CLASSIFICATION OF THORAX DISEASES FROM CHEST X-RAY IMAGES

Sharad Jayusukhbhai Dobariya

CLASSIFICATION OF THORAX DISEASES FROM CHEST X-RAY IMAGES

A Project

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

---

by

Sharad Jaysukhbhai Dobariya

December 2023

CLASSIFICATION OF THORAX DISEASE FROM CHEST X-RAY IMAGES

A Project

Presented to the

Faculty of

California State University,

San Bernardino

by

Sharad Jaysukhbhai Dobariya

December 2023

Approved by:

ABSTRACT

Chest X-ray images are crucial for medical decisions and patient care. However, their manual interpretation is time-consuming and prone to human error. This project aims to create an automated system that uses deep learning techniques to classify thorax disease from chest X-ray images. We are using the NIH Chest X-Ray Dataset, which contains many annotated images, as input data for this project. This approach uses UNet architecture as its classification layer. UNet architecture is well-known for its efficiency in image segmentation. Adding residual blocks enhances this approach's ability to classify images. The goal of this project is to create a robust and accurate classification model that uses UNet’s unique capabilities for feature representation and extraction. This would allow accurate discrimination between different forms of thorax diseases with high precision.

This project shows the effectiveness of UNet architecture with residual block for accurately classifying thorax disease types. These techniques combined produced superior results to many other architectures for medical image analysis, underscoring their importance.

ACKNOWLEDGEMENTS

We present with great enthusiasm and pride our comprehensive project report entitled “Thorax Disease Classification from Chest X-Ray Images.” This project is a testament to the fact that we have been exploring deep learning and machine learning fields with exciting potential for academic and professional growth throughout its entirety. This journey would not have been possible without the constant support, expert guidance, and collective wisdom from peers as well as various mentors.

My heart is filled with gratitude as a scholar at California State University-San Bernardino when I think about Dr. Jennifer Jin, who is instrumental in spearheading the research project that I am currently working on. Her leadership and guidance are crucial to our success. Our committee members Dr. Bilal Khan and Dr. Yan Zhang also deserve special recognition as outstanding mentors. Their knowledge helps us to achieve academic goals and venture boldly into emerging technology with passion.

California State University is a leader in educational innovation and deserves our gratitude. We can take advantage of cutting-edge technologies by leveraging their innovative offerings and utilizing the university's commitment to provide avant-garde courses. We are fortunate to be able to expand our technical and academic prowess every day in such a progressive academic environment.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

Background

Medical image classification is a key area of research in image processing, machine learning and computer vision. Accurate and efficient classification has become more important as technology advances and large datasets of medical images are available. The classification of chest X-rays is crucial in the diagnosis and treatment of various thoracic disorders. The NIH chest X ray dataset is a prominent dataset in the field. It contains over 100,000 anonymized images of chest X rays from more than 30,000 different patients. Deep learning models are used to extract relevant features.

Literature Review

Chest X-rays are one of the most frequently performed diagnostic imaging studies. Large datasets and deep learning techniques, automated analysis of chest X-rays has become an expanding area of research. Wang et al [1] provides valuable insight into the challenges and possible solutions of chest X-ray classification. They act as benchmarks to measure performance of various classification models on ChestX-ray8 dataset. Wang et. al. [1] introduces ChestXray8, a chest X-ray image database that is open to the public and contains over 100,000 images with disease labels. Their paper also provides benchmarks for weakly supervised classification and localization using convolutional networks (CNNs). This shows that their network could identify disease markers on images of X-rays without detailed annotations.

Yao et al. [2] tackles the problem of medical diagnosis using X-rays by identifying dependencies between various disease labels. This knowledge is then uses to train a model using deep learning from scratch, using correlation among diseases for training data. They take into account the fact that multiple conditions can coexist and interact, resulting in a more accurate and nuanced diagnosis system. Their methodology demonstrates how learning intricate correlations among diseases can enhance diagnostic accuracy.

The study by Rajpurkar et al. [3] shows that a deep-learning algorithm are capable of detecting pneumonia in chest X-ray images with accuracy levels higher than those achieves by radiologists. They focuses only on pneumonia detection by using 121 layer CNN training models. Their authors shows the potential AI has in medical diagnosis applications, such as ChestX ray14 dataset which is an extension of ChestX ray8 dataset.

Ronneberger et al.[4] introduces U-Net, a CNN architecture developed specifically for biomedical image segmentation. U-Net stands out in medical imaging because its architecture works efficiently even with limited training samples while producing precise segmentations results. Furthermore, the model structure facilitates precise localization of structures within biomedical images. Siddique, N. et al's [5] paper on U-Net and its iterations provides a thorough investigation of how U-Net architecture has been utilized for medical image segmentation applications. They discuss its theory behind construction as well as modifications for enhanced performance as well as applications of U-Net in medical imaging.

Khanna et al’s [6] research employs a variant of U-Net architecture which utilizes residual learning (Residual U-Net) for lung segmentation on CT images. By building deeper networks with residual connections that utilizes U-Net's enhanced learning capacities and performance for segmenting medical images.

Deep learning's success in medical imaging goes beyond architectures alone, however. He et al.’s [7] residual learning concept enables deeper networks without encountering typical vanishing gradient issue. Furthermore, Kingma & Ba [8] presents training strategies such as Adam optimizer that proves instrumental in ensuring convergence of their networks.

Data augmentation techniques should not be underestimated either; they expand and diversify training data in order to prevent overfitting. Shorten & Khoshgoftaar [9] provides an exhaustive survey of various augmentation techniques utilizes in deep learning environments.

Prechelt [10] discusses early stopping as an invaluable way of increasing performance and understanding when to end training, offering invaluable guidance.

Significance

Accurate classification of chest X-ray images has far-reaching ramifications for healthcare providers and patient care. Being able to quickly detect and classify thoracic diseases automatically can aid radiologists in early diagnosis of pneumonia, tuberculosis, and lung cancer - leading to timely interventions, improved treatment planning, improved patient outcomes, and faster interventions overall. Furthermore, automating image classification tasks helps relieve medical professionals of manual analysis tasks; freeing them up for more critical cases or complex diagnoses.

In this project, we seek to address the challenges associate with chest X-ray classification using a combination of UNet architecture and residual layers. Originally has proposed for biomedical image segmentation, this model has shown its promise in capturing fine-grain details and spatial information. By adding residual layers - which enable gradient information propagation as well as deep network training - to our model's design we hope to enhance its performance and robustness while improving classification accuracy for thoracic diseases seen on chest X-ray images. We hypothesize that using both models together can produce improved classification accuracy when classifying diseases seen on chest X-ray images.

Purpose

The purpose of this project is twofold. Firstly, we aim to optimize the classification model for the NIH Chest X-ray dataset by systematically exploring various factors such as hyperparameters, image sizes, and input configurations. Through a comprehensive experimentation process, we can identify the optimal settings that yield the highest classification accuracy. Secondly, we seek to compare the performance of the UNet model with residual layers against other popular convolutional neural network (CNN) architectures, such as VGG and ResNet, on the ChestX-ray dataset. This comparative analysis provides insights into the strengths and weaknesses of different models for chest X-ray classification.

By achieving these objectives, we hope to contribute to the advancement of medical image classification in the context of chest X-ray analysis. The findings of this project can potentially benefit radiologists, healthcare providers, and researchers working in the field of thoracic disease diagnosis. The developed model has the potential to assist in the accurate and timely classification of chest X-ray images, leading to improved diagnostic accuracy, reduced workload for medical professionals, and enhanced patient care.

CHAPTER TWO

OVERALL DESCRIPTION

Hardware Requirement

- Memory: 32 GB (minimum)
- Graphics Card: NVIDIA Tesla P100
- CPU: Intel Core i5 or above, Apple M series
- OS: Windows, Mac OS, Linux

The implementation of the classification model and the experimentation process in this research project requires specific software and programming languages. The following software and language requirements has utilized to develop and evaluate the proposed approach.

Software Requirements

Python, an open-source programming language, serves as the primary language for this research project. Python provides a wide range of libraries and frameworks that are essential for deep learning, image processing, and scientific computing tasks.

The following software components were used:

Python

Version 3.7 or higher is used as the programming language for model implementation, data manipulation, and experimentation.

Deep Learning Frameworks

The project utilizes TensorFlow (version 2.4.0) and Keras (version 2.4.3), popular deep learning frameworks, to build, train, and evaluate the classification model. These frameworks offer a high-level API for efficient implementation of neural networks.

Jupyter Notebook

Jupyter Notebook (version 6.1.4) provides an interactive development environment for coding, documentation, and experimentation. It allows for seamless integration of code, visualizations, and textual explanations.

Data Manipulation Libraries

The NumPy library (version 1.19.2) is used for efficient handling and manipulation of numerical data, while the Pandas library (version 1.2.0) facilitate data pre-processing and analysis tasks.

Image Processing Libraries

OpenCV (version 4.5.1) and PIL (Python Imaging Library, version 8.1.0) are employed for image loading, resizing, and augmentation operations. These libraries provide essential functionalities for pre-processing the chest X-ray images.

Visualization Libraries

Matplotlib (version 3.3.2) and seaborn (version 0.11.0) are utilized for generating visualizations, plotting results, and analyzing the performance of the classification model.

Programming Language Requirements

The research project requires proficiency in the following languages and concepts:

Python

A solid understanding of Python programming language is essential for implementing the classification model, pre-processing the dataset, and conducting experiments. Proficiency in Python's syntax, data structures, and libraries is necessary for efficient coding and data manipulation.

Deep Learning Concepts

A comprehensive knowledge of deep learning concepts, including convolutional neural networks (CNNs), optimization algorithms, loss functions, and regularization techniques, is crucial for designing and training the classification model. Understanding these concepts enable the utilization of appropriate architectural choices and optimization strategies.

Image Processing Knowledge

Familiarity with image processing techniques, such as image resizing, normalization, and data augmentation, is necessary for pre-processing the chest X-ray images. This knowledge facilitates the preparation of the dataset for effective model training and evaluation.

Experimental Design and Analysis

A solid grasp of experimental design principles and statistical analysis is essential for conducting rigorous experiments, comparing model performance, and drawing meaningful conclusions from the results. Applying appropriate statistical tests and visualization techniques allow for insightful interpretation of the experimental findings.

Dataset Information

This research project uses the NIH Chest X-Ray dataset, collected and maintained by the National Institutes of Health (NIH). This vast repository includes anonymized chest X-ray images has taken from over 30,000 patients gather through natural language processing analysis of radiology reports associated with each image for additional diagnostic information.

Dataset Characteristics

- The NIH Chest X-Ray Dataset comprises over 100,000 chest X-ray images taken of patients' thoracic regions by physicians across the US and labelled with various lung conditions such as pneumonia, tuberculosis and lung cancer - with each image labelled to reflect its relevance within this dataset.
- Multi-label classification is necessary as images may display multiple pathologies at once.
- Our dataset features numerous pathology classes to provide an inclusive representation of thoracic diseases encountered during medical practice.
- Data frames represent datasets. Here we explore each column to better understand its contents:

- Image Index - Representing a unique identifier or filename associated with medical images
- Finding Labels: This column describes any findings or abnormalities detected within medical images. In the first row, multiple findings are separated using "|", such as Infiltration, Mass Nodule and Pleural Thickening presence; for rows two and three the label reads "No Finding," meaning there were no abnormalities uncovered from that particular medical image.
- Follow-Up #: It displays the date or duration since initial examination; for instance, in row one there was 11 follow-up examinations since initial exam.
- Pertaining Patient I.D: Each unique patient identification number. Pertaining Patient Age During Examinations.
- Patient Gender (M for male, F for female) and View Position are essential information about medical images taken, which indicate gender as well as view position (AP for anterior-posterior and PA for posterior-anterior images respectively).
- OriginalImage[Width and Height]: These columns indicate the dimensions (width and height) of an original medical image
- OriginalImagePixelSpacing[x,y]: These columns demonstrate how pixels space out across both dimensions in its original image.
- Finding labels

- "No Finding" With over 60,361 instances has recorded so far, this label indicates no abnormalities or findings are discovered within any image X-ray taken for that patient.
- "Infiltration'" has an infiltrate count of 9,547 and indicates the presence of abnormal infiltrates within lung tissue.
- "Atelectasis" has atelectasis count of 4,215 which suggests collapsed lung tissue as the culprit for infiltrations and atelectasis respectively.
- "Effusion" occurs 3,955 times and indicates an accumulation of fluid in the pleural cavity.
- "Nodule" occurs 2,705 times; this indicates small abnormal growths or nodules present within a lung tissue sample.
- "Pneumothorax" appears 2,194 times in our datasets and represents air or gas build-up within the pleural cavity leading to lung collapse, while "Mass" shows 2,139 appearances to indicate abnormal masses or tumors present within lungs.
- "Effusion|Infiltration" appears 1,603 times, suggesting both effusion and infiltration findings were present in equal numbers.
- "Atelectasis|Infiltration" has 1,350 counts indicating similar findings of both types.
- "Consolidation" occurs 1,310 times, signaling consolidation of lung tissue often caused by pneumonia.

- "Atelectasis|Effusion" occurs 1,165 times - suggesting an overlap in symptoms between atelelectasis and effusion findings.
- "Pleural_Thickening" appears 1,126 times, signifying thickening of the pleural membranes surrounding the lung, while "Cardiomegaly" shows up 1 093 times; an enlarged heart would qualify.
- "Emphysema": This term appears 892 times, signifying lung tissue damage and increase airspace volume.
- "Infiltration|Nodule" has 829 instances recorded indicating both nodules and infiltrations as findings.

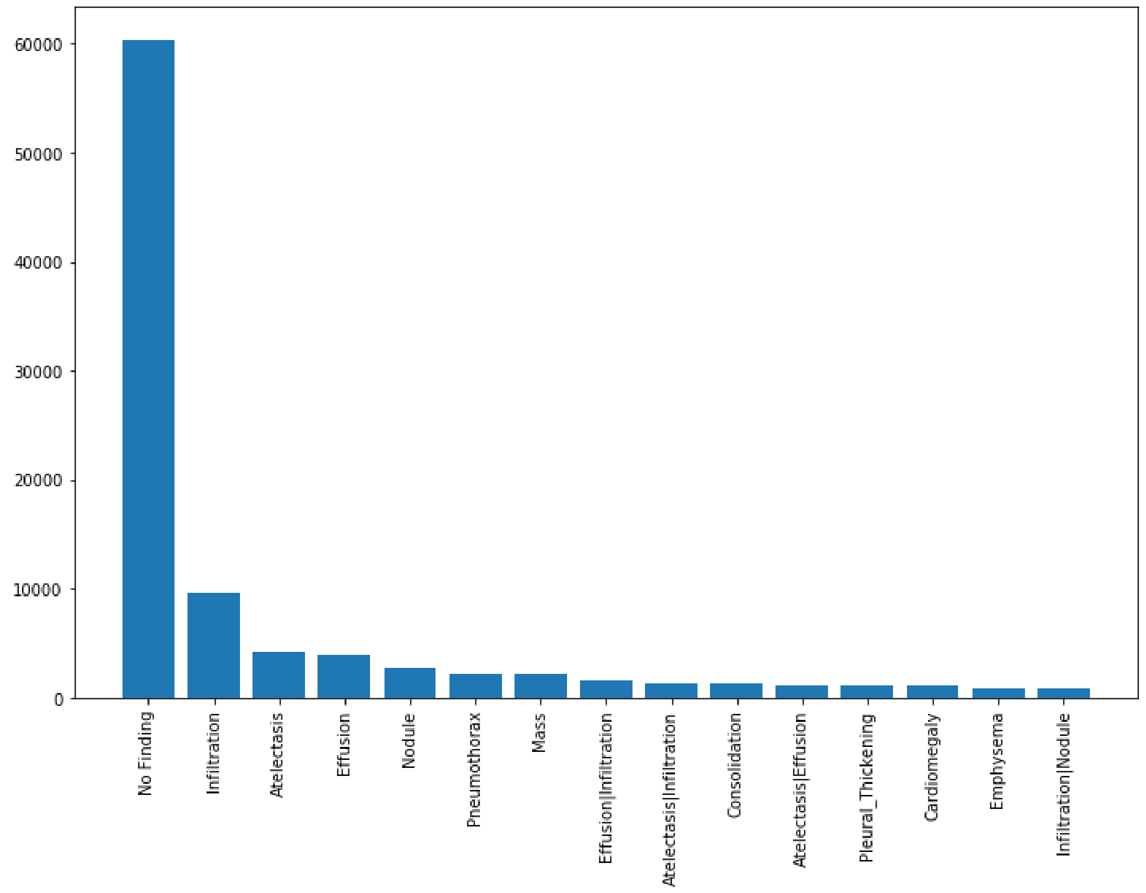

Figure 1: Data Set

This Figure 1 offers insights into the distribution and prevalence of abnormal findings or conditions found within an X-ray dataset. It helps in comprehending relative frequency across conditions found within images taken of anatomical parts.

<u>Dataset Split</u>

- Data is split into subsets for training, validation, and testing to assess the performance. The training set, comprising 70%, has used to train the models. 10% has used for hyperparameter tuning, model selection, or validation, while

20% is a sample independent to measure the overall performance of the classification model. Below Figure 2 Shows percentage of various conditions available in training data frame.

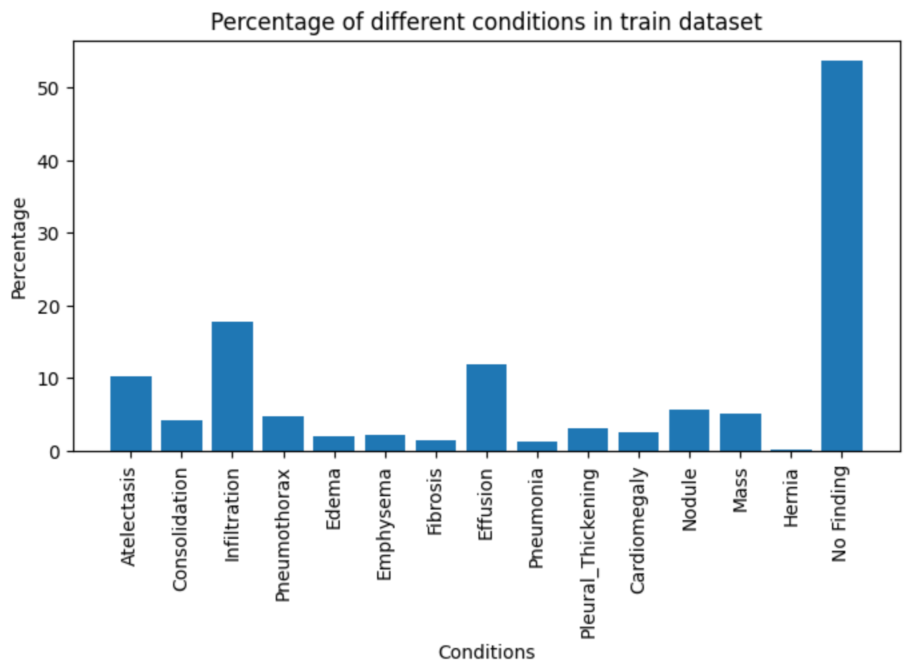

Figure 2: Percentage of Different Conditions in Dataset

]

Data Analysis

- An exploratory analysis has performed to gain insights into the distribution and co-occurrence patterns among various classes in terms of pathologies. This process provides insight into multi-label classification challenges as well as guidance in selecting suitable evaluation metrics.

Dataset Limitations

- It is essential to recognize that this dataset has certain constraints. Since labels have been collected from radiology reports, there may be inherent uncertainties with diagnoses; furthermore, without an "diagnosis confidence" attribute to define label accuracy is difficult; and conflicts among physicians regarding certain diagnoses could potentially cause mislabelled images.
- Despite these constraints, this dataset offers an unparalleled opportunity for training and testing deep learning models for chest X-ray classification.

Technology Stack and its Application

This chapter provides the general background behind the technology has been utilized for image classification in this project, using machine learning and deep learning techniques like UNet for faster training times and optimal results.

Python Programming Language

Python serves as our primary programming language to implement and analyze our classification model and conduct data analysis. Notorious for its flexibility and ease-of-use, this widely popular programming language offers a comprehensive library and framework specifically developed to address scientific computing tasks as well as deep learning tasks for machine learning tasks and image processing tasks.

Deep Learning Framework

Deep learning frameworks such as TensorFlow and Keras are integral parts of the technology stack, offering high-level APIs and prebuilt functions for designing, training, and evaluating neural network models efficiently. By capitalizing on the power of these framework researchers were able to quickly design complex deep learning architectures while optimizing model performance effectively as they handle large image datasets efficiently.

Jupyter Notebook

Jupyter Notebook is an integral tool in aiding the interactive development, experimentation, and documentation of our classification model. Thanks to its web interface with code support for code, visualizations, and text explanations as well as its seamless environment for implementation and testing various components it allows researchers to iteratively refine code, visualize intermediate results visually as well as effectively communicate findings to one another.

Image Processing Libraries

Popular image processing libraries such as OpenCV and PIL (Python Imaging Library) are being utilized for preprocessing and manipulating chest X-ray images, providing functions including image loading, resizing, augmenting data sets to boost model generalization as well as assuring compatibility of images with classification models. Researchers utilizes these libraries as they enable data preparation while performing data augmentation to increase generalization capacity as well as to ensure images could work within classification models.

Data Manipulation Libraries

NumPy and Pandas libraries are essential in handling numerical data efficiently and manipulating it effectively, offering efficient numerical operations as well as array manipulation capabilities, while Pandas provides data preprocessing, cleaning, and analysis functionality. Researchers can effectively preprocess their dataset and draw meaningful insights out of it for training/evaluation, using these libraries.

Visualization Libraries

Visualization libraries such as Matplotlib and Seaborn provides researchers with tools for producing informative plots, charts, and graphs to display data visually, evaluate model performance, present their research findings, or communicate interpretation and communication of results more easily. They provide various forms of visualization such as bar plots, scatter plots, and heatmaps which make results easy to comprehend and communicate through visualization techniques such as bar plots.

Flow of the Project

Data Loading and Preprocessing

- To start out the code is loaded using **pd.read_csv** and **glob** function. The CSV containing metadata about chest X-ray images is read into pandas DataFrame (**all_xray_df**), and then extracted image paths are stored in a dictionary (**all_image_paths**).

- Image paths are then added to the DataFrame using the **map** function, creating a column titled 'path' in it.

Data Processing

- One-Hot Encoding: Since the classification is a multiclass problem, the code performs one-hot encoding for the target labels. A list of all possible labels (**all_labels**) is defined, and for each label, a new column is added to the DataFrame with a value of 1 if the label is present in the 'Finding Labels' column and 0 otherwise. This step ensures the representation of the labels in a suitable format for training the classification model.
- Balancing the Dataset: To address class imbalance, the code balances the dataset by sampling all examples from the DataFrame. The sampling is performed based on the sample weights, which are calculated using the number of findings associated with each image. The resulting balanced dataset is stored back into all_xray_df.

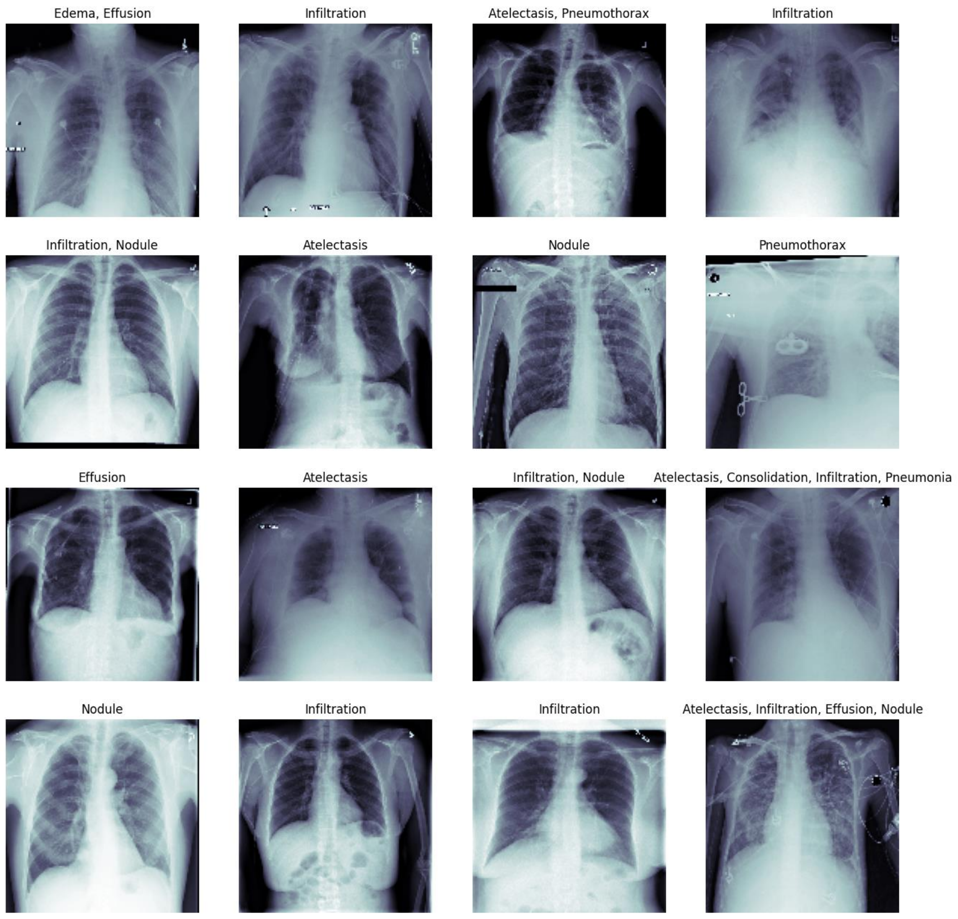

Figure 3: Images of the Chest X-ray and Corresponding Output Labels

This figure 3 Shows some of the images of the Chest X-ray from dataset and depicts corresponding output labels to respective image.

Train-Test Split

- This code utilizes the **train_test_split** function from the **sklearn.model_selection** to divide data sets into training and validation sets using 0.3 as test size. As a result, training dataFrames (train_df) are created and stored into **train_df** and **valid_df** folders respectively.

- For convenience, training and validation sets sizes are listed to give an overall view of their data split.

Data Generator

- The code sets up data generators using the ImageDataGenerator class from Keras. These generators used to load and preprocess the images on the fly during model training.
- The Core_IDG object is then instantiated which defines data augmenter transformations and normalization options.
- To generate train and validation generators for training a neural network, two DataFrames named train_df and valid_df are passed as input to flow_from_dataframe method in their respective DataFrames - these provide batches of preprocessed images paired with labels in a format compatible with neural network training.
- These generators have been configured to resize images to their specified IMAGE_SIZE, convert them to grayscale format, randomly shuffling data and using an adjustable batch size.

Model Training and Testing

- This code implements a machine-learning model training and testing phase. It includes defining a model structure (e.g., UNet with residual layer), compiling it by using an appropriate optimizer/loss functions combination, fitting it to the training data using the fit method, and validating/assessing it on test or validation sets using the evaluate method.

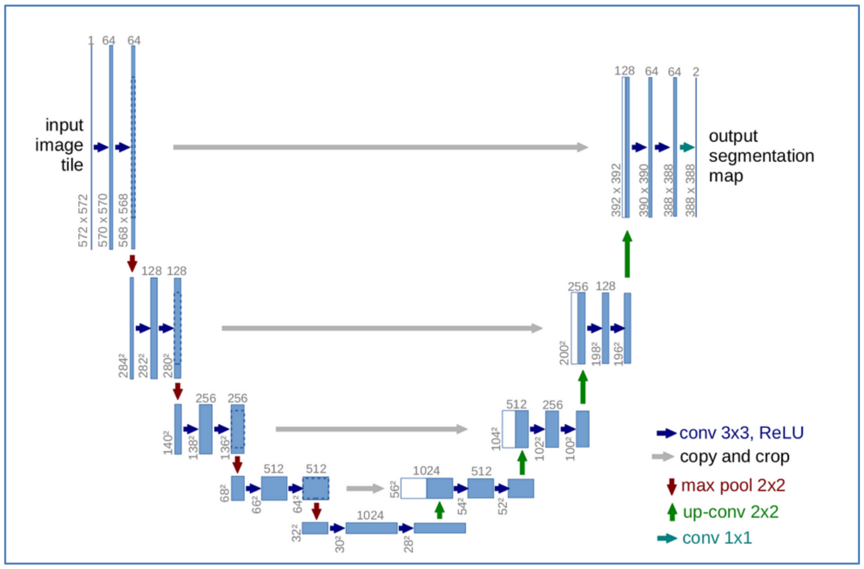

Figure 4: Unet Architecture

At first glance, Figure 4 appears to have a "U-shaped" form. The architecture is symmetrical and consists primarily of two parts: the left part, called the contracting path is made up of the general convolutional processes; the right, expansive path is made up of transposed 2d layers.

Above U-net Architecture (example 32x32 pixels at the lowest resolution). Each blue box represents a feature map with multiple channels. On the top of each box, you will find several channels. The lower left corner of the box is the x-y size. The white boxes are copied feature maps. The arrows indicate the different operations.

- Along with the Unet, the classification layer is enhanced by residual blocks or skip connections that feature as follows.

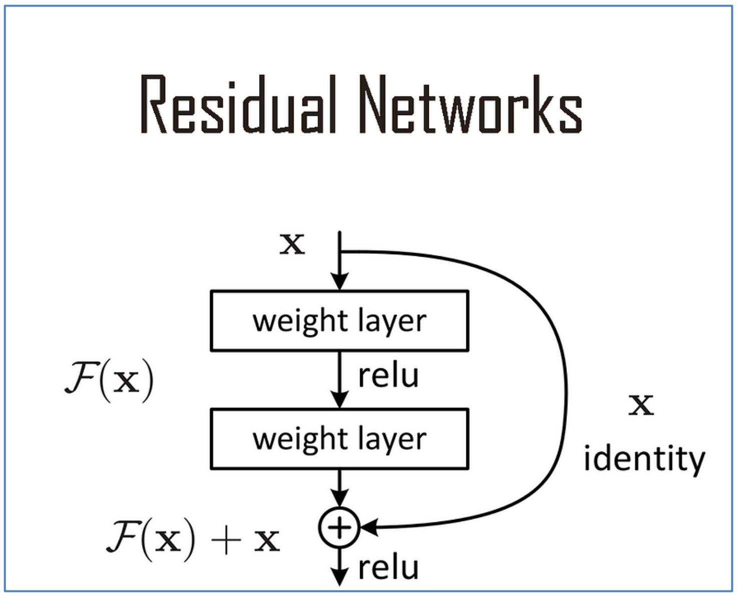

Figure 5: Residual Network

Figure 5 shows a single residual block with skip connection. Skip connections refer to any mechanism whereby the output from one layer is fed directly to a later layer, in U-Net architecture for image segmentation the use of long skip connections to avoid passage through deeper levels and preserve spatial information that might otherwise become lost within its layers.

CHAPTER THREE

METHOD

Image Transformation

Samplewise Centering and Standardization

ImageDataGenerator can be utilized by setting its **samplewise_center =True** and **samplewise_std_normalization = True** parameters to optimize the convergence and performance of models. This transformation centers each image around zero while scaling its pixels with unit variance, improving convergence and performance of models.

Rotation

ImageDataGenerator is configured with **rotation_range=3**, randomly rotating each image by 3 degrees during training to add variability and make its model more robust to changes in test data.

Reflective Fill Mode

ImageDataGenerator sets its **fill_mode** parameter to 'reflect'. This setting fills any holes created during rotations or shifts with values taken from adjacent pixels - helping preserve image content without creating artificial patterns at its boundaries.

Horizontal and Vertical Flipping

ImageDataGenerator allows random horizontal flipping during training using its **horizontal_flip=True** parameter, providing additional variations to image orientation that help broaden a model's ability to generalize across different orientations. Vertical flipping may also be applied as needed.

Width and Height Shifts

The **width_shift_range** and **height_shift_range** parameters of ImageDataGenerator have been set to 0.1 and 0.05 respectively to enable random translations of images horizontally and vertically within certain ranges, aiding learning models with features insensitive to small translations.

Zooming

ImageDataGenerator sets its zoom_range parameter to 0.15; this augmentation technique randomly applies zooming transformations to images within this range, magnifying or diminishing their content within it. Zooming helps the model learn to focus on various parts of an image while improving its ability to detect and classify objects at different scales.

Models

This research project employs a model that combines U-Net architecture with a classification layer to optimize image segmentation tasks. U-Net architecture is an established approach for image segmentation that involves two networks, an encoder network, and a decoder network, with the encoder network collecting high-level features from an input image while its decoder reconstructs segmented output images.

U-Net architecture can be especially effective at image segmentation due to its ability to capture fine-grained structures and accurately localize objects. This is achieved via skip connections that enable decoder access and the use of low-level feature maps from the encoder. By including such connections, the U-Net model can make use of both high-level features as well as lower ones, providing precise localization while simultaneously capturing fine details in segmented regions.

In this research project, U-Net architecture has enhanced by adding a classification layer. With this addition, the model not only performs segmentation but can also classify segmented regions into specific categories for classification. By combining segmentation and classification capabilities into one comprehensive solution, this model gains the capacity to offer detailed analysis and understanding of image content.

U-Net architecture, as a convolutional neural network, excels at extracting powerful feature representations from images. The convolutional layers in this model capture complex patterns, textures, and structures present in an input image to form powerful discriminative features that improve accuracy when discriminating between classes accurately.

U-Net architecture's encoder-decoder structure also facilitates the capture of multiscale context information. While an encoder network collects high-level features from high-level connections, its counterpart (decoder network) combines them with low-level features from skip connections for a more comprehensive analysis of both local and global contexts, leading to improved segmentation and classification performance.

Training of a combined U-Net and classification model should be carried out comprehensively. Loss functions such as cross-entropy loss or dice loss provides optimal training conditions, considering both segmentation and classification objectives simultaneously. Through joint training, optimization can occur across the whole model for improved overall performance.

By employing the U-Net architecture with a classification layer, this research project achieves improved segmentation and classification results. The encoder-decoder structure of the model allows precise localization and segmentation of objects while its classification layer adds the capability of classifying segments into specific categories - making this model suitable for performing image analysis tasks of all sorts - providing valuable insights into both content and characteristics of input images.

In this project, we use Binary accuracy as an approach to one vs rest classification. With 15 binary classifiers total, each class of interest being labeled positive, and all others labeled negative; we then calculate Macro-accuracy which provides a useful measure for evaluating multi-class classification models when classes are unequally distributed. Macro-accuracy is calculated by averaging the precision or recall for each class. Macro-accuracy is particularly useful when evaluating multi-class classification models with many classes, as it's often difficult to visually inspect performance for individual classes individually. Macro-accuracy provides one metric to summarize all performance across all classes; binary accuracy can also be calculated this way.

Optimizer

Optimizers are essential in updating the weights and biases of neural networks during training; they adjust their parameters based on gradients computed from backpropagation algorithms that calculate loss function gradients with respect to model parameters. Optimizer selection has a significant influence over convergence speed as well as final model performance.

Adam (Adaptive Moment Estimation) is one of the more frequently used optimizers in this project, serving as an adaptive learning rate optimization algorithm that combines the benefits of two other popular optimizers such as AdaGrad and RMSProp. Adam maintains adaptive learning rates for each parameter by calculating exponential moving averages of both first-order moments (i.e. mean) and second-order moments (uncentered variance) of gradient gradients.

Learning Rate Schedule

To ensure the training does not overfit, we use the Python function build_lrfn to generate a custom learning rate schedule using TensorFlow for training deep learning models. This schedule includes three stages.

- Ramp-up Period (rampup_epochs): Gradually increases learning rate from lr_start to lr_max over an arbitrary number of rampup_epochs.
- Sustain Period (sustain_epochs) maintains it at maximum for specified time-period(s).
- Exponential Decay: After the sustain period, exponential decay reduces the learning rate from its maximum towards the minimum, with its rate being determined by lr_exp_decay.

TensorFlow's LearningRateScheduler callback utilizes this function, which takes an epoch number as input and returns its learning rate for that epoch. lrfn acts as the actual learning rate function that returns the rate at each start of the training epoch.

Here is an outline of the parameters and their function in creating a schedule:

- Lr_start: The initial learning rate at the start of training (start of ramp-up).
- Lr_max: The maximum learning rate reached after the ramp-up period has ended.
- Lr_min: Lower bound learning rate after decay has occurred.
- Lr_rampup_epochs: The number of epochs over which the learning rate linearly ramps up to its maximum value of Lr_max.
- Lr_sustain_epochs: The number of epochs during which Lr_max remains steady throughout.

- lr_exp_decay: This factor determines how often learning rate multipliers are applied during ramp-up and sustain periods (exponential decay rate).

As soon as we fed the LearningRateScheduler's output lr_schedule into TensorFlow's model training's callbacks parameter of the fit method, TensorFlow automatically adjusted learning rates according to lrfn function for every epoch.

Activation Function

The ReLU (Rectified Linear Unit) activation function is one of the more frequently employed activation functions. ReLU is defined as f(x) = max(0,x), where x represents the input to neurons. ReLU's piecewise linear function returns positive input values (x >= 0) while negative ones return nothing (x = 0).

ReLU activation function offers many advantages for neural network training. First, its computational efficiency makes it ideal when training deep neural networks with many layers, especially with many layers. Second, ReLU helps alleviate the vanishing gradient problem by allowing gradient flow during backpropagation, leading to effective learning at deeper layers. Thirdly, ReLU introduces sparsity into networks by setting negative values to zero; this sparsity can assist regularization efforts as it reduces overfitting for improved generalization performance.

ReLU may have been used, among other activation functions, in this project; examples include sigmoid, tanh, and Leaky ReLU. The former two functions are frequently seen in earlier neural network architectures for mapping input values between 0-1; for tanh, this range extends between -1 and 1. Leaky ReLU is a variation on ReLU that introduces a small negative slope when negative inputs arrive which helps avoid "dying ReLU" syndrome where neurons stop responding to input.

The choice of activation function depends on the specific characteristics of a problem, network architecture, and desired properties of a model. Experimentation with various activation functions is frequently conducted to ascertain which yields the best performance and convergence results for model performance and convergence.

This research project seeks to select an activation function that introduces nonlinearity into its model, facilitates effective learning of complex patterns, eliminates the vanishing gradient problem, and facilitates regularization to facilitate improved generalization. Through such activation functions, models can capture intricate relationships within data for improved classification performance.

CHAPTER FOUR

RESULTS

Below table 1 shows the training progress of the deep learning model over 100 epochs are summarized in the following table:

Table 1: Training Outputs

Epoch	Learning Rate	Loss	Binary Accuracy	AUC	Val Loss	Val Binary Accuracy	Val AUC
1	2.00e-06	0.4457	0.9074	0.6811	0.2258	0.9216	0.8008
2	1.425e-05	0.2382	0.9186	0.7870	0.2143	0.9263	0.8360
3	2.65e-05	0.2291	0.9220	0.8068	0.2077	0.9271	0.8528
4	3.875e-05	0.2243	0.9233	0.8179	0.2098	0.9267	0.8498
5	5.10e-05	0.2212	0.9239	0.8245	0.2083	0.9268	0.8526
6	6.325e-05	0.2181	0.9247	0.8314	0.2043	0.9287	0.8565
7	7.55e-05	0.2156	0.9253	0.8364	0.2067	0.9276	0.8558
8	8.775e-05	0.2135	0.9258	0.8415	0.2044	0.9281	0.8625
9	0.0001	0.2118	0.9264	0.8447	0.2060	0.9273	0.8581
10	8.00e-05	0.2099	0.9269	0.8496	0.1994	0.9301	0.8662
11	6.40e-05	0.2079	0.9275	0.8534	0.2028	0.9287	0.8659
12	5.12e-05	0.2077	0.9274	0.8546	0.2011	0.9295	0.8673
13	4.096e-05	0.2069	0.9278	0.8557	0.1993	0.9297	0.8685
14	3.2768e-05	0.2059	0.9281	0.8578	0.2006	0.9294	0.8690
15	2.62144e-05	0.2058	0.9278	0.8590	0.1986	0.9302	0.8707
16	2.097152e-05	0.2052	0.9281	0.8594	0.1994	0.9291	0.8704
17	1.677721e-05	0.2052	0. 9279	0.8603	0.2015	0.9289	0.8675

This Table 1 shows that how throughout the training process, Training Losses have been steadily decreasing over time, signaling that the model is learning. Binary accuracy remains stable around 92% while AUC is steadily rising. Validation loss start out at 0.2258 and decreases gradually to 0.1994 by the 17th epoch. Validation binary accuracy remains relatively constant between 92.1% to 93% while Validation AUC increases from 0.8008 to 0.8704.

Binary Accuracy

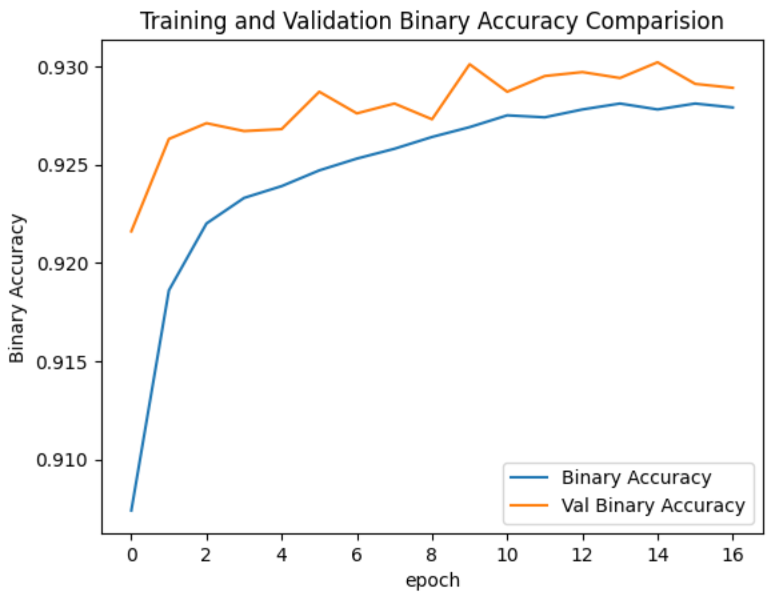

Figure 6: Training and Validation Binary Accuracy

This metric indicates the proportion of correct predictions the model makes on the training dataset. The values are between 0 and 1, with 1 indicating perfect accuracy. In the table, the binary accuracy starts at 0.9074 in epoch 1 and generally increases over time, reaching 0.9281 by epoch 16. This suggests that as the model trains, it is getting better at correctly classifying the training data. The validation binary accuracy also starts lower at 0.9216 in epoch 1 and increases over time, peaking at 0.9302 in epoch 15. It's a measure of the model's generalization; an increasing trend here is a good sign that the model is not just memorizing the training data, but rather learning general patterns that apply to new data as well.

The close alignment of binary accuracy and validation binary accuracy is a positive sign, indicating that the model is likely generalizing well and not overfitting significantly to the training data (where overfitting is when a model performs well on training data but poorly on new, unseen data). Overfitting would be suggested if the training accuracy continued to improve while the validation accuracy started to decrease or stagnate.

AUC

This is a performance measurement for classification problems at various threshold settings. The AUC represents the likelihood that the model will rank a randomly chosen positive instance higher than a randomly chosen negative one. For binary classification, an AUC of 0.5 suggests no discriminative power (equivalent to random guessing), while an AUC of 1.0 indicates perfect discrimination. In this table, the AUC starts at 0.6811 and consistently rises to 0.8603 by epoch 17 on the training data. This increase means the model's ability to discriminate between the classes is improving as training progresses. While The Val AUC starts at 0.8008 and, like the training AUC, increases throughout the epochs, reaching 0.8675 by epoch 17. The AUC and Val AUC are particularly important metrics when dealing with imbalanced datasets or when the costs of false positives and false negatives are different. A model that produces a higher

AUC value is generally considered better at predicting true positives while minimizing false positives.

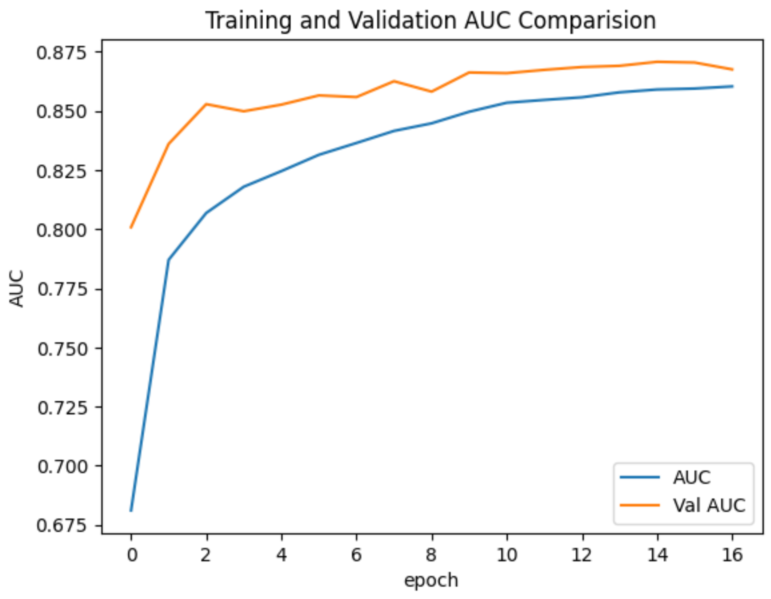

Figure 7: Training and Validation AUC

Both the AUC and Val AUC in the table show an upward trend, indicating that the model's predictive performance is improving over time. It is also good to observe that the AUC and Val AUC are quite close to each other across epochs, which suggests that the model is generalizing well and not just fitting to the noise in the training data. A consistent gap between AUC and Val AUC could indicate overfitting, but that does not appear to be the case with the provided data.

Loss

This is a measure of how well the model is performing on the training dataset. It is typically calculated using a loss function, which quantifies the difference between the predicted values and the actual values. The goal of the training is to minimize this loss. In the table provided, the loss starts at 0.4457 in the first epoch and consistently decreases to 0.2052 by epoch 17. This decrease in loss indicates that the model is getting better at making predictions on the training data, as it's learning the patterns and reducing errors over time. In the table, the validation loss starts at 0.2258 and, with some fluctuations, generally decreases to 0.1986 by epoch 15 before slightly increasing to 0.2015 in epoch 17. The trend in the validation loss provides insights into whether the model is generalizing well. Ideally, both the training loss and validation loss decreases together, and by the final epoch, they are relatively low and close to each other, which would indicate good model performance without overfitting.

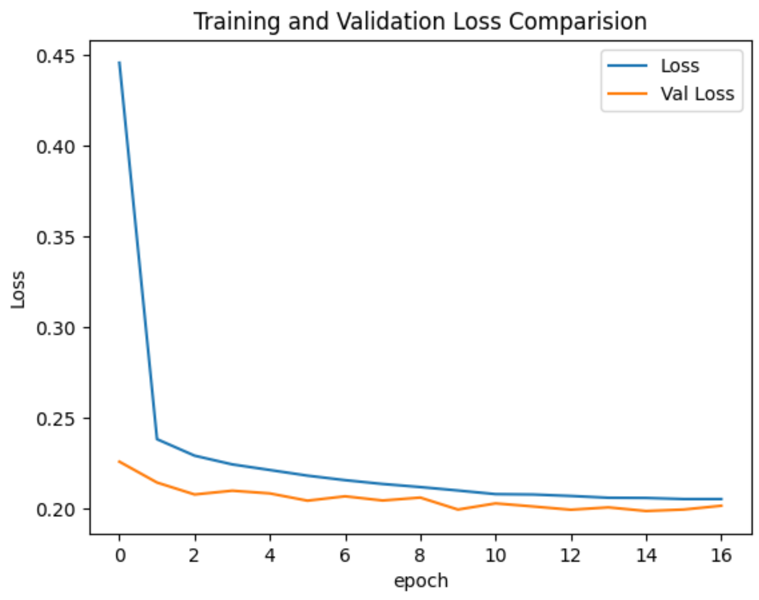

Figure 8: Training and Validation loss.

Overall, the results indicate that the deep learning model performs well in the binary classification task, demonstrating consistent improvement in both loss and accuracy metrics over the course of training. The increasing AUC values further support the model's effectiveness in distinguishing between positive and negative instances.

ROC Curve

Receivers Operating Characteristic (ROC) curves are produced for multiple classes when undertaking binary classification tasks. Below, FPR (false positive rate), TPR (true positive rate), and threshold values are presented for each class: each ROC curve represents the performance of the model for a specific class; FPR represents the ratio of false positives to actual negatives while TPR shows the ratio of true positives to actual positives; threshold values indicate probability threshold above which an instance can be classified as positive.

ROC curves offer valuable insights into a model's performance at different threshold values for each class. By altering these threshold values, its performance can be optimized based on specific requirements such as prioritizing high recall or minimizing false positives.

Figure 9 shows the ROC Curve depicted below; Edema (0.90) and Pneumonia (0.81) both boast high AUC scores, suggesting that this model excels at accurately recognizing these conditions; conversely, lower AUC scores such as Infiltration (0.66) or Fibrosis (0.65) may not be identified by this approach as reliably. The model's AUC for "No Finding" indicates its ability to accurately detect an absence of findings.

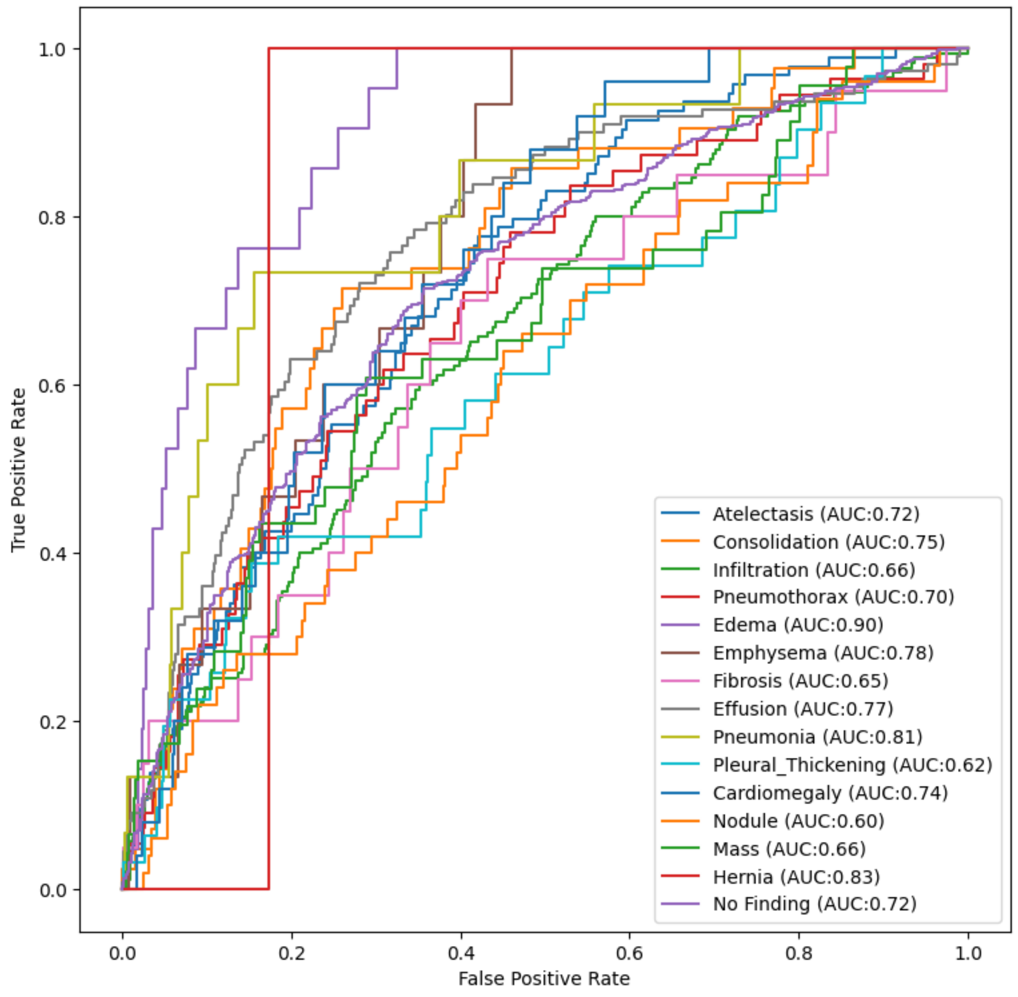

Figure 9: ROC curve for each metric

Table 2 describes that, In most of the cases our model has improved accuracy as compared to average AUC value given by author of this dataset Wang et al.[1] while in other paper Yao et al.[2] have consistently demonstrated high AUC values across various pathologies, demonstrating their model Our Model's robust performance across various conditions such as Edema and

Hernia. Specifically, its AUC values fluctuate among pathologies, surpassing other models about Edema and Hernia diagnosis.

Table 2: AUC-Roc Curve Value Comparison

Pathology	**Wang et al.[1]**	**Yao et al.[2]**	**Our Model**
Atelectasis	0.716	0.772	0.72
Cardiomegaly	0.807	0.904	0.74
Effusion	0.784	0.859	0.77
Infiltration	0.609	0.695	0.66
Mass	0.706	0.792	0.66
Nodule	0.671	0.717	0.60
Pneumonia	0.633	0.713	0.81
Pneumothorax	0.806	0.841	0.70
Consolidation	0.708	0.788	0.75
Edema	0.835	0.882	0.90
Emphysema	0.815	0.829	0.78
Fibrosis	0.769	0.767	0.65
Pleural Thickening	0.708	0.765	0.62
Hernia	0.767	0.914	0.83
No Finding	--	0.76	0.72

CHAPTER FIVE

CONCLUSION

In this study, we have created a classification model using UNet architecture with an embedded classification layer. Our experiments have revealed that this approach proved superior when classifying different thoracic pathologies as well as health findings.

UNet architecture has enabled this model to efficiently extract both local and global features from input images. Encoder networks efficiently extract high-level features by downsampling while decoder networks upsampled spatial information; this allows for greater feature representation as well as improved differentiation among various thoracic pathologies.

Adam optimizes our model during training. Adam combines the benefits of both AdaGrad and RMSProp optimizers by adapting learning rates for parameters based on past gradients; this optimization algorithm helps to accelerate convergence while simultaneously improving model performance during development.

To increase the generalization ability of our model, we use data augmentation techniques such as rotation, horizontal flip, and zoom during training to enlarge diversity in training data sets and teach robust features to the model.

The model has trained using a cross-entropy loss function, which penalizes any differences between its predicted probabilities and actual ground truth labels. This loss function has enabled rapid learning processes while encouraging precise classification.

Overall, our experiments demonstrate the efficiency of UNet architecture combined with a classification layer in classifying thoracic pathologies on chest X-ray images. This model achieves promising results, which shows its promise as an aid for medical professionals in diagnosing and identifying various thoracic pathologies accurately.

Future projects may focus on further optimizing the model by exploring different architectures, fine-tuning hyperparameters, and including additional clinical information to increase accuracy and interpretability. Furthermore, evaluating its performance on larger datasets can give more insights into its practical use and generalizability.

REFERENCES

[1] Wang, X., Peng, Y., Lu, L., Lu, Z., Bagheri, M., & Summers, R. M. (2017). ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 2097-2106) arXiv:1705.02315.

[2] Li Yao, Eric Poblenz, Dmitry Dagunts, Ben Covington, Devon Bernard, Kevin Lyman.Learning to diagnose from scratch by exploiting dependencies among labels arXiv:1710.10501

[3] Rajpurkar, P., Irvin, J., Zhu, K., Yang, B., Mehta, H., Duan, T., ... & Ball, R. L. (2017). CheXNet: Radiologist-level pneumonia detection on chest X-rays with deep learning. arXiv:1711.05225

[4] Ronneberger, O., Fischer, P., & Brox, T. (2015, October). U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and computer-assisted intervention (pp. 234-241). Springer,Cham. arXiv:1505.04597

[5] Siddique, Nahian, et al. “U-Net and its variants for medical image segmentation: theory and applications.” arXiv:2011.01118

[6] Khanna, A.; Londhe, N.D.; Gupta, S.; Semwal, A. A deep Residual U-Net convolutional neural network for automated lung segmentation in computed tomography images. Biocybern. Biomed. Eng. 2020, 40, 1314–1327, https://doi.org/10.1016/j.bbe.2020.07.007.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90

[8] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv:1412.6980.

[9] Shorten and Khoshgoftaar J Big Data (2019) 6:60 https://doi.org/10.1186/s40537-019-0197-0.

[10] Prechelt, L. (1998). Early stopping—but when? In Neural Networks: Tricks of the trade (pp. 55-69). Springer, Berlin, Heidelberg.