DEEP LEARNING BASED MUTLI CLASS DISEASES CLASSIFICATION FROM CHEST, X-RAY IMAGES

Mr.G.Suresh[1], Jangidi Veena[2], Kalthireddy Akshaya[2], Gangavarapu Pavani[2], Kailam Tulasi[2]

[1]Department of Electronics and Communication Engineering, Geethanjali Institute of Science and Technology, Nellore.

ABSTARCT: Chest X-rays are indispensable for diagnosing a wide range of diseases, play a pivotal role in diagnosing various diseases, but traditional methods relying on visual interpretation can introduce subjectivity and inefficiency. For instance, diagnosing pneumonia, especially among pediatric patients, poses significant challenges. but these methods can be subjective and inefficient. According to a study published in the British Medical Journal (BMJ), misinterpretation rates for chest X-rays range from 2% to 20%, depending on the condition and the experience of the radiologist. In response to these challenges, this study proposes a novel approach utilizing Convolutional Neural Networks (CNNs) to enhance diagnostic accuracy. CNNs have demonstrated remarkable capabilities in image recognition tasks, as evidenced by their widespread use in various fields, including medical imaging. By harnessing the power of CNNs, this method aims to mitigate the impact of human subjectivity and improve the efficiency of disease diagnosis. This has the potential to revolutionize pulmonary disease diagnosis, leading to faster treatment initiation and better patient outcomes. Furthermore, the scalability of CNN-based approaches allows for broader implementation across healthcare facilities, thereby benefiting a larger population. This represents a significant advancement in the field of radiology and has the potential to optimize healthcare resources and improve overall healthcare delivery.

Keywords: Multi-class classification, Disease classification, Chest X-ray images, Diagnostic accuracy, Pneumonia diagnosis, Misinterpretation rates, British Medical Journal (BMJ), Radiologist experience, Image recognition

1.INTRODUCTION

1.1 Overview

Chest X-ray images are widely used for the detection and diagnosis of lung diseases due to their non-invasive nature and ability to provide crucial information about the condition of the lungs. These images are instrumental in identifying abnormalities, such as infections, tumours, and other pulmonary disorders. Multiclass disease classification from chest X-ray images is a challenging task that has gained significant attention in the field of medical imaging. this task is to automatically categorize the chest X-ray images into multiple classes, such as pneumonia, tuberculosis, lung cancer, and other pulmonary diseases. Multiclass disease classification from chest X-ray images aims to develop a robust and accurate system for automatically identifying various abnormalities and diseases present in chest X-ray scans. This project is crucial in the field of medical imaging as it can significantly aid radiologists and clinicians in making timely and accurate diagnoses, leading to improved patient outcomes. The successful development and deployment of accurate multiclass disease classification models from chest X-ray images have the potential to greatly benefit the field of radiology. These models can aid radiologists in their clinical decision-making processes by providing them with automated support for

disease identification and classification. Furthermore, these models can help improve the efficiency of healthcare systems by reducing the time required for disease diagnosis and enabling early detection and intervention, ultimately leading to better patient outcomes.

1.2 Motivation

Early detection and accurate diagnosis of diseases are essential for effective treatment and management. Chest X-rays are commonly used to screen for a variety of respiratory and cardiovascular conditions, including pneumonia, tuberculosis, and lung cancer. By developing accurate multiclass classification models, healthcare professionals can identify these diseases at an early stage, leading to timely interventions and improved patient Multiclass disease classification from chest X-ray images can help in the development of personalized treatment plans. Different diseases require different treatment approaches, and accurate classification can ensure that patients receive the most appropriate care. For example, a patient with pneumonia may require antibiotics, while a patient with lung cancer may need chemotherapy or surgery. By accurately identifying the disease, healthcare providers can tailor treatment plans to individual patients, leading to better outcomes and reduced healthcare costs. Multiclass disease classification from chest X-ray images can help in the detection of rare and emerging diseases. For example, during the COVID-19 pandemic, chest X-rays were used to screen for and diagnose cases of the virus. By developing accurate multiclass classification models, healthcare providers can quickly identify cases of emerging diseases, allowing for timely interventions and public health measures. The motivation behind this endeavour is to improve patient outcomes, enhance personalized medicine, advance medical research, monitor and manage chronic conditions, and detect rare and emerging diseases. By accurately classifying chest X-ray images, healthcare providers can ensure that patients receive the most appropriate care, leading to better outcomes and reduced healthcare costs.

1.3 Problem Statement

The problem statement addresses the need for a solution that can overcome the challenges faced in disease classification, such as the complexity and variability of chest X-ray images, the presence of multiple disease classes, and the potential for misdiagnosis or missed diagnoses. The existing methodology aims to bridge the gap between the current state of disease classification and the desired state of accurate and efficient diagnosis. The multiclass disease classification from chest X-ray images is to accurately and efficiently classify diseases based on the X-ray images. The goal is to develop a reliable system that can assist doctors in diagnosing diseases from chest X-ray images, ultimately improving patient care and outcomes.

2. LITERATURE SURVEY

2.1 INTRODUCTION

Many studies have been conducted with the help of different AI-based methods. Numerous studies produced encouraging findings for using AI tools in classifying of lung diseases depending on certain features extracted from lung images. Deep learning (DL) which is a part of machine learning that uses artificial neural networks. DL is used widely in image classification, and recognition. Classifying of lung diseases is a challenging problem that requires specialized techniques. Deep learning approaches usually use a "convolutional neural network" (CNN). In Several image processing tasks, deep learning has demonstrated promising results, including

classifying of lung diseases in recent years. To distinguish between infected and healthy lung tissue. Eswara Rao proposed model had the highest training and testing accuracy of 97.2% and 98.8% on the crd dataset Koyyada survey presented in three folds. The first is an exploration of how researched had progressed from classic feature engineering approaches to deep learned methods Hussein was proposed for identifying covid-19 infection in chest x-rays. The custom-cnn model consists of eight weighted layers and utilizes strategies liked dropout and batch normalization to enhance performance and reduce overfitting Adjei-Mensah proposed a multi-eca attention mechanism embedded in the cov-fed to enhance feature maps of chest x-ray scans for classification without compromising performance

2.2 RELATED WORKS

Eswara Rao et. al [1] proposed model had the highest training and testing accuracy of 97.2% and 98.8% on the crd dataset and a training and testing loss of 0.02, 0.01. Upon various experiments, the proposed model had proven have been more accurate, portable, and memory-efficient than other deep and machine learned models for respiratory disease detection.

Koyyada et. al [2] survey presented in three folds. The first is an exploration of how researched had progressed from classic feature engineering approaches to deep learned methods; the second was how these were used to identified the listed diseases used radiology images such as chest x-rays (cxrs); and the third was the future path way of researched to detect these diseases.

Hussein et. al [3] was proposed for identifying covid-19 infection in chest x-rays. The custom-cnn model consists of eight weighted layers and utilizes strategies liked dropout and batch normalization to enhance performance and reduce overfitting. Their proposed approach achieved a classification accuracy of 98. 19% and aimed to accurately classify covid-19, normal, and pneumonia samples.

Khalif et. al [4] introduced a novel methodology that combines the strength of deep convolutional neural networks (dcnns) and generative adversarial networks (gans).They proposed study helps to mitigate the lack of labelled coronavirus (covid-19) images, which had been a standard limitation in related researched, and improved the model's ability to distinguish between covid-19-related patterns and healthy lung images. A thorough understanding of the model's performance in real-world scenarios was also provided by the study's meticulous evaluation of the model's performance used a variety of metrics, including accuracy, precision, recall, and f1-score.

Parthasarathy et. al [5] presented a new computer aided diagnosis used harris hawks optimizer with deep learned (cad-hhodl) method for pneumonia detection on cxr images.Their cad-hhodl method investigates the cxr images for the recognition and classification of pneumonia.To obtained this, the cad-hhodl method performed image pre-processing used a median filtering (mf) approached.For feature extraction, the residual network (resnet50) model was used.Their simulation values inferred the high performance of the cad-hhodl method over other techniques.

Mirza et. al [6] proposed that cnn-lstm model with neutrosophic fuzzy logic shows better accuracy with 98. 6% which was 4. 4 % higher when compared with covid caps, bayesian cnn , deep feature + svm and dcnn.Their study represented a major advancement in the creation of sophisticated and trustworthy diagnostic instruments for effective healthcare administration during times of worldwide health emergencies.

Malik et. al [7] proposed that experimentally presents that the fl framework achieves better results with these models trained by sharing data.These findings would provide medical institutions with the confidence they needed to apply collaborative methods and harness private data to fast construct a credible model for identifying multiple chest diseases.

Adjei-Mensah et. al [8] proposed a multi-eca attention mechanism embedded in the cov-fed to enhance feature maps of chest x-ray scans for classification without compromising performance. Extensive experiments were conducted on several public medical datasets and compared with some state-of-the-art models.

Islam et. al [9] proposed a novel approached called dcnn-vit-gru, which combines deep convolutional neural networks (cnns) with gated recurrent units (grus) and the vision transformer (vit) model for the accurate detection and classification of lung abnormalities. They evaluated the performance of their proposed approached on diverse datasets containing cases of lung abnormalities. Through cross-validation, our dcnn-vit-gru model achieved impressive weighted mean accuracy of 99% and 99. 86% for two distinct datasets, demonstrating its superior performance. Furthermore, in hold-out validation on separate datasets, the model achieved accuracies of 99. 09% and 99. 87%, respectively.

Jiang et. al [10] proposed transdd-pvt attained sota performance on the chestx-ray14 dataset, achieving a mean area under the receiver operating characteristic (auc) of 83. 1% across all 14 classes. Also, our method achieves 94. 31% accuracy and 93. 31% sensitivity on three-class classifications. Extensive experiments conducted on several datasets demonstrate the powerful ability of our trans to improve the performance of thoracic diseases classification. It could serve as a plug-and-play structure to improve the classification performance of both cnns and recent transformer-based backbones.

3.PROPOSED METHODOLOGY

3.1 OVERVIEW

The script utilizes the Tkinter library to create a graphical user interface (GUI) application for classifying diseases from chest X-ray images. The application's interface allows users to upload a directory containing chest X-ray images, preprocess the data, train machine learning models (specifically, Support Vector Machine and Convolutional Neural Network models), make predictions on test images, and visualize model performance. The GUI interface is constructed using Tkinter, providing users with an intuitive platform to interact with the application. Upon launching the application, users are presented with options to upload the dataset, preprocess the data, build an SVM model, build a CNN model, upload test data for prediction, and view accuracy comparison graphs.

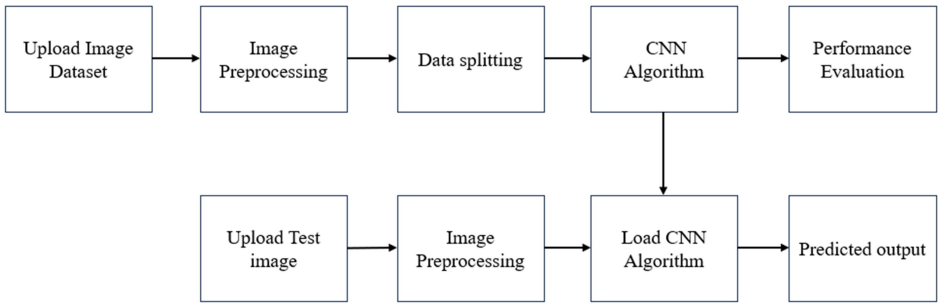


Figure: Block diagram of proposed model

The data preprocessing step involves resizing the images, converting them to NumPy arrays, normalizing the pixel values, shuffling the data, and splitting it into input features (X) and labels (Y). This ensures that the data is ready for training and testing. The application supports building two types of models: SVM and CNN. For the SVM model, scikit-learn is utilized to train the model on the preprocessed data. Performance metrics such as accuracy, precision, recall, and F1-score are computed and displayed, along with a confusion matrix and classification report. Similarly, for the CNN model, Keras is employed to define the model architecture comprising convolutional and pooling layers followed by fully connected layers. The model is compiled using the Adam optimizer and categorical cross-entropy loss function. After training the model, accuracy is displayed, and the trained model is saved for future use.Once the models are trained, users can upload test images for prediction. The CNN model is then used to predict the disease from the test image, and the predicted disease is displayed alongside the original image. Furthermore, the application offers graphical visualization capabilities. Users can view graphs depicting the accuracy and loss of the CNN model over epochs, providing insights into the model's training progress.

3.2 IMAGE PROCESSING

3.2.1 IMREAD

`imread` is a function commonly used in image processing libraries like OpenCV or MATLAB to read images from files into a data structure that can be manipulated. It loads the image as a matrix of pixel values representing the intensity or color at each point in the image. This function is fundamental for most image processing tasks as it allows the input of images for further analysis, transformation, or enhancement. The resulting matrix can be processed using a variety of techniques like filtering, edge detection, or feature extraction to extract useful information from the image.

3.2.2 RESIZE

In image processing, resizing refers to the operation of changing the dimensions of an image. This can involve increasing or decreasing the size of the image, typically by interpolation to estimate the values of new pixels based on the original image. Resizing is commonly used to prepare images for display at different resolutions, aspect ratios, or for fitting into a specific space. It can also be a preprocessing step for other operations such as object detection or recognition, where a standard input size is required.

3.2.3 RESHAPE

In image processing, `reshape` is a function used to change the dimensions of an image matrix while preserving the total number of elements. This operation is often used to convert between different image representations, such as reshaping a color image from a 3D matrix (height x width x channels) to a 2D matrix (height*width x channels), or reshaping an image into a different size for processing or display purposes. Reshaping can also be used to prepare image data for input into machine learning models by converting it into the required format.

3.2.4 FLATTEN

In image processing, "flatten" typically refers to converting a multi-dimensional image, such as a color image represented in three color channels (e.g., red, green, blue), into a single channel image. This process collapses the image into a two-dimensional array, where each pixel is represented by a single value. For example, in the case of a color image, the RGB values of each pixel can be averaged or combined using a formula to produce a single intensity value per pixel. Flattening can be useful for certain image processing tasks that do not require color information, such as edge detection or image compression.

3.3 CONVOLUTIONAL NEURAL NETWORK (CNN)

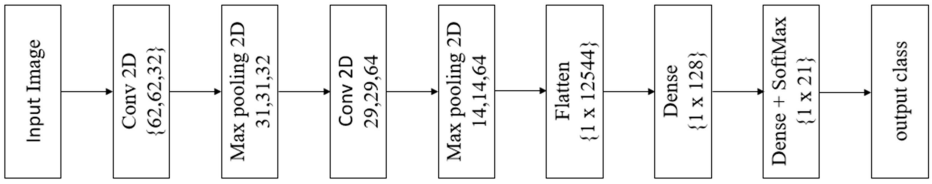


Figure: Proposed CNN Architecture

CNN Basics

According to the facts, training and testing of proposed model involves in allowing every source image via a succession of convolution layers by a kernel or filter, rectified linear unit (ReLU), max pooling, fully connected layer and utilize SoftMax layer with classification layer to categorize the objects with probabilistic values ranging from [0,1]. Convolution layer as is the primary layer to extract the features from a source image and maintains the relationship between pixels by learning the features of image by employing tiny blocks of source data. It's a mathematical function which considers two inputs like source image $I(x, y, d)$ where x and y denotes the spatial coordinates i.e., number of rows and columns. d is denoted as dimension of an image (here $d = 3$, since the source image is RGB) and a filter or kernel with similar size of input image and can be denoted as $F(k_x, k_y, d)$.

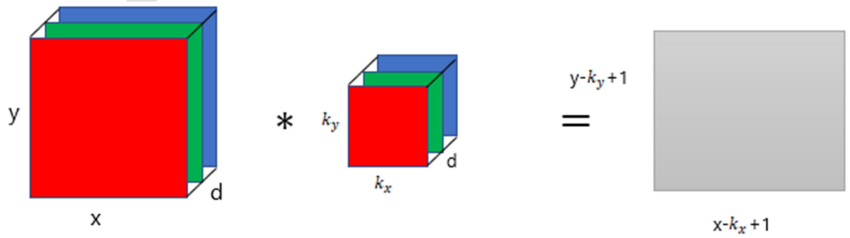
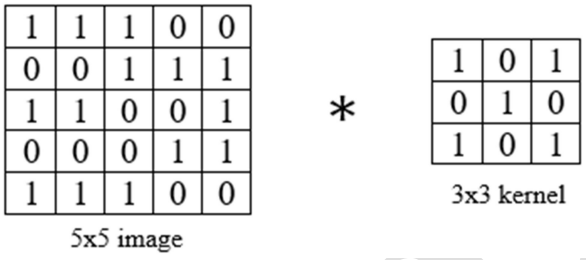
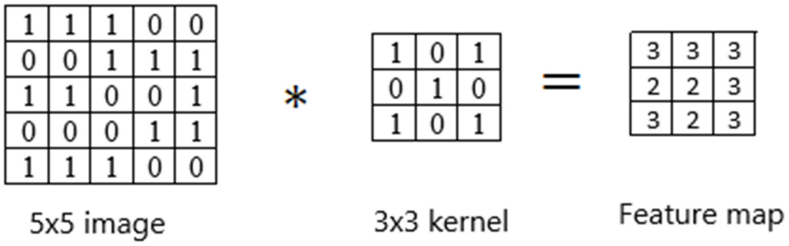


Fig. 4.3: Representation of convolution layer process.

The output obtained from convolution process of input image and filter has a size of $C\left((x-k_x+1),(y-k_y+1),1\right)$, which is referred as feature map. Let us assume an input image with a size of 5×5 and the filter having the size of 3×3. The feature map of input image is obtained by multiplying the input image values with the filter values.



(a)



(b)

Fig. 4.4: Example of convolution layer process (a) an image with size $\mathbf{5 \times 5}$ is convolving with $\mathbf{3 \times 3}$ kernel (b) Convolved feature ma

CONVOLUTIONAL LAYER

In Convolutional Neural Networks (CNNs), the convolution layer serves as the fundamental building block for feature extraction from input data, especially in image recognition tasks. This layer applies a set of learnable filters, also known as kernels, to the input data through a convolution operation. Each filter detects specific patterns or features within the input, such as edges, textures, or shapes, by sliding across the input spatially and computing element-wise multiplications and summations. As a result, the convolutional layer generates feature maps that highlight relevant spatial information present in the input. Additionally, CNNs typically incorporate activation functions, such as ReLU (Rectified Linear Unit), to introduce non-linearity and capture complex relationships within the data. By stacking multiple convolutional layers along with pooling layers for dimensionality reduction and regularization techniques like dropout, CNNs can effectively learn hierarchical representations of data, enabling them to extract intricate features and achieve state-of-the-art performance in various computer vision tasks.

MAXPOOILING LAYER

The Max Pooling layer in Convolutional Neural Networks (CNNs) is a critical component for downsampling and extracting the most salient features from feature maps generated by convolutional layers. This layer operates by partitioning the input feature map into non-overlapping regions, typically small rectangular areas, and then selecting the maximum value within each region. By retaining only the maximum activation, MaxPooling

effectively reduces the spatial dimensions of the feature maps while preserving the most significant features. This downsampling process helps in reducing computational complexity, alleviating overfitting, and enhancing translation invariance within the network. MaxPooling layers are typically interspersed between convolutional layers in CNN architectures, facilitating hierarchical feature extraction and spatial abstraction, ultimately contributing to the network's ability to learn robust representations of input data for tasks such as image classification, object detection, and semantic segmentation.

RELU ACTIVATION

In Convolutional Neural Networks (CNNs), the Rectified Linear Unit (ReLU) activation function is a crucial element that introduces non-linearity to the network's computations. ReLU operates by replacing negative values in the input tensor with zeros while leaving positive values unchanged. This simple yet effective activation function helps CNNs learn complex representations by enabling faster convergence during training and mitigating the vanishing gradient problem. ReLU's computational efficiency and ability to model complex relationships make it a popular choice for CNN architectures, facilitating the learning of intricate features in large-scale datasets. By promoting sparse activation and allowing the network to learn more robust representations, ReLU activation contributes significantly to the success of CNNs in various computer vision tasks, including image classification, object detection, and semantic segmentation.

FLATTEN LAYER

The Flatten layer in Convolutional Neural Networks (CNNs) plays a crucial role in transforming the multi-dimensional feature maps generated by the preceding convolutional and pooling layers into a one-dimensional vector. This transformation is essential for connecting the convolutional layers to the fully connected layers in the network's architecture. The Flatten layer reshapes the output volume into a linear array, maintaining the spatial relationships learned by the convolutional layers while preparing the data for processing by the dense layers. By flattening the feature maps, the network can effectively capture high-level abstractions and patterns across the entire input space, facilitating tasks such as classification and regression. The Flatten layer serves as a bridge between the convolutional feature extraction layers and the fully connected layers, enabling CNNs to leverage hierarchical representations and learn complex relationships within the data for accurate prediction and inference.

DENSE LAYER

In Convolutional Neural Networks (CNNs), the Dense layer, also known as a fully connected layer, represents a key component in processing extracted features for classification or regression tasks. Unlike convolutional layers that learn spatial hierarchies of features, Dense layers connect every neuron in one layer to every neuron in the next layer, forming a densely connected network. Each neuron in a Dense layer receives input from all neurons in the preceding layer and computes a weighted sum, followed by an activation function such as ReLU or softmax. This process allows the network to learn complex patterns and relationships in the high-dimensional feature space obtained from the convolutional and pooling layers. Dense layers are typically placed at the end of the CNN architecture, serving as the final stage where the network aggregates learned features to make predictions. By leveraging the interconnectedness of neurons in Dense layers, CNNs can effectively model

intricate dependencies within the data, enabling tasks such as image classification, object detection, and semantic segmentation with high accuracy and efficiency.

FULLY CONNECTED LAYER

In Convolutional Neural Networks (CNNs), fully connected layers, also referred to as dense layers, are critical components responsible for processing high-level features extracted by preceding convolutional and pooling layers. Unlike convolutional layers that capture spatial hierarchies, fully connected layers connect every neuron in one layer to every neuron in the next layer, forming a densely connected network. Each neuron in a fully connected layer receives input from all neurons in the preceding layer and computes a weighted sum, followed by an activation function such as ReLU or softmax. This process enables the network to learn complex patterns and relationships in the feature space, facilitating tasks such as image classification, object detection, and semantic segmentation. Fully connected layers are typically positioned at the end of the CNN architecture, aggregating learned features and making final predictions based on the extracted representations. Leveraging the interconnectedness of neurons, fully connected layers empower CNNs to model intricate dependencies within the data, thereby achieving accurate and efficient performance across a variety of visual recognition tasks.

SOFTMAX CLASSIFIER

In Convolutional Neural Networks (CNNs), the softmax classifier is a vital component utilized for multiclass classification tasks, often employed at the output layer of the network. The softmax function converts raw scores or logits generated by the preceding layers into probabilities representing the likelihood of each class. It accomplishes this by exponentiating the input values and normalizing them across all classes, ensuring that the probabilities sum up to one. This normalization enables the model to interpret the outputs as probabilities, facilitating intuitive interpretation and enabling straightforward selection of the most probable class. The softmax classifier is crucial for CNNs in tasks such as image classification, where it provides a principled approach to probabilistically assign class labels to input images based on the learned features. By utilizing the softmax function, CNNs can effectively handle multiclass classification problems, making them a cornerstone in various applications ranging from medical diagnosis to natural language processing.

ADAM OPTIMIZATION

Adam optimization, a popular algorithm in Convolutional Neural Networks (CNNs), is an adaptive learning rate optimization method designed to efficiently update network parameters during training. Adam combines the advantages of adaptive methods such as RMSprop with momentum techniques, resulting in faster convergence and improved performance. The algorithm maintains separate adaptive learning rates for each parameter, adjusting them based on past gradients and second moments of gradients. This adaptivity makes Adam well-suited for training CNNs, as it dynamically adjusts learning rates according to the specific requirements of each parameter, leading to more stable and efficient convergence. By incorporating momentum to prevent oscillations and adapting learning rates to different parameters, Adam optimization enhances the training process of CNNs, enabling them to effectively learn complex patterns and achieve superior performance across various tasks, including image classification, object detection, and semantic segmentation.

BINARY CROSS ENTROPY LOSS REDUCTION

Binary cross-entropy loss reduction in Convolutional Neural Networks (CNNs) is a critical aspect of training models for binary classification tasks. This loss function measures the dissimilarity between predicted and actual binary labels by computing the cross-entropy between the predicted probability distribution and the true binary labels. During training, CNNs aim to minimize this loss function using optimization algorithms like stochastic gradient descent (SGD) or Adam. By iteratively updating model parameters based on gradients computed from the binary cross-entropy loss, CNNs learn to make accurate predictions for binary classification problems, such as identifying whether an image contains a particular object or not. Binary cross-entropy loss reduction enables CNNs to effectively differentiate between positive and negative examples, optimizing model performance and enhancing its ability to generalize to unseen data.

TRAINING USING MULTIPLE EPOCHS

Training Convolutional Neural Networks (CNNs) using multiple epochs is a fundamental strategy in deep learning that involves iterating over the entire dataset multiple times during training. Each epoch consists of forward and backward passes through the network, where input data is propagated forward to compute predictions, and then gradients are computed backward to update the model parameters using optimization algorithms like stochastic gradient descent (SGD) or Adam. By training over multiple epochs, CNNs gradually refine their parameters, learning complex patterns and representations within the data. This iterative process allows the network to adjust its weights and biases to minimize the loss function, optimizing its ability to make accurate predictions. Training for multiple epochs is crucial for ensuring that the model captures diverse patterns present in the dataset and generalizes well to unseen data. It helps CNNs converge to an optimal solution, improving their performance across various computer vision tasks, including image classification, object detection, and semantic segmentation.

ADVANTAGES OF CNN

Hierarchical Feature Learning: CNNs can automatically learn hierarchical representations of data. They extract low-level features like edges and textures in earlier layers and progressively learn more abstract and complex features in deeper layers. This hierarchical approach enables CNNs to effectively capture the intricate structures and patterns present in images and other forms of data.

Translation Invariance: CNNs leverage convolutional operations and pooling layers to achieve translation invariance, meaning they can recognize patterns regardless of their location in the input space. This property makes CNNs robust to translations, rotations, and other transformations in the input data, enhancing their generalization ability.

Parameter Sharing: CNNs utilize weight sharing across spatial locations, reducing the number of parameters compared to fully connected networks. By sharing parameters, CNNs can efficiently learn from large datasets and generalize well to unseen data without overfitting.

Sparse Connectivity: In CNNs, neurons in each layer are only connected to a small region of the input volume, as determined by the receptive field size of the convolutional filters. This sparse connectivity reduces the computational burden and memory requirements of the network, enabling efficient processing of high-dimensional data such as images and videos.

Local Receptive Fields: CNNs exploit local connectivity and receptive fields to capture spatial dependencies and local patterns within the input data. By focusing on small regions of the input at a time, CNNs can effectively extract features while preserving spatial information, enabling them to handle complex tasks like object recognition and segmentation.

State-of-the-Art Performance: CNNs have demonstrated state-of-the-art performance across a wide range of computer vision tasks, including image classification, object detection, semantic segmentation, and image generation. Their ability to learn hierarchical representations and leverage spatial relationships within data makes them indispensable in many real-world applications.

4. RESULTS AND DISCUSSION

4.1 IMPLEMENTATION DESCRIPTION

The script for a graphical user interface (GUI) application using the Tkinter library. This application is aimed at performing multiclass disease classification from chest X-ray images using deep learning techniques, specifically Convolutional Neural Networks (CNN) and Support Vector Machine (SVM).

- Imports: The script starts with importing necessary libraries including Tkinter for GUI, file handling, data manipulation, visualization, machine learning models (such as SVM from scikit-learn and CNN from Keras), and image processing libraries like OpenCV and scikit-image.
- GUI Initialization: The main GUI window is created with specific dimensions and a title.
- Global Variables: Global variables are declared to store filenames, model accuracies, classifiers, and other metrics.
- Functions:
    1. upload(): Allows the user to select a directory containing chest X-ray images.
    2. preprocess(): Preprocesses the dataset by resizing images, normalizing pixel values, shuffling data, and encoding labels.
    3. SVM(): Builds an SVM model for classification, or loads a pre-trained model if available. It trains the model, evaluates its performance, and displays metrics like accuracy, precision, recall, and confusion matrix.
    4. buildCNNModel(): Builds a CNN model for classification, or loads a pre-trained model if available. It trains the model, evaluates its performance, and displays accuracy, precision, recall, and confusion matrix.
    5. predict(): Allows the user to upload a test X-ray image, makes predictions using the trained model, and displays the predicted disease on the image.
    6. graph(): Displays a graph showing the accuracy and loss during training of the CNN model.
- GUI Layout: Buttons are placed on the GUI window for uploading data, preprocessing, building SVM/CNN models, predicting disease, and displaying graphs. Text widget is used to display output messages and results.
- GUI Configuration: Fonts, button placements, text widget dimensions, and window background color are configured.
- Main Loop: The main GUI event loop is started using main.mainloop()

4.2 RESULTS AND DESCRIPTION

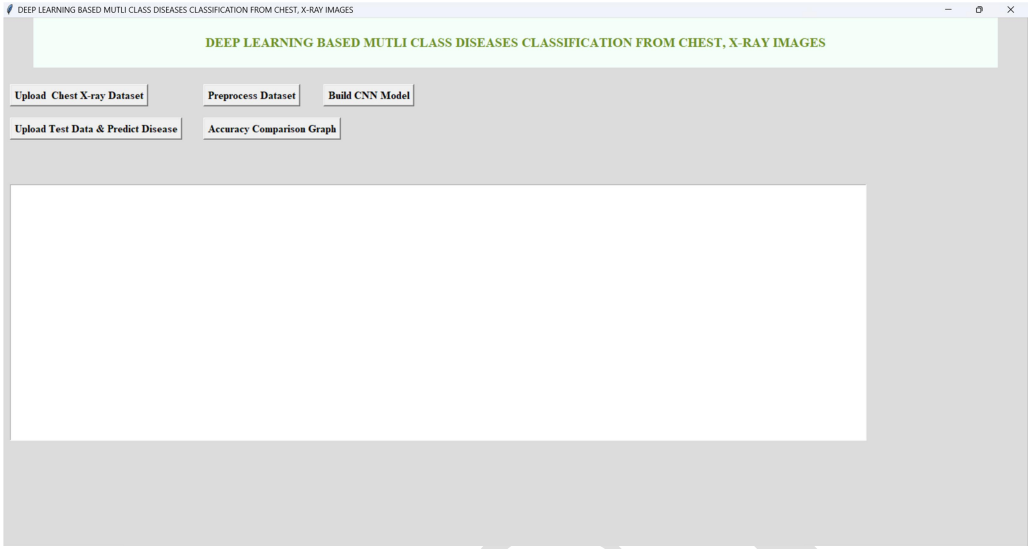


Figure 1: Sample UI used for Multi class disease classification

Figure 1 Sample UI used for multi-class disease classification presents the user interface designed for multi-class disease classification. It includes input fields for data, options for selecting classes or types of diseases, buttons for initiating the classification process, and areas for displaying results.

Figure 2 Simple File dialog box before uploading dataset figure displays a simple file dialog box that appears before uploading a dataset. It allows users to browse their local file system to select the dataset they want to upload for processing or analysis.

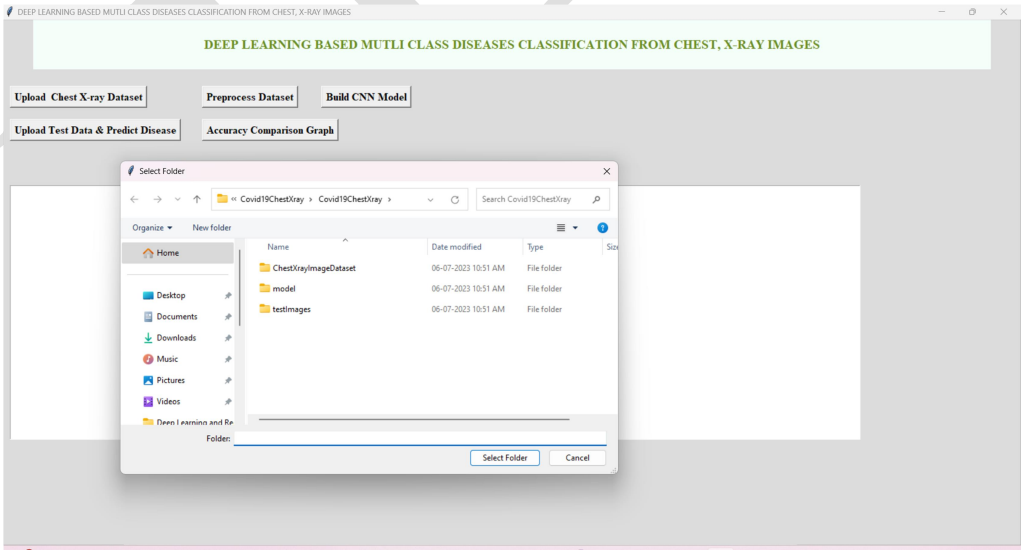


Figure 2: Simple Filedialogbox before uploading dataset

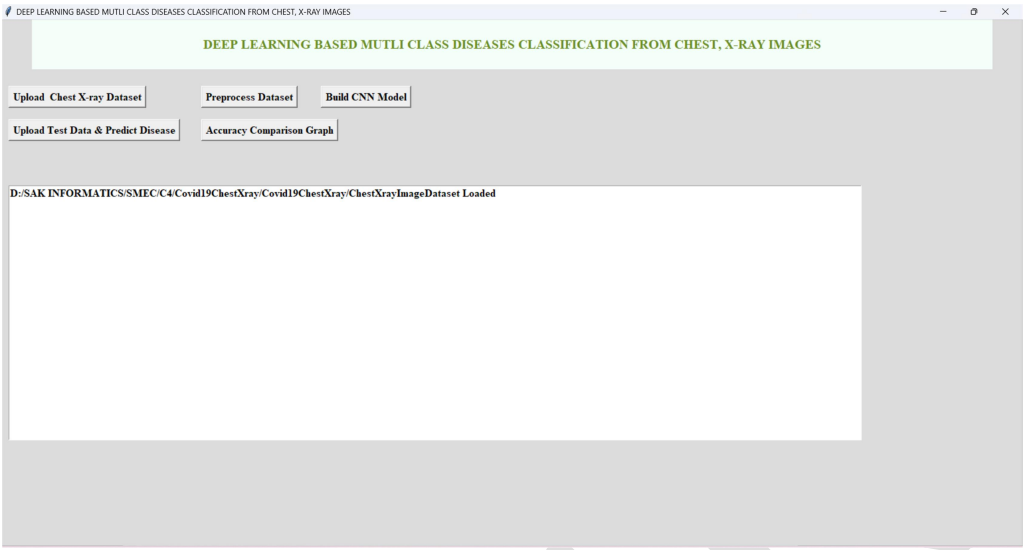


Figure 3: UI shows the path of the dataset after uploading dataset

Figure 3 UI shows the path of the dataset after uploading dataset figure showcases the user interface after a dataset has been uploaded. It includes information about the uploaded dataset, such as its file path or name, and possibly additional options for preprocessing or analyzing the data.

Figure 4: Model Summary of proposed CNN model figure provides a summary of the proposed Convolutional Neural Network (CNN) model architecture. It includes details such as the number of layers, layer types (convolutional, pooling, fully connected), activation functions, input and output shapes, and the total number of parameters.

Figure 5 Performance evolution of proposed CNN Model figure illustrates the performance evolution of the proposed CNN model over time or iterations. It includes metrics such as accuracy, loss, precision, recall, or F1 score plotted against epochs or iterations during the training process.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 62, 62, 32)        896
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 31, 31, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 29, 29, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 14, 14, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 12544)             0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               1605760
_________________________________________________________________
dense_2 (Dense)              (None, 21)                2709
=================================================================
Total params: 1,627,861
Trainable params: 1,627,861
Non-trainable params: 0
_________________________________________________________________
None
```

Figure 4: Model Summary of proposed CNN model

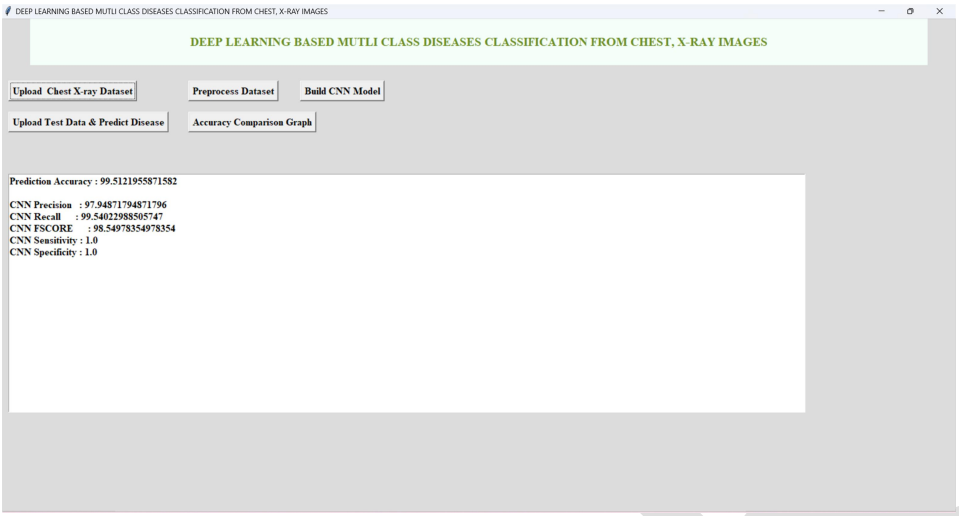


Figure 5: Performance evolution of proposed CNN Model

Figure 6 Confusion matrix of proposed CNN Confusion matrix figure displays the confusion matrix generated by the proposed CNN model. A confusion matrix is a table used to describe the performance of a classification model on a set of test data for which the true values are known.

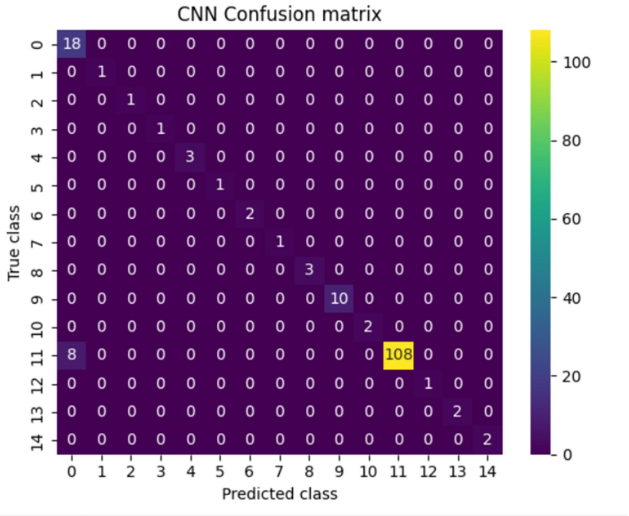


Figure 6: Confusion matrix of proposed CNN Confusion matrix

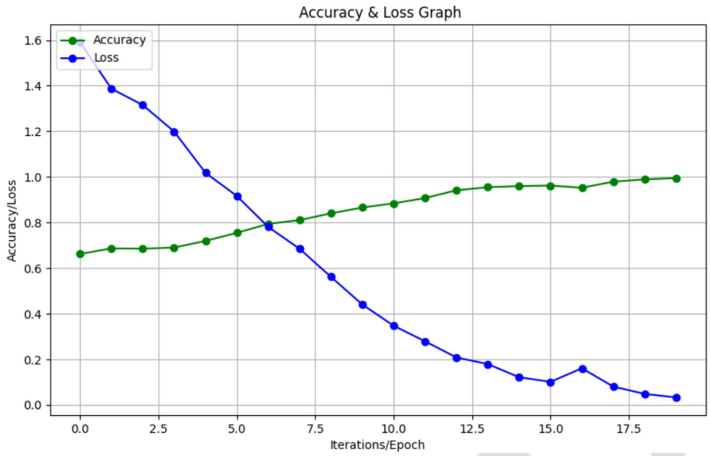


Figure 7: Accuracy and loss graph of CNN iteration wise

Figure 7 Accuracy and loss graph of CNN iteration-wise figure presents the accuracy and loss graph of the CNN model plotted against iterations or epochs during the training process. It helps in visualizing how the accuracy and loss change over the course of training, which is crucial for assessing the model's performance and identifying any overfitting or underfitting issues.

Figure 8 Predicted output using CNN Model figure shows the predicted output generated by the CNN model. It could be a visual representation of the classification results, possibly showing images with predicted labels or a table summarizing the predictions made by the model for different input data samples.

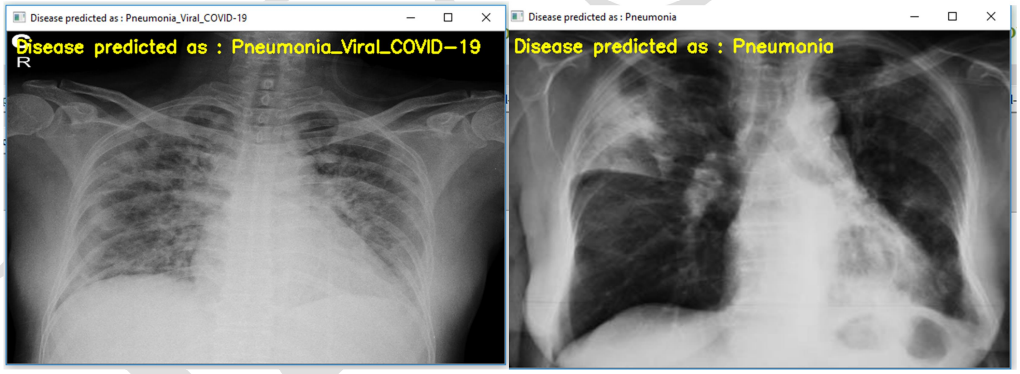


Figure 8: Predicted output using CNN Model

REFERENCES

[1] Eswara Rao, G. V., and B. Rajitha. "HQF-CC: hybrid framework for automated respiratory disease detection based on quantum feature extractor and custom classifier model using chest X-rays." International Journal of Information Technology (2024): 19.

[2] Koyyada, Shiva Prasad, and Thipendra P. Singh. "A Systematic Survey of Automatic Detection of Lung Diseases from Chest X-Ray Images: COVID-19, Pneumonia, and Tuberculosis." SN Computer Science 5, no. 2 (2024): 229.

[3] Hussein, Ahmad MohdAziz, Abdulrauf Garba Sharifai, Osama Moh'D. Alia, Laith Abualigah, Khaled H. Almotairi, Sohaib KM Abujayyab, and Amir H. Gandomi. "Auto-detection of the coronavirus disease

by using deep convolutional neural networks and X-ray photographs." Scientific reports 14, no. 1 (2024): 534.

[4] Khalif, Ku Muhammad Naim Ku, Woo Chaw Seng, Alexander Gegov, Ahmad Syafadhli Abu Bakar, and Nur Adibah Shahrul. "Integrated Generative Adversarial Networks and Deep Convolutional Neural Networks for Image Data Classification: A Case Study for COVID-19." Information 15, no. 1 (2024): 58.

[5] Parthasarathy, V., and S. Saravanan. "Computer aided diagnosis using Harris Hawks optimizer with deep learning for pneumonia detection on chest X-ray images." International Journal of Information Technology (2024): 1-7.

[6] Mirza, Olfat M., and Ahmed H. Samak. "Neutrosophic Fuzzy Logic-Based Hybrid CNN-LSTM for Accurate Chest X-ray Classification in COVID-19 Prediction." Sciences 18, no. 1 (2024): 14.

[7] Malik, Hassaan, and Tayyaba Anees. "Federated learning with deep convolutional neural networks for the detection of multiple chest diseases using chest x-rays." Multimedia Tools and Applications (2024): 1-29.

[8] Adjei-Mensah, Isaac, Xiaoling Zhang, Isaac Osei Agyemang, Sophyani Banaamwini Yussif, Adu Asare Baffour, Bernard Mawuli Cobbinah, Collins Sey, Linda Delali Fiasam, Ijeoma Amuche Chikwendu, and Joseph Roger Arhin. "Cov-Fed: Federated learning-based framework for COVID-19 diagnosis using chest X-ray scans." Engineering Applications of Artificial Intelligence 128 (2024): 107448.

[9] Islam, Md Khairul, Md Mahbubur Rahman, Md Shahin Ali, S. M. Mahim, and Md Sipon Miah. "Enhancing lung abnormalities diagnosis using hybrid DCNN-ViT-GRU model with explainable AI: A deep learning approach." Image and Vision Computing (2024): 104918.

[10] Srinivasarao, G., Penchaliah, U., Devadasu, G. et al. Deep learning based condition monitoring of road traffic for enhanced transportation routing. J Transp Secur 17, 8 (2024). https://doi.org/10.1007/s12198-023-00271-3

[11] Jiang, Xiaoben, Yu Zhu, Yatong Liu, Gan Cai, and Hao Fang. "TransDD: A transformer-based dual-path decoder for improving the performance of thoracic diseases classification using chest X-ray." Biomedical Signal Processing and Control 91 (2024): 105937.