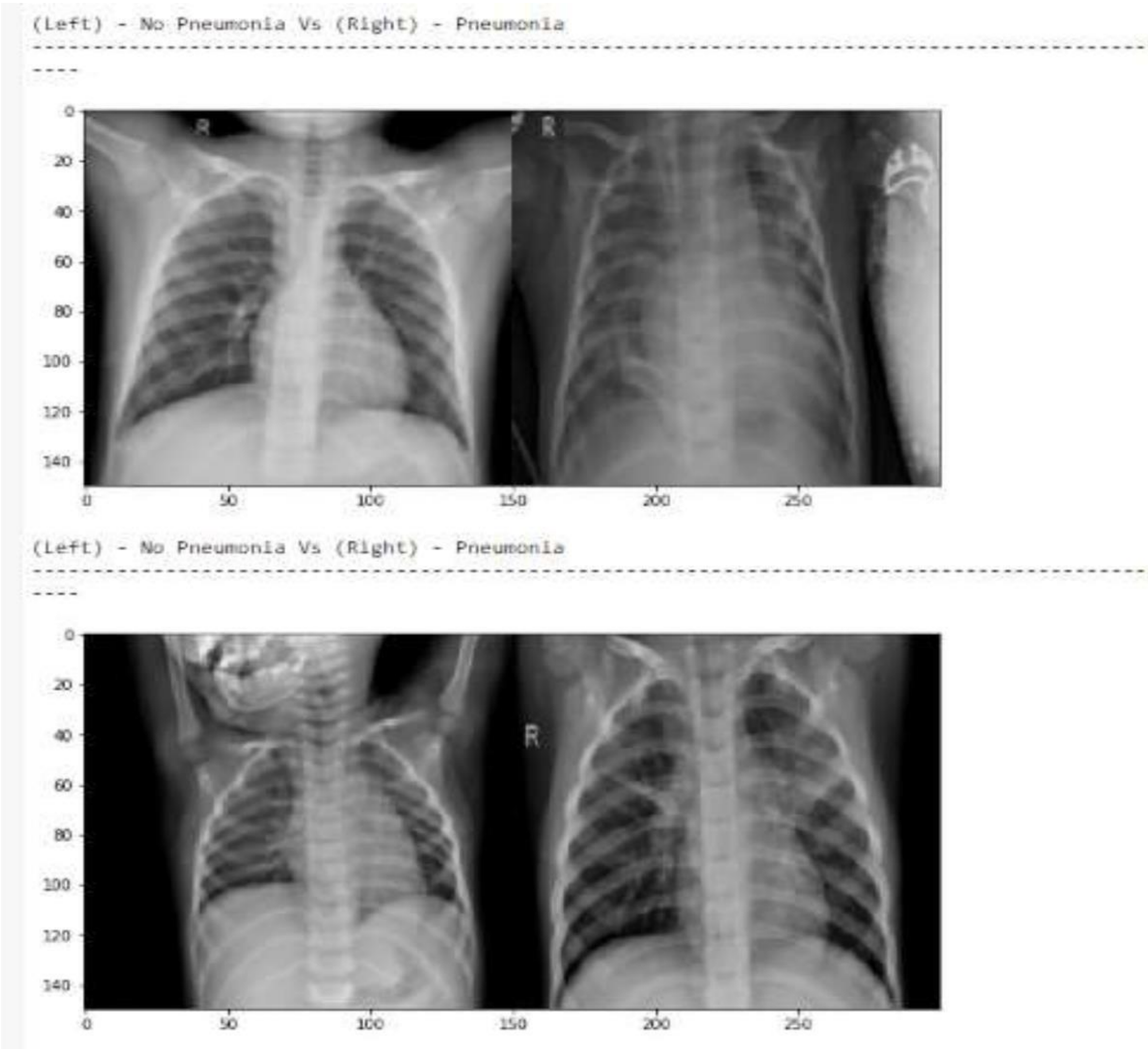
```
In [55]: y_train = to_categorical(y_train, 2)
         y_test = to_categorical(y_test, 2)
```

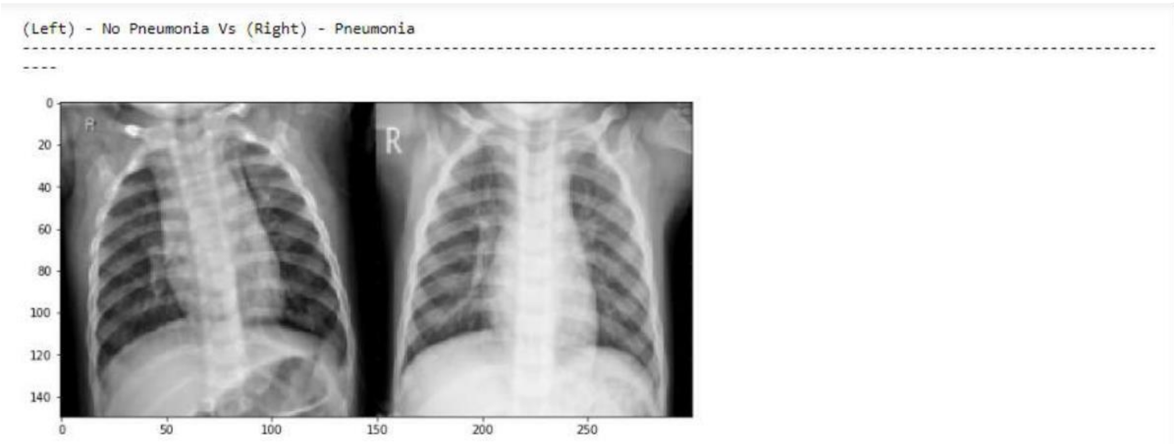
Using matplotlib to demonstrate pneumonia and no pneumonia images side by side.

```
In [17]: Pimages = os.listdir(TRAIN_DIR + "PNEUMONIA")
         Nimages = os.listdir(TRAIN_DIR + "NORMAL")
```

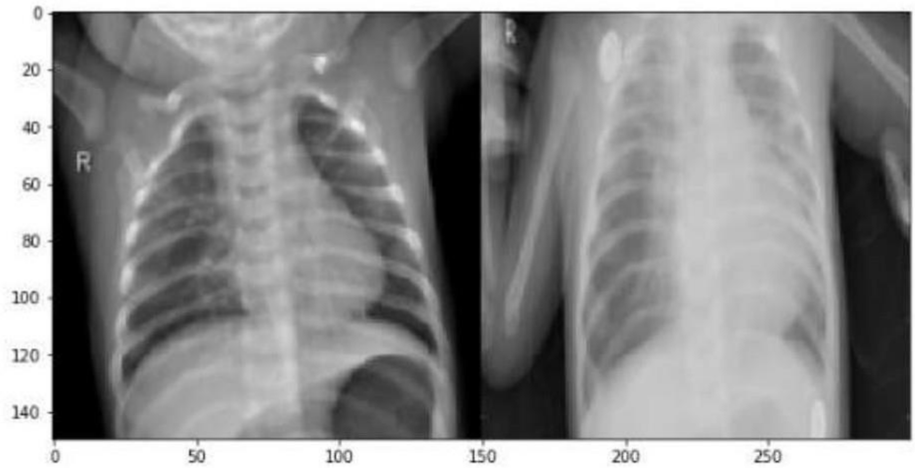
```
In [20]: import matplotlib.pyplot as plt
         def plotter(i):
             imagep1 = cv2.imread(TRAIN_DIR+"PNEUMONIA/"+Pimages[i])
             imagep1 = skimage.transform.resize(imagep1, (150, 150, 3) , mode = 'reflect')
             imagen1 = cv2.imread(TRAIN_DIR+"NORMAL/"+Nimages[i])
             imagen1 = skimage.transform.resize(imagen1, (150, 150, 3))
             pair = np.concatenate((imagen1, imagep1), axis=1)
             print("(Left) - No Pneumonia Vs (Right) - Pneumonia")
             print("-----------------------------------------------------------------------------
             plt.figure(figsize=(10,5))
             plt.imshow(pair)
             plt.show()
         for i in range(5,10):
             plotter(i)
```

In the below images, it can be seen that the right-side images are chest x-rays of the person diagnosed with pneumonia. It is differentiated by observing the increased amount of cloudiness in the lungs.

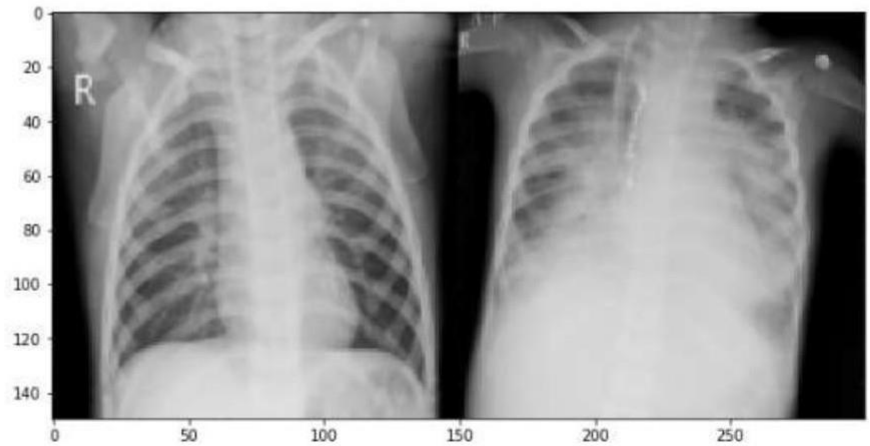

(Left) - No Pneumonia Vs (Right) - Pneumonia
(Left) - No Pneumonia Vs (Right) - Pneumonia


(Left) - No Pneumonia Vs (Right) - Pneumonia

```
(Left) - No Pneumonia Vs (Right) - Pneumonia
----------------------------------------------------------------------------------------------------
----
```



```
(Left) - No Pneumonia Vs (Right) - Pneumonia
----------------------------------------------------------------------------------------------------
----
```



Models often benefit from reducing the learning rate once learning stagnates. For this, we used ReduceLROnPlateau which monitors accuracy setting the factor by which to reduce learning rate as 0.1. Verbose is 1: update messages. Patience: number of epochs that produced the monitored quantity with no improvement after which training will be stopped.

```
In [21]: from keras.callbacks import ReduceLROnPlateau , ModelCheckpoint , LearningRateScheduler
         lr_reduce = ReduceLROnPlateau(monitor='val_acc', factor=0.1, epsilon=0.0001, patience=1, verbose=1)

/anaconda3/lib/python3.7/site-packages/keras/callbacks.py:1335: UserWarning: `epsilon` argument is deprecated and will be remov
ed, use `min_delta` instead.
  warnings.warn('`epsilon` argument is deprecated and '
```

Transfer learning

Transfer learning is a machine learning approach where a model created for a particular task is adapted as the initial architecture for a model aimed at a different task. Widely employed in deep learning, particularly for tasks in computer vision and natural language processing, this strategy harnesses pre-trained models. It takes advantage of the considerable computational and time resources needed to construct neural network models for these challenges, yielding significant performance enhancements for related problems.

We specified the transfer learning weights filepath and called ModelCheckPoint to save model to the given filepath after every epoch.

```
In [22]: filepath="transferlearning_weights.hdf5"
         checkpoint = ModelCheckpoint(filepath, monitor='val_acc', verbose=1, save_best_only=True, mode='max')
```

Inception v3 is a widely-used image recognition model that has been shown to attain greater than 78.1% accuracy on the ImageNet dataset. The model is the culmination of many ideas developed by multiple researchers over the years.

Importing and preparing the base InceptionV3 model

```
In [39]: from keras.applications.inception_v3 import InceptionV3
         # create the base pre-trained model
         base_model = InceptionV3(weights=None, include_top=False , input_shape=(150, 150,3))
```

```
In [40]: x = base_model.output
         x = Dropout(0.5)(x)
         x = GlobalAveragePooling2D()(x)
         x = Dense(128, activation='relu')(x)
         x = BatchNormalization()(x)
         predictions = Dense(2, activation='sigmoid')(x)
```

Configuring the base model weights and compiling the model set for accuracy prediction

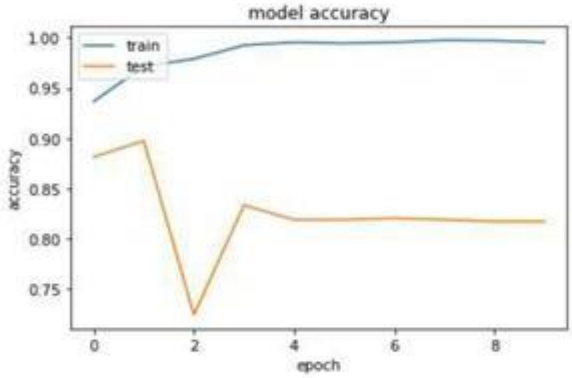
```
In [42]: base_model.load_weights("chest_xray/inception_v3_weights.h5")

In [43]: model = Model(inputs=base_model.input, outputs=predictions)

In [44]: model.compile(loss='categorical_crossentropy',
                       optimizer='adam',
                       metrics=['accuracy'])
```

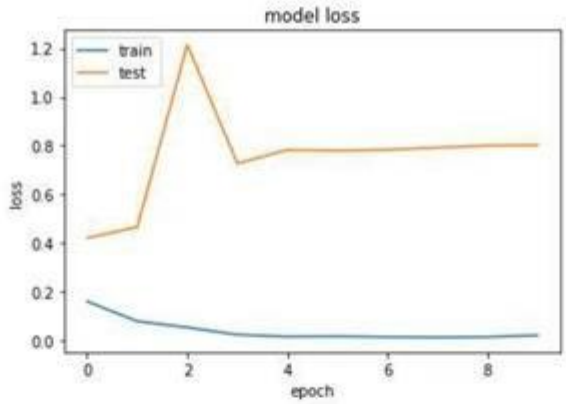
```
In [45]: print(model.summary())

Model: "model_1"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            (None, 150, 150, 3)  0
__________________________________________________________________________________________________
conv2d_1 (Conv2D)               (None, 74, 74, 32)   864         input_1[0][0]
__________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 74, 74, 32)   96          conv2d_1[0][0]
__________________________________________________________________________________________________
activation_1 (Activation)       (None, 74, 74, 32)   0           batch_normalization_1[0][0]
__________________________________________________________________________________________________
conv2d_2 (Conv2D)               (None, 72, 72, 32)   9216        activation_1[0][0]
__________________________________________________________________________________________________
batch_normalization_2 (BatchNor (None, 72, 72, 32)   96          conv2d_2[0][0]
__________________________________________________________________________________________________
activation_2 (Activation)       (None, 72, 72, 32)   0           batch_normalization_2[0][0]
__________________________________________________________________________________________________
conv2d_3 (Conv2D)               (None, 72, 72, 64)   18432       activation_2[0][0]
__________________________________________________________________________________________________
batch_normalization_3 (BatchNor (None, 72, 72, 64)   192         conv2d_3[0][0]
__________________________________________________________________________________________________
activation_3 (Activation)       (None, 72, 72, 64)   0           batch_normalization_3[0][0]
__________________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D)  (None, 35, 35, 64)   0           activation_3[0][0]
__________________________________________________________________________________________________
```

Setting up batch size and number of epochs and calling model.fit to train the model and storing the output in history variable. The batch size has been set up to be 64 and number of epochs are 10.

```
In [46]: batch_size = 64
         epochs = 10

In [47]: history = model.fit(X_train, y_train, validation_data = (X_test , y_test) ,callbacks=[lr_reduce,checkpoint] ,
                             epochs=epochs)

WARNING:tensorflow:From /anaconda3/lib/python3.7/site-packages/tensorflow/python/ops/math_grad.py:1250: add_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Train on 5216 samples, validate on 624 samples
Epoch 1/10
5216/5216 [==============================] - 692s 133ms/step - loss: 0.1600 - acc: 0.9367 - val_loss: 0.4200 - val_acc: 0.8814

Epoch 00001: val_acc improved from -inf to 0.88141, saving model to transferlearning_weights.hdf5
Epoch 2/10
5216/5216 [==============================] - 761s 146ms/step - loss: 0.0782 - acc: 0.9714 - val_loss: 0.4654 - val_acc: 0.8974

Epoch 00002: val_acc improved from 0.88141 to 0.89744, saving model to transferlearning_weights.hdf5
Epoch 3/10
5216/5216 [==============================] - 793s 152ms/step - loss: 0.0527 - acc: 0.9789 - val_loss: 1.2146 - val_acc: 0.7244
```

Using matplotlib to trace the model accuracy and model loss with the respect to the number of epochs





Representing the model accuracy by plotting the confusion matrix with x axis is the predicted result and y axis as the true result.

First quadrant represents that patient is not diagnosed with pneumonia and model predicted for the same.

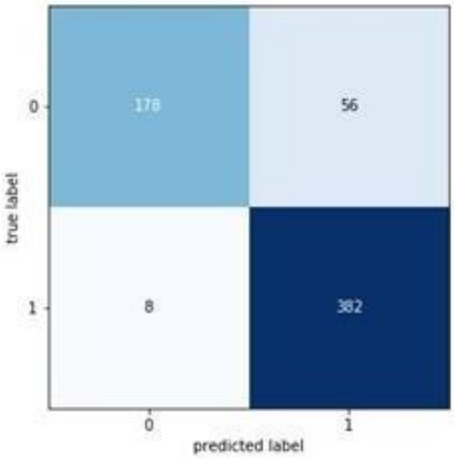
Second quadrant shows the number of cases where person is not diagnosed with pneumonia but model predicted that patient has pneumonia.

Third quadrant is most critical one as it shows the number of cases where the patient is diagnosed with pneumonia and model predicted opposite.

Second quadrant shows the number of cases where person is diagnosed with pneumonia and model predicts for the same.

```
In [50]: from sklearn.metrics import confusion_matrix
         pred = model.predict(X_test)
         pred = np.argmax(pred,axis = 1)
         y_true = np.argmax(y_test,axis = 1)
```

```
In [52]: CM = confusion_matrix(y_true, pred)
         from mlxtend.plotting import plot_confusion_matrix
         fig, ax = plot_confusion_matrix(conf_mat=CM ,  figsize=(5, 5))
         plt.show()
```



Accuracy of the model when the person is not actually suffering from pneumonia came out to be 76%

```
In [1]: 178/(178+56)
Out[1]: 0.7606837606837606
```

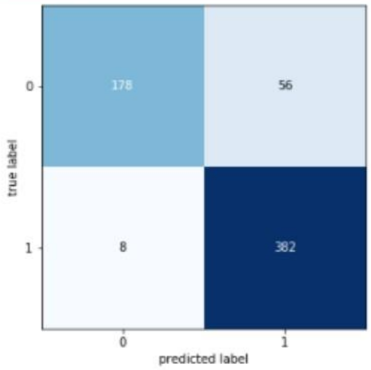
Accuracy of the model when the person is actually suffering from pneumonia came out to be 98%

```
In [54]: 382/(382+8)
Out[54]: 0.9794871794871794
```

International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 11 | November - 2023 SJIF Rating: 8.176 ISSN: 2582-3930

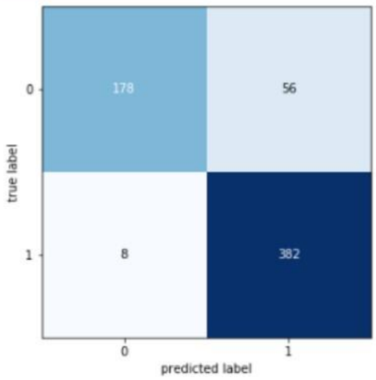
```
In [47]: fx_test , fy_test = get_data("Median/")
fy_test = to_categorical(fy_test, 2)
fx_test=fx_test.reshape(624,150,150,3)

fpred = model.predict(fx_test)
fpred = np.argmax(fpred,axis = 1)
fy_true = np.argmax(fy_test,axis = 1)
CM = confusion_matrix(y_true, pred)
from mlxtend.plotting import plot_confusion_matrix
fig, ax = plot_confusion_matrix(conf_mat=CM ,  figsize=(5, 5))
plt.show()
```

```
100%|
| 234/234 [00:10<00:00, 21.91it/s]
100%|
| 390/390 [00:16<00:00, 23.22it/s]
```




```
In [49]: fx_test , fy_test = get_data("Blurring/")
fy_test = to_categorical(fy_test, 2)
fx_test=fx_test.reshape(624,150,150,3)

fpred = model.predict(fx_test)
fpred = np.argmax(fpred,axis = 1)
fy_true = np.argmax(fy_test,axis = 1)
CM = confusion_matrix(y_true, pred)
from mlxtend.plotting import plot_confusion_matrix
fig, ax = plot_confusion_matrix(conf_mat=CM ,  figsize=(5, 5))
plt.show()
```

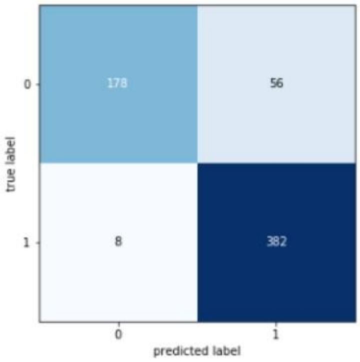
```
100%|
| 234/234 [00:09<00:00, 24.07it/s]
100%|
| 390/390 [00:16<00:00, 23.76it/s]
```



International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 11 | November - 2023 SJIF Rating: 8.176 ISSN: 2582-3930

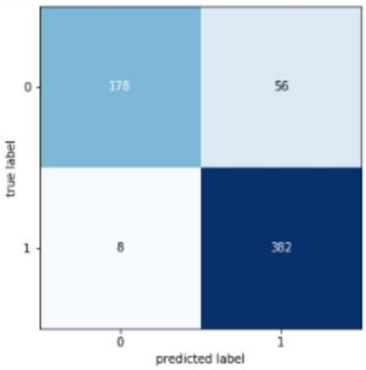
In [48]:
```
fx_test , fy_test = get_data("Bilateral/")
fy_test = to_categorical(fy_test, 2)
fx_test=fx_test.reshape(624,150,150,3)

fpred = model.predict(fx_test)
fpred = np.argmax(fpred,axis = 1)
fy_true = np.argmax(fy_test,axis = 1)
CM = confusion_matrix(y_true, pred)
from mlxtend.plotting import plot_confusion_matrix
fig, ax = plot_confusion_matrix(conf_mat=CM ,  figsize=(5, 5))
plt.show()
```

```
100%|
| 234/234 [00:11<00:00, 20.74it/s]
100%|
| 390/390 [00:16<00:00, 24.35it/s]
```

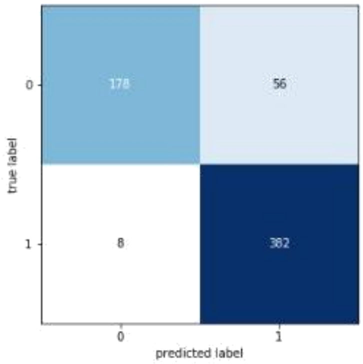


In [46]:
```
fx_test , fy_test = get_data("Affine/")
fy_test = to_categorical(fy_test, 2)
fx_test=fx_test.reshape(624,150,150,3)

fpred = model.predict(fx_test)
fpred = np.argmax(fpred,axis = 1)
fy_true = np.argmax(fy_test,axis = 1)
CM = confusion_matrix(y_true, pred)
from mlxtend.plotting import plot_confusion_matrix
fig, ax = plot_confusion_matrix(conf_mat=CM ,  figsize=(5, 5))
plt.show()
```

```
100%|
| 234/234 [00:10<00:00, 21.34it/s]
100%|
| 390/390 [00:17<00:00, 22.75it/s]
```



```
In [45]: fx_test , fy_test = get_data("2DConv/")
         fy_test = to_categorical(fy_test, 2)
         fx_test=fx_test.reshape(624,150,150,3)

         fpred = model.predict(fx_test)
         fpred = np.argmax(fpred,axis = 1)
         fy_true = np.argmax(fy_test,axis = 1)
         CM = confusion_matrix(y_true, pred)
         from mlxtend.plotting import plot_confusion_matrix
         fig, ax = plot_confusion_matrix(conf_mat=CM ,  figsize=(5, 5))
         plt.show()

100%|████████████████████████████████████████████████████████████
██████| 234/234 [00:10<00:00, 23.28it/s]
100%|████████████████████████████████████████████████████████████
██████| 390/390 [00:20<00:00, 19.27it/s]
```



Results

Therefore, by implementing a Convolutional Neural Network on the provided dataset, we attain a 97% accuracy in correctly identifying cases diagnosed with pneumonia and a 76% accuracy in correctly classifying cases that are not diagnosed with pneumonia.

Conclusion

This research represents a modest stride in leveraging the combined potential of image processing and deep learning within the medical domain, with much more anticipated in the future. The model developed demonstrated notable success, achieving an accuracy of approximately 98% when diagnosing pneumonia in patients. The synergy of Transfer Learning and image processing algorithms served as the primary catalyst for the research's accomplishments. It is evident that Transfer Learning is poised to be a pivotal force propelling the success and widespread adoption of machine learning and deep learning within various industries, particularly in the field of medicine.

REFERENCES

[1] Souza, J. C., Diniz, J. O. B., Ferreira, J. L., da Silva, G. L. F., Silva, A. C., & de Paiva,

A. C. (2019). An automatic method for lung segmentation and reconstruction in chest X-Ray using deep neural networks. Computer Methods and Programs in Biomedicine.

[2] Zar, H. J. (2017). Bacterial and viral pneumonia: New insights from the Drakenstein Child Health Study. Paediatric respiratory reviews, 24, 8-10.

[3] Dear, J. D. (2014). Bacterial pneumonia in dogs and cats.

[4] Wilmes, D., Coche, E., Rodriguez-Villalobos, H., & Kanaan, N. (2018). Bacterial pneumonia in kidney transplant recipients. Respiratory medicine, 137, 89-94.

[5] Henig, O., & Kaye, K. S. (2017). Bacterial pneumonia in older adults. Infectious Disease Clinics, 31(4), 689-713.

[6] Wong, J. L., & Evans, S. E. (2017). Bacterial pneumonia in patients with cancer: novel risk factors and management. Clinics in chest medicine, 38(2), 263-277.

[7] Das, D., & Howlett, D. C. (2009). Chest X-ray manifestations of pneumonia. Surgery- Oxford International Edition, 27(10), 453-455.

[8] Falsey, A. R. (2007). Community-acquired viral pneumonia. Clinics in geriatric medicine, 23(3), 535-552.

[9] Nayak, S., Pradhan, D., Singh, H., & Reddy, M. S. (2019). Computational screening of potential drug targets for pathogens causing bacterial pneumonia. Microbial pathogenesis, 130, 271-282.

[10] Erdem, S. B., Can, D., Girit, S., Çatal, F., Şen, V., Pekcan, S., & Ersu, R. (2018). Does atopy affect the course of viral pneumonia? Allergologia et immunopathologia, 46(2), 119- 126.