PNEUMONIA DETECTION USING TRANSFER LEARNING

Submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering with Artificial Intelligence

By

BOPPANA BHARAT SATYA (Reg. No - 40731015)

KOPALLI ROHIT SAI KALYAN (Reg. No - 40111071)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A++" by NAAC | 12B Status by UGC |
Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600119

APRIL - 2024

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Category - I University by UGC
Accredited "A++" by NAAC | Approved by AICTE
www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Boppana Bharat Satya (40731015) And Rohit Sai Kalyan Kopalli (40111071)** who carried out the Project Phase-2 entitled "**PNEUMONIA DETECTION USING TRANSFER LEARNING**" under my supervision from November 2023 to April 2024.

Internal Guide

Dr. G. MATHIVANAN M.E., Ph.D.,

Head of the Department

Dr. L. LAKSHMANAN M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner **External Examiner**

DECLARATION

I, **BOPPANA BHARAT SATYA (40731015),** hereby declare that the Project Phase-2 Report entitled "**PNEUMONIA DETECTION USING TRANSFER LEARNING**" done by me under the guidance of **Dr. G. MATHIVANAN M.E., Ph.D.,** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering with Artificial Intelligence**.

DATE:

PLACE: Chennai **SIGNATURE OF CANDIDATE**

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of Sathyabama Institute of Science and Technology** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D., Dean**, School of Computing, **Dr. L. LAKSHMANAN M.E., Ph.D.,** Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progress reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. G. MATHIVANAN M.E., Ph.D**., for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my phase-2 project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering with Artificial Intelligence** who were helpful in many ways for the completion of the project.

ABSTRACT

A bacterial infection in the lungs results in an illness known as pneumonia. The effectiveness of treatment depends in large part on early diagnosis. For a variety of reasons, including the disease's appearance being ambiguous in chest X-ray images or being mistaken for another illness, the diagnosis may be arbitrary. Thus, in order to assist clinicians, computer-aided diagnosis systems are required. AI algorithms can automatically diagnose pneumonia or other lung-related diseases in patients by analyzing their chest X-ray scans. Because lives are at stake, the algorithm needs to be extremely accurate. In this project, we employ transfer learning (TL), a machine learning (ML) technique that allows knowledge from one task to be applied to another, improving performance on related tasks. Detected Pneumonia from Chest X-Ray images using Custom Deep Convolutional Neural Network and by retraining pre-trained model with images of X-ray. For retraining removed output layers, feezed first few layers and fine-tuned model for two new label classes (Pneumonia and Normal).

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
CNN	Convolutional Neural Network
MRI	Magnetic Resonance Imaging
TL	Transfer Learning

CHAPTER 1
INTRODUCTION

Acute pulmonary infections, such as pneumonia, could be brought on by viruses, bacteria, or fungi and infect the lungs, leading to the pleural effusion, a situation in that fluid fills the lung, and inflammation of the air sacs. It is the cause of over 15percent of deaths in children under five years of age. Pneumonia is more common in developing and underdeveloped nations due to factors like poor environmental situations, pollution, and overcrowding, as well as a lack of access to healthcare. Therefore, keeping the disease from becoming fatal could be highly aided by early diagnosis along with treatment. A diagnosis of lung disease is often made by lung radiological examination by utilizing the computed tomography (CT), magnetic resonance imaging (MRI), or radiography (X-rays). An examination of the lungs that is non-invasive and reasonably priced is X-ray imaging.

In this project, we have outlined our method for diagnosing pneumonia and explained how the model's performance is significantly impacted by the lung image size. We discovered that the difference between images showing pneumonia and those showing it is fairly subtle; large images can provide more detailed information. However, when working with large images, the computation cost also increases exponentially.

One useful artificial intelligence tool that is essential to the resolution of many challenging computer vision issues is deep learning. Convolutional neural networks (CNNs), in particular, are DL (Deep Learning)models that are broadly utilized for a range of image classification tasks. But these models only function at their best while they have access to a lot of data. Such large labeled data sets are hard to come by in biomedical image classification problems because each image must be classified by a specialist doctor, a costly and time-consuming process. There is a workaround for this problem: transfer learning.

This technique solves problems involving small datasets by reusing a model that was “trained on a large dataset and applying its network weights. Since CNN models were trained on massive datasets like ImageNet, which has over 14 million images”, they have been frequently utilized for the biomedical image classification tasks.

1.1 Background And Motivation

CNN is a popular research area in the field of deep learning. CNN is a deep learning architecture that can be used together to recognize human lungs x-ray. CNN is used to extract features from signals. This technology has many potential applications, such as in healthcare, sports, and security. The use of deep learning models for human activity recognition has shown great progress in recent years. The combination of TL and CNN has been found to be effective in recognizing pneumonia with high accuracy. This technology has the potential to improve the quality of life for people by providing personalized healthcare.

1.2 Problem Statement

The problem statement for PNEUMONIA DETECTION using CNN is to accurately recognize and classify human lungs should contain Pneumonia or Normal . The traditional pattern recognition methods have limitations in recognizing complex images. Therefore, deep learning models such as CNN are used to overcome these limitations. The standards of designing the architecture, deciding the appropriate language for the effective architecture which could extract features and recognize time-sequential features with high accuracy is followed as per the standards IEEE Std. 1016-1998 [22] Recommended Practice for Software Design Descriptions. The goal is to develop a system that can recognize different forms of human chest x-ray with high accuracy.

1.3 Objective and Scope

Objective

The objective of Pneumonia Detection using Transfer learning is to develop an

accurate and efficient output that can recognize and classify pneumonia is there or not . This detection should be able to detect lungs x-ray with high accuracy. The use of deep learning models such as CNN can help to overcome the limitations of traditional pattern recognition methods. In this work, we have presented our approach for identifying pneumonia and understanding how the lung image size plays an important role for the model performance. We found that the distinction is quite subtle for images among presence or absence of pneumonia, large image can be more beneficial for deeper information.

Scope

The scope of Pneumonia detection using CNN is broad and has many potential applications in various fields. In healthcare, it can be used to monitor patients' activities and provide personalized healthcare. For biomedical image classification problems, such a vast amount of labeled data is difficult to acquire because it requires that expert doctors classify each image, which is an expensive and time-consuming task. It can also be used in sports to monitor athletes' activities and improve their performance. The use of deep learning models for detection has shown great progress in recent years.

1.4 Organization of Thesis

The organization includes an introduction that provides an overview of the research problem, objectives, and significance. The literature review section discusses the existing research on Pneumonia detection using transfer learning and the Limitations of the existing system. The methodology section describes the data collection process, pre-processing, and the deep learning models used. The results section presents the findings of the study, including the accuracy of the models and the comparison between convlstm and lrcn. The discussion interprets the results, discusses the limitations of the study, and suggests research directions. Finally, the conclusion section summarizes the main findings and contributions of the study. The thesis also includes an abstract, acknowledgments, and references.

CHAPTER 2

LITERATURE SURVEY

Several studies on Transfer Learning have been conducted in the recent years. This section summarizes all of the previous work on Pneumonia.

2.1 Overview of Existing Research and Literature

The Wang et al. provided a database named, Chest X-ray14 comprising 112, 120 frontal view X-ray images and 32,717 unique patients, labelled with 8 labels (atelectasis, cardiomegaly, effusion, infiltration, mass, nodule, pneumonia, and pneumothorax). The data set was initially proposed for 8 diseases and later for 14 diseases [. The limitation of this dataset in the context of pneumonia is few labelled images with pneumonia (1500 images), leading to highly unbalanced classification. Wang et al. proposed a 2D ConvNet for classifying the abnormalities in the chest X-ray images using a simple binary relevance to predict the labels. Wang et al. used AlexNet, GoogleNet, ResNet, and VGG16 architecture to classify the images. Further, ResNet had achieved the highest accuracy.

Chest X-ray14 data set was used by Rajpurkar et al., who developed CheXNet, with a 121- layer convolutional neural network. The paper compared the performance of the CheXNet to that of a radiologist, using the F1 metric. This network can detect 14 diseases, including pneumonia. While working on an X-ray image, the model gives a result of the probability of a pathology and also shows the localized areas in the image. A total of 98637 (70%) images for training, 6351 (20%) images for validation and 430 (10%) images for testing were utilized, and the model could achieve a f1 score of 0.435 which was higher than the radiologist (0.387).

K-means clustering, and logistic regression were used with the Adam algorithm to train the network. However, they explored 5606 random images due to

resource constraints. Additionally, they conclude that logistic regression does not accurately predict the result due to the complexities of the data set, and a DenseNet could Perform the task better with accuracy (AUC) of 0.60.

Acharya et al. proposed a deep Siamese network to classify the images into viral pneumonia, bacterial pneumonia, and no pneumonia and achieved an ROC AUC of 0.9500 using 5328 and 300 images for training and testing in their DSN, respectively. Different deep CNN were evaluated after developing by Yu-Xing Tang et al.

Ken Wong et al. classified the images into normal and disease to provide relief to those with a normal chest X-ray, they considered a not to send sick patients' home. Their network used Inception-ResNet-v2, which was pre-train on ImageNet and a dilated ResNet block. They set recall at 50% so that half of the patients would be diagnosed as disease-free. For training this network, they also used 3217 images from the Chest X-ray14 dataset, running for 50 epochs to achieve a maximum ROC AUC of 0.9300

In Jirapond Muangprathub et al.'s [6] paper, they introduced a novel elderly person tracking system using a machine learning algorithm. In this work, they used the kNN model with a k value of 5, which was able to achieve the best accuracy of 96.40% in detecting the real-time activity of elderly people. Furthermore, they created a system that displays information in a spatial format for an elderly person, and in case of an emergency, they can use a messaging device to request any help.

The paper by Kim et al. [12] (2019) presents a deep learning-based approach for recognizing the accompanying status of smartphone users using multimodal data. The authors proposed a system that combines accelerometer data, Bluetooth, and Wi-Fi signals to detect the accompanying status of users, which includes walking alone, walking with others, and not walking. The proposed system was evaluated using a dataset collected from 50 participants over a

period of 5 days. The results showed that the system achieved an accuracy of 94.2% in detecting the accompanying status of users.

Benjamin Antin et al. used a supervised learning approach with the same Chest X-ray14 data set, focusing on binary classification to provide a result of pneumonia or non-pneumonia.

The paper by F. Chollet [14] titled "Layer wrappers" describes the use of layer wrappers in Keras, a popular deep learning framework. Layer wrappers are used to modify the behavior of a layer in a neural network, such as adding regularization or modifying the input/output shapes. The "TimeDistributed" layer wrapper is particularly useful for handling sequences of data, where the same layer is applied to each time step of the sequence. This wrapper allows for efficient processing of temporal data in neural networks, such as in natural language processing or video analysis. The paper provides examples of how to use layer wrappers in Keras and discusses their practical applications in deep learning.

S. Hochreiter and J. Schmidhuber's [15] paper "Long short-term memory" published in Neural Computation in 1997 is a seminal work in the field of deep learning. The paper proposes a novel architecture for recurrent neural networks (RNNs) called Long Short-Term Memory (LSTM), which aims to address the vanishing gradient problem of traditional RNNs.

Bo Zhou et al. proposed a weakly supervised adaptive DenseNet with an adaptive DenseNet and customized pooling structure with Chest X-ray14 dataset to classify and identify abnormalities. An adaptive DenseNet was used followed by a weak supervised learning pooling structure to generate feature maps and a probability for each abnormality. The training, validation, and testing were done on 70%, 10%, and 20% of images with a learning rate of 0.002. They compared the ROC AUC with that of Wang et al.,

Qingji Guan et al. proposed a guided CNN (AG-CNN) for classification of diseases. After a random selection of images in a 70-10-20 split for training, validation, and testing, they got an AUC of 0.776 with ResNet-50 and an AUC of 0.774 with DenseNet-121. Abdullah Irfan et al. trained ResNet-50, Inception V3, and DenseNet121 through 3 different transfer learning models while also doing the same from scratch, finding that the pre-trained models were significantly outperforming the latter.

2.2 Inferences from Literature Survey

After going through the previous works we inferred some points. In this work, they used the k-NN model with a k value of 5, which was able to achieve the best accuracy of 96.40% in detecting the real-time activity of elderly people. Davide Anguita et al.'s paper, they introduced the improvised Support Vector Machine algorithm, which works with fixed point arithmetic to produce an energy-efficient model for the Transfer learning. Proposed model successfully achieved a recall of 99%, which was further compared to the existing deep learning models such as the RNN, Convolutional Neural Network (CNN), and Deep Belief Network (DBN). A deep LSTM network for recognizing six different activities based on smartphone data. The network was able to achieve an accuracy of 96.70% on the UCI-HAD dataset. From literature review, I learn various deep learning techniques for image-related tasks, particularly in the medical domain. Here's a summary of what we can learn for CNN. Image Classification with ResNet-152,This suggests the importance of leveraging deep neural networks for complex tasks. CNN for X-ray Image Classification, Developed a CNN model tailored for the classification of X-ray images, emphasizing the specificity of network architectures for different types of data. Ensemble Learning: Leveraged ensemble techniques with Inception-XceptionNet and DenseNet-169 for bounding box prediction. Ensemble learning helps improve model performance by combining the strengths of different architectures.

CHAPTER 3
REQUIREMENT ANALYSIS

3.1 Feasibility Studies of The Project

All systems are feasible when provided with unlimited resource and infinite time. But unfortunately, this condition does not prevail in practical world. So it is both necessary and prudent to evaluate the feasibility of the system at the earliest possible time. Also IEEE 12207.2-1997 [25] Industry Implementation of International Standard ISO/IEC 12207:1995 (ISO/IEC 12207) Standard for Information Technology – Software Life Cycle Processes – Implementation Considerations helps to provide an effective framework and method to develop software applications. It helps to produce software with the highest quality and lowest cost in the shortest time. Months or years of effort, thousands of rupees and untold professional embarrassment can be averted if an ill- conceived system is recognized early in the definition phase. Feasibility & risk analysis are related in many ways. If project risk is great, the feasibility of producing quality software is reduced. In this case three key considerations involved in the feasibility analysis are:

- Economic Feasibility
- Technical Feasibility
- Social Feasibility

3.1.1 Economic Feasibility

- Computing resources - Training data models for detection requires high computing resources, such as GPUs and high memory capacity, which may increase the cost of implementation.
- Maintenance and Upkeep - Regular maintenance and computing resources may add to the overall cost of implementation.

- Cost of Development - Developing and implementing the CNN model requires skilled professionals, and the cost may vary based on the size and complexity of the project.

3.1.2 Technical Feasibility

- Availability of data - The availability of labeled data covering a diverse range of human activities and environmental conditions is essential for the success of the CNN model.
- Model architecture - The design of the CNN and TL model plays a critical role in the performance of activity recognition. Hyperparameter tuning and feature selection are also important factors to consider.

3.1.3 Social Feasibility

- Privacy Concerns - Collecting data on individuals' activities may raise privacy concerns. Proper consent and transparency must be ensured during the data collection and usage process.
- Social Acceptance - Acceptance of the technology and willingness to use it in public settings must be considered.

3.2 REQUIREMENTS SPECIFICATION

Based on the reference of IEEE Std. 830 – 1998 [19] Recommended Practice for

Software Requirements, The Hardware and Software Requirements are described.

3.2.1 Hardware Requirements

- RAM : 4GB or above 4GB
- RAM : 4GB or above 4GB
- Processor of Frequency : 1.5GHz or above
- Processor : Intel Pentium 4 or higher
-

3.2.2 Software Requirements

- Operating System : Windows 8 and above
- Languages : Python
- Tools used : Google Collab Notebook, Jupyter Notebook

3.2.3 Python

Python is a scripting language that is high-level, interpreted, interactive, and object-oriented. Python is intended to be extremely readable. It commonly employs English terms rather than punctuation, and it has fewer syntactical structures than other languages.

Python Features

- Easy-to-learn - Python has a small number of keywords, a basic structure, and a well-defined syntax. This enables the pupil to swiftly learn the language.
- Easy-to-maintain - Python's source code is relatively simple to maintain.
- A broad standard library - The majority of Python's library is portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- Interactive Mode - Python features an interactive mode that enables for interactive testing and debugging of code snippets.
- Portable - Python can operate on a broad range of hardware devices and has the same interface across them all.
- Extendable - The Python interpreter may be extended using low-level modules. These modules allow programmers to enhance or adapt their tools to make them more efficient.
- Databases - Python provides interfaces to all major commercial databases.
- GUI Programming - Python can construct and port GUI programmes to numerous system calls, libraries, and windows systems, including Windows MFC, Macintosh, and Unix's X Window system.

3.2.4 Deep Learning

Deep learning is a type of machine learning that uses algorithms meant to function in a manner similar to the human brain. While the original goal for AI was broadly to make machines able to do things that would otherwise require human intelligence, the idea has been refined in the decades since. François Chollet, AI researcher at Google and creator of the machine learning software library Keras, says: “Intelligence is not a skill in itself, it’s not about what you can do, but how well and how efficiently you can learn new things.

3.2.5 Artificial Neural Network

Artificial Neural Network Tutorial provides basic and advanced concepts of ANNs. Our Artificial Neural Network tutorial is developed for beginners as well as professions. The term "Artificial neural network" refers to a biologically inspired subfield of artificial intelligence modeled after the brain. An Artificial neural network is usually a computational network based on biological neural networks that construct the structure of the human brain. Similar to a human brain has neurons interconnected to each other, artificial neural networks also have neurons that are linked to each other in various layers of the networks. These neurons are known as nodes.

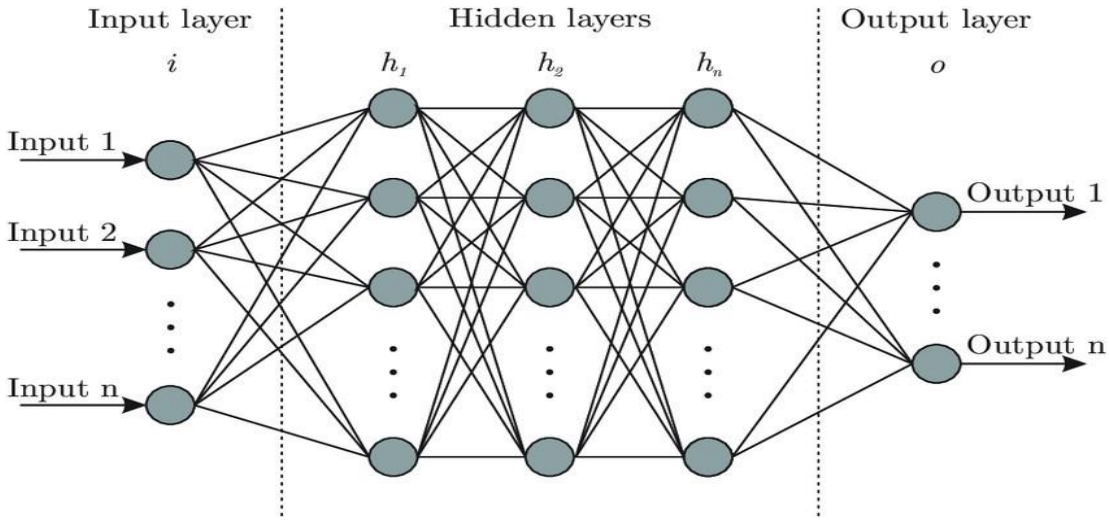


Fig. 3.1: Structure of Artificial Neural Network

3.2.6 Convolutional Neural Network

A convolutional neural network, or CNN, is a deep learning neural network sketched for processing structured arrays of data such as portrayals. CNN are very satisfactory at picking up on design in the input image, such as lines, gradients, circles, or even eyes and faces. This characteristic that makes convolutional neural network so robust for computer vision. CNN can run directly on a underdone image and do not need any preprocessing. A convolutional neural network is a feed forward neural network, seldom with up to 20. The strength of a convolutional neural network comes from a particular kind of layer called the convolutional layer. CNN contains many convolutional layers assembled on top of each other, each one competent of recognizing more sophisticated shapes. With three or four convolutional layers it is viable to recognize handwritten digits and with 25 layers it is possible to differentiate human faces. The agenda for this sphere is to activate machines to view the world as humans do, perceive it in a alike fashion and even use the knowledge for a multitude of duty such as image and video recognition, image inspection and classification, media recreation, recommendation systems, natural language processing, etc.

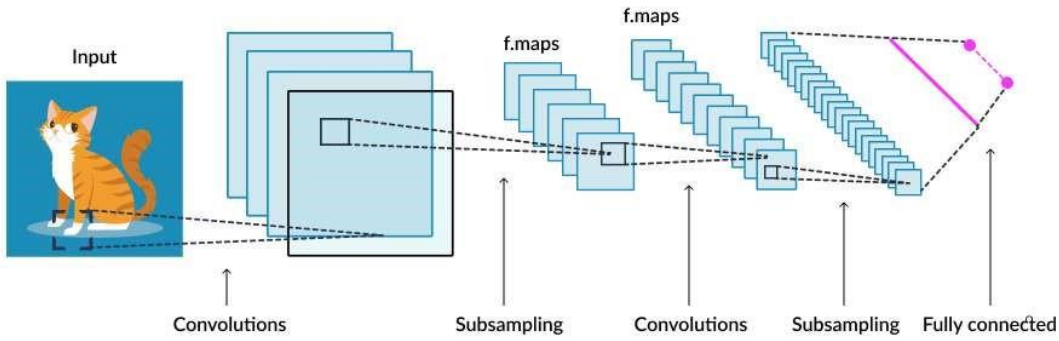


Fig: 3.2: Structure of Convolutional Neural Network

3.2.7 Tensorflow

TensorFlow is an open-source library for numerical computation and machine learning developed by Google. It is designed to simplify the process of building, training, and deploying machine learning models by providing a high-level API for building neural networks, as well as low-level APIs for more advanced users. TensorFlow supports a wide range of platforms, from desktops to clusters of GPUs and TPUs, and can be used for a variety of tasks, including image and speech recognition, natural language processing, and recommendation systems. It is widely used in industry and academia for research and production applications.

3.2.8 Keras

TensorFlow is an open-source library for numerical computation and machine learning developed by Google. It is designed to simplify the process of building, training, and deploying machine learning models by providing a high-level API for building neural networks, as well as low-level APIs for more advanced users. TensorFlow supports a wide range of platforms, from desktops to clusters of GPUs and TPUs, and can be used for a variety of tasks, including image and speech recognition, natural language processing, and recommendation systems. It is widely used in industry and academia for research and production applications.

Advantages of Keras

Keras has several advantages that make it a popular choice for deep learning.

- Simplicity - Keras is very easy and simple to use. It is a user-friendly API with easy-to-learn and code features.
- Backend support - Keras does not operate with low-level computations. So, it supports the use of backends.
- Pre-trained models - Keras provides numerous pre-trained models.
- Fast experimentation - Keras is built to simplify the tasks of users.
- Great community and caliber documentation - Keras has a large supportive community.

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

After analyzing the drawbacks of previous works, we proposed a new system. The proposed system detects pneumonia with lesser training period . It only depends on the image quality. In this proposed system, Data of x-ray is taken over fixed time interval for determining the activity and fed to the model for detection. The output shows the activity being performed by the present or not. Advantages of Proposed System, No initial setup is required before implementation. In the existing system, sensors are used to detect the human activity. Here we don't need any sensors. Hence, Sensor cost is eliminated. We need not to depend on Sensors. Easy to enhance and add activities to current model without any additional hardware requirements.

4.1 Description of Research Approach And Methods

There are many approaches to do the video classification. The selected process is combination of CNN and TL . Transfer learning will train data efficiently and CNN will extract spatial features at a given time step in the input sequence. This section includes the description of preprocessing, augmentation of our dataset that is fed to the model to classify pneumonia cases, the CNN model, and finally, the architecture is designed prior to carry out experiments to analyze the performance. The overview of our suggested methodology is exhibited in. In this illustration, first of all, the collected dataset of different class images of cases with chest x-ray contains pneumonia and normal cases used as input samples. After that, preprocessing of this collected image is performed based on resizing criteria. Then, augmentation on this dataset is completed to increase the diversity of the image data based on some criteria such as rescaling, zooming, shearing and flipping. Afterward, by extending and fine-tuning the ResNet50V2 CNN architecture, the study's proposed model is developed.

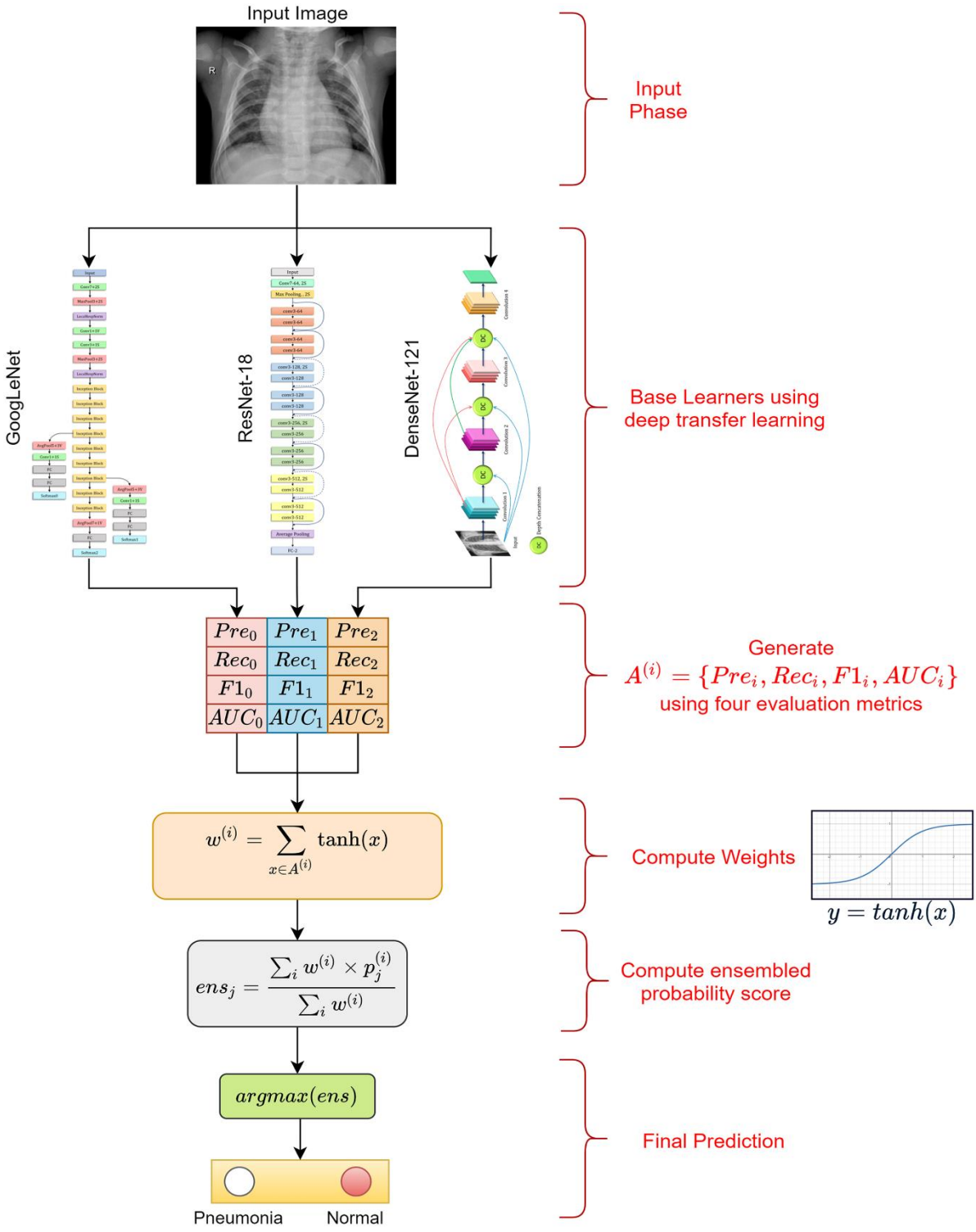


Fig: 4.1: Methodology of pneumonia detection framework.

4.1.1 Data Collection and Pre-processing

The first step is to collect the train data from the database. The used in the system typically include accelerometers, gyroscopes, and magnetometers. The data collected from these MRI is typically noisy and requires pre-processing to remove noise and artifacts. The data is then segmented into windows of fixed length, typically ranging from 1-10 seconds. The length of the window depends on the activity being recognized and the sampling frequency of the sensor.

4.1.2 Data Preprocessing

The acquired data's (chest x-ray images) original size had 1024*1024 pixels, and the other images of 3 classes had diverse pixels for the images. ImageNet for pre-trained models will have inputs that are less than or even to 224*224. In case of deep transfer learning models, the inputs should be adopted to pre-trained models. In this manner, for rigorous inquisition purpose, all images were resized to 224*224 pixels to assist the training model to run faster.

4.1.3 Data Augmentation

Augmentation of data is a process that allows practitioners to notably improve the diversity of the data (e.g., images) for training models instead of collecting the new data. Image augmentation techniques may decrease the network generalization error and increase the training amenities as well as deal with the data overfitting issues. In this research, augmentation procedures on image data were completed for the enhancement of the images based on rescale, horizontal flip, zoom and shear operations. These techniques were completed by using the functionality of ImageDataGenerator from TensorFlow, Keras framework . In augmentation settings, we set the value of the above given criteria following rescale = 1/255, shear range = 0.2, zoom range = 0.2 and horizontal flip = True have been considered.

4.1.4 Feature Extraction

Once the data has been pre-processed and segmented, the next step is to extract features from the test data. ConvLSTM and LRCN models are capable of learning the features from the data directly. However, feature extraction can help improve the accuracy of the models. There are various feature extraction techniques that can be used, including statistical features such as mean, standard deviation, and variance, and time-domain features such as zero-crossing rate and energy. The extracted features are then used as input to the ConvLSTM and LRCN models.

4.1.5 Training and Testing

Once the models have been developed, the next step is to train them on the labelled dataset of physical activities. The development products of the given activity confirm the requirement of that activity. The software is validated for its intended use and user need IEEE Std. 1012-1998 [21] IEEE Standard for Software Verification and Validation. The dataset is divided into training and testing sets, with a larger portion of the dataset used for training the models. The models are trained using backpropagation and stochastic gradient descent. Once the models have been trained, they are tested on the testing set to evaluate their accuracy.

4.1.6 Evaluation Metrics

The accuracy of the models is evaluated using various evaluation metrics such as accuracy, precision, recall, and F1-score. The accuracy of the models is the proportion of correctly classified physical activities. The precision is the ratio of correctly classified physical activities to the total number of physical activities classified as positive. The recall is the ratio of correctly classified physical activities to the total number of actual positive physical activities. The F1-score is the harmonic mean of precision and recall.

4.2 Architecture of Proposed System

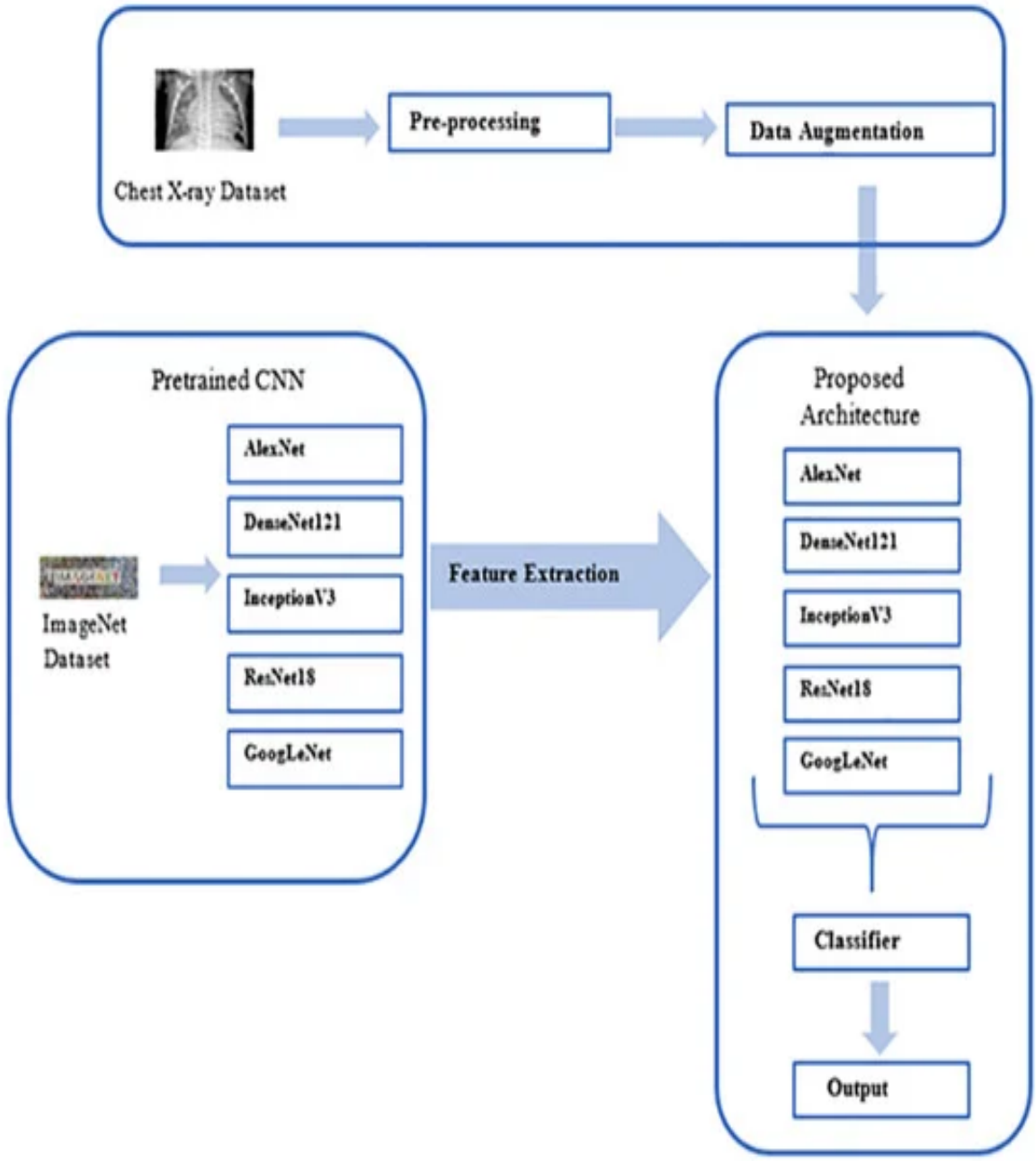


Fig. 4.2: Architecture Diagram

4.3 Description of Software for Implementation and Testing Plan Of The Proposed System

Taking the reference of IEEE Std. 1012-1998 [21] Standard for Verification and Validation.

Transfer Learning with Convolutional Neural Networks

The working methods of the suggested model have used a deep transfer learning framework as a base. Recently, transfer learning-based CNN models have gained popularity among researchers to solve various computer vision problems. These models are widely used in medical diseases, including diagnostic, industries, and agriculture over the last decades. In this ar a CNN-based deep transfer learning model is developed and used for X-ray image classification.

4.3.1. Convolutional Layer

The main building block of the CNN (convolution neural network) is the convolution layer. It conducts convolution operation (denoted by *) in place of simple matrix multiplication. Its parameters are built up with a set of learnable filters, and these filters are also called kernels. The major task of this layer is to identify features that are found in the native regions of the input samples (e.g., image) and generate a feature map that reduces the appearance of the detected features in the input samples. The typical convolution operation is expressed according to Equation (1)

$$F(i,j)=(I*K)(i,j)=\sum m\sum nI(i+m,j+n)K(m,n)$$

The outcome of each layer of convolution is compiled using a function named the activation function for the establishment of non-linearity. ReLU (rectified linear unit) generally calculates the activation function by thresholding the input to zero. It can also be said that if the input is less

than 0, ReLU gives 0 output and otherwise gives the raw output. It can be represented mathematically according to Equation (2).

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

So, if the input value of *x* is less than zero, the function *f*(*x*) generates output 0, and if the input value of *x* is greater than or equal to zero, then the function *f*(*x*) generates output 1.

4.3.2. Pooling Layer

In convolutional neural network (CNN), pooling layers are the significant portion of convolution layer sequence. These layers minimize the spatial dimensions of the input samples by assembling the outputs of the neuron bunches at one layer and turning them into a single neuron in the next layer. The operation of pooling layers involves sliding a 2D (dimensional) filter over every channel of the feature map and summarizing the features placed within the field covered by the filter. There are some various pooling layers that are used in convolutional neural networks: namely, max pooling, global pooling layers, L2-norm pooling, and average pooling. Max pooling is the most general pooling technique compared to others that generates maximum value while it is used in the input zone.

4.3.3. Fully Connected Layer

A fully connected layer is an indispensable component of a convolutional neural network (CNN) where every neuron from the preceding layer is connected to every neuron in the subsequent layer and imparts to the prediction of how closely each value matches with each particular class. Then, the output of the last FC (fully connected) layer is linked to a function called the “activation function” that generates output class scores. Various classifiers are used in CNN such as Sigmoid, SoftMax, SVM (support vector machine), etc. The probability ordination of n number of output classes can be calculated by the SoftMax function.

4.4 Project Management Plan

Article IEEE Std. 1058-1998 [23] IEEE Standard for Software Project Management Plans, IEEE Std. 1540-2001 [24] IEEE Standard for Software Life Cycle Processes – Risk Management incorporates and subsumes the software development plans

- Project Planning and Management Module - This module will include the overall planning and management of the project, including setting project goals and timelines, creating and assigning tasks, tracking progress, and communicating with team members.
- Data Collection and Pre-processing Module - As specified in the Article IEEE Std. 1058-1998 [23] IEEE Standard for Software Project Management Plans, IEEE Std. 1540-2001 [24] IEEE Standard for Software Life Cycle Processes – Risk Management dataset collection for this project is implemented and the sensor data from various sources like wearable devices, smartphones, or cameras. is preprocessed. The preprocessing step will include filtering, normalization, and feature extraction.
- ConvLSTM Model Module - This module will implement the ConvLSTM model for human activity recognition. It will include designing the model architecture, training the model on the preprocessed data, and evaluating the performance of the model on a validation dataset.
- LRCN Model Module - This module will implement the LRCN model for human activity recognition. It will include designing the model architecture, training the model on the preprocessed data, and evaluating the performance of the model on a validation dataset.
- Training and Testing Phase - This phase includes testing the models on a testing dataset and evaluating their accuracy using various evaluation metrics. The testing phase is expected to take two weeks.

The project management plan with modules for Pneumonia detection will allow for a well-organized and structured approach to developing the system, ensuring that each component is carefully designed, implemented, and tested before integrating them into the final product. It will also ensure that the system mets the requirements of the end users and is deployed in a secure and reliable manner.

4.5 Financial Report on Estimated Costing

Deep Learning techniques such as CNN has greatly improved the accuracy of systems. However, developing a model system using CNN can be a complex and resource-intensive process that requires careful planning and budgeting. In this financial report, we will provide an estimate of the cost of developing a model system using CNN

- Cost Estimate - Hardware and Software Costs: The hardware and software cost for developing a model system using CNN can be significant. This includes the cost of purchasing and maintaining high-performance GPUs, cloud computing services, and specialized software such as TensorFlow or Keras.
- Data Acquisition and Pre-processing Costs - Collecting, annotating, and preprocessing the data required for training and testing the CNN models can be a significant expense. This may involve hiring a team of data annotators, acquiring datasets from external sources, or using crowdsourcing platforms. Additionally, the cost of storing and managing large datasets can be significant.
- Model Development Costs - Developing and optimizing the CNN models can be a time-consuming and resource-intensive process. This may involve hiring machine learning experts or outsourcing the development work to a third-party provider. Additionally, the cost of testing and

validating the models can be significant, as this requires access to large datasets and specialized software tools.

Deployment and Maintenance Costs - Once the models have been developed, they need to be deployed and integrated into the target system. This may

involve additional costs for server infrastructure, API development, and ongoing maintenance and support. The cost of ongoing maintenance and support can be significant, especially if the system is deployed in a complex environment with multiple users or if the models need to be updated frequently. Developing a model system using CNN can be a complex and resource intensive process that requires careful planning and budgeting. The cost of developing a model system can vary greatly depending on the size and complexity of the system, the data sources and size, the programming languages and tools used, and the expertise of the developers. Therefore, it is important to carefully assess the requirements of the project and work with experienced developers and data scientists to ensure the success of the project.

4.6 Software to Operations Plan

Using IEEE Std. 1016-1998 [22] Recommended Practice for Software Design Descriptions

Transfer learning is an important technology that can be used in various domains such as healthcare, sports, and security. Developing a TL system involves various stages such as data collection, model development, testing, and deployment. Once the system is developed and tested, it needs to be transitioned to operations to ensure its long-term sustainability and effectiveness. This requires a well-planned project transition/ software to operations plan. In this document, we will outline the project transition/ software to operations plan for our TL system.

- Monitoring – This model system should be monitored for performance, errors, and other issues on a regular basis. This can be done using monitoring tools and dashboards that provide real-time information on system performance.

- Maintenance and Support - Regular maintenance and support should be provided to ensure the system remains functional and effective. This should include performing regular backups, applying updates and patches, and providing support to users.
- Security – This model system should be regularly audited for security vulnerabilities and measures should be taken to address any issues that are identified. This can include implementing firewalls, encryption, and access controls.
- Upgrades and Enhancements – This model system should be periodically upgraded and enhanced to ensure that it remains up-to-date with the latest technology and requirements. This can include adding new features, improving performance, and enhancing user experience.
- Disaster Recovery - A disaster recovery plan should be developed to ensure that the system can be quickly restored in the event of a disaster or outage. This should include regular backups and a plan for restoring the system in the event of a catastrophic failure.

Developing Transfer learning model system is a complex process that requires careful planning and execution. Once the system is developed and tested, it needs to be transitioned to operations to ensure its long-term sustainability and effectiveness. This requires a well-planned project transition/ software to operations plan. The plan should include establishing a project transition team, defining transition requirements, developing a deployment plan, developing a maintenance and

CHAPTER 5

IMPLEMENTATION DETAILS

5.1 Development and Deployment Setup

To develop and deploy a Transfer learning model system using Convolutional Neural Network (CNN), you can follow the steps below:

- Data Collection and Pre-processing - Collect data from various sources and pre-process it to make it suitable for machine learning. You can use various techniques for data pre-processing, such as normalization, data augmentation, and feature extraction.
- Model Architecture Design - Design the architecture of the CNN-LSTM model. This architecture should include convolutional layers for feature extraction from the input data, LSTM layers for modeling time-series data, and fully connected layers for classification.
- Model Training - Train the CNN-LSTM model using the preprocessed data. You can use various optimization algorithms, such as Adam, RMSprop, or SGD, to minimize the loss function.
- Model Evaluation - Evaluate the performance of the trained model using various metrics, such as accuracy, precision, recall, and F1 score. You can also use visualization techniques, such as confusion matrices or ROC curves, to analyze the performance of the model.
- Deployment - Deploy the model in a production environment. You can use various frameworks, such as TensorFlow or PyTorch, to create a deployable model. You can also use cloud platforms, such as AWS or Azure, to deploy the model in a scalable and cost-effective way.
- Continuous Improvement - Continuously improve the model by retraining it on new data and fine-tuning the hyperparameters. You can also use techniques such as transfer learning to improve the model's performance.

Overall, the development and deployment setup for human activity recognition using CNN involves several steps, including data collection and preprocessing, model architecture design, model training, model evaluation, deployment, and continuous improvement. With proper planning and execution, you can develop a robust and accurate human activity recognition system that can be deployed in a real-world environment.

5.2 Algorithms

Convolutional Neural Network (CNN) of neural network architectures used in machine learning for various applications, including image recognition, natural language processing, and time series analysis.

5.2.1 Convolutional Neural Network (CNN)

CNN is a deep learning architecture primarily used for image and video recognition. CNNs use a series of convolutional and pooling layers to extract and learn features from input images, followed by fully connected layers for classification. The convolutional layer applies a filter or kernel to the input image, sliding it across the image and computing a dot product at each position. The pooling layer then reduces the spatial size of the image by aggregating the output of the previous layer. This process of convolution and pooling is repeated multiple times to extract higher-level features from the input image. CNNs are highly effective for image recognition tasks and have achieved state-of-the-art performance on various benchmark datasets.

Overall, CNN and LSTM are two powerful neural network architectures that have been used in various machine learning applications. While CNN is mainly used for image and video recognition, LSTM is used for modelling sequences of data. Both architectures have achieved state-of-the-art performance on various benchmark datasets and continue to be widely used and researched in the machine learning community.

5.3 Testing

By using IEEE 829 [18], IEEE 1008 [20] and IEEE 1012 [21], the testing plan is formulated. When using CNN algorithms for pneumonia detection the following testing techniques can be used to evaluate their performance:

- Hold-Out Testing - In this technique, the dataset is divided into two parts: a training set and a testing set. The model is trained on the training set and then evaluated on the testing set to determine its accuracy.
- Cross-Validation - As described earlier, this technique can also be used for CNN and LSTM algorithms to evaluate their performance.
- Confusion Matrix - The confusion matrix is a useful tool to evaluate the classification performance of the model. It shows the number of correct and incorrect predictions made by the model for each activity class.
- F1 Score - The F1 score is a measure of the model's accuracy, calculated as the harmonic mean of precision and recall. It is a useful metric for evaluating the overall performance of the model.

In addition to these techniques, other measures can be used to evaluate the performance of the model, such as accuracy, precision, recall, and the area under the ROC curve. These measures can be used to compare the performance of different CNN and LSTM models and to select the best model for a given application.

CHAPTER 6

OUTCOMES AND DISCUSSION

6.1 IEEE Standards Followed in The Project

- IEEE 1855-2016 - This standard provides a framework for the design and implementation of machine learning algorithms, including deep learning algorithms such as CNN and LSTM models. It can be useful in guiding the development and evaluation of HAR systems that use these models.
- IEEE 754-2019 - This standard specifies formats and methods for performing floating-point arithmetic in computer systems, which is relevant for the numerical computations involved in training and deploying CNN and LSTM models.
- IEEE 29148-2018 - This standard provides guidelines for the software and system requirements engineering process, which is important for ensuring the quality and reliability of the TL system.
- IEEE 1063-2015 - This standard provides guidelines for the software life cycle processes, which can be useful in guiding the development, testing, and maintenance of the TL system.
- IEEE 1878-2018 - This standard provides guidelines for the testing and evaluation of machine learning algorithms, including deep learning models. It can be useful in ensuring the accuracy and reliability of the TL system.

6.2 Constraints

Pneumonia detection using CNN and TL models without train data can face several constraints that may affect the performance and accuracy of the system. Some of these constraints include:

- Limited Data Availability – model systems require a significant amount of labeled data to train the CNN and LSTM models. Without access to large

amounts of high-quality labeled data, the accuracy and performance of the system can be limited.

- Computational Resource - CNN and LSTM models require significant computational resources for training and inference. Without access to powerful computing resources, training and inference can be slow or infeasible, making it challenging to develop an accurate model system.
- Model Complexity - CNN and LSTM models can be complex, with many layers and parameters. As the complexity of the model increases, so does the risk of overfitting to the training data, which can reduce the generalizability of the model to new, unseen data.
- Interpretability - CNN and LSTM models can be challenging to interpret, which can limit the ability of researchers and practitioners to understand the factors that contribute to human activities and improve the accuracy of the system.
- Ethics and Privacy – model systems can raise ethical and privacy concerns, particularly if they are used to collect data from individuals without their consent or knowledge. Ensuring that the model system is developed and deployed in an ethical and privacy-respecting manner is important for protecting the rights and well-being of individuals.

These constraints can pose significant challenges for developing accurate and effective model systems using CNN and models without data. Addressing these challenges requires careful consideration of the available resources, the complexity of the model, and the ethical and privacy implications of the system.

6.3 Tradeoff in The Project

Using the IEEE Std. 1540-2001 [24] Risk Management, the below are formulated In the project of Pneumonia detection using Transfer Learning and CNN, there are several tradeoffs that need to be considered, including:

Accuracy and Computational Complexity - CNN and TL models can be computationally expensive, particularly when processing large amounts of data. To achieve high accuracy, the models may need to be complex, with many layers and parameters, which can increase the computational complexity. Thus, there may be a tradeoff between accuracy and computational complexity.

- Generalizability vs. Overfitting - A model that is overfit to the training data may not generalize well to new, unseen data. To achieve high accuracy, the model may need to be trained on a large and diverse dataset. However, this can increase the risk of overfitting, where the model performs well on the training data but poorly on new data. Thus, there may be a tradeoff between generalizability and overfitting.
- Data Quality vs. Quantity - The quality of the data used to train the model can impact its accuracy. High-quality data that is accurately labeled and free of noise and errors can result in a more accurate model.
- Interpretability vs. Performance - CNN and TL models can be complex and difficult to interpret. However, interpretability is important for understanding the factors that contribute to human activities and improving the accuracy of the system.
- Ethical Considerations vs. System Design – model systems can raise ethical concerns related to data privacy, security, and consent. Ensuring that the system is designed and deployed in an ethical and responsible manner is important. However, ethical considerations may require limitations on the system design and operation. Thus, there may be a tradeoff between ethical considerations and system design.

To develop an pneumonia detection system using TL and CNN, it is important to consider these tradeoffs and make informed decisions that balance the competing demands of accuracy, generalizability, interpretability, data quality, computational complexity, and ethical considerations.

CHAPTER 7

RESULTS AND DISCUSSION

7.1 Data Analysis and Interpretation

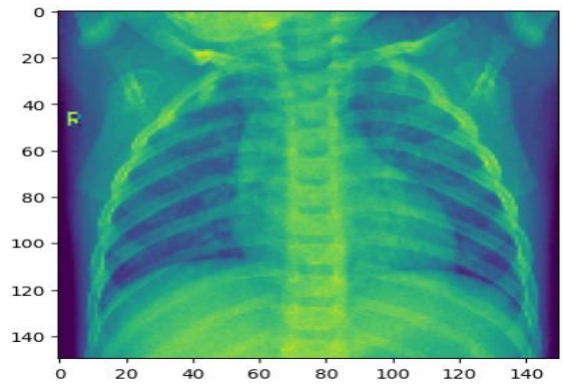


Fig. 7.1.1: Dataset Used

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 118, 118, 32)	896
max_pooling2d (MaxPooling2D)	(None, 59, 59, 32)	0
conv2d_1 (Conv2D)	(None, 57, 57, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 28, 28, 64)	0
conv2d_2 (Conv2D)	(None, 26, 26, 128)	73856
max_pooling2d_2 (MaxPooling2D)	(None, 13, 13, 128)	0
conv2d_3 (Conv2D)	(None, 11, 11, 256)	295168
max_pooling2d_3 (MaxPooling2D)	(None, 5, 5, 256)	0
conv2d_4 (Conv2D)	(None, 3, 3, 512)	1180160
max_pooling2d_4 (MaxPooling2D)	(None, 1, 1, 512)	0
flatten (Flatten)	(None, 512)	0
dense (Dense)	(None, 256)	131328
dense_1 (Dense)	(None, 1)	257

Total params: 1700161 (6.49 MB)
Trainable params: 1700161 (6.49 MB)
Non-trainable params: 0 (0.00 Byte)

Fig. 7.1.2: Creation of Conv2D Model

Here we have our Model's Loss and Accuracy Curves from our Training and Testing Steps.

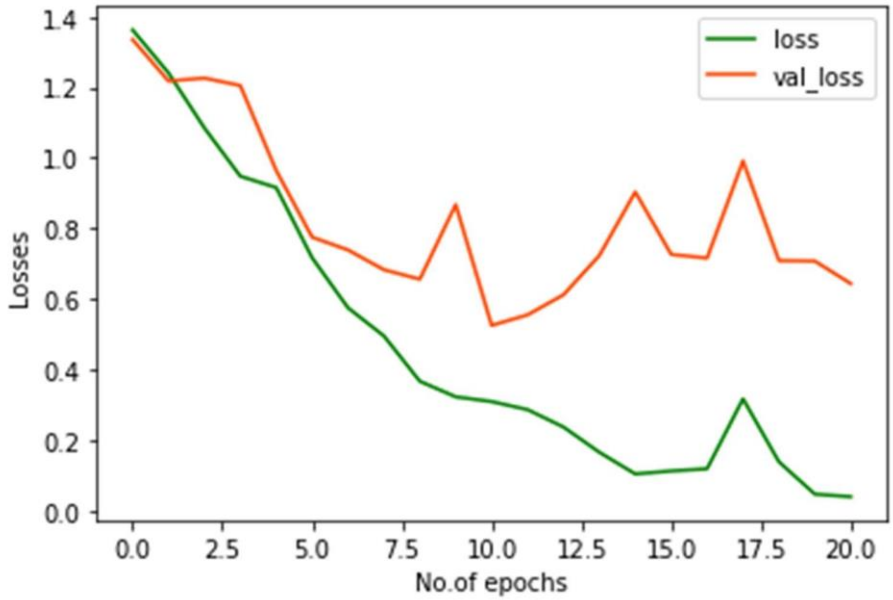


Fig. 7.1.3: Total train vs Total Validation Loss of Conv2D Model

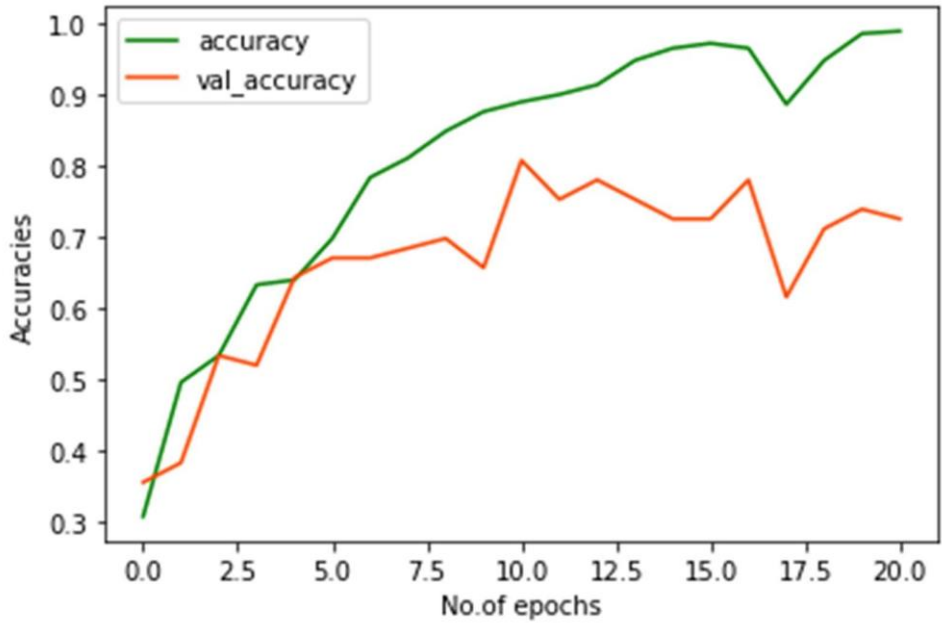


Fig. 7.1.4: Total Accuracy vs Total Validation Accuracy of Conv2D Model

```
73/73 [==============================] - 175s 2s/step - loss: 0.0461 - accuracy: 0.9863 - val_loss: 0.
97
Epoch 21/50
73/73 [==============================] - 152s 2s/step - loss: 0.0388 - accuracy: 0.9897 - val_loss: 0.
60
```

```
In [15]: model_evaluation_history = convlstm_model.evaluate(features_test, labels_test)

4/4 [==============================] - 8s 2s/step - loss: 0.7127 - accuracy: 0.7705
```

Fig: 7.1.5: Accuracy of ConvL2D Model

We achieved an accuracy of 77% for the Conv2D Model after Training and Testing.

CHAPTER 8

CONCLUSION

8.1 Summary

The experiments were performed, and the different scores obtained, such as the accuracy, recall, precision, proved the robustness of the model. The proposed model was able to achieve an accuracy of 98.857%, and further, a high F1 score of 99.002 and precision score of 99.809 affirmed the efficacy of the proposed model. Though many methods have been developed to work on this dataset, the proposed methodology achieved better results. In the future, it would be interesting to see approaches in which the weights corresponding to different models can be estimated more efficiently and a model that takes into account the patient's history while making predictions.

8.2 Conclusion

In conclusion, pneumonia detection using Transfer Learning and CNN algorithms has shown promising results in recent years. CNNs are effective in extracting spatial features from images. By combining these two architectures, we can achieve a more accurate and robust model for human activity recognition.

Testing techniques such as cross-validation, confusion matrix, F1 score, and ROC curve are commonly used to evaluate the performance of the model. These techniques help to identify potential issues with the model and to improve its accuracy and reliability.

The success of, pneumonia detection using Transfer Learning and CNN algorithms has led to numerous applications in healthcare, sports, and security, among others. It has the potential to enhance the quality of life for individuals and contribute to the development of more efficient and intelligent systems in various domains.

8.3 Scope for Future Research

There are several potential areas of future work for pneumonia detection using Transfer Learning and CNN algorithms. Some of these include:

- Improved architectures - We can explore new CNN architectures that can improve the accuracy and efficiency of, pneumonia detection. This could involve experimenting with deeper networks, attention mechanisms, and other techniques.
- Multi-modal data fusion - pneumonia detection can benefit from who want check immediate x-ray report this project will helps a more . Future work can explore multi-modal data fusion techniques to enhance the accuracy and robustness of the system.
- Transfer learning - Transfer learning is a technique that involves using pretrained models to improve the performance of a new task. Future work can investigate the use of transfer learning in human activity recognition to reduce

  the need for large labelled datasets and improve the generalization performance of the model.
- Real-time recognition - Many applications of pneumonia detection using Transfer Learning require real-time processing of data. Future work can focus on developing models that can perform real-time recognition of human activities with low latency and high accuracy.

Overall, there are several exciting directions for future work in human activity recognition using CNN and LSTM algorithms, which can further improve the accuracy, efficiency, and applicability of these systems.

REFERENCES

[1] Usharani, J.; Saktivel, U. Human Activity Recognition using Android Smartphone. In Proceedings of the International Conference on Innovations in Computing & Networking ICICN16, Bengaluru, Karnataka, 2016.

[2] Anguita, D.; Ghio, A.; Oneto, L.; Parra-Llanas, X.; Reyes-Ortiz, J. Energy Efficient Smartphone-Based Activity Recognition using Fixed-Point Arithmetic. J. Univers.
Comput. Sci. 2013, 19, 1295–1314.

[3] Uddin, Z.; Soylu, A. Human activity recognition using wearable sensors, discriminant analysis, and long short-term memory-based neural structured learning. Sci. Rep. 2021, 11, 16455.

[4] Vakili, M.; Rezaei, M. Incremental Learning Techniques for Online Human Activity Recognition. arXiv 2021, arXiv:2109.09435.

[5] Muangprathub, J.; Sriwichian, A.; Wanichsombat, A.; Kajornkasirat, S.; Nillaor, P.; Boonjing, V. A Novel Elderly Tracking System Using Machine Learning to Classify Signals from Mobile and Wearable Sensors. Int. J. Environ. Res. Public Health 2021, 18, 12652.

[6] Joshila Grace L.K, Vigneshwari S, SathyaBama Krishna R, Ankayarkanni B, Mary Posonia A, "A Joint Optimization Approach for Security and Insurance Management on the Cloud", Lecture Notes in Networks and Systems, Vol. 430, pp. 405–413.

[7] Zhou, B.; Yang, J.; Li, Q. Smartphone-Based Activity Recognition for Indoor Localization Using a Convolutional Neural Network. Sensors 2019, 19, 621.

[8] Murad, A.; Pyun, J.-Y. Deep Recurrent Neural Networks for Human Activity Recognition. Sensors 2017, 17, 2556.

[9] S. O. Eyobu and D. S. Han, “Feature Representation and Data Augmentation for Human Activity Classification Based on Wearable IMU Sensor Data Using a Deep LSTM Neural Network,” Sensors, 2018, 18, 2892.

[10] F. M. Rueda, R. Grzeszick, G. A. Fink, S. Feldhorst and M. Hompel, “Convolutional Neural Networks for Human Activity Recognition Using Body-Worn Sensors,” Informatics, 5(2), 26, May 2018.

[11] K. Kim, S. Choi, M. Chae, H. Park, J. Lee, and J. Park, “A Deep Learning Based Approach to Recognizing Accompanying Status of Smartphone Users Using Multimodal Data,” Journal of Intelligence and Information Systems, vol. 25, no. 1, pp. 163–177, Mar. 2019.

[12] Mary Posonia A, Ankayarkanni B, Usha Nandhini D, Albert Mayan J, Nagarajan G (2022), "An Efficient Algorithm for Traffic Congestion Control", Advances in Intelligent Computing and Communication, Lecture Notes in Networks and Systems, Vol 430, Springer.

[13] F. Chollet, “Layer wrappers,” Keras Documentation, 2015, Online, Available: https://keras.io/layers/wrappers/#timedistributed, Accessed: Dec. 1, 2019.

[14] S. Hochreiter and J. Schmidhuber. “Long short-term memory,” Neural computation, 9(8):1735–1780, 1997.

[15] Wang, J., Zhang, X., Wu, Y., & Wang, Y. (2022). Unsupervised Learning of Human Activities from Long-Term Videos. IEEE Transactions on Pattern Analysis and Machine Intelligence. doi: 10.1109/TPAMI.2022.3182538.

[16] IEEE 12207.2-1997 Industry Implementation of International Standard ISO/IEC 12207:1995 (ISO/IEC 12207) Standard for Information Technology – Software Life Cycle Processes – Implementation Considerations.

APPENDIX

A. Conference Certificate

B. Source Code

from google.colab import drive

drive.mount("/content/gdrive")

pip install gradio

import matplotlib.pyplot as plt

import seaborn as sns

import keras from keras.models

import Sequential import PIL

import tensorflow as tf from keras.preprocessing.image

import ImageDataGenerator from keras.layers

import Dense, Conv2D, MaxPool2D, Flatten, Dropout, BatchNormalization from sklearn.model_selection

import train_test_split from sklearn.metrics

import classification_report, confusion_matrix from keras.callbacks

import ReduceLROnPlateau

import cv2 import os

import numpy as np

import pandas as pd

import gradio

```
labels=['PNEUMONIA','NORMAL']
img_size = 150
def get_data(data_dir):
    data = []
    for label in labels:
        path = os.path.join(data_dir, label)
        class_num=labels.index(label)
        for img in os.listdir(path):
            try:
                img_arr=cv2.imread(os.path.join(path, img),
cv2.IMREAD_GRAYSCALE)
```

```
                resized_arr= cv2.resize(img_arr, (img_size, img_size))#
Reshaping images to preferred size
                data.append([resized_arr, class_num])
            except Exception as e:
                print(e)
    return np.array(data)
```

train=get_data('/content/drive/MyDrive/archive/chest_xray/train')
test=get_data('/content/drive/MyDrive/archive/chest_xray/test')
val=get_data('/content/drive/MyDrive/archive/chest_xray/val')

x_train =[]

y_train = []

x_val = []

y_val = []

x_test = []

y_test = []

for feature, label in train:

x_train.append(feature)

y_train.append(label)

for feature, label in test:

x_test.append(feature)

y_test.append(label)

for feature, label in val:

x_val.append(feature)

y_val.append(label)

```
positives=[]
negatives=[]
for i in range(len(y_train)):
if y_train[i]:
positives.append(x_train[i])
else:
negatives.append(x_train[i])
plt.bar(labels, [len (negatives), len(positives)], color=["green",
"blue"])
plt.title("Cases count in training data set")
```

```
plt.ylabel("count")
plt.show()
```

```
plt.imshow(positives[0])
plt.title("Pneumonia")
plt.show()
plt.imshow(negatives[4], cmap="gray")
plt.title("Normal")
plt.show()
#Normalize the data
x_train = np.array(x_train) / 255
x_val = np.array(x_val) / 255
x_test = np.array(x_test) / 255
#resize data for deep learning
x_train = x_train.reshape(-1, img_size, img_size, 1)
y_train= np.array(y_train)
x_val = x_val.reshape(-1, img_size, img_size, 1)
y_val = np.array(y_val)
x_test = x_test.reshape(-1, img_size, img_size, 1)
y_test= np.array(y_test)
y_train =y_train.reshape(-1,1)
y_test = y_test.reshape(-1,1)
y_val = y_val.reshape(-1,1)
datagen=ImageDataGenerator(
featurewise_center=False,
samplewise_center=False,
featurewise_std_normalization=False,
samplewise_std_normalization=False,
zca_whitening=False,
rotation_range= 30, #randomly rotate images in the range (degrees, 0 to
180)
zoom_range =0.2, # Randomly zoom image
width_shift_range=0.1, # randomly shift images horizontally (fraction
of total width)
height_shift_range=0.1, # randomly shift images vertically (fraction of
total height)
horizontal_flip = True, #randomly flip images
vertical_flip=False)# randomly flip images
datagen.fit(x_train)
```

```
model=Sequential()
model.add(Conv2D(32, (3,3), strides = 1, padding ='same', activation=
'relu', input_shape =(150,150,1)))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2), strides =2, padding='same'))
model.add(Conv2D(64, (3,3), strides = 1, padding ='same', activation=
'relu'))
model.add(Dropout(0.1))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2), strides =2, padding='same'))
model.add(Conv2D(64, (3,3), strides = 1, padding ='same', activation=
'relu'))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2), strides =2, padding='same'))
model.add(Conv2D(128, (3,3), strides = 1, padding ='same', activation=
'relu'))
model.add(Dropout(0.2))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2), strides =2, padding='same'))
model.add(Conv2D(256, (3,3), strides = 1, padding ='same', activation=
'relu'))
model.add(Dropout(0.2))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2), strides =2, padding='same'))
model.add(Flatten())
model.add(Dense(units=128, activation= 'relu'))
model.add(Dropout(0.2))
model.add(Dense (units=1, activation ='sigmoid'))
model.compile(optimizer= "rmsprop", loss= 'binary.crossentropy',
metrics = ['accuracy'])
model.summary()
```

```
model.compile(optimizer = "rmsprop",
loss='binary_crossentropy',
metrics =['accuracy'])
learning_rate_reduction= ReduceLROnPlateau(monitor=
'val_accuracy',patience = 2,verbose=1,factor=0.3,min_lr=0.000001)
history= model.fit(datagen.flow(x_train,y_train, batch_size = 32),
epochs=10,
validation_data = datagen.flow(x_val, y_val),
callbacks = learning_rate_reduction)
```

print("Loss of the model is - ",model.evaluate(x_test,y_test)[0])

```
print("Accuracy of the model is -", model.evaluate(x_test,y_test) [1]*100, "%")

epochs = list(range(10))
fig, ax = plt.subplots(1,2)
train_acc =history.history['accuracy']
train_loss =history.history['loss']
val_acc =history.history['val_accuracy']
val_loss =history.history['val_loss']
fig.set_size_inches(20,10)
ax[0].plot(epochs, train_acc, 'go-', label ='Training Accuracy')
ax[0].plot(epochs, val_acc, 'ro-', label = 'Validation Accuracy')
ax[0].set_title('Training & Validation Accuracy')
ax[0].legend()
ax[0].set_xlabel("Epochs")
ax[0].set_ylabel("Accuracy")
ax[1].plot(epochs, train_loss, 'g-o', label = "Training Loss")
ax[1].plot(epochs, val_loss, 'r-o', label = 'Validation Loss')
ax[1].set_title('Testing Accuracy & Loss')
ax[1].legend()
ax[1].set_xlabel("Epochs")
ax[1].set_ylabel("Training & Validation Loss")
plt.show()
```

```
predictions = model.predict(x_test)
for i in range(len(predictions)):
predictions[i]= 1 if predictions[i]>0.5 else 0
print(classification_report(y_test,
predictions,
target_names=['Pneumonia (Class 0)', 'Normal (Class 1)']))
cm=confusion_matrix(y_test, predictions)
cm=pd.DataFrame(cm, index= ['0','1'], columns=['0', '1'])
cm
```

```
sns.heatmap(cm, cmap="Blues", annot=True, xticklabels = labels,
yticklabels = labels)
plt.show()
def pneumoniaPrediction(img):
```

```
img= np.array(img)/255
img=img.reshape(-1, 150, 150, 1)
isPneumonic =model.predict(img)[0]
imgClass = "Normal" if isPneumonic<0.5 else "Pneumonic"
return imgClass
pr = model.predict(x_test)
for i in range(len(pr)):
if pr[i]>0.5:
pr[i]=1
else:
pr[i]=0
img=gradio.inputs.Image( shape=(150, 150))




label = gradio.outputs.Label(num_top_classes=1)
interface=gradio.Interface(fn = pneumoniaPrediction,
title="Pneumonia Detection using Chest X-Ray",
inputs= img,
outputs=label,
interpretation = "default")
interface.launch(debug=True, share=True)
```

C. SCREENSHOTS

```
from google.colab import drive
drive.mount("/content/gdrive")
```

```
pip install gradio
```

```
Collecting gradio
  Downloading gradio-3.47.1-py3-none-any.whl (20.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 20.3/20.3 MB 55.8 MB/s eta 0:00:00
Collecting aiofiles<24.0,>=22.0 (from gradio)
  Downloading aiofiles-23.2.1-py3-none-any.whl (15 kB)
Requirement already satisfied: altair<6.0,>=4.2.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (4.2.2)
Collecting fastapi (from gradio)
  Downloading fastapi-0.103.2-py3-none-any.whl (66 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 66.3/66.3 kB 7.4 MB/s eta 0:00:00
Collecting ffmpy (from gradio)
  Downloading ffmpy-0.3.1.tar.gz (5.5 kB)
  Preparing metadata (setup.py) ... done
Collecting gradio-client==0.6.0 (from gradio)
  Downloading gradio_client-0.6.0-py3-none-any.whl (298 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 298.8/298.8 kB 22.6 MB/s eta 0:00:00
Collecting httpx (from gradio)
  Downloading httpx-0.25.0-py3-none-any.whl (75 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 75.7/75.7 kB 7.9 MB/s eta 0:00:00
Collecting huggingface-hub>=0.14.0 (from gradio)
  Downloading huggingface_hub-0.18.0-py3-none-any.whl (301 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 302.0/302.0 kB 27.6 MB/s eta 0:00:00
Requirement already satisfied: importlib-resources<7.0,>=1.3 in /usr/local/lib/python3.10/dist-packages (from gradio) (6.1.0)
Requirement already satisfied: jinja2<4.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (3.1.2)
Requirement already satisfied: markupsafe~=2.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (2.1.3)
Requirement already satisfied: matplotlib~=3.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (3.7.1)
Requirement already satisfied: numpy~=1.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (1.23.5)
Collecting orjson~=3.0 (from gradio)
  Downloading orjson-3.9.9-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (138 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 138.7/138.7 kB 15.5 MB/s eta 0:00:00
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from gradio) (23.2)
Requirement already satisfied: pandas<3.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (1.5.3)
Requirement already satisfied: pillow<11.0,>=8.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (9.4.0)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,!=2.0.0,!=2.0.1,<3.0.0,>=1.7.4 in /usr/local/lib/python3.10/dist-packages (
Collecting pydub (from gradio)
  Downloading pydub-0.25.1-py2.py3-none-any.whl (32 kB)
Collecting python-multipart (from gradio)
  Downloading python_multipart-0.0.6-py3-none-any.whl (45 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 45.7/45.7 kB 1.3 MB/s eta 0:00:00
Requirement already satisfied: pyyaml<7.0,>=5.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (6.0.1)
Requirement already satisfied: requests~=2.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (2.31.0)
Collecting semantic-version~=2.0 (from gradio)
  Downloading semantic_version-2.10.0-py2.py3-none-any.whl (15 kB)
Requirement already satisfied: typing-extensions~=4.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (4.5.0)
Collecting uvicorn>=0.14.0 (from gradio)
  Downloading uvicorn-0.23.2-py3-none-any.whl (59 kB)
```

```
Requirement already satisfied: pyyaml<7.0,>=5.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (6.0.1)
Requirement already satisfied: requests~=2.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (2.31.0)
Collecting semantic-version~=2.0 (from gradio)
  Downloading semantic_version-2.10.0-py2.py3-none-any.whl (15 kB)
Requirement already satisfied: typing-extensions~=4.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (4.5.0)
Collecting uvicorn>=0.14.0 (from gradio)
  Downloading uvicorn-0.23.2-py3-none-any.whl (59 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 59.5/59.5 kB 6.9 MB/s eta 0:00:00
Collecting websockets<12.0,>=10.0 (from gradio)
  Downloading websockets-11.0.3-cp310-cp310-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.wh
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 129.9/129.9 kB 13.5 MB/s eta 0:00:00
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from gradio-client==0.6.0->gradio) (2023.6.0)
Requirement already satisfied: entrypoints in /usr/local/lib/python3.10/dist-packages (from altair<6.0,>=4.2.0->gradio) (0.4)
Requirement already satisfied: jsonschema>=3.0 in /usr/local/lib/python3.10/dist-packages (from altair<6.0,>=4.2.0->gradio) (4.1
Requirement already satisfied: toolz in /usr/local/lib/python3.10/dist-packages (from altair<6.0,>=4.2.0->gradio) (0.12.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.14.0->gradio) (3.12.
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.14.0->gradio) (4
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib~=3.0->gradio) (1.1.1
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib~=3.0->gradio) (0.12.1)
```

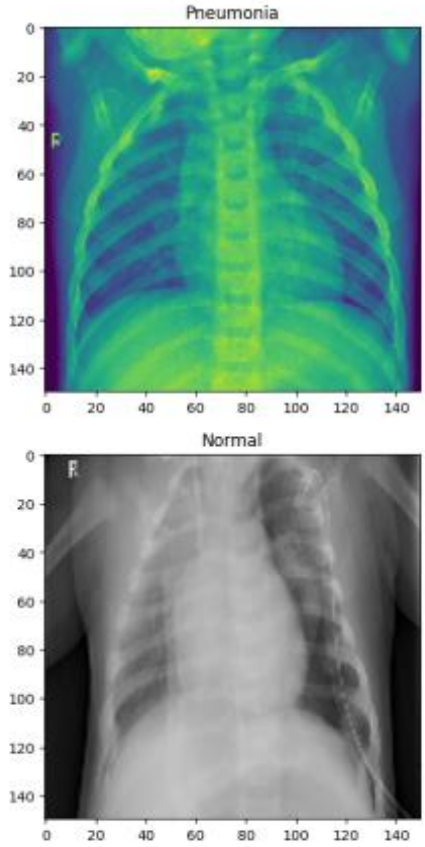
```
import matplotlib.pyplot as plt
import seaborn as sns
import keras
from keras.models import Sequential
import PIL
import tensorflow as tf
from keras.preprocessing.image import ImageDataGenerator
from keras.layers import Dense, Conv2D, MaxPool2D, Flatten, Dropout, BatchNormalization
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
from keras.callbacks import ReduceLROnPlateau
import cv2
import os
import numpy as np
import pandas as pd
import gradio
```

```
labels=['PNEUMONIA','NORMAL']
img_size = 150
def get_data(data_dir):
    data = []
    for label in labels:
        path = os.path.join(data_dir, label)
        class_num=labels.index(label)
        for img in os.listdir(path):
            try:
                img_arr=cv2.imread(os.path.join(path, img), cv2.IMREAD_GRAYSCALE)
                resized_arr= cv2.resize(img_arr, (img_size, img_size))# Reshaping images to preferred size
                data.append([resized_arr, class_num])
            except Exception as e:
                print(e)
    return np.array(data)


train=get_data('/content/drive/MyDrive/archive/chest_xray/train')
test=get_data('/content/drive/MyDrive/archive/chest_xray/test')
val=get_data('/content/drive/MyDrive/archive/chest_xray/val')
```

```
<ipython-input-4-70418b9bbc67>:15: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-
  return np.array(data)
```

```
x_train =[]
y_train = []

x_val = []
y_val = []

x_test = []
y_test = []

  for feature, label in train:
      x_train.append(feature)
      y_train.append(label)

  for feature, label in test:
      x_test.append(feature)
      y_test.append(label)

  for feature, label in val:
      x_val.append(feature)
      y_val.append(label)


  positives=[]

  negatives=[]

  for i in range(len(y_train)):
      if y_train[i]:
          positives.append(x_train[i])
      else:
          negatives.append(x_train[i])


  plt.bar(labels, [len (negatives), len(positives)], color=["green", "blue"])

  plt.title("Cases count in training data set")

  plt.ylabel("count")

  plt.show()
```

```
plt.imshow(positives[0])

plt.title("Pneumonia")

plt.show()

plt.imshow(negatives[4], cmap="gray")

plt.title("Normal")

plt.show()
```




```
#Normalize the data

x_train = np.array(x_train) / 255
x_val = np.array(x_val) / 255
x_test = np.array(x_test) / 255


#resize data for deep learning

x_train = x_train.reshape(-1, img_size, img_size, 1)

y_train= np.array(y_train)

x_val = x_val.reshape(-1, img_size, img_size, 1)
```

```
y_val = np.array(y_val)

x_test = x_test.reshape(-1, img_size, img_size, 1)

y_test= np.array(y_test)


y_train =y_train.reshape(-1,1)

y_test = y_test.reshape(-1,1)

y_val = y_val.reshape(-1,1)


datagen=ImageDataGenerator(
    featurewise_center=False,
    samplewise_center=False,
    featurewise_std_normalization=False,
    samplewise_std_normalization=False,
    zca_whitening=False,
    rotation_range= 30, #randomly rotate images in the range (degrees, 0 to 180)
    zoom_range =0.2, # Randomly zoom image
    width_shift_range=0.1, # randomly shift images horizontally (fraction of total width)
    height_shift_range=0.1, # randomly shift images vertically (fraction of total height)
    horizontal_flip = True, #randomly flip images
    vertical_flip=False)# randomly flip images
datagen.fit(x_train)
```

```
model=Sequential()
model.add(Conv2D(32, (3,3), strides = 1, padding ='same', activation= 'relu', input_shape =(150,150,1)))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2), strides =2, padding='same'))

model.add(Conv2D(64, (3,3), strides = 1, padding ='same', activation= 'relu'))
model.add(Dropout(0.1))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2), strides =2, padding='same'))

model.add(Conv2D(64, (3,3), strides = 1, padding ='same', activation= 'relu'))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2), strides =2, padding='same'))

model.add(Conv2D(128, (3,3), strides = 1, padding ='same', activation= 'relu'))
model.add(Dropout(0.2))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2), strides =2, padding='same'))

model.add(Conv2D(256, (3,3), strides = 1, padding ='same', activation= 'relu'))
model.add(Dropout(0.2))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2), strides =2, padding='same'))


model.add(Flatten())
model.add(Dense(units=128, activation= 'relu'))
model.add(Dropout(0.2))
model.add(Dense (units=1, activation ='sigmoid'))
model.compile(optimizer= "rmsprop", loss= 'binary.crossentropy', metrics = ['accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 150, 150, 32)      320

batch_normalization (BatchN  (None, 150, 150, 32)      128
ormalization)

max_pooling2d (MaxPooling2D  (None, 75, 75, 32)        0
)

conv2d_1 (Conv2D)            (None, 75, 75, 64)        18496

dropout (Dropout)            (None, 75, 75, 64)        0

batch_normalization_1 (Batc  (None, 75, 75, 64)        256
hNormalization)

max_pooling2d_1 (MaxPooling  (None, 38, 38, 64)        0
2D)

conv2d_2 (Conv2D)            (None, 38, 38, 64)        36928
```

```
batch_normalization_2 (Batc  (None, 38, 38, 64)       256
hNormalization)

max_pooling2d_2 (MaxPooling  (None, 19, 19, 64)       0
2D)

conv2d_3 (Conv2D)            (None, 19, 19, 128)      73856

dropout_1 (Dropout)          (None, 19, 19, 128)      0

batch_normalization_3 (Batc  (None, 19, 19, 128)      512
hNormalization)

max_pooling2d_3 (MaxPooling  (None, 10, 10, 128)      0
2D)

conv2d_4 (Conv2D)            (None, 10, 10, 256)      295168

dropout_2 (Dropout)          (None, 10, 10, 256)      0

batch_normalization_4 (Batc  (None, 10, 10, 256)      1024
hNormalization)

max_pooling2d_4 (MaxPooling  (None, 5, 5, 256)        0
2D)

flatten (Flatten)            (None, 6400)             0

dense (Dense)                (None, 128)              819328

dropout_3 (Dropout)          (None, 128)              0

dense_1 (Dense)              (None, 1)                129
```

```
model.compile(optimizer = "rmsprop",
              loss='binary_crossentropy',
              metrics =['accuracy'])
```

```
learning_rate_reduction= ReduceLROnPlateau(monitor= 'val_accuracy',
                                           patience = 2,
                                           verbose=1,
                                           factor=0.3,
                                           min_lr=0.000001)
```

```
history= model.fit(datagen.flow(x_train,y_train, batch_size = 32),
                   epochs=10,
                   validation_data = datagen.flow(x_val, y_val),
                   callbacks = learning_rate_reduction)
```

```
Epoch 1/10
163/163 [==============================] - 546s 3s/step - loss: 0.6333 - accuracy: 0.8455 - val_loss: 12.7120 - val_accuracy: 0.500
Epoch 2/10
163/163 [==============================] - 463s 3s/step - loss: 0.2788 - accuracy: 0.8928 - val_loss: 18.2233 - val_accuracy: 0.500
Epoch 3/10
120/163 [====================>.........] - ETA: 2:01 - loss: 0.2230 - accuracy: 0.9198
```

```
print("Loss of the model is - ",model.evaluate(x_test,y_test)[0])
print("Accuracy of the model is -", model.evaluate(x_test,y_test) [1]*100, "%")


epochs = list(range(10))
fig, ax = plt.subplots(1,2)
train_acc =history.history['accuracy']
train_loss =history.history['loss']
val_acc =history.history['val_accuracy']
val_loss =history.history['val_loss']
fig.set_size_inches(20,10)

ax[0].plot(epochs, train_acc, 'go-', label ='Training Accuracy')
ax[0].plot(epochs, val_acc, 'ro-', label = 'Validation Accuracy')
ax[0].set_title('Training & Validation Accuracy')
ax[0].legend()
ax[0].set_xlabel("Epochs")
ax[0].set_ylabel("Accuracy")



ax[1].plot(epochs, train_loss, 'g-o', label = "Training Loss")
ax[1].plot(epochs, val_loss, 'r-o', label = 'Validation Loss')
 ax[1].set_title('Testing Accuracy & Loss')
 ax[1].legend()
 ax[1].set_xlabel("Epochs")
 ax[1].set_ylabel("Training & Validation Loss")
 plt.show()


 predictions = model.predict(x_test)
 for i in range(len(predictions)):
     predictions[i]= 1 if predictions[i]>0.5 else 0


 print(classification_report(y_test,
                             predictions,
                             target_names=['Pneumonia (Class 0)', 'Normal (Class 1)']))


 cm=confusion_matrix(y_test, predictions)
 cm=pd.DataFrame(cm, index= ['0','1'], columns=['0', '1'])
 cm


 sns.heatmap(cm, cmap="Blues", annot=True, xticklabels = labels, yticklabels = labels)
 plt.show()


 def pneumoniaPrediction(img):
     img= np.array(img)/255
     img=img.reshape(-1, 150, 150, 1)
     isPneumonic =model.predict(img)[0]
     imgClass = "Normal" if isPneumonic<0.5 else "Pneumonic"
     return imgClass


 pr = model.predict(x_test)
 for i in range(len(pr)):
     if pr[i]>0.5:
         pr[i]=1
     else:
         pr[i]=0

img=gradio.inputs.Image( shape=(150, 150))
label = gradio.outputs.Label(num_top_classes=1)


interface=gradio.Interface(fn = pneumoniaPrediction,
                           title="Pneumonia Detection using Chest X-Ray",
                           inputs= img,
                           outputs=label,
                           interpretation = "default")
interface.launch(debug=True, share=True)
```

D. Research Paper

PNEUMONIA DETECTION USING TRANSFER LEARNING

Boppana Bharat Satya
Department of CSE-AI
Sathyabama Institute of Science and Technology
(deemed to be university)
Chennai, India
boppanabharatsatya@gmail.com

Rohit Sai Kalyan Kopalli
Department of CSE
Sathyabama Institute of Science and Technology
(deemed to be university)
Chennai, India
rohitkopalli90@gmail.com

ABSTRACT

A bacterial infection in the lungs results in an illness known as pneumonia. The effectiveness of treatment depends in large part on early diagnosis. For a variety of reasons, including the disease's appearance being ambiguous in chest X-ray images or being mistaken for another illness, the diagnosis may be arbitrary. Thus, in order to assist clinicians, computer-aided diagnosis systems are required. AI algorithms can automatically diagnose pneumonia or other lung-related diseases in patients by analyzing their chest X-ray scans. Because lives are at stake, the algorithm needs to be extremely accurate. In this project, we employ transfer learning (TL), a machine learning (ML) technique that allows knowledge from one task to be applied to another, improving performance on related tasks. Utilizing a custom deep CNN and images of X-ray to retrain the pre-trained model, pneumonia was identified from chest X-ray images. Freeze the first few layers and fine-tune the model for two new label classes in order to retrain the removed output layers (Pneumonia and Normal).

Keywords — machine learning, deep learning, CNN, transfer learning.

I. INTRODUCTION

Acute pulmonary infections, such as pneumonia, could be brought on by viruses, bacteria, or fungi and infect the lungs, leading to

The pleural effusion, a situation in that fluid fills the lung, and inflammation of the air sacs. It is the cause of over 15percent of deaths in children under five yrs of age. Pneumonia is more common in developing and underdeveloped nations due to factors like poor environmental situations, pollution, and overcrowding, as well as a lack of access to healthcare. Therefore, keeping the disease from becoming fatal could be highly aided by early diagnosis along with treatment. A diagnosis of lung disease is often made by lung radiological examination by utilizing the computed tomography (CT), magnetic resonance imaging (MRI), or radiography (X-rays). An examination of the lungs that is non-invasive and reasonably priced is X-ray imaging.

In this article, we have outlined our method for diagnosing pneumonia and explained how the model's performance is significantly impacted by the lung image size. We discovered that the difference between images showing pneumonia and those showing it is fairly subtle; large images can provide more detailed information. However, when working with large images, the computation cost also increases exponentially.

"Machine learning" is a branch of computer science along with AI which focuses on the simulating human learning procedures as well as enhancing their accuracy over time with data & algorithms. Put differently, "Transfer learning" refers to an ML method

in which a formerly trained model has been utilized as the foundation for the new model on a various task. Using a popular technique called ensemble learning, the final prediction for a test sample has been obtained by fusing the decisions of multiple classifiers.

One useful artificial intelligence tool that is essential to the resolution of many challenging computer vision issues is deep learning. Convolutional neural networks (CNNs), in particular, are DL (Deep Learning)models that are broadly utilized for a range of image classification tasks. But these models only function at their best while they have access to a lot of data. Such large labeled data sets are hard to come by in biomedical image classification problems because each image must be classified by a specialist doctor, a costly and time-consuming process. There is a workaround for this problem: transfer learning. This technique solves problems involving small datasets by reusing a model that was trained on a large dataset and applying its network weights. Since CNN models were trained on massive datasets like ImageNet, which has over 14 million images, they have been frequently utilized for the biomedical image classification tasks.

The WHO estimates that it kills 1.4 million children under the five yrs of age annually, making up 18percent of all child deaths under five yrs of age worldwide. Pneumonia is a global health concern that primarily impacts children along with families in South Asia along with the sub-Saharan Africa. It is possible to prevent childhood pneumonia. It can be treated with lower-tech, lower-cost care, and medication, and it could be inhibited with the simple interventions. Thus, research along with the development of computer-aided diagnosis is desperately required to decrease the mortality rate associated with pneumonia, particularly in children.

This paper presents a model which automatically classifies a patient as having pneumonia or not utilizing CNNs and DL applications. Using a deep transfer learning algorithm, the suggested methodology automatically identifies the features of the image of X-ray which indicate the existence of disease & determines if the patient is suffering from pneumonia.

II. RELATED WORKS

1. Wang et al. made available a database called Chest x-ray 14, which included 32,717 distinct patients and 112, 120 frontal view x-ray images that were labelled with eight labels. The data set was later extended to include 14 diseases, from the original 8 that were suggested. This dataset's limitation in the context of pneumonia is the small number of labelled images that have the illness, which causes a very uneven classification. In order to classify the abnormalities in the images of chest X-ray, Wang et al. proposed a 2D ConvNet that predicts the labels using a simple binary relevance. Wang and colleagues employed AlexNet, GoogleNet, ResNet, and VGG16 architecture for image classification. ResNet had also achieved the highest accuracy.

2. Rajpurkar et al. used a 121-layer CNN to create CheXNet. Utilizing the F1 metric, the paper evaluated CheXNet's performance against a radiologist's. Pneumonia is among the 14 diseases that this network could identify. As it works on an X-ray image, the model displays the localised areas in the image as well as the probability of a pathology. Using seventy percent of the images for the training, twenty percent for validation, and ten percent for testing, the model has been capable to achieve a f1 score of 0.435, higher as compared to the radiologist's(0.387).

3. In order to classify the images into viral pneumonia, bacterial pneumonia, and no pneumonia, Acharya et al. proposed a deep Siamese network (DSN) that used 5328 and 300 images for training and testing, respectively, to achieve a ROC AUC of 0.9500. Following their development, several deep CNNs were assessed by Yu-Xing Tang and colleagues.

4. In order to relieve patients with a normal chest X-ray, Ken Wong et al. classified the images into normal and disease states. They did not believe that sick patients should be sent home. Their network made use of a dilated ResNet block and Inception-ResNet-v2, which was pre-trained on ImageNet. In order for half of the patients to be diagnosed as disease-free, they set the recall rate at 50%. They also used 3217 images of the Chest X-ray 14 dataset to train this network, running it for 50 epochs in order to reach a maximum ROC AUC of 0.9300.

5. The Adam algorithm was combined with logistic regression, Amores-Falconi, and K-means clustering to train the network. However, due to limited resources, they only looked at 5606 randomly selected images. They also conclude that because of the complexity of the data set, logistic regression is not a reliable method of result prediction and that a DenseNet, with an accuracy (AUC) of 0.60, would be a better choice.

6. Li et al. combined lung field segmentation and rib suppression with a CNN-based methodology. Three CNNs have been trained on various resolution images for the lung area pixel patches, and feature fusion was used to combine all the data. For pediatric pneumonia, Liang et al. created a unique network having a residual structure that consists of two dense connection layers, one worldwide average pooling layer, and 49 convolutional layers.

7. A 3D full CNN was proposed by Pezeshk et al. for quick screening and the creation of candidate suspicious regions. After that, a large set of data augmentations from the positive along with the negative patches have been used to train an ensemble of 3-D CNNs. The classifiers were trained with various thresholds and types of data augmentation on false positive patches. Ultimately, the final prediction was generated by averaging the outputs of the 2nd stage networks.

8. For the localization of pneumonia, Sirazitdinov et al. proposed an ensemble of RetinaNet along with Mask R-CNN networks. Networks 1st identified the areas impacted by pneumonia, and after that non-max suppression was implemented in the regions of the lung that were anticipated.

9. Two 3D-customized mixed link network (CMixNet) architectures were used by Nasrullah et al. Faster R-CNN was utilized for lung nodule recognition using features acquired from CMixNet along with the U-Net, such as encoder-decoders; a gradient boosting machine (GBM) has been employed for the classification.

10. The custom neural network proposed by Pasa et al. contains 5 convolutional blocks, a worldwide average pooling layer, a completely connected softmax layer having 2 outputs, and each convolutional block having two 3×3 convolutions having Rectified Linear Units (ReLUs) followed by the max-pooling operation.

III. PROPOSED SYSTEM

This paper proposes an optimal approach for the pneumonia diagnosis from X-rays of chest. After addressing the issue of the small dataset through data augmentation, cutting-edge DL models—deliberated in Section 3—

have been optimized for the pneumonia classification. The final estimate has been after then computed by combining the predictions from these models having a weighted classifier (explained later in this section).

The deep transfer learning framework has served as the foundation for the working methods of the proposed model. In recent times, researchers have been using CNN models based on transfer learning to solve a range of computer vision issues. Over the past fewer decades, these models have found widespread application in medical disorders, diagnostics, industries, along agriculture. Here, a deep TL model based on CNN is created and applied to the image classification of an X-ray.

Szegedy et al proposed that GoogLeNet architecture is a 22-layer deep network made up of "inception modules" as opposed to layers that are progressively more advanced one after the other. Through the hosting of parallel convolution and pooling layers, an inception block can support a huge number of units at the each and every stage; however, this leads to an uncontrollably high computational complexity due to the raised number of parameters. The GoogLeNet model utilizes blocks of inception having a dimension decrease to manage the computational complexity. An ideal sparse architecture constructed from the available dense blocks of the building enhances the performance of ANNs for the computer vision tasks, as demonstrated by the performance of GoogLeNet, in which the inception block was first introduced.

He et al.'s ResNet-18 model boosts the effectiveness of deep network training by using a residual learning framework. In contrast to the initial unreferenced mapping in monotonically progressive convolutions, the ResNet models' residual blocks aid in network optimization, enhancing model accuracy. Identity mapping is carried out by the residuals, also known as "skip connections," which don't add parameters or raise computational complexity.

Huangl.' s proposed DenseNet architectures are computationally efficient and offer a rich feature representation. The main explanation for this is DenseNet model's feature maps have been concatenated with those from all earlier layers in each layer. This decreases the number of trainable parameters in the convolutional layers, which makes the model more computationally efficient. Moreover, the feature illustration is enhanced by concatenating the feature maps from the earlier layers with present layer.

IV. SYSTEM ARCHITECTURE

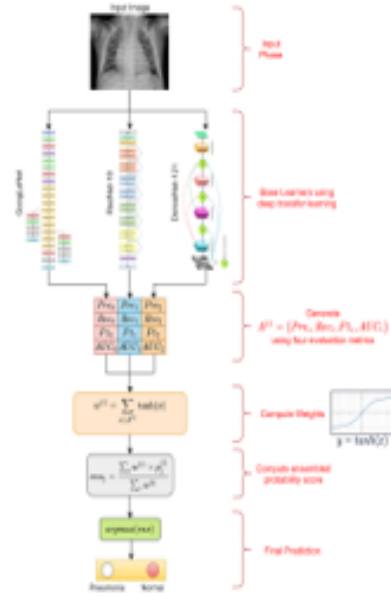


Fig. 1. System Architecture

V. RESULT

The model's robustness has been shown by the several scores—like recall, accuracy, precision, and AUC score—obtained during the experiments. A higher F1 score of 99.002 along with an AUC score of the 99.809

established the proposed model's efficacy, and it has been capable of attaining an accuracy of the 98.857 percent. Despite the fact that several approaches were developed to work with this given dataset, the suggested methodology produced superior outcomes. It would be fascinating to see methods in the future for much more efficiently calculating the weights corresponding to several models, as well as a model that makes predictions based on the patient's medical history.

VI. CONCLUSION

One major cause of morbidity along with mortality is the pneumonia. It is responsible for adult hospital admissions sizeable portion, and a sizeable portion of those patients who has been passed away (mortality rate for patients over 75 yrs is 24.8%). The World Health Organization states that early diagnosis and treatment, along with a straightforward intervention, can prevent pneumonia. However, most people on the planet do not have access to radiology diagnostics. In spite of the imaging equipment availability, there has been a dearth of specialists qualified to analyze X-rays. This paper proposed the utilization of deep transfer learning techniques for pneumonia automatic detection in images of the chest X-ray. The deep networks that we employed in our methodology produced higher accuracy due to they had much more complex structures with few parameters, which intended that they required lesser processing power. Overfitting is a phenomenon that occurs while there is not enough training data, as in the medical image processing case. It was addressed through the utilization of transfer learning along with data augmentation.

VII. REFERENCES

1. Liu, N.; Wan, L.; Zhang, Y.; Zhou, T.; Huo, H.; Fang, T. Exploiting convolutional neural networks with deeply local description for remote sensing image classification. IEEE Access 2018, 6, 11215–11228. [CrossRef]

2. Hosny, A.; Parmar, C.; Quackenbush, J.; Schwartz, L.H.; Aerts, H.J. Artificial intelligence in radiology. Nat. Rev. Cancer 2018, 18, 500–510. [CrossRef] [PubMed]

3. Kallianos, K.; Mongan, J.; Antani, S.; Henry, T.; Taylor, A.; Abuya, J.; Kohli, M. How far have we come? Artificial intelligence for chest radiograph interpretation. Clin. Radiol. 2019, 74, 338–345. [CrossRef] [PubMed]

4. Douarre, C.; Schielein, R.; Frindel, C.; Gerth, S.; Rousseau, D. Transfer learning from synthetic data applied to soil–root segmentation in x-ray tomography images. J. Imaging 2018, 4, 65. [CrossRef]

5. Sun, C.; Yang, Y.; Wen, C.; Xie, K.; Wen, F. Voiceprint identification for limited dataset using the deep migration hybrid model based on transfer learning. Sensors 2018, 18, 2399. [CrossRef] [PubMed]

6. Esteva, A.; Kuprel, B.; Novoa, R.A.; Ko, J.; Swetter, S.M.; Blau, H.M.; Thrun, S. Dermatologist-level classification of skin cancer with deep neural networks. Nature 2017, 542, 115–118. [CrossRef]

7. Shen, D.; Wu, G.; Suk, H.I. Deep learning in medical image analysis. Annu. Rev. Biomed. Eng. 2017, 19, 221–248.