INTEGRATION OF ADVANCED IMAGING TECHNOLOGIES FOR THE PREDICTIVE ANALYSIS AND DETECTION OF COVID-19

CO8811 – PROJECT REPORT

Submitted by

SHAMITHA H	**211420118045**
SRUTHINAYA K	**211420118048**
THEKSHITHA N	**211420118054**

in partial fulfillment for the award the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER AND COMMUNICATION ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

MARCH 2024

BONAFIDE CERTIFICATE

Certified that this project report **"INTEGRATION OF ADVANCED IMAGING TECHNOLOGIES FOR THE PREDICTIVE ANALYSIS AND DETECTION OF COVID-19"** is the bonafide work of **SHAMITHA H (211420118045), SRUTHINAYA K (211420118048) AND THEKSHITHA N (211420118054) who** carried out the project work under my supervision.

SIGNATURE

Dr. B. ANNI PRINCY M.E., Ph.D.,

HEAD OF THE DEPARTMENT

PROFESSOR,
COMPUTER AND COMMUNICATION ENGINEERING,
PANIMALAR ENGINEERING COLLEGE,
NAZARATHPETTAI, POONAMALLEE,
CHENNAI- 600123.

SIGNATURE

V. MUTHU M. Tech (Ph.D.)

SUPERVISOR

ASSISTANT PROFESSOR,
ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING,
PANIMALAR ENGINEERING COLLEGE,
NAZARATHPETTAI, POONAMALLEE,
CHENNAI- 600123.

Certified that the above candidate(s) was/ were examined in the End Semester Project Viva-Voce Examination held on.....

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We express our deep gratitude to our respected Secretary andCorrespondent **Dr. P. CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like to extend our heartfelt and sincere thanks to our Directors Tmt. **C.VIJAYARAJESWARI, Dr. C. SAKTHIKUMAR, M.E., Ph.D**., and **Dr. SARANYASREE SAKTHIKUMAR B.E., M.B.A., Ph.D**., for providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr. K. MANI, M.E., Ph.D.,** for his timely concern and encouragement provided to us throughout the course.

We thank our HOD of Computer and Communication Engineering Department, **Dr. B. ANNI PRINCY, M.E., Ph.D**., Professor, for the support extended throughout the project.

We would like to thank our supervisor, **Mr. V. MUTHU M. Tech (Ph.D.),** Assistant Professor, and all the faculty members of the Department of Artificial Intelligence and Machine Learning for their advice and suggestions for the successful completion of the project.

SHAMITHA H

SRUTHINAYA K

THEKSHITHA N

ABSTRACT

The COVID-19 pandemic, which began in Wuhan, China, emerged as a global health crisis, causing widespread disruptions. Efforts to combat its impact include the development of crucial vaccines like Sputnik V, Novavax and Sanofi Pasteur. In the year 2021 the worldwide COVID-19 cases were declared as zero but at the year 2023 the covid cases were started rising and every day hundreds of cases were recorded in hospitals. So, we decided to predict and detect the COVID-19 using CT-Scan images, Thermal camera images and Difficulty in breath with the help of deep learning and machine learning algorithms. We integrated advanced imaging technologies to enhance predictive analysis and detection capabilities. Our research aims to provide a comprehensive solution for early identification and analysis of COVID-19. Thermal cameras offer real-time fever detection, CT scan provide detailed imaging of the respiratory system. While, Difficulty in breath is the symptom of COVID-19. We are using combined CNN and RNN algorithm for COVID-19 detection using CT scan images, fever detection using thermal camera images and RNN algorithm for Difficulty in breath detection. This integrated approach holds significant potential for improving public health outcomes by facilitating early intervention and targeted management strategies.

Keywords: COVID-19, CT-Scan, Thermal Camera, Deep Learning, Machine Learning, Early Detection.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVATIONS

S.NO	ABBREVATION	EXPANSION
1	CNN	Convolutional Neural Network
2	RNN	Recurrent Neural Network
3	LSTM	Long Short-Term Memory
4	CT	Computed Tomography
5	COVID-19	COronaVIrus Disease of 2019
6	RELU	Rectified Linear Unit

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

This project aims to develop a multi-modal approach for the prediction and detection of COVID-19 infection, fever, and difficulty in breathing using various modalities. Firstly, utilizing Computed Tomography (CT) images, a Convolutional Neural Network (CNN) algorithm combined with a Recurrent Neural Network (RNN) algorithm will be employed to predict and detect COVID-19 infection. Secondly, thermal camera images will be utilized for fever prediction, where a CNN-RNN algorithm will extract features and model sequences to predict fever. Lastly, audio signals will be analyzed using an RNN algorithm to detect difficulty in breathing, enhancing early detection and monitoring capabilities. This multi-modal approach intends to provide a comprehensive framework for accurate and timely diagnosis of COVID-19 and associated symptoms, aiding in effective healthcare management.

1.2 PROBLEM STATEMENT

- **COVID-19 Detection using CNN combined RNN Algorithm with CT Images**

  Develop a deep learning model that can accurately detect COVID-19 infection from CT scan images. Utilize Convolutional Neural Networks (CNNs) for feature extraction from CT images. Implement Recurrent Neural Networks (RNNs) for sequence modeling and capturing temporal dependencies in image data.

- **Fever Detection using CNN combined RNN Algorithm with Thermal Camera Images**

  Create a deep learning framework for fever detection based on thermal camera images. Apply CNNs to extract relevant features from thermal images capturing body temperature. Incorporate RNNs to analyse temporal patterns in temperature variations for accurate fever detection.

- **Difficulty in Breath Detection using RNN Algorithm with Audio**

  Develop an RNN-based algorithm to detect difficulties in breathing using audio signals. Collect a dataset of audio recordings containing breathing patterns, including normal and abnormal instances. Design a recurrent neural network architecture capable of learning from sequential audio data to identify breathing abnormalities.

1.3 OBJECTIVES

The main objective of this project is to develop a comprehensive solution for the early prediction and detection of COVID-19 using deep learning and machine learning algorithms. Leveraging advanced imaging technologies such as thermal cameras, CT scans, and symptom assessment like shortness of breath, the project aims to provide a robust system capable of identifying potential COVID-19 cases promptly. By integrating multiple modalities and employing sophisticated algorithms, the project seeks to enhance predictive analysis and detection capabilities, thereby facilitating early intervention and targeted management strategies to mitigate the spread of the virus.

1.4 METHODOLOGY

Data Collection: Gather a diverse dataset of CT images, thermal camera images, and audio recordings related to COVID-19 diagnosis and symptom detection.

Preprocessing: Preprocess the data to enhance quality, normalize features, and prepare it for model training.

Model Development:

- For COVID-19 detection, design a CNN-RNN architecture to process CT images and predict infection probabilities.
- For fever detection, create a similar CNN-RNN framework tailored to thermal camera images for fever identification.
- For breath difficulty detection, develop an RNN model capable of analysing sequential audio data to detect abnormal breathing patterns.

Training and Evaluation: Train the models using appropriate loss functions and optimization techniques. Evaluate model performance using metrics such as accuracy, precision, recall, and F1 score.

Integration: Integrate the developed models into a unified system for comprehensive COVID-19 diagnosis and symptom monitoring.

1.5 LANGUAGE USED

Deep Learning Model Development

Python: Python is the most common language for deep learning tasks due to its extensive libraries and frameworks specifically designed for machine learning and deep learning. Libraries like TensorFlow, PyTorch, and Keras are widely used for building and training deep learning models.

Python Libraries

1. **TensorFlow** or **PyTorch**: It will be used as the primary deep learning framework for building and training convolutional neural networks (CNNs), recurrent neural networks (RNNs), and combined CNN-RNN models. These frameworks provide extensive support for implementing various neural network architectures and offer efficient computation on both CPUs and GPUs.
2. **Keras:** Keras, as a high-level neural networks API, will be utilized for developing and prototyping CNN and RNN models. Keras provides a user-

friendly interface for building neural networks with minimal code, making it suitable for rapid experimentation and model iteration.

3. **OpenCV**: OpenCV will be employed for image processing tasks, including loading, preprocessing, and augmenting image datasets used in conjunction with CNN and combined CNN-RNN models. OpenCV offers a wide range of functions and algorithms for image manipulation and analysis, facilitating efficient data preprocessing pipelines.
4. **Librosa**: Librosa will be utilized for audio feature extraction and analysis in conjunction with RNN algorithms. Librosa provides functionalities for loading audio data, extracting features such as spectrograms or Mel-frequency cepstral coefficients (MFCCs), and performing audio signal processing tasks required for training RNN models.
5. **Matplotlib** or Seaborn for data visualization
6. **NumPy** and **Pandas** for data manipulation and analysis

CHAPTER 2

LITERATURE SURVEY

2.1 INTRODUCTION

This literature survey delves into the integration of advanced imaging technologies for predictive analysis and detection of COVID-19, aiming to address the pressing need for accurate and efficient diagnostic methods amidst the ongoing pandemic. By examining the current landscape of diagnostic approaches and their limitations, we highlight the potential of advanced imaging modalities, including thermal cameras and CT scans, to enhance detection capabilities. Leveraging the power of artificial intelligence (AI) algorithms, particularly Convolutional Neural Networks (CNNs) combined with Recurrent Neural Networks (RNNs), we propose innovative solutions for comprehensive COVID-19 diagnosis and monitoring. Specifically, we explore the utilization of CNN-RNN models for analysing CT images and thermal camera images for COVID-19 detection, alongside RNN algorithms for difficulty in breathing (DIB) detection using audio data. This literature survey aims to provide insights into emerging research directions and contribute to the development of effective strategies for combating the spread of the virus.

2.2 REVIEW

1. **Title**: CovFrameNet: An Enhanced Deep Learning Framework for COVID-19 Detection

   Author: Olaide Nathaniel Oyelade, Absalom El-Shamir Ezugwu and Haruna Chiroma

   Year: 2021

Abstract

The novel coronavirus, also known as COVID-19, is a pandemic that has weighed heavily on the socio-economic affairs of the world. Research into the production of relevant vaccines is progressively being advanced with the development of the Pfizer and BioNTech, AstraZeneca, Moderna, Sputnik V, Janssen, Sinopharm, Valneva, Novavax and Sanofi Pasteur vaccines. There is, however, a need for a computational intelligence solution approach to mediate the process of facilitating quick detection of the disease. Different computational intelligence methods, which comprise natural language processing, knowledge engineering, and deep learning, have been proposed in the literature to tackle the spread of coronavirus disease. More so, the application of deep learning models has demonstrated an impressive performance compared to other methods. This paper aims to advance the application of deep learning and image pre-processing techniques to characterise and detect novel coronavirus infection. Furthermore, the study proposes a framework named CovFrameNet., which consist of a pipelined image pre-processing method and a deep learning model for feature extraction, classification, and performance measurement. The novelty of this study lies in the design of a CNN architecture that incorporates an enhanced image pre-processing mechanism. The National Institutes of Health (NIH) Chest X-Ray dataset and COVID-19 Radiography database were used to evaluate and validate the effectiveness of the proposed deep learning model. Results obtained revealed that the proposed model achieved an accuracy of 0.1, recall/precision of 0.85, F-measure of 0.9, and specificity of 1.0. Thus, the study's outcome showed that a CNN-based method with image pre-processing capability could be adopted for the pre-screening of suspected COVID-19 cases, and the confirmation of RT-PCR-based detected cases of COVID-19.

Limitation

The CNN architecture proposed in this study was designed based on the authors' expertise in neural network architectures. Although the number of parameters was memory demanding, this could be further scaled down by optimising hyperparameters, which will eliminate operations that do not significantly contribute to the algorithm.

2. **Title:** Screening of COVID-19 Suspected Subjects Using Multi-Crossover Genetic Algorithm Based Dense Convolutional Neural Network.

   Author: Dilbag Singh, Vijay Kumar, Manjit Kaur, Mohamed Yaseen Jabarulla and Heung-No Lee

   Year: 2021

   Abstract

   Fast and accurate screening of novel coronavirus (COVID-19) suspected subjects plays a vital role in timely quarantine and medical care. Deep transfer learning-based screening models on chest X-ray (CXR) are effective for countering the COVID-19 outbreak. However, an efficient screening of COVID-19 is still a huge task due to the spatial complexity of CXRs. In this paper, a dense convolutional neural network (DCov-Net) based transfer learning model is proposed for the screening of COVID-19 suspected subjects using CXR images. A modified multi-crossover genetic algorithm (MMCGA) is then proposed to tune the hyper-parameters of DCov-Net. Majority of the existing COVID-19 diagnosis models are not interpretable as they do not provide any transparency to the users. Therefore, the concept of heat-maps is used to achieve explainability and interpretability. MMCGA based DCov-Net is implemented on a multiclass dataset that contains four different classes. Experimental results reveal that MMCGA based DCov-Net achieves better performance than the existing models. The proposed MMCGA based DCov-Net can be utilized for initial

screening of COVID-19 suspected subjects with an accuracy of 99.34 ± 0.51%.

Limitations

Existing COVID-19 screening models suffer from underfitting issues due to the lack of labelled datasets - The existing models have focused on COVID-19 screening as a binary or three-class problem, but there exists a similarity between CXRs of COVID-19 suspected subjects and CXRs of patients who have other lung diseases - The database used is completely imbalanced, leading to the utilization of data augmentation to balance the database - Suggestions for further research include extending the proposed model by designing an evolving deep transfer learning model and integrating image preprocessing techniques to address noise and poor visibility issues in real-time applications.

3. **Title:** A Comprehensive Review of Deep Learning-Based Methods for COVID-19 Detection Using Chest X-Ray Images.

   Author: Saeed S. Alahmari, Baderaldeen Altazi, Jisoo Hwang, Samuel Hawkins and Tawfiq Salem

   Year: 2022

   Abstract

   The novel coronavirus disease 2019 (COVID-19) added tremendous pressure on healthcare services worldwide. COVID-19 early detection is of the utmost importance to control the spread of the coronavirus pandemic and to reduce pressure on health services. There have been many approaches to detect COVID-19; the most commonly used one is the nasal swab technique. Before that was available chest X-ray radiographs were used. X-ray radiographs are a primary care method to reveal lung infections, which allows physicians to assess and plan a course of treatment. X-ray machines are prevalent, which makes this method a preferable first approach for the

detection of new diseases. However, this method requires a radiologist to assess each chest X-ray image. Therefore, different automated methods using machine learning techniques have been proposed to assist in speeding up diagnoses and improving the decision-making process. In this paper, we review deep learning approaches for COVID-19 detection using chest X-ray images. We found that the majority of deep learning approaches for COVID-19 detection use transfer learning. A discussion of the limitations and challenges of deep learning in radiography images is presented. Finally, we provide potential improvements for higher accuracy and generalisability when using deep learning models for COVID-19 detection.

Limitations

Deep learning models for COVID-19 detection in chest X-ray images face challenges like limited labelled data, potential bias, and interpretability concerns. Issues such as class imbalance, ethical considerations, and difficulties in model transferability across imaging modalities need careful attention. Striking a balance between model performance and ethical deployment, addressing interpretability concerns, and ensuring generalizability remain key focal points for improvement in this field.

4. **Title:** Deep Neural Network with Hyperparameter Tuning on Early Detection for Symptom Recognition in Suspected Covid-19.

   Author: Djuniadi, Nur Iksan, Alfa Faridh Suni, Ahmad Fashiha Hastawan

   Year: 2023

   Abstract

   The Coronavirus outbreak (COVID-19) is still a concern for the world according to WHO. Although this virus has been controlled, prevention efforts are still being carried out. Prevention of the virus can be done by identifying patients with symptoms such as fever, respiratory distress, and sore throat. This research aims to develop an early detection system through

the recognition of symptoms of COVID-19 infection using thermal camera sensors combined with the DNN using Hyperparameter Tuning. The result is the DNN algorithm can be proposed as the right algorithm to detect someone suspected of COVID-19.

Limitations

Data availability and diversity – limited access to diverse and comprehensive dataset may impact the model's generalizability.

5. **Title:** A Machine Learning Approach as an Aid for Early COVID-19 Detection

   Author: Roberto Martinez-Velazquez, Diana P. Tobón V, Alejandro Sanchez, Abdulmotaleb El Saddik and Emil Petriu

   Year: 2021

   Abstract

   The novel coronavirus SARS-CoV-2 that causes the disease COVID-19 has forced us to go into our homes and limit our physical interactions with others. Economies around the world have come to a halt, with non-essential businesses being forced to close in order to prevent further propagation of the virus. Developing countries are having more difficulties due to their lack of access to diagnostic resources. In this study, we present an approach for detecting COVID-19 infections exclusively on the basis of self-reported symptoms. Such an approach is of great interest because it is relatively inexpensive and easy to deploy at either an individual or population scale. Our best model delivers a sensitivity score of 0.752, a specificity score of 0.609, and an area under the curve for the receiver operating characteristic of 0.728. These are promising results that justify continuing research efforts towards a machine learning test for detecting COVID-19.

Limitations

The limitations include the difficulty in conducting a proper evaluation due to the pandemic, potential for improved results with a larger dataset, need for effective prioritization of access to COVID-19 tests, and establishment of a baseline for future research.

6. **Title:** Detection of COVID-19 using deep learning techniques and classification methods

   Author: Çinare Oğuz, Mete Yağanoğlu

   Year: 2021

   Abstract

   Since the patient is not quarantined during the conclusion of the Polymerase Chain Reaction (PCR) test used in the diagnosis of COVID-19, the disease continues to spread. In this study, it was aimed to reduce the duration and amount of transmission of the disease by shortening the diagnosis time of COVID-19 patients with the use of Computed Tomography (CT). In addition, it is aimed to provide a decision support system to radiologists in the diagnosis of COVID-19. In this study, deep features were extracted with deep learning models such as ResNet-50, ResNet101, AlexNet, Vgg-16, Vgg-19, GoogLeNet, SqueezeNet, Xception on 1345 CT images obtained from the radiography database of Siirt Education and Research Hospital. These deep features are given to classification methods such as Support Vector Machine (SVM), k Nearest Neighbour (kNN), Random Forest (RF), Decision Trees (DT), Naive Bayes (NB), and their performance is evaluated with test images. Accuracy value, F1-score and ROC curve were considered as success criteria. According to the data obtained as a result of the application, the best performance was obtained with ResNet-50 and SVM method. The accuracy was 96.296%, the F1-score was 95.868%, and the AUC value was 0.9821. The deep learning model and classification method

examined in this study and found to be high performance can be used as an auxiliary decision support system by preventing unnecessary tests for COVID-19 disease.

Limitations

The limitations include potential biases from open-source databases, small dataset sizes, data imbalance, lack of information on run times, and the need for future studies with diverse datasets and additional lung diseases.

7. **Title:** Deep Learning with hyper-parameter tuning for COVID-19 Cough Detection

   Author: Sunil Rao, Vivek Narayanaswamy, Michael Esposito, Jayaraman Thiagarajan, Andreas Spanias

   Year: 2022

   Abstract

   As the COVID-19 pandemic continues, rapid non-invasive testing has become essential. Recent studies and benchmarks motivated the use of modern artificial intelligence (AI) tools that utilize audio waveform spectral features of coughing for COVID19 diagnosis. In this paper, we describe the system we developed for COVID-19 coughing detection. We utilize features directly extracted from the coughing audio and use deep learning algorithms to develop automated diagnostic tools for COVID-19. In particular, we develop a novel modification of the VGG-13 deep learning architecture for audio analysis that uses log-Mel spectrograms and uses a combination of binary cross entropy and focal losses for training. This unique modification enabled the model to achieve highly robust classification on the DiCOVA 2021 COVID-19 dataset. We also explore the use of data augmentation and an ensembling strategy to further improve the performance on the validation and the blind test datasets. Our model achieved an average validation AUC of 82.23% and a test AUC of 78.3% at a sensitivity of 80.49%.

Limitations

While leveraging deep learning with hyper-parameter tuning for COVID-19 cough detection holds promise, this approach faces limitations such as the scarcity of labelled datasets specifically focused on COVID-19 coughs, leading to potential model overfitting and generalization challenges. Additionally, the dynamic nature of cough sounds across individuals and the evolving characteristics of the virus may hinder the model's ability to consistently and accurately identify COVID-19 cases. Addressing these limitations requires concerted efforts in data collection, standardization, and continual adaptation of models to changing disease characteristics.

8. **Title:** An AI-enabled pre-trained model-based Covid detection model using chest X-ray images

   Author: Rajeev Kumar Gupta, Nilesh Kunhare, Nikhlesh Pathik, Babita Pathik

   Year: 2022

   Abstract

   The year 2020 and 2021 was the witness of Covid 19 and it was the leading cause of death throughout the world during this time period. It has an impact on a large geographic area, particularly in countries with a large population. Due to the fact that this novel coronavirus has been detected in all countries around the world, the World Health Organization (WHO) has declared Covid-19 to be a pandemic. This novel coronavirus spread quickly from person to person through the saliva droplets and direct or indirect contact with an infected person. The tests carried out to detect the Covid-19 are time-consuming and the primary cause of rapid growth in Covid19 cases. Early detection of Covid patient can play a significant role in controlling the Covid chain by isolation the patient and proper treatment at the right time. Recent research on Covid-19 claim that Chest CT and X-ray images can be

used as the preliminary screening for Covid-19 detection. This paper suggested an Artificial Intelligence (AI) based approach for detecting Covid-19 by using X-ray and CT scan images. Due to the availability of the small Covid dataset, we are using a pre-trained model. In this paper, four pre-trained models named VGGNet-19, ResNet50, InceptionResNetV2 and MobileNet are trained to classify the X-ray images into the Covid and Normal classes. A model is tuned in such a way that a smaller percentage of Covid cases will be classified as Normal cases by employing normalization and regularization techniques. The updated binary cross entropy loss (BCEL) function imposes a large penalty for classifying any Covid class to Normal class. The experimental results reveal that the proposed InceptionResNetV2 model outperforms the other pre-trained model with training, validation and test accuracy of 99.2%, 98% and 97% respectively.

Limitations

Small dataset size, Dataset imbalance leading to biased model, Model's inability to learn patterns due to smaller number of layers and iterations, decreased model performance in testing phase, Need for further evaluation on real-world datasets.

9. **Title:** Covid-19 classification using sigmoid based hyper-parameter modified DNN for CT scans and chest X-rays

   Author: B Anilkumar, K Srividya and A Mary Sowjanya

   Year: 2022

   Abstract

   Coronavirus disease (COVID-19) is an infectious disease caused by the SARS-CoV-2 virus. Diagnosis of Computed Tomography (CT), and Chest X-rays (CXR) contains the problem of overfitting, earlier diagnosis, and mode collapse. In this work, we predict the classification of the Corona in CT and CXR images. Initially, the images of the dataset are pre-processed

using the function of an adaptive Gaussian filter for de-nosing the image. Once the image is pre-processed it goes to Sigmoid Based Hyper-Parameter Modified DNN(SHMDNN). The hyperparameter modification makes use of the optimization algorithm of adaptive grey wolf optimization (AGWO). Finally, classification takes place and classifies the CT and CXR images into 3 categories namely normal, Pneumonia, and COVID-19 images. Better accuracy of 99.9% is reached when compared to different DNN networks.

Limitations

Despite the potential of using a sigmoid-based hyper-parameter modified deep neural network (DNN) for COVID-19 classification in CT scans and chest X-rays, this approach faces limitations. Challenges include the need for large and diverse datasets to ensure model generalization, potential biases introduced during training, interpretability issues associated with complex DNN architectures, and the necessity for extensive computational resources. Addressing these limitations is crucial for enhancing the reliability and practical applicability of the proposed methodology in real-world healthcare scenarios.

10. Title: Deep Learning Approach for COVID-19 Detection Based on X-Ray Images

Author: Hayat O.alasasfeh, Taqwa Alomari, MS Ibbini

Year: 2021

Abstract

COVID-19 is an infectious disease that has invaded the world since 2019 starting from China. It is caused by a new member of the corona viruses' family that attacks the respiratory system of living body leading to fever, cough, general tiredness and to death in worst case scenarios. The disease is commonly detected by RT-PCR tests, which is time consuming and relatively expensive. The need for faster, cheaper, and more precise

diagnostic tool raised the demand for the utilization of technology and artificial intelligence in this purpose. This study aims to build a robust deep learning algorithm using convolutional neural networks (CNNs) that is capable to classify chest X-ray images into COVID19, viral pneumonia, and normal cases. X-ray is a safe and inexpensive imaging system that is available at all hospitals and healthcare centres. A novel CNN model built from scratch (COV-X) has been introduced and shown 94% accuracy in the classification purpose. VGG16, VGG19, RESNET50, XCEPTION, and MOBILENET are pre-trained models that have shown an accuracy of 95%, 95%, 33%, 65%, and 77%, respectively. Findings prove that deep learning is an effective technique for early detection of COVID-19, it provides automatic detection with high reliability to help the healthcare professions and avoid the pandemic from spreading more.

Limitations

Small dataset and low computational power, need for enhancements for higher accuracy with a larger dataset, increasing the depth of the neural networks does not necessarily improve the model's performance, more robust models can be achieved by taking the health history of the patients into account, along with their current status and symptoms, adding more classes to be diagnosed, especially viral diseases with similar symptoms as COVID-19, future work involving large datasets and utilizing patient history of infection.

CHAPTER 3

SYSTEM DEVELOPMENT

The system design for the project encompasses several key components focused on COVID-19 detection using deep learning models from CT images, fever detection from thermal camera images, and difficulty in breath detection from audio signals. The initial step involves gathering diverse datasets comprising CT scan images, thermal camera images, and audio recordings related to breathing patterns. These datasets are then preprocessed to ensure data quality and uniformity. Preprocessing tasks include resizing CT and thermal images, normalizing audio signals, and organizing the data into suitable training, validation, and testing sets. The project employs distinct deep learning architectures tailored to each task. For COVID-19 detection, a combination of Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) is utilized. The CNN extracts relevant features from CT scan images, while the RNN captures temporal dependencies and sequence information. Similarly, the fever detection model utilizes a CNN-RNN hybrid approach to analyse thermal images over time for fever identification. The breath difficulty detection model focuses on RNN-based algorithms capable of analysing sequential audio data to detect abnormal breathing patterns. Each model undergoes rigorous training using appropriate loss functions, optimization algorithms, and hyperparameters. The training process involves feeding the pre-processed data into the respective model architectures to learn and generalize patterns. Following training, model performance is evaluated using a range of metrics such as accuracy, precision, recall and F1 score. A user-friendly interface is designed to allow users to interact with the system, input relevant data (CT images, thermal images, audio recordings), and view diagnostic results.

3.1 PROPOSED SYSTEM ARCHITECTURE DESIGN

Our proposed system consists of three modules such as:

a) Detection of COVID-19 using CT-Scan images with CNN combined RNN algorithm

   Convolutional Neural Network (CNN) This component of the algorithm is responsible for extracting relevant features from CT scan images. CNNs are specialized neural networks designed for image processing tasks, capable of capturing spatial patterns and hierarchical representations within images.

   Recurrent Neural Network (RNN) The RNN component is integrated with the CNN to model temporal dependencies and sequences within the image data. RNNs are suitable for handling sequential data and capturing patterns over time. In the context of COVID-19 detection, RNNs can help analyze how features in CT images evolve across multiple scans, aiding in identifying infection patterns characteristic of COVID-19.

b) Fever Detection using Thermal Camera images with CNN combined RNN algorithm

   Convolutional Neural Network (CNN) Similar to the COVID-19 detection algorithm, this CNN component processes thermal camera images to extract features related to body temperature patterns. CNNs are adept at analyzing spatial features in images, crucial for identifying fever-related patterns.

   Recurrent Neural Network (RNN) The RNN layer in this algorithm analyzes temporal variations in temperature captured by the thermal images. By incorporating an RNN, the algorithm can track temperature changes over time, distinguishing normal variations from fever patterns indicative of potential infections.

c) Difficulty in breath Detection using audio with RNN algorithm

   Recurrent Neural Network (RNN) In this algorithm, an RNN is employed to process sequential audio data capturing breathing sounds. RNNs are well-

suited for sequential data analysis and can learn temporal patterns in audio signals.

Attention Mechanisms or Long Short-Term Memory (LSTM) Attention mechanisms or LSTM cells may be incorporated within the RNN architecture. Attention mechanisms allow the model to focus on relevant parts of the audio sequence, aiding in detecting abnormalities in breathing patterns. LSTM cells, known for their ability to capture long-term dependencies, can improve the model's ability to recognize subtle variations and trends in breathing sounds associated with difficulty in breath detection.

a) Detection of COVID-19 using CT-Scan images with CNN combined RNN algorithm

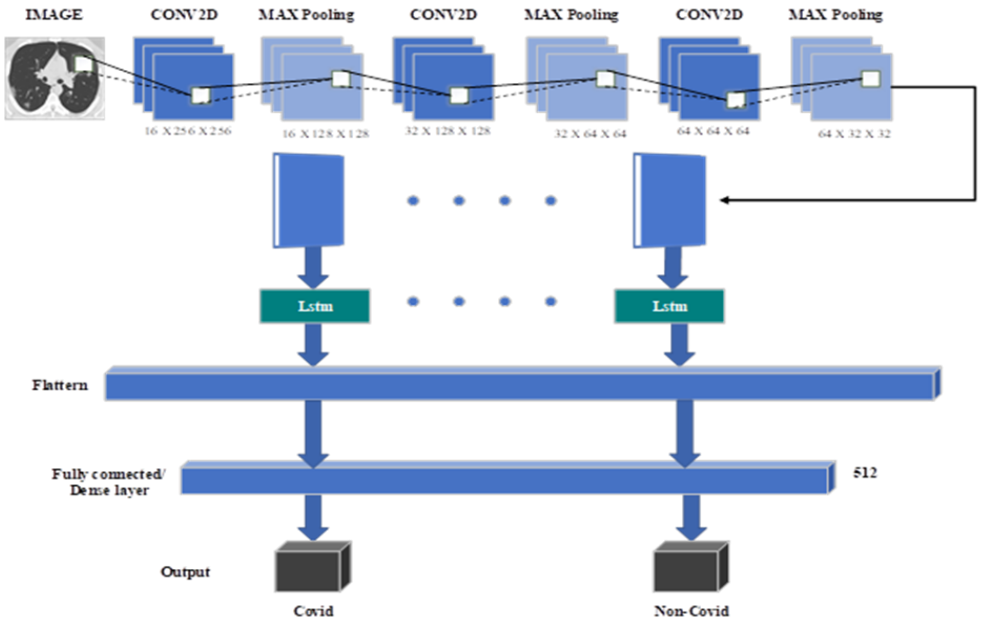

Figure 3.1 Block Diagram of COVID-19 detection with CNN combine RNN algorithm using CT-Scan images.

IMAGE

This is the input image, which is a CT scan image. It has a dimension of 26x26x3, which means it has 26 pixels in height and width, and 3 channels for red, green, and blue colors.

CONV2D

This is a convolutional layer, which applies a set of filters to the input image to detect edges, shapes, and patterns. Each filter produces a feature map, which is a representation of the input image. The dimension of the feature map depends on the number of filters, the size of the filters, and the stride and padding parameters $\text{Output size}=(126-3+2\times0)+1=24$ So the output feature map size is $24\times24\times32$ (32 feature maps). In this block diagram, the first convolutional layer has 32 filters of size 3x3, and produces a feature map of size 26x26x32.

MAX Pooling

This is a pooling layer, which reduces the size of the feature map by applying a max operation over a sliding window. This helps to reduce the computational cost and avoid overfitting. The size of the output depends on the size and stride of the window. For this layer: $\textbf{Output size= }(\mathbf{(24-2)/2)+1=12}$ So the output feature map size is $12\times12\times32$.

LSTM

This is a long short-term memory network, which is a type of recurrent neural network that can handle sequential data. It has a memory cell that can store and update information over time, and a set of gates that control the flow of information. It can learn long-term dependencies and capture temporal patterns. It has several components including input gate, forget gate, and output gate. They receive the features from the max pooling layers and output a vector of size 64.

Flatten

This is an operation that converts the 2D matrix output of the LSTM networks into a 1D vector. This is necessary to connect the LSTM networks with the fully connected layer. The flattened vector has a size of 128, which is the sum of the

sizes of the two LSTM outputs. If the output from the LSTM networks is M rows and N columns, the flattened vector will have a size of $M \times N$.

Fully connected/Dense layer

The Fully Connected or Dense layer serves as a high-capacity processing unit within the neural network. With 512 output units and employing the Rectified Linear Unit (ReLU) activation function, this layer performs a linear transformation followed by a non-linear activation. This enables the model to learn complex mappings between the flattened LSTM outputs and the desired output classes (Covid or Non-Covid), facilitating higher-level feature learning and abstraction.

Output

The final output layer consists of two units, representing the classes "Covid" and "Non-Covid." Using The final layer uses the SoftMax activation function to convert the output of the previous layer into probabilities. This ensures that the probabilities sum up to one, making it suitable for multi-class classification tasks. The class with the highest probability is considered the predicted class, providing a probabilistic assessment of whether the input CT scan image indicates the presence of COVID-19 or not.

b) Fever Detection using Thermal Camera images with CNN combined RNN algorithm

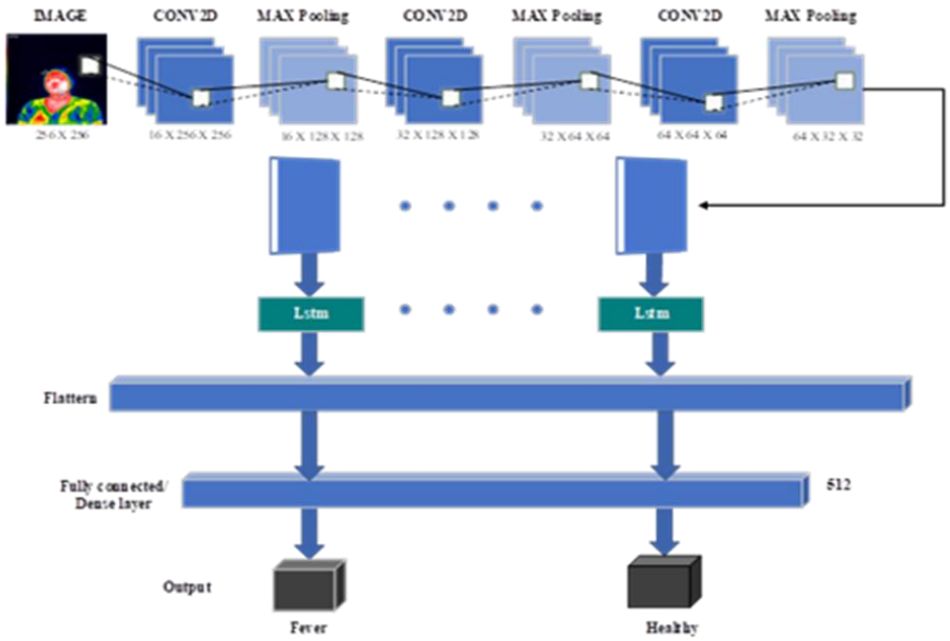

Figure 3.2 Block Diagram of Fever Detection using Thermal Camera images with CNN combined RNN algorithm

IMAGE

This is the input image, which is a thermal or heat signature image. It has a dimension of 26x26x3, which means it has 26 pixels in height and width, and 3 channels for red, green, and blue colors.

CONV2D

This is a convolutional layer, which applies a set of filters to the input image to detect edges, shapes, and patterns. Each filter produces a feature map, which is a representation of the input image. The dimension of the feature map depends on the number of filters, the size of the filters, and the stride and padding parameters. For this layer: $\text{Output size} = (126-3+2\times0) +1=24$ So the output feature map size is $24\times24\times32$ (32 feature maps). In this block diagram, the first

convolutional layer has 32 filters of size 3x3, and produces a feature map of size 26x26x32.

MAX Pooling

This is a pooling layer, which reduces the size of the feature map by applying a max operation over a sliding window. This helps to reduce the computational cost and avoid overfitting. The size of the output depends on the size and stride of the window. For this layer: **Output size= $((24-2)/2)+1=12$** So the output feature map size is $12\times12\times32$.

LSTM

This is a long short-term memory network, which is a type of recurrent neural network that can handle sequential data. It has a memory cell that can store and update information over time, and a set of gates that control the flow of information. It can learn long-term dependencies and capture temporal patterns. It has several components including input gate, forget gate, and output gate. In this block diagram, there are two LSTM networks in parallel, each with 64 hidden units. They receive the features from the max pooling layers and output a vector of size 64.

Flatten

This is an operation that converts the 2D matrix output of the LSTM networks into a 1D vector. This is necessary to connect the LSTM networks with the fully connected layer. The flattened vector has a size of 128, which is the sum of the sizes of the two LSTM outputs. If the output from the LSTM networks is $\boldsymbol{M}$ rows and $\boldsymbol{N}$ columns, the flattened vector will have a size of $\boldsymbol{M\times N}$.

Fully connected/Dense layer

The Fully Connected or Dense layer serves as a high-capacity processing unit within the neural network. With 512 output units and employing the Rectified

Linear Unit (ReLU) activation function, this layer performs a linear transformation followed by a non-linear activation. This enables the model to learn complex mappings between the flattened LSTM outputs and the desired output classes (Healthy or Unhealthy), facilitating higher-level feature learning and abstraction.

Output

The final output layer consists of two units, representing the classes "Healthy" and "Unhealthy." Using The final layer uses the SoftMax activation function to convert the output of the previous layer into probabilities. This ensures that the probabilities sum up to one, making it suitable for multi-class classification tasks. The class with the highest probability is considered the predicted class, providing a probabilistic assessment of whether the input Thermal image indicates the presence of Fever or not.

c) Difficulty in breath Detection using audio with RNN algorithm

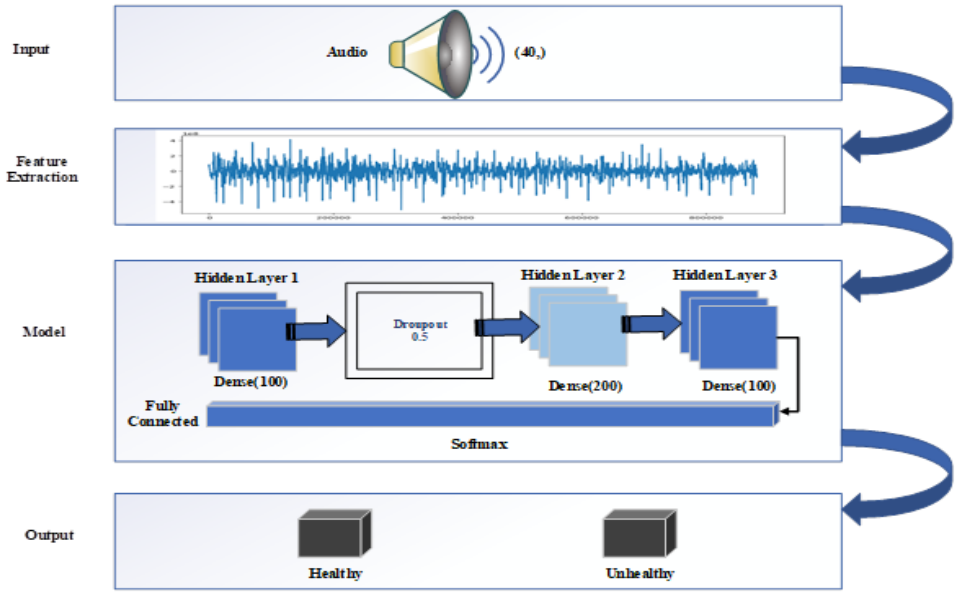

Figure 3.3 Difficulty in breath Detection using audio with RNN Algorithm

Input

The system takes an audio signal as input, which is represented by a speaker icon and a waveform at the top of the diagram.

Feature Extraction

This step involves converting the raw audio data into a more manageable and meaningful format, as depicted by the waveform turning into a structured format. Feature extraction from audio signals is a critical step in many signal processing and machine learning tasks. It involves distilling the raw audio data into a set of features that capture the essential characteristics of the signal, which can then be used for further analysis or classification. A brief explanation of some common types of features extracted from audio:

- Time-Domain Features: These are derived directly from the audio waveform in the time domain.
- Frequency-Domain Features: These are obtained by transforming the audio signal into the frequency domain using techniques like the Fourier Transform.
- Cepstral Features: These are features based on the cepstrum of the signal, which is the inverse Fourier transform of the logarithm of the spectrum. The most common cepstral feature is the Mel-Frequency Cepstral Coefficients (MFCCs), which represent the short-term power spectrum of a sound.
- Rhythm Features: These capture the rhythmic content of the audio and can include features like Tempo, Beat, and Rhythm Patterns.

These features are then typically fed into a machine learning model to perform tasks such as speech recognition, music genre classification, or health monitoring of machinery through sound analysis. Each feature

captures a different aspect of the audio signal, and together they provide a comprehensive representation for analysis.

Model

The extracted features are fed into a model, which is a neural network with three hidden layers and a softmax output layer. The model learns to recognize patterns in the features and assign probabilities to the two possible classes: “Healthy” or “Unhealthy”.

-The model consists of three hidden layers with dropout to prevent overfitting.
-Hidden Layer 1 has 100 dense units.
-Hidden Layer 2 has 200 dense units.
-Hidden Layer 3 also has 100 dense units.
-There is a dropout of 0.5 after the first hidden layer to reduce overfitting.

- Hidden Layer 1: The first layer of the neural network consists of 100 densely connected neurons. It receives the extracted features as input.
- Dropout 0.5: To prevent overfitting, a dropout layer is applied after the first hidden layer. Dropout randomly sets a fraction (in this case, 50%) of the input units to 0 at each update during training, which helps to prevent complex co-adaptations on training data.
- Hidden Layer 2: The second hidden layer has 200 densely connected neurons. It processes the data received from the first layer.
- Hidden Layer 3: The third hidden layer again consists of 100 densely connected neurons. It further refines the data processing.

Output

After processing through the fully connected layers and SoftMax activation function, the model classifies the audio as either “Healthy” or “Unhealthy”.

3.2 MODEL DESIGN

3.2.1 Detection of COVID-19 using CT-Scan images with CNN combined RNN algorithm

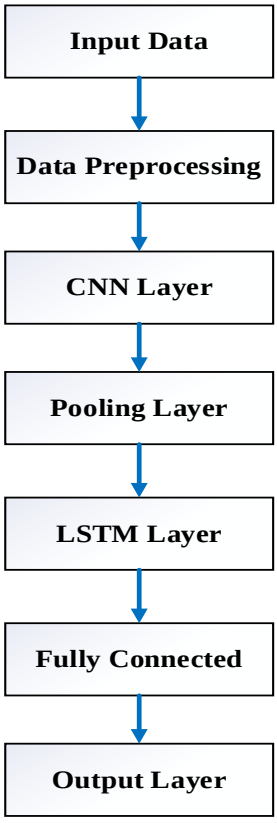

Figure 3.4 Flowchart for detection of COVID-19 using CT-Scan

Input Data

Figure 3.5 is one of the input images, which is a CT scan image. It has a dimension of 26x26x3, which means it has 26 pixels in height and width, and 3 channels for red, green, and blue colors.

Figure 3.5 Input Image of CT-Scan to detect COVID-19

Data Preprocessing

Data preprocessing is essential for neural network models. It involves cleaning data (removing noise, handling missing values, and outliers), normalizing data (scaling to a standard range), and augmenting data (generating extra samples with transformations like rotation or flipping). These steps ensure data quality, improve convergence during training, and enhance model robustness and generalization.

CNN Layer

Convolutional layers extract features, activation functions introduce non-linearity like ReLU, and dropout regularizes by randomly dropping connections to prevent overfitting in convolutional neural networks (CNNs). These components work together to enhance model performance, especially in tasks like image recognition and computer vision. The output feature map size is calculated using the formula:

$$\text{Output size}=(\text{Input size} - \text{Filter size} + 2 \times \text{Padding}/\text{Stride})+1$$

For this layer: $\text{Output size}=(126-3+2\times0)+1=24$ So the output feature map size is $24\times24\times32$ (32 feature maps). In this block diagram, the first convolutional layer has 32 filters of size 3x3, and produces a feature map of size 26x26x32.

Pooling Layer

This is a pooling layer, which reduces the size of the feature map by applying a max operation over a sliding window. This helps to reduce the computational cost and avoid overfitting. The size of the output depends on the size and stride of the window. The output size calculation is given by:

$$\text{Output size}= ((\text{Input size} - \text{Window size})/\text{Stride}) +1$$

For this layer: $\mathbf{Output\ size= ((24-2)/2) +1=12}$ So the output feature map size is $12\times12\times32$.

LSTM

This is a long short-term memory network, which is a type of recurrent neural network that can handle sequential data. It has a memory cell that can store and update information over time, and a set of gates that control the flow of information. It can learn long-term dependencies and capture temporal patterns. It has several components including input gate, forget gate, and output gate, and its output is calculated as:

$$ht=tanh\ (Ct)\cdot sigmoid(ot)$$

where ht is the output, Ct is the cell state, and ot is the output gate. In this block diagram, there are two LSTM networks in parallel, each with 64 hidden units. They receive the features from the max pooling layers and output a vector of size 64.

Fully connected

The Fully Connected or Dense layer serves as a high-capacity processing unit within the neural network. With 512 output units and employing the Rectified Linear Unit (ReLU) activation function, this layer performs a linear transformation followed by a non-linear activation. This enables the model to learn complex mappings between the flattened LSTM outputs and the desired output classes (Covid or Non-Covid), facilitating higher-level feature learning and abstraction. The output of the layer can be calculated as:

$$Output=ReLU(W\cdot Input+b)$$

where $\mathbf{W}$ represents the weights matrix, $\mathbf{Input}$ is the input vector, $\mathbf{b}$ is the bias vector, and $\mathbf{ReLU}$ is the activation function.

Output

The final output layer consists of two units, representing the classes "Covid" and "Non-Covid." Using The final layer uses the SoftMax activation function to

convert the output of the previous layer into probabilities. The SoftMax function is defined as:

$$P(\text{class}_i)=\frac{e^{zi}}{\sum_{j=1}^{n} e^{zj}}$$

where **zi** is the output of the previous layer for class **i**, and **n** is the total number of classes. This ensures that the probabilities sum up to one, making it suitable for multi-class classification tasks. The class with the highest probability is considered the predicted class, providing a probabilistic assessment of whether the input CT scan image indicates the presence of COVID-19 or not.

3.2.2 Fever Detection using Thermal Camera images with CNN combined RNN algorithm

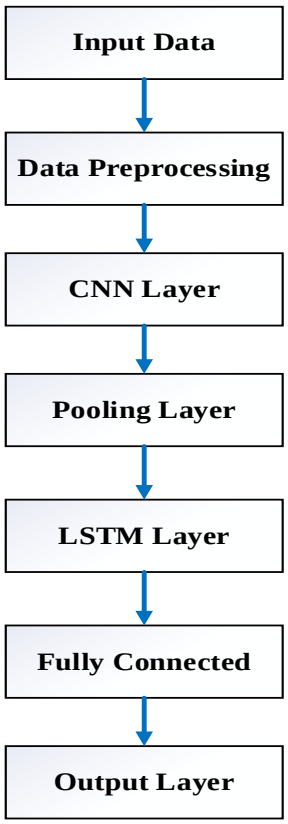

Figure 3.6 Flowchart for detection of Fever

Input Data

Figure 3.7 is one of the input images, which is a thermal or heat signature image. It has a dimension of 26x26x3, which means it has 26 pixels in height and width, and 3 channels for red, green, and blue colors.

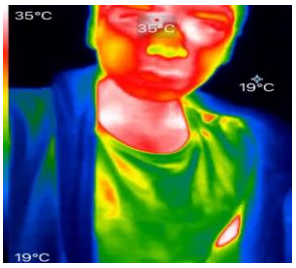

Figure 3.7 Thermal Camera Image input for Fever detection

CNN Layer

Convolutional layers extract features, activation functions introduce non-linearity like ReLU, and dropout regularizes by randomly dropping connections to prevent overfitting in convolutional neural networks (CNNs). These components work together to enhance model performance, especially in tasks like image recognition and computer vision. The output feature map size is calculated using the formula:

$$\text{Output size}=(\text{Input size} - \text{Filter size} + 2 \times \text{Padding}/\text{Stride})+1$$

For this layer: $\text{Output size}=(126-3+2\times0)+1=24$ So the output feature map size is $24\times24\times32$ (32 feature maps). In this block diagram, the first convolutional layer has 32 filters of size 3x3, and produces a feature map of size 26x26x32.

Pooling Layer

This is a pooling layer, which reduces the size of the feature map by applying a max operation over a sliding window. This helps to reduce the computational cost and avoid overfitting. The size of the output depends on the size and stride of the window. The output size calculation is given by:

$$\text{Output size} = ((\text{Input size} - \text{Window size})/\text{Stride}) + 1$$

For this layer: $\mathbf{Output\ size} = ((\mathbf{24} - \mathbf{2})/\mathbf{2}) + \mathbf{1} = \mathbf{12}$ So the output feature map size is $12 \times 12 \times 32$.

LSTM Layer

This is a long short-term memory network, which is a type of recurrent neural network that can handle sequential data. It has a memory cell that can store and update information over time, and a set of gates that control the flow of information. It can learn long-term dependencies and capture temporal patterns. It has several components including input gate, forget gate, and output gate, and its output is calculated as:

$$h_t = \tanh(C_t) \cdot \text{sigmoid}(o_t)$$

where h_t is the output, C_t is the cell state, and o_t is the output gate. In this block diagram, there are two LSTM networks in parallel, each with 64 hidden units. They receive the features from the max pooling layers and output a vector of size 64.

Fully connected

The Fully Connected or Dense layer serves as a high-capacity processing unit within the neural network. With 512 output units and employing the Rectified Linear Unit (ReLU) activation function, this layer performs a linear transformation followed by a non-linear activation. This enables the model to learn complex mappings between the flattened LSTM outputs and the desired output classes (Healthy or Unhealthy), facilitating higher-level feature learning and abstraction. The output of the layer can be calculated as:

$$\text{Output} = \text{ReLU}(W \cdot \text{Input} + b)$$

where **W** represents the weights matrix, **Input** is the input vector, **b** is the bias vector, and **ReLU** is the activation function.

Output

The final output layer consists of two units, representing the classes "Healthy" and "Unhealthy." Using The final layer uses the SoftMax activation function to convert the output of the previous layer into probabilities. The SoftMax function is defined as:

$$P(\text{class}_i)=\frac{e^{zi}}{\sum_{j=1}^{n} e^{zj}}$$

where **zi** is the output of the previous layer for class **i**, and **n** is the total number of classes. This ensures that the probabilities sum up to one, making it suitable for multi-class classification tasks. The class with the highest probability is considered the predicted class, providing a probabilistic assessment of whether the input Thermal image indicates the presence of Fever or not.

3.2.3 Difficulty in breath Detection using audio with RNN algorithm

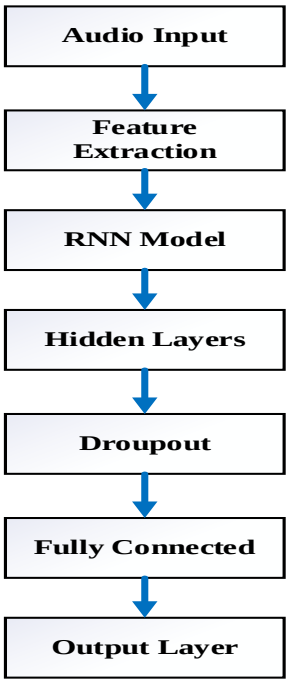

Figure 3.8 Flowchart for detection of difficulty in breath

Input

The system takes an audio signal as input, which is represented by a speaker icon and a waveform at the top of the diagram. Figure 3.9 is one of the input audios

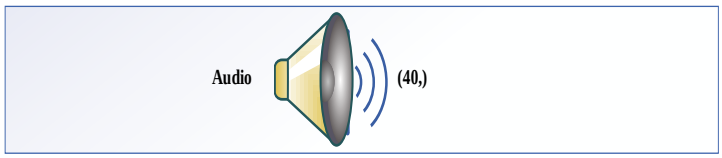

Figure 3.9: Audio input to detect difficulty in breath

Feature Extraction

This step involves converting the raw audio data into a more manageable and meaningful format, as depicted by the waveform turning into a structured format. Feature extraction from audio signals is a critical step in many signal processing and machine learning tasks. It involves distilling the raw audio data into a set of features that capture the essential characteristics of the signal, which can then be used for further analysis or classification. Figure 3.10 shows the wave diagram extracted from input audios. A brief explanation of some common types of features extracted from audio:

- Time-Domain Features: These are derived directly from the audio waveform in the time domain.
- Frequency-Domain Features: These are obtained by transforming the audio signal into the frequency domain using techniques like the Fourier Transform.
- Cepstral Features: These are features based on the cepstrum of the signal, which is the inverse Fourier transform of the logarithm of the spectrum. The most common cepstral feature is the Mel-Frequency Cepstral Coefficients (MFCCs), which represent the short-term power

spectrum of a sound.

- Rhythm Features: These capture the rhythmic content of the audio and can include features like Tempo, Beat, and Rhythm Patterns.

These features are then typically fed into a machine learning model to perform tasks such as speech recognition, music genre classification, or health monitoring of machinery through sound analysis. Each feature captures a different aspect of the audio signal, and together they provide a comprehensive representation for analysi

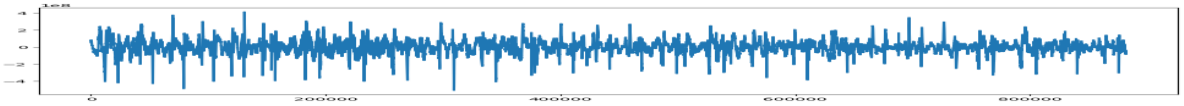

Figure 3.10 Feature Extraction wave diagram

Model

The extracted features are fed into a model, which is a neural network with three hidden layers and a softmax output layer. The model learns to recognize patterns in the features and assign probabilities to the two possible classes: “Healthy” or “Unhealthy”.

Hidden Layers and Dropout

- The model consists of three hidden layers with dropout to prevent overfitting.
-Hidden Layer 1 has 100 dense units.
-Hidden Layer 2 has 200 dense units.
-Hidden Layer 3 also has 100 dense units.
-There is a dropout of 0.5 after the first hidden layer to reduce overfitting.

- Hidden Layer 1: The first layer of the neural network consists of 100 densely connected neurons. It receives the extracted features as input.
- Dropout 0.5: To prevent overfitting, a dropout layer is applied after the

first hidden layer. Dropout randomly sets a fraction (in this case, 50%) of the input units to 0 at each update during training, which helps to prevent complex co-adaptations on training data.

- Hidden Layer 2: The second hidden layer has 200 densely connected neurons. It processes the data received from the first layer.
- Hidden Layer 3: The third hidden layer again consists of 100 densely connected neurons. It further refines the data processing.

Output

After processing through the fully connected layers and softmax activation function, the model classifies the audio as either “Healthy” or “Unhealthy”.

3.3 DATASET DESCRIPTION

Source

The datasets utilized in this project were obtained from publicly accessible repositories hosted on Kaggle and GitHub, two prominent platforms for sharing and discovering data-related resources. Kaggle, known for its diverse collection of datasets and competitions, provided datasets relevant to the project's scope, while GitHub, a platform widely used for collaborative development and open-source projects, offered additional datasets that complemented the analysis. Leveraging these repositories ensured access to high-quality and varied datasets, contributing to the robustness and depth of the project's data analysis and outcomes.

Description

The dataset encompasses a comprehensive collection of medical data crucial for COVID-19 detection and diagnosis, including CT (computed tomography) images, thermal camera images, and audio recordings. These diverse data modalities offer a multi-dimensional view of patients' conditions, enabling

researchers and healthcare professionals to develop advanced algorithms and systems for accurate and efficient detection of COVID-19 cases. The integration of CT images provides detailed insights into lung abnormalities associated with the virus, while thermal camera images can reveal temperature variations indicative of infection. Additionally, audio recordings may contain cough or breathing patterns that can serve as valuable indicators for diagnosis. This dataset plays a pivotal role in advancing the field of medical imaging and artificial intelligence for combating the COVID-19 pandemic effectively.

Data Pre-Processing

Prior to model training, the CT and thermal images were resized to a uniform dimension of 256x256 pixels and normalized to a pixel intensity range of [0, 1]. Audio recordings were converted to spectrograms for feature extraction.

Data Split

The datasets were divided into training (80%) and validation (20%).

Data Augmentation

Data augmentation techniques such as rotation, flipping, and zooming were applied to the image datasets to increase sample diversity.

Size and Format

CT Image Dataset

The dataset comprises 3227 CT images in PNG format (256x256 pixels), 80% of which (2581 images) are allocated for training purposes, while the remaining 20% (646 images) are designated for validation to ensure the effectiveness and accuracy of model training and evaluation.

Thermal Camera Image Dataset

The thermal camera image dataset, containing 2000 images in JPG format, follows the same split, with 80% (1600 images) for training and 20% (400 images) for validation.

Audio Dataset

The audio dataset, comprising 1500 recordings in WAV format, also adheres to the 80-20 rule, with 80% (1200 recordings) allocated for training and 20% (300 recordings) for validation. This systematic data split strategy ensures balanced representation across training and validation sets, facilitating accurate and effective model training and assessment across all data modalities.

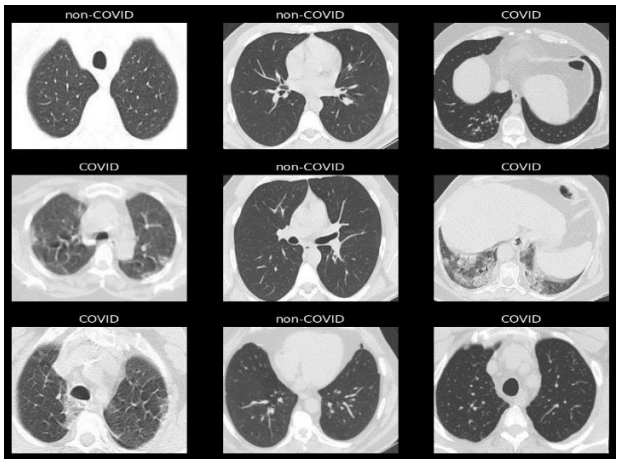

Figure 3.11 Some of the images of CT Image Dataset

Figure 3.11 showcases a curated selection from a CT Image Dataset, contrasting the pulmonary manifestations in patients diagnosed with COVID-19 against those without the infection. The ‘COVID’ labeled scans reveal the typical radiographic features associated with the virus, such as ground-glass opacities and consolidation, while the ‘non-COVID’ scans depict normal, healthy lung parenchyma. This visual compilation aids in the comparative study of radiological findings, serving as a valuable asset for medical research and diagnostic algorithm development.

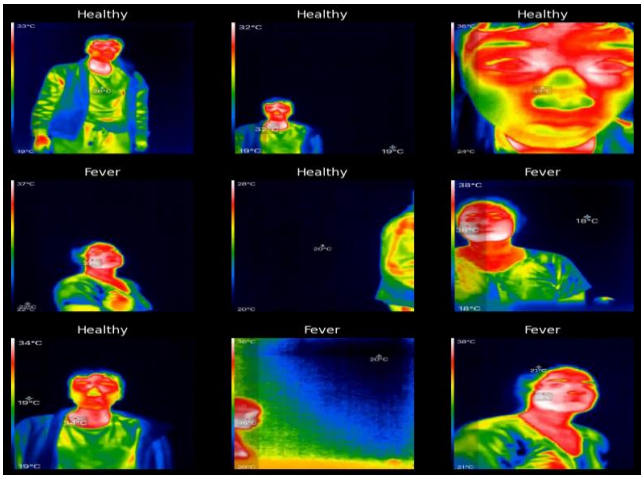

Figure 3.12 Some of the images of Thermal Camera Image Dataset

Figure 3.12 displays a selection from the Thermal Camera Image Dataset, capturing the variance in body temperatures of individuals classified as 'Healthy' or exhibiting 'Fever'. These thermal images provide a non-invasive method to detect elevated body temperatures, a common symptom of various infections, including COVID-19. The dataset is instrumental for developing thermal screening tools that offer rapid preliminary assessments in public health monitoring."

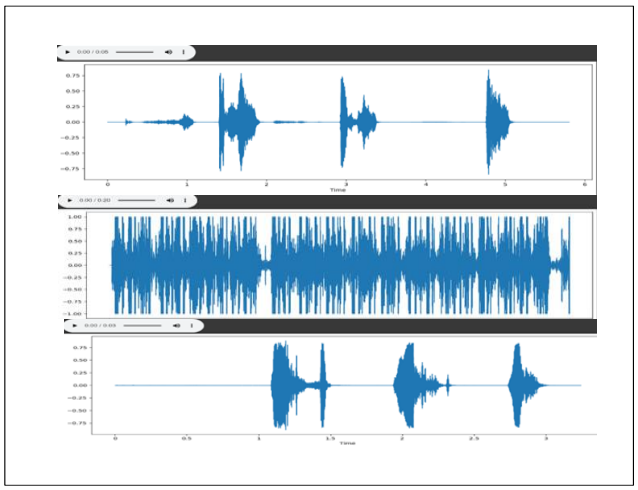

Figure 3.13 Some of the audio images of Audio Dataset

Figure 3.13 illustrates a series of audio waveforms from the Audio Dataset, providing a visual representation of sound recordings. Each waveform captures the unique temporal dynamics of audio signals, reflecting variations in amplitude and frequency that correspond to different acoustic properties. This dataset is essential for auditory research and the development of sound analysis algorithms.

CHAPTER 4

PERFORMANCE ANALYSIS

For the project involving COVID-19 detection using a CNN combined with an RNN algorithm with CT images, fever detection using a CNN combined with an RNN algorithm with thermal camera images, and difficulty in breath detection using an RNN algorithm with audio.

Accuracy

Measure the overall correctness of predictions across all classes (COVID-19 positive/negative, fever present/absent, difficulty in breath detected/not detected). This metric assesses the effectiveness of the models in making correct classifications.

Accuracy = ((Number of Correct Predictions)/ (Total Number of Predictions)) × 100%

To find Number of Correct predictions: Number of Correct Prediction = Accuracy × Total Number of Predictions

Precision

Specifically for COVID-19 detection, precision would indicate the ratio of true positive COVID-19 cases to all cases predicted as positive. It ensures that identified COVID-19 cases are genuine and minimizes false positives.

$$Precision=TP/(TP+FP)$$

Recall (Sensitivity)

Measure the ability of the models to detect positive instances correctly. In the context of COVID-19 detection, recall would show how well the model identifies actual COVID-19 cases among all true COVID-19 cases.

$$Recall=TP/(TP+FN)$$

where:

TP = True positive

TN = True negative

FN = False negative

FP = False positive

F1 Score

The F1 score balances precision and recall, providing a single measure that considers both false positives and false negatives. It's especially useful when there's an imbalance between classes.

$$F1\ score = 2\times Precision+ Recall/Precision \times Recall$$

These performance measures collectively provide insights into the models' capabilities in detecting COVID-19, fever, and difficulty in breath using different types of data inputs and algorithms. Evaluation using these metrics helps in assessing the models' accuracy, sensitivity, specificity, and overall effectiveness in the intended detection tasks.

Reference	Model	Accuracy (%)
Proposed	CNN-RNN	97.45
Anwar T et al. [2]	EfficientNetB4	89.70
Fakhfakh et al. [8]	ProgNet	92.40
XinZhang et al. [32]	CNN	92.60
Khuzani et al. [36]	CNN	94.05

Table 4.1 Model Performance on Detection of COVID-19

Training and Validation

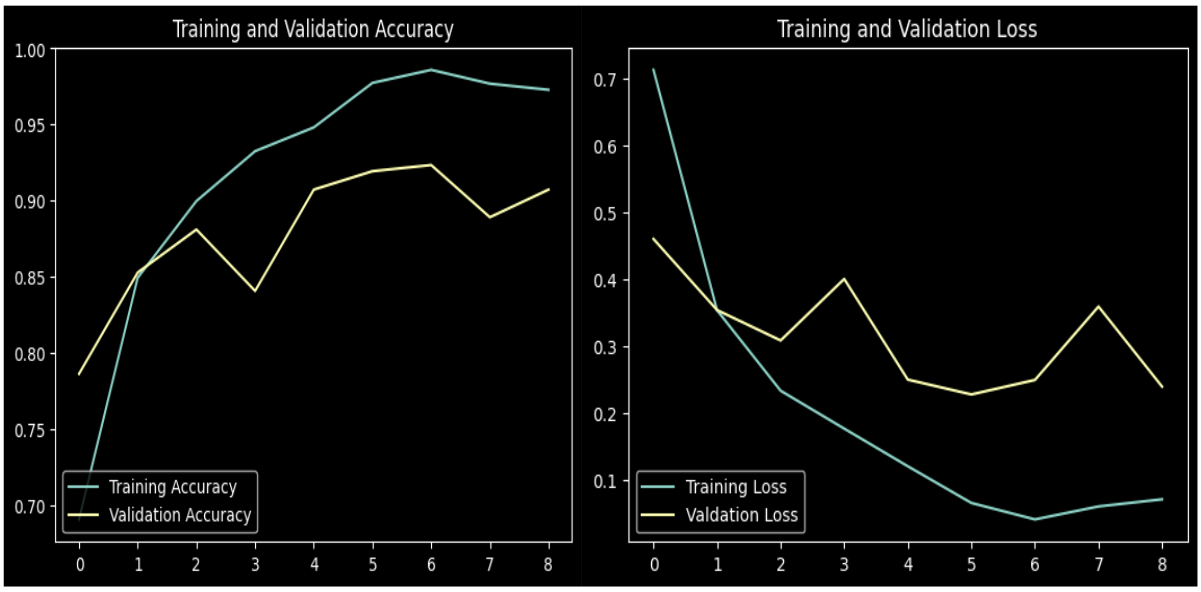

Figure 4.1 Training and Validation (Loss & accuracy) of Detection of Covid-19 using CT images

Figure 4.1 illustrates the training and validation metrics for a deep learning model designed to detect COVID-19 from CT images. The graphs depict a clear trend of decreasing loss and increasing accuracy over successive epochs, indicating robust model learning and generalization capabilities. The training loss and accuracy reflect the model's performance on the dataset it was trained on, while the validation loss and accuracy provide insight into the model's predictive power on unseen data. These trends are essential for evaluating the model's potential for reliable clinical application in diagnosing COVID-19.

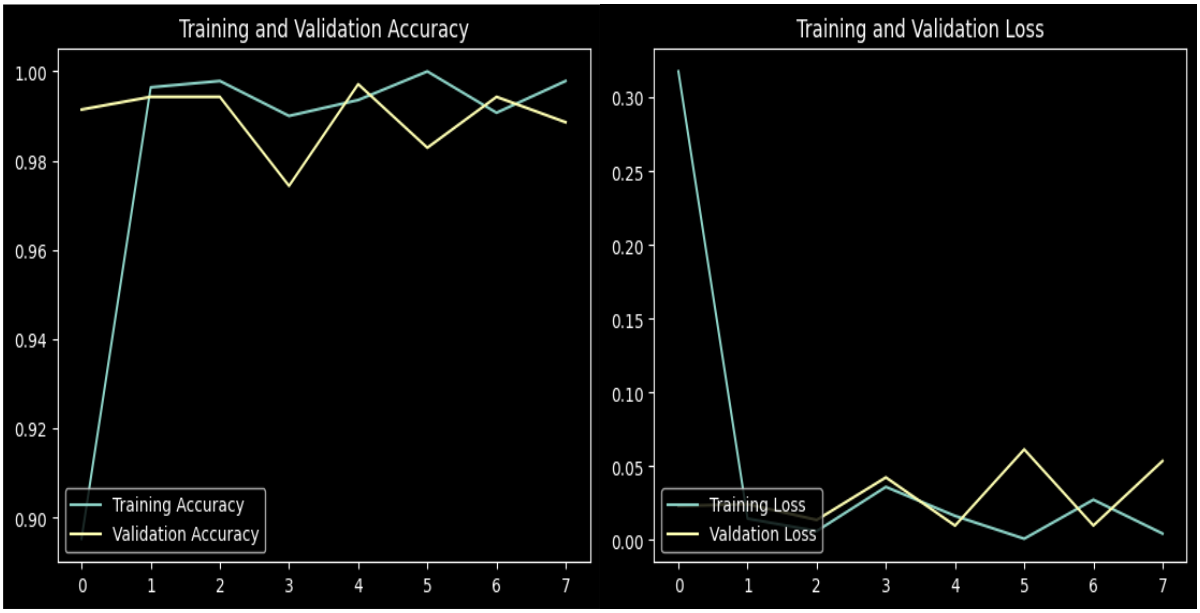

Figure 4.2 Training and Validation (Loss & accuracy) of Detection of Fever using Thermal images

Figure 4.2 presents the training and validation curves for a machine learning model tasked with detecting fever from thermal images. The loss and accuracy graphs demonstrate the model's learning trajectory, with the training loss decreasing and the training accuracy increasing over time, indicating successful feature learning. The validation loss and accuracy provide insights into the model's generalization ability, crucial for its application in real-world fever screening scenarios. This model represents a significant step towards automated, non-contact fever detection, which is vital for public health monitoring, especially in the context of the COVID-19 pandemic.

a) Detection of COVID-19 using CT-Scan images with CNN combined RNN algorithm

- **Covid Positive**

Figure 4.3 CT-Scan image input of COVID-19 Positive patient

```
1/1 [==============================] - 0s 83ms/step
This image is most likely COVID .
<matplotlib.image.AxesImage at 0x7d414bce4d30>
```

Figure 4.4 Output of Covid positive person input image

Figure 4.3 shows CT-Scan image input of a COVID-19 positive patient, displaying characteristic lung involvement with ground-glass opacities indicative of viral infection. Figure 4.4 Processed output image from the CT-Scan data, highlighting regions affected by COVID-19 with increased radiodensity in the lung parenchyma.

- **Covid Negative:**

Figure 4.5 CT-Scan image input of COVID-19 Negative patient

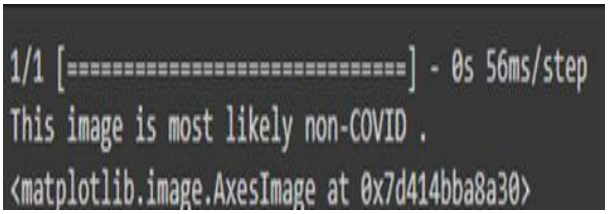

Figure 4.6 Output of Covid negative person input image

Figure 4.5 CT-Scan image input of a COVID-19 negative patient, showing clear lung fields without the radiographic signs typically associated with COVID-19 infection. Figure 4.6 Output image after analysis of the COVID-19 negative patient's CT-Scan, confirming the absence of infection-related abnormalities in the lung parenchyma.

b) Fever Detection using Thermal Camera images with CNN combined RNN algorithm

- **HEALTHY (No Fever)**

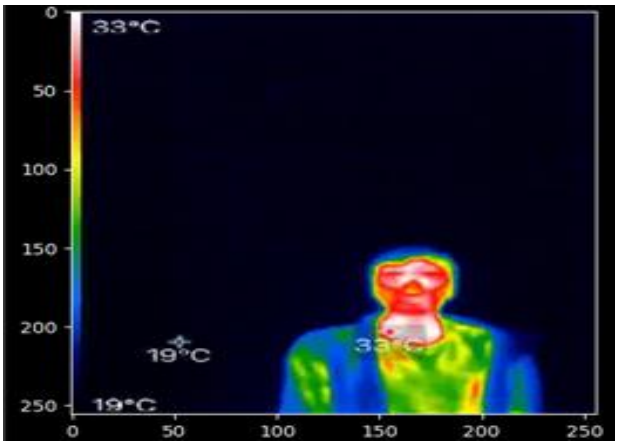

Figure 4.7 Thermal image input of Healthy person

```
1/1 [==============================] - 0s 157ms/step
This image is most likely Healthy .
<matplotlib.image.AxesImage at 0x7fcbaaaf8940>
```

Figure 4.8 Output of healthy person input image

Figure 4.7 This thermal image captures the facial temperature distribution of a healthy individual. The colour gradient remains consistent and within normal ranges, indicating no significant hotspots that would suggest a fever or underlying health issue. Figure 4.8 The output image, derived from the thermal input, uses advanced algorithms to analyze and confirm the healthy status of the individual. It shows a uniform temperature profile with no anomalies, supporting the initial assessment of a healthy, non-febrile condition.

- **UNHEALTHY (Fever)**

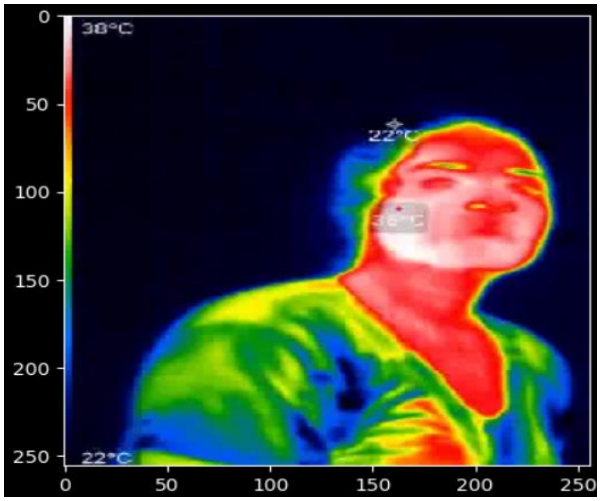

Figure 4.9 Thermal image input of Healthy person

```
1/1 [==============================] - 0s 54ms/step
This image is most likely Fever .
<matplotlib.image.AxesImage at 0x7cac67211180>
```

Figure 4.10 Output of unhealthy person input image

Figure 4.9 The thermal input image of an unwell person presents a distinct pattern of elevated temperatures, particularly around the forehead and sinus areas, which are common fever indicators. The colour intensities suggest a higher-than-normal body temperature. Figure 4.10 In the output image, the analysis software has flagged the areas with abnormal temperature elevations, typically associated with a feverish state. The highlighted zones provide a clear visual cue for medical professionals to identify potential health concerns.

c) Difficulty in breath Detection using audio with RNN algorithm

➢ HEALTHY (Normal in Breathing)

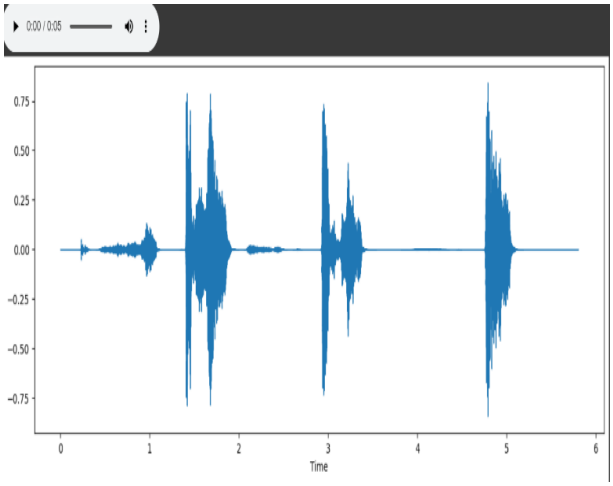

Figure 4.11 Audio input of Normal person (Covid Negative)

```
[57] print(prediction_class)

     ['healthy']
```

Figure 4.12 Output of healthy person audio input

Figure 4.11 This figure represents the audio waveform input of a normal, healthy individual. The consistent wave patterns indicate regular breathing rhythms and the absence of respiratory distress typically associated with COVID-19. Figure 4.12 The output image showcases the results of the audio analysis, confirming the healthy status of the individual. The analysis software has not detected any anomalies or irregularities in the breathing sounds that would suggest a respiratory condition.

➢ **UNHEALTHY (Abnormal in Breathing)**

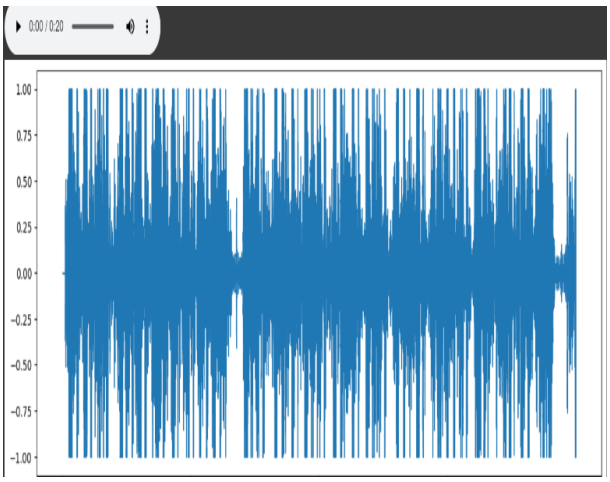

Figure 4.13 Audio input of Abnormal person (Covid Positive)

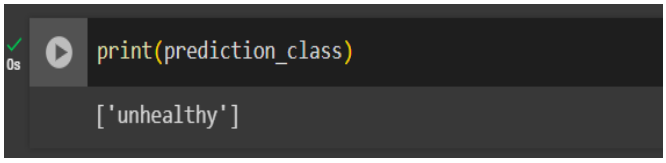

Figure 4.14 Output of healthy person audio input

Figure 4.13 This figure displays the audio waveform of a person with abnormal breathing patterns. The erratic and disrupted waveforms suggest respiratory difficulties, which are common symptoms in COVID-19 positive patients. Figure 4.14 The output image depicts the analytical results of the audio input, highlighting the irregularities in the breathing pattern. The software has identified these deviations from normal as potential indicators of a respiratory condition, such as COVID-19.

CHAPTER 5

CONCLUSION AND FUTURE ENHANCEMENT

5.1 CONCLUSION

In conclusion, the proposed integrated system for COVID-19 detection, utilizing advanced imaging technologies and machine learning algorithms, holds significant promise in enhancing diagnostic capabilities. The combined use of Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) algorithms enables a comprehensive and accurate approach to identifying COVID-19 symptoms from CT scan images, thermal camera images, and audio signals indicative of respiratory distress. The system's ability to provide early and non-invasive detection, along with its potential to reduce the workload on healthcare professionals, positions it as a valuable tool in the ongoing global fight against the pandemic. However, it is crucial to emphasize that the system's outputs should be considered in consultation with trained medical clinicians, underscoring the collaborative role of artificial intelligence in healthcare decision-making.

5.2 FUTURE ENHANCEMENT

Future work on this integrated system could focus on several avenues to enhance its applicability and impact:

Clinical Validation: Conducting rigorous clinical trials to validate the system's performance in real-world healthcare settings is essential. This includes assessing its effectiveness across diverse populations and healthcare environments.

Continuous Model Improvement: Iterative refinement of the machine learning models, incorporating additional data and adapting to emerging COVID-19 variants, can enhance the system's accuracy and robustness over time.

Remote Patient Monitoring: Exploring the potential for remote patient monitoring by integrating the system into telehealth platforms could extend its reach and utility, allowing for proactive healthcare management.

Integration with Electronic Health Records (EHR): Integrating the system with electronic health record systems can streamline information exchange between the diagnostic tool and healthcare providers, facilitating more seamless patient care.

Ethical and Privacy Considerations: Further research is needed to address ethical concerns and privacy considerations associated with the use of patient data in AI-driven healthcare applications, ensuring compliance with regulatory standards.

Global Collaboration: Collaborative efforts with healthcare institutions, research organizations, and technology companies globally can contribute to the continuous improvement and widespread adoption of the proposed system.

Incorporating these future directions will not only strengthen the proposed system's capabilities but also contribute to the broader advancements in AI-assisted healthcare, especially in the context of infectious disease detection and management.

ANNEXURE

SAMPLE CODE

a) Detection of COVID-19 using CT-Scan images with CNN combined RNN algorithm

```
import NumPy as np
import matplotlib.pyplot as plt
import keras
from keras.models import Sequential, Model
from keras.layers import Conv2D, MaxPool2D, Dropout, Flatten, Dense,
LSTM, Input, concatenate, Reshape

from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing import image
from keras.optimizers import Adam
from keras.layers import BatchNormalization

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
```

```
train_ds = tf.keras.utils.image_dataset_from_directory(
 "/content/CTScan/train",
  validation_split=0.2,
  subset="training",
  seed=123,
  image_size=(256,256),
  batch_size=16)

val_ds = tf.keras.utils.image_dataset_from_directory(
  "/content/CTScan/train",
  validation_split=0.2,
  subset="validation",
  seed=123,
  image_size=(256, 256),
  batch_size=16)

class_names = train_ds.class_names
print(class_names)

import matplotlib.pyplot as plt
plt.style.use('dark_background')

plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
 for i in range(9):
 ax = plt.subplot(3, 3, i + 1)
 plt.imshow(images[i].numpy().astype("uint8"))
 plt.title(class_names[labels[i]])
 plt.axis("off")
```

```
AUTOTUNE = tf.data.AUTOTUNE
train_ds =
train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)

normalization_layer = layers.Rescaling(1./255)

normalized_ds = train_ds.map(lambda x, y: (normalization_layer(x), y))
image_batch, labels_batch = next(iter(normalized_ds))
first_image = image_batch[0]
print(np.min(first_image), np.max(first_image))

cnn_input = Input(shape=(256, 256, 3))

# CNN model
model = Sequential([
    layers.Rescaling(1./255, input_shape=(256, 256, 3)),
    layers.Conv2D(16, (3,3), activation='relu'),
    layers.MaxPooling2D(2,2),
    layers.Conv2D(32, (3,3), activation='relu'),
    layers.MaxPooling2D(2,2),
   layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    layers.Flatten(),
    layers.Dense(512, activation='relu'),
    layers.Dense(1, activation='sigmoid')
])
cnn_output = model(cnn_input)
```

 # Reshape for LSTM input

reshaped_input = Reshape((256, 256 * 3))(cnn_input)

 # RNN model

rnn_model = Sequential()

rnn_model.add(LSTM(256, return_sequences=True, input_shape=(256, 256 * 3)))

rnn_model.add(LSTM(256))

rnn_output = rnn_model(reshaped_input)

 # Combine the outputs

combined = concatenate([cnn_output, rnn_output])

combined = Dense(128, activation='relu')(combined)

combined = Dropout(rate=0.5)(combined)

combined_output = Dense(1, activation='sigmoid')(combined)

 # Create the combined model

combined_model = Model(inputs=cnn_input, outputs=combined_output)

 # Hyperparameter tuning

adam_optimizer = Adam(lr=0.001) # Adjust learning rate if needed

combined_model.compile(loss='binary_crossentropy', optimizer=adam_optimizer, metrics=['accuracy'])

 # Print the summary of the combined model

```
combined_model.summary()
callback = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
patience=3)

model.compile(optimizer='adam',

              loss='binary_crossentropy',
              metrics=['accuracy'])


history = model.fit(
  train_ds,
  validation_data = val_ds,
  callbacks=[callback],
  epochs = 10
)

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, label='Training Accuracy')
plt.plot(epochs, val_acc, label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.legend(loc='lower left')
plt.figure()
```

```
plt.plot(epochs, loss, label='Training Loss')
plt.plot(epochs, val_loss, label='Valdation Loss')
plt.legend(loc='lower left')
plt.title('Training and Validation Loss')

plt.show()

img_path = "/content/CTScan/train/non-COVID/Non-Covid (640).png"

img = tf.keras.utils.load_img(
    img_path, target_size=(256, 256)
)

img_array = tf.keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0)

predictions = model.predict(img_array)

score = predictions[0]

print(
    "This image is most likely {} with {:.2f} percent confidence."
    .format(class_names[round(score[0])], 100 * np.max(score))
)

plt.imshow(img)
```

b) Fever Detection using Thermal Camera images with CNN combined RNN algorithm

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten,
Dense, LSTM
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
from sklearn.model_selection import train_test_split

import os
from PIL import Image

import numpy as np
import matplotlib.pyplot as plt
import keras
from keras.models import Sequential, Model
from keras.layers import Conv2D, MaxPool2D, Dropout, Flatten, Dense,
LSTM, Input, concatenate, Reshape
from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing import image
from keras.optimizers import Adam
from keras.layers import BatchNormalization

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
```

```
train_ds = tf.keras.utils.image_dataset_from_directory(
  "/content/drive/MyDrive/1. Project Details/Dataset/Thermal/train",
  validation_split=0.2,
  subset="training",
  seed=123,
  image_size=(256,256),
  batch_size=16)

val_ds = tf.keras.utils.image_dataset_from_directory(
  "/content/drive/MyDrive/1. Project Details/Dataset/Thermal/train",
  validation_split=0.2,
  subset="validation",
  seed=123,
  image_size=(256, 256),
  batch_size=16)

class_names = train_ds.class_names
print(class_names)

import matplotlib.pyplot as plt
plt.style.use('dark_background')

plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
  for i in range(9):
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(images[i].numpy().astype("uint8"))
    plt.title(class_names[labels[i]])
    plt.axis("off")
```

```
AUTOTUNE = tf.data.AUTOTUNE
train_ds =
train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)

normalization_layer = layers.Rescaling(1./255)

normalized_ds = train_ds.map(lambda x, y: (normalization_layer(x), y))
image_batch, labels_batch = next(iter(normalized_ds))
first_image = image_batch[0]
print(np.min(first_image), np.max(first_image))

cnn_input = Input(shape=(256, 256, 3))

# CNN model
model = Sequential([
    layers.Rescaling(1./255, input_shape=(256, 256, 3)),
    layers.Conv2D(16, (3,3), activation='relu'),
    layers.MaxPooling2D(2,2),
    layers.Conv2D(32, (3,3), activation='relu'),
    layers.MaxPooling2D(2,2),
   layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    layers.Flatten(),
    layers.Dense(512, activation='relu'),
    layers.Dense(1, activation='sigmoid')
])
cnn_output = model(cnn_input)
```

 # Reshape for LSTM input

reshaped_input = Reshape((256, 256 * 3))(cnn_input)

 # RNN model

rnn_model = Sequential()

rnn_model.add(LSTM(256, return_sequences=True, input_shape=(256, 256 * 3)))

rnn_model.add(LSTM(256))

rnn_output = rnn_model(reshaped_input)

 # Combine the outputs

combined = concatenate([cnn_output, rnn_output])

combined = Dense(128, activation='relu')(combined)

combined = Dropout(rate=0.5)(combined)

combined_output = Dense(1, activation='sigmoid')(combined)

 # Create the combined model

combined_model = Model(inputs=cnn_input, outputs=combined_output)

 # Hyperparameter tuning

adam_optimizer = Adam(lr=0.001) # Adjust learning rate if needed

combined_model.compile(loss='binary_crossentropy', optimizer=adam_optimizer, metrics=['accuracy'])

 # Print the summary of the combined model

combined_model.summary()

```
callback = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
patience=3)

model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

history = model.fit(
  train_ds,
  validation_data = val_ds,
  callbacks=[callback],
  epochs = 10
)

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, label='Training Accuracy')
plt.plot(epochs, val_acc, label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.legend(loc='lower left')
plt.figure()

plt.plot(epochs, loss, label='Training Loss')
plt.plot(epochs, val_loss, label='Valdation Loss')
```

```
plt.legend(loc='lower left')
plt.title('Training and Validation Loss')

plt.show()

img_path = "/content/drive/MyDrive/1. Project
Details/Dataset/Thermal/train/Fever/b (133).jpg"

img = tf.keras.utils.load_img(
    img_path, target_size=(256, 256)
)

img_array = tf.keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0)

predictions = model.predict(img_array)

score = predictions[0]

print(
    "This image is most likely {} with {:.2f} percent confidence."
    .format(class_names[round(score[0])], 100 * np.max(score))
)

plt.imshow(img)
```

c) Difficulty in breath Detection using audio with RNN algorithm

```
pip install --upgrade numpy librosa

from google.colab import drive
drive.mount('/content/drive')

import matplotlib.pyplot as plt

filename='/content/drive/MyDrive/1. Project
Details/Dataset/breathing/healthy/1005_Negative_male_59.wav'

import IPython.display as ipd
import librosa
import librosa.display

import librosa.display

plt.figure(figsize=(14,5))
data, sample_rate = librosa.load(filename)
librosa.display.waveshow(data, sr=sample_rate)
ipd.Audio(filename)

filename='/content/drive/MyDrive/1. Project
Details/Dataset/breathing/unhealthy/107_2b4_Ar_mc_AKGC417L.wav'
plt.figure(figsize=(14,5))
data,sample_rate=librosa.load(filename)
librosa.display.waveshow(data,sr=sample_rate)
ipd.Audio(filename)
```

```
from scipy.io import wavfile as wav
wave_sample_rate, wave_audio=wav.read(filename)

import pandas as pd

metadata=pd.read_csv('/content/drive/MyDrive/1. Project
Details/Dataset/breathing/healthy_unhealthy.csv')
metadata.head(10)

metadata['Class'].value_counts()

import librosa
audio_file_path='/content/drive/MyDrive/1. Project
Details/Dataset/breathing/healthy/183_1b1_Tc_sc_Meditron.wav'
librosa_audio_data,librosa_sample_rate=librosa.load(audio_file_path)

print(librosa_audio_data)

import matplotlib.pyplot as plt
# Original audio with 1 channel
plt.figure(figsize=(12, 4))
plt.plot(librosa_audio_data)

from scipy.io import wavfile as wav
wave_sample_rate, wave_audio = wav.read(audio_file_path)

wave_audio

import matplotlib.pyplot as plt
```

```
# Original audio with 2 channels
plt.figure(figsize=(12, 4))
plt.plot(wave_audio)

import librosa
import numpy as np
mfccs = librosa.feature.mfcc(y=librosa_audio_data,
sr=librosa_sample_rate, n_mfcc=40)
print(mfccs.shape)

mfccs

import pandas as pd
import os
import librosa

audio_dataset_path='/content/drive/MyDrive/1. Project
Details/Dataset/breathing/healthy_unhealthy'
metadata=pd.read_csv('/content/drive/MyDrive/1. Project
Details/Dataset/breathing/healthy_unhealthy.csv')
metadata.head()

def features_extractor(file):
    audio, sample_rate = librosa.load(file_name, res_type='kaiser_fast')
    mfccs_features = librosa.feature.mfcc(y=audio, sr=sample_rate,
n_mfcc=40)
    mfccs_scaled_features = np.mean(mfccs_features.T,axis=0)
```

```
    return mfccs_scaled_features

import os

file_path = '/content/drive/MyDrive/1. Project
Details/Dataset/breathing/healthy_unhealthy/0_Negative_male_26.wav'

if os.path.exists(file_path):
    print("File exists.")
else:
    print("File does not exist.")

!pip install resampy

import librosa
import resampy

file_path = '/content/drive/MyDrive/1. Project
Details/Dataset/breathing/healthy_unhealthy/0_Negative_male_26.wav'

try:
    audio, sample_rate = librosa.load(file_path, res_type='kaiser_fast')
    print("File loaded successfully.")
except Exception as e:
    print(f"Error loading file: {e}")

import numpy as np
from tqdm import tqdm
### Now we iterate through every audio file and extract features
```

```
### using Mel-Frequency Cepstral Coefficients
extracted_features=[]
for index_num,row in tqdm(metadata.iterrows()):
    file_name =
os.path.join(os.path.abspath(audio_dataset_path),str(row["Name"]))
    final_class_labels=row["Class"]
    data=features_extractor(file_name)
    extracted_features.append([data,final_class_labels])

extracted_features_df=pd.DataFrame(extracted_features,columns=['featur
e','Class'])
extracted_features_df.head()

X=np.array(extracted_features_df['feature'].tolist())
y=np.array(extracted_features_df['Class'].tolist())

X.shape

y=np.array(pd.get_dummies(y))
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelEncoder
labelencoder=LabelEncoder()
y=to_categorical(labelencoder.fit_transform(y))

y

y.shape

from sklearn.model_selection import train_test_split
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=0)

X_train

y

X_train.shape

X_test.shape

y_train.shape

y_test.shape

import tensorflow as tf
print(tf.__version__)

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Dropout,Activation,Flatten
from tensorflow.keras.optimizers import Adam
from sklearn import metrics

import numpy as np
import matplotlib.pyplot as plt
import keras
from keras.models import Sequential, Model
from keras.layers import Conv2D, MaxPool2D, Dropout, Flatten, Dense, LSTM, Input, concatenate, Reshape
```

```
from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing import image
from keras.optimizers import Adam
from keras.layers import BatchNormalization
import tensorflow as tf
from tensorflow.keras.layers import Dense
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential

num_labels=y.shape[1]

model=Sequential()
###first layer
model.add(Dense(100,input_shape=(40,)))
model.add(Activation('relu'))
model.add(Dropout(0.5))
###second layer
model.add(Dense(200))
model.add(Activation('relu'))
model.add(Dropout(0.5))
###third layer
model.add(Dense(100))
model.add(Activation('relu'))
model.add(Dropout(0.5))
```

```
###final layer
model.add(Dense(num_labels))
model.add(Activation('softmax'))

model.summary()

adam_optimizer = Adam(lr=0.001)  # Adjust learning rate if needed

model.compile(loss='categorical_crossentropy',metrics=['accuracy'],opti
mizer='adam')

from tensorflow.keras.callbacks import ModelCheckpoint
from datetime import datetime

num_epochs = 100
num_batch_size = 32

checkpointer =
ModelCheckpoint(filepath='saved_models/audio_classification.hdf5',
                        verbose=1, save_best_only=True)
start = datetime.now()

history=model.fit(X_train, y_train, batch_size=num_batch_size,
epochs=num_epochs, validation_data=(X_test, y_test),
callbacks=[checkpointer], verbose=1)

duration = datetime.now() - start
print("Training completed in time: ", duration)
```

```
test_accuracy=model.evaluate(X_test,y_test,verbose=0)
print(test_accuracy[1])

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, label='Training Accuracy')
plt.plot(epochs, val_acc, label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.legend(loc='lower left')
plt.figure()

plt.plot(epochs, loss, label='Training Loss')
plt.plot(epochs, val_loss, label='Valdation Loss')
plt.legend(loc='lower left')
plt.title('Training and Validation Loss')

plt.show()

import librosa
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model
from sklearn.preprocessing import LabelEncoder
```

```
tf.keras.models.load_model

metadata['Class'].unique()

model.input_shape

filename="/content/drive/MyDrive/1. Project
Details/Dataset/breathing/healthy_unhealthy/1082_Positive_male26.wav"
prediction_feature=features_extractor(filename)
prediction_feature=prediction_feature.reshape(1,-1)
model.predict(prediction_feature)
prediction_feature.shape

filename="/content/drive/MyDrive/1. Project
Details/Dataset/breathing/unhealthy/107_2b4_Ar_mc_AKGC417L.wav"
audio, sample_rate = librosa.load(filename, res_type='kaiser_fast')
mfccs_features = librosa.feature.mfcc(y=audio, sr=sample_rate,
n_mfcc=40)
mfccs_scaled_features = np.mean(mfccs_features.T,axis=0)

print(mfccs_scaled_features)
mfccs_scaled_features=mfccs_scaled_features.reshape(1,-1)
print(mfccs_scaled_features)
print(mfccs_scaled_features.shape)
predicted_label = model.predict(mfccs_scaled_features)
predicted_label = np.argmax(predicted_label, axis=1)
prediction_class = labelencoder.inverse_transform(predicted_label)

print(prediction_class)
```

REFERENCE

1. A. Borghesi and R. Maroldi, “COVID-19 outbreak in italy: Experimental chest x-ray scoring system for quantifying and monitoring disease progression,” La Radiologia Med., vol. 125, no. 5, pp. 509–513, 2020.
2. Anwar T., Zakir S. (2020). Deep Learning Based Diagnosis of COVID-19 Using Chest CT-scan Images. In IEEE International Multitopic Conference (INMIC), Bahawalpur, Pakistan, 5-7 Nov. 2020. IEEE. 10.1109/INMIC50486.2020.9318212
3. B. Cheng, et al., “Panoptic- deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 12475–12485.
4. C. Huang et al., “Clinical features of patients infected with 2019 novel coronavirus in Wuhan, China,” Lancet, vol. 395, no. 10223, pp. 497–506, 2020.
5. D. A. D. Júnior et al., “Automatic method for classifying COVID-19 patients based on chest X-ray images, using deep features and PSO-optimized xgboost,” Expert Syst. Appl., vol. 183, 2021, Art. no. 115452.
6. D. Apostolopoulos and T. A. Mpesiana, “COVID-19: Automatic detection from X-ray images utilizing transfer learning with convolutional neural networks,” Phys. Eng. Sci. Med., vol. 43, no. 2, pp. 635–640, 2020.
7. D.-P Fan et al., “Inf-Net: Automatic COVID-19 lung infection segmentation from CT images,” IEEE Trans. Med. Imag., vol. 39, no. 8, pp. 2626–2637, Aug. 2020.
8. Fakhfakh M., Bouaziz B., Gargouri F., Chaari L. (2020). ProgNet: COVID-19 Prognosis Using Recurrent and Convolutional Neural Networks.

9. G. Gaál, B. Maga, and A. Lukács, “Attention U-Net based adversarial architectures for chest X-ray lung segmentation,” 2020, arXiv:2003.10304.

10. J. Zhang et al., “Viral pneumonia screening on chest X-ray images using confidence-aware anomaly detection,” IEEE Trans. Med. Imag., vol. 40, no. 3, pp. 879–890, Mar. 2021.

11. Jacobi, M. Chung, A. Bernheim, and C. Eber, “Portable chest x-ray in coronavirus disease-19 (COVID-19): A pictorial review,” Clin. Imag., vol. 64, pp. 35–42, 2020.

12. L. Brunese, F. Mercaldo, A. Reginelli, and A. Santone, “Explainable deep learning for pulmonary disease and coronavirus COVID-19 detection from X-rays,” Comput. Methods Prog. Biomed., vol. 196, 2020, Art. no. 105608.

13. L. Orioli, M. P. Hermans, J.-P. Thissen, D. Maiter, B. Vandeleene, and J.-C. Yombi, “COVID-19 in diabetic patients: Related risks and specifics of management,” Annales d’Endocrinologie, vol. 81, no. 2, pp. 101–109, 2020.

14. L. R. Baltazar et al., “Artificial intelligence on COVID-19 pneumonia detection using chest Xray images,” PLoS One, vol. 16, no. 10, 2021, Art. no. e0257884.

15. L. Wang, Z. Q. Lin, and A. Wong, “COVID-Net: A tailored deep convolutional neural network design for detection of COVID-19 cases from chest X-ray images,” Sci. Rep., vol. 10, no. 1, pp. 1–12, 2020.

16. M. Berrimi, S. Hamdi, R. Y. Cherif, A. Moussaoui, M. Oussalah, and M. Chabane, “COVID-19 detection from xray and CT scans using transfer learning,” in Proc. Int.Conf. Women Data Sci. Taif Univ., 2021, pp. 1–6.

17. M. D. Hasan et al., “CVR-Net: A deep convolutional neural network for coronavirus recognition from chest radiography images,” 2020, arXiv:2007.11993.

18. M. Farooq and A. Hafeez, “COVID- ResNet: A deep learning framework for screening of COVID19 from radiographs,” 2020, arXiv:2003.14395.

19. M. L. Holshue et al., “First case of 2019 novel coronavirus in the United States,” New England J. Med., vol. 382, pp. 929–936, 2020.

20. M. Teymouri et al., “Recent advances and challenges of RT-PCR tests for the diagnosis of COVID-19,” Pathol. - Res. Pract., vol. 221, 2021, Art. no. 153443.

21. M. Z. Alom, M. Rahman, M. S. Nasrin, T. M. Taha, and V. K. Asari, COVID_MTNet: Covid-19 detection with multi-task deep learning approaches, 2020, arXiv:2004.03747.

22. Md. M. Ahsan et al., “Study of different deep learning approach with explainable ai for screening patients with COVID-19 symptoms: Using CT scan and chest X-ray image dataset,” 2020, arXiv:2007.12525.

23. N. Saeedizadeh et al., “COVID TV- Unet: Segmenting COVID-19 chest CT imagesusing connectivity-imposed U-Net,” Comput. Methods Programs Biomed. Update 1, 2021 Art. no. 100007.

24. P. K. Sethy and S. K. Behera, “Detection of coronavirus disease (COVID-19) based on deep features,” Preprints, vol. 2020030300, 2020, Art. no. 2020.

25. R. Selvan et al., “Lung segmentation from chest X-rays using variational data imputation,” 2020, arXiv:2005.10052.

26. S. Basu, S. Mitra, and N. Saha, “Deep learning for screening COVID-19 using chest X-ray images,” in Proc. IEEE Symp. Ser. Comput. Intell., 2020, pp. 2521–2527.

27. S. Kadry et al., “Development of a machine-learning system to classify lung CT scan images into normal/COVID-19 class,” 2020, arXiv:2004.13122.

28. Sarkar et al., “Detection of COVID-19 from chest X-rays using deep learning: Comparing cognex visionpro deep learning 1.0 software with open-source convolutional neural networks,” 2020, arXiv:2008.00597.

29. Shelke et al., “Chest X-ray classification using deep learning for automated COVID-19 screening,” SN Comput. Sci., vol. 2, no. 4, 2021, Art. no. 300.

30. Symptoms of Coronavirus, https://www.cdc.gov/ coronavirus/2019-ncov/ symptoms-testing/ symptoms.html (Last Accessed: 11.05.2020)

31. W. Taylor et al., “A review on the state of the art in non contact sensing for COVID-19,” Sensors, vol. 20, 2020, Art. no. 5665.

32. W.-J. Guan et al., “Clinical characteristics of coronavirus disease 2019 in China,” New England J. Med., vol. 382, no. 18, pp. 1708–1720, 2020.

33. Xin Zhang et al., “CXR-Net: A Multitask Deep Learning Network for Explainable and Accurate Diagnosis of COVID-19 Pneumonia From Chest X-Ray Images,” IEEE Journal of biomedical and health informatics, VOL. 27, NO. 2, Feb.2023.

34. Yang, P., Wang, X. COVID-19: a new challenge for human beings. Cell Mol Immunol 17, 555–557 (2020). https://doi.org/10.1038/s41423-020-0407-x

35. Z. Cao, T. Liao, W. Song, Z. Chen, and C. Li, “Detecting the shuttlecock for a badminton robot: A yolo based approach,” Expert Syst. Appl., vol. 164, 2020, Art. no. 113833.

36. Zargari Khuzani A., Heidari M., Shariati A. (2020). COVID-classifier: An Automated Machine Learning Model to Assist in the Diagnosis of COVID-19 Infection in Chest X-ray Images. medRxiv: 10.1101/2020.05.09.20096560