DESIGN AND IMPLEMENTATION OF A VISION TRANSFORMER FOR PNEUMONIA DETECTION

A PROJECT REPORT

Submitted by

N S S PAVAN KUMAR [RA2111003010891]

T VIKASH PATRA [RA2111003010894]

Under the Guidance of

Mrs. VETRISELI D

Assistant Professor,

Department of Computing Technologies

in partial fulfillment of the requirementsfor the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE ENGINEERING

DEPARTMENT OF COMPUTING TECHNOLOGIES,

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE ANDTECHNOLOGY

KATTANKULATHUR- 603 203

NOVEMBER 2024

Department of Computational Intelligence
SRM Institute of Science & Technology
Declaration Form

Degree/ Course : B.Tech, Computer Science Engineering

Student Name : NSS Pavan Kumar, T Vikash Patra

Registration Number : RA2111003010891, RA2111003010894

Title of Work : Design and Implementation of a Vision Transformer for Pneumonia Detection.

I / We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism**, as listed in the University Website, Regulations, and the Education Committee guidelines.

I / We confirm that all the work contained in this assessment is my / our own except where indicated, and that I / We have met the following conditions:

- Clearly referenced / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g.fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook /University website

I understand that any false claim for this work will be penalized in accordance with theUniversity policies and regulations.

DECLARATION:
I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above. Student 1 : Student 2 :

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY, KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that 18CSP107L - Minor Project report titled "**Design and Implementation of a Vision Transformer for pneumonia detection** " is the bonafide work of "**NSS Pavan Kumar [RA2111003010894], T Vikash Patra [RA2111003010894]"** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE
Mrs. Vetriselvi D
Assistant Professor
Department of Computing Technology

SIGNATURE
Dr. Niranjana G
Head of the Department
Department of Computing Technology

ACKNOWLEDGEMENTS

We express our humble gratitude to **Dr. C. Muthamizhchelvan**, Vice-Chancellor, SRM InstituteofScience and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to **Dr. T. V. Gopal ,** Dean-CET, SRM Institute of Science and Technology, for his invaluable support.

We wish to thank **Dr. Revathi Venkataraman**, Professor and Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work.

We encompass our sincere thanks to , **Dr. M. Pushpalatha**, Professor and Associate Chairperson, School of Computing and **Dr. C.Lakshmi,** Professor and Associate Chairperson, School of Computing, SRM Institute of Science and Technology, for their invaluable support.

We are incredibly grateful to our Head of the Department, **Dr. Niranjana G**, Professor, Department of Computing Technology, SRM Institute of Science and Technology, for her suggestions and encouragement at all stages of the project work.

We want to convey our thanks to our Project Coordinators **Dr.R.Vidhya**, Assistant Professor, **Dr.M.Arul Prakash**, Assistant Professor, **Dr.M.Revathi**, Assistant Professor, **Dr M Baskar**, Panel Head, **Dr P Renuka Devi** and **Mrs Vetriselvi D**, Panel Members, Department of Computing Technologies, SRM Institute of Science and Technology, for their input during the project reviews and support.

We register our immeasurable thanks to our Faculty Advisor, **Dr Sindhu C**, Department of Computing Technologies, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to our guide, **Mrs Vetriselvi D**, Department of Computing Technologies, SRM Institute of Science and Technology, for providing us with an opportunity to pursue our project under his / her mentorship. He / She provided us with the freedom and support to explore the research topics of our interest. His / Her passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank all the staff and students of Computational Intelligence, School of Computing, S.R.M Institute of Science and Technology, for their help during our project. Finally, we would like to thank our parents, family members, and friends for their unconditional love, constant support and encouragement

NSS Pavan Kumar [RA2111003010891]
T Vikash Patra [RA2111003010894]

ABSTRACT

Pneumonia continues to be one of the most costly diseases that affect the population, and it is also one of the most common causes of death. However, in order to reduce the number of deaths and improve the prognosis of patients, it is imperative that they receive a diagnosis as soon as possible. During conventional diagnostic procedures, such as a physical examination followed by a chest x-ray, the interpretation of these pictures is typically done manually. As a result, it is prone to errors and quite slow. This is especially true in locations where there are a limited number of radiologists or where they are unable to afford. In the past several years, a substantial amount of attention has been directed into deep learning-based algorithms, particularly Convolutional Neural Networks (CNN), for the purpose of automatically detecting pneumonia from medical images.

The results of this endeavour have been noted as being positive. CNN models, on the other hand, have a restriction in that, because of their local receptive fields, it is impossible for them to capture long-range dependencies in the image. Chest X-ray pictures are used in this research to demonstrate a unique application of Vision Transformer (ViT), which was initially developed as a model for natural image categorisation. The purpose of this application is to diagnose pneumonia. Vision transformer (ViT) architecture makes primary use of the self-attention mechanism, which assists in the context-aware representation of aspects of the image. This is due to the fact that in medical image analysis, irregular patterns are frequently dispersed across the image and are located quite a distance apart from one another. We trained a ViT model on a public cough chest X-ray database in order to accomplish this goal while working on this project.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Pneumonia is a life-threatening infection that primarily affects the lungs, causing inflammation of the air sacs. It is a leading cause of morbidity and mortality globally, particularly among vulnerable populations such as young children, the elderly, and immunocompromised individuals. According to the World Health Organization (WHO), pneumonia accounts for approximately 14% of all deaths in children under the age of five, making it a critical global health issue. Timely and accurate diagnosis is vital for effective treatment, but in many parts of the world, the lack of trained healthcare professionals and diagnostic tools hinders early detection.

Traditional pneumonia diagnosis relies heavily on chest X-rays, which require expert interpretation by radiologists. However, in low-resource settings, the availability of radiologists is limited, often leading to delayed diagnosis and treatment. This problem is further compounded in overburdened healthcare systems, where misdiagnosis or late diagnosis can have fatal consequences. Artificial intelligence (AI), specifically in the form of deep learning models, offers a solution by automating the diagnostic process, making it faster, more accessible, and potentially more accurate.

This project focuses on developing a Vision Transformer (ViT) model to automate the detection of pneumonia from chest X-ray images. Vision Transformers represent a novel approach in the field of computer vision, as they deviate from the traditional convolutional neural network (CNN) architectures that have been widely used for image classification tasks. By utilizing a self-attention mechanism, Vision Transformers can capture long-range dependencies in images more effectively than CNNs, making them suitable for complex medical imaging tasks like pneumonia detection.

This project aims to contribute to the ongoing efforts in AI-driven healthcare solutions by offering an accurate, scalable, and accessible tool for pneumonia detection. The use of Vision Transformers represents a significant advancement in medical diagnostics, providing healthcare professionals with a reliable tool that can improve patient outcomes and reduce diagnostic delays, especially in resource-constrained settings.

1.1 General

The primary objective of this project is to develop a deep learning model that can automatically detect pneumonia from chest X-ray images. Pneumonia detection is a crucial task in medical diagnostics, as it allows for timely intervention and treatment. The current reliance on radiologists for interpreting X-rays poses challenges, particularly in rural or underdeveloped regions, where access to trained medical professionals is limited.

Vision Transformers offer a new approach to image analysis by replacing the convolutional layers in traditional CNN models with a self-attention mechanism. This allows the model to focus on different parts of the image simultaneously, capturing both global and local patterns. The Vision Transformer model processes an entire image as a sequence of patches, similar to how transformers process words in natural language processing tasks. This structure enables the model to understand the relationships between different regions of the image, which is particularly important for medical imaging, where subtle details can influence a diagnosis.

This project utilizes a publicly available dataset of labeled chest X-ray images, with both healthy cases and pneumonia-positive cases. The dataset is preprocessed to ensure consistency in image size and quality, and then fed into the Vision Transformer model for training. The model is trained to distinguish between normal chest X-rays and those showing signs of pneumonia. Throughout the training process, various techniques such as data augmentation and hyperparameter tuning are applied to optimize the model's performance.

Upon completion, the Vision Transformer model is evaluated on a test set, with its performance measured in terms of accuracy, precision, recall, and F1 score. Achieving a high level of accuracy is crucial, as false negatives (missed pneumonia cases) could lead to dangerous delays in treatment, while false positives could result in unnecessary medical interventions.

The significance of this project lies in its potential to revolutionize the way pneumonia is diagnosed. By automating the diagnostic process, the Vision Transformer model can reduce the workload on radiologists and provide rapid, accurate diagnoses, even in areas where healthcare resources are scarce. Furthermore, the scalability of AI models makes them an ideal solution for global health challenges like pneumonia, as they can be deployed in hospitals, clinics, and even mobile health units**.**

1.2 Motivation

The motivation behind this project is multifaceted, encompassing both the personal and societal impacts of pneumonia and the advancements in AI technology that make this project feasible. Pneumonia is a disease that affects millions of people every year, with high mortality rates in low- and middle-income countries. Despite being treatable with antibiotics, pneumonia continues to claim lives, particularly when diagnosis is delayed or missed altogether. In many regions, access to diagnostic tools and skilled medical professionals is limited, leading to poor outcomes for patients.

As an aspiring computer scientist with a focus on AI and its applications in healthcare, I am driven by the potential of AI to address some of the world's most pressing health challenges. This project is motivated by the belief that technology can play a critical role in bridging the gap between healthcare needs and available resources. The development of an AI model that can assist in pneumonia detection aligns with the broader goals of improving healthcare accessibility and quality, particularly in underserved areas.

On a technical level, the motivation for using Vision Transformers stems from their recent success in a variety of image classification tasks. Transformers, originally developed for natural language processing, have shown great promise in computer vision by overcoming some of the limitations of traditional CNNs. While CNNs are highly effective at capturing local patterns in images, they struggle with understanding global context, especially in medical images where details in one region of the image may depend on features elsewhere in the image. Vision Transformers, by contrast, excel at capturing these long-range dependencies, making them an ideal choice for this project.

Additionally, the success of this project could open the door to further research in applying Vision Transformers to other medical imaging tasks, such as cancer detection or organ segmentation. The versatility of the transformer architecture, combined with the growing availability of medical image datasets, creates exciting possibilities for the future of AI in healthcare.

1.3 Sustainable Development Goal of the Project

This project aligns with the United Nations Sustainable Development Goal (SDG) 3: Good Health and Well-being. SDG 3 seeks to ensure healthy lives and promote well-being for people of all ages by addressing various health challenges, including communicable diseases like pneumonia. Specifically, this project contributes to Target 3.8 of SDG 3, which focuses on achieving universal health coverage and access to quality essential healthcare services.

Pneumonia is a preventable and treatable disease, yet it continues to be a leading cause of death in low-income countries due to a lack of access to timely and accurate diagnostic services. By developing an AI-based tool for pneumonia detection, this project has the potential to make high-quality diagnostic services more accessible, particularly in resource-poor settings. The scalability of AI models means that once developed, they can be deployed in various healthcare environments, from urban hospitals to rural clinics, without requiring extensive infrastructure or human expertise.

The project also addresses another important aspect of SDG 3: reducing the financial burden of healthcare. Traditional diagnostic methods, such as chest X-rays interpreted by radiologists, can be costly and time-consuming. In contrast, an AI-driven approach offers a cost-effective alternative that can be implemented at scale. This is particularly important for low- and middle-income countries, where healthcare resources are often stretched thin.

Moreover, this project contributes to global efforts to reduce child mortality, as pneumonia is one of the leading causes of death among children under five. By providing an accurate and efficient method of diagnosing pneumonia, this AI tool can help healthcare providers administer timely treatments, thereby reducing the risk of complications and fatalities.

In conclusion, the Vision Transformer model for pneumonia detection not only advances the field of medical imaging but also directly supports the goals of global health equity and accessibility. It offers a sustainable, scalable solution that can be deployed in diverse healthcare settings, ultimately contributing to the achievement of SDG 3 and improving health outcomes for millions of people worldwide**.**

1.4 Vision Transformer

The Vision Transformer (ViT) is a groundbreaking model architecture in computer vision that applies the transformer mechanism, widely successful in natural language processing, to image analysis tasks. Traditionally, image classification models relied on convolutional neural networks (CNNs), which are structured to extract spatial features from images by applying local filters across regions. Vision Transformers diverge from this approach by treating images as sequences, similar to how transformers process words in text. This shift enables the model to capture both local and global features across an entire image, an essential property for detailed image analysis tasks.

In a Vision Transformer, an image is divided into fixed-size patches, each of which is flattened into a vector, akin to a "token" in text processing. These vectors are then projected into embeddings of fixed size, allowing for consistent representation across the model. To retain spatial information, each embedding is augmented with a positional encoding, which gives the model a sense of where each patch is located within the larger image. This structured approach allows the Vision Transformer to process the image patches as a sequence, incorporating the spatial relationships that define the image.

Patch Extraction: Initially, the input image is segmented into non-overlapping patches, typically sized 16×16 pixels. Each of these patches is flattened into a one-dimensional vector. A linear transformation is then applied to these vectors to create a lower-dimensional representation, converting the 2D image data into a sequence of patch embeddings. This method enables the model to treat each patch as an individual token, analogous to words in a sentence.

Positional Encoding: To maintain the spatial context of the patches, positional encodings are added to each patch embedding. These encodings provide crucial information regarding the position of each patch within the original image, which is essential for the model to understand the layout of features.

Transformer Encoder Layers: The backbone of the ViT consists of multiple layers of Transformer encoders. Each encoder is composed of two main components: a multi-head self-attention mechanism and a feed-forward neural network. The self-attention mechanism allows the model to evaluate the relevance of different patches relative to each other, effectively capturing long-range dependencies. This ability stands in contrast to CNNs, which may overlook such global

relationships. The output from the self-attention layer is then processed by a feed-forward network, followed by normalization and residual connections to enhance training stability.

Classification Mechanism: The output from the final Transformer layer corresponding to a special [CLS] token, representing the entire image, is passed to a classification head. This head, typically a simple feed-forward network, generates the final class predictions based on the aggregated information from the patches.

Central to the ViT architecture is the self-attention mechanism, which assesses and adjusts the relevance of each patch relative to others. This mechanism allows the model to weigh certain patches more heavily, depending on the task, effectively prioritizing the most relevant areas of an image. For example, in medical imaging, this ability enables the model to focus on regions within a chest X-ray that may indicate abnormalities, which can be crucial for disease detection. Through the layers of the transformer encoder, which applies self-attention and feed-forward operations iteratively, the model learns complex relationships across patches, refining its understanding of the entire image structure.

One of the major benefits of Vision Transformers over CNNs is their ability to handle long-range dependencies within an image, making them well-suited for scenarios where critical information might be spread across multiple areas. This architecture provides a holistic view, allowing the model to capture global patterns that CNNs might miss due to their reliance on local convolutions. Vision Transformers have demonstrated state-of-the-art performance on large image datasets, such as ImageNet, especially when pre-trained on extensive data and fine-tuned for specific tasks.

However, ViTs come with their own challenges. Unlike CNNs, which benefit from inherent inductive biases such as translation invariance, Vision Transformers require large datasets for training to achieve optimal performance. The flexibility of transformers makes them highly expressive, but also data-hungry, as they rely less on these built-in assumptions. Furthermore, Vision Transformers are computationally intensive and require significant memory, particularly during training. Despite these challenges, Vision Transformers are proving to be an important innovation in computer vision, setting a new standard for tasks where both local detail and global context matter deeply.

CHAPTER 2

LITERATURE SURVEY

Literature surveys are a critical component of any research project, as it provides an understanding of the current state of knowledge, identifies research gaps, and sets the stage for the development of new methodologies or models. In this project, the literature survey focuses on the application of deep learning models for medical image analysis, specifically in the context of pneumonia detection using chest X-rays.

Existing research in the field of computer vision and medical image analysis has largely focused on convolutional neural networks (CNNs), which have shown significant success in image classification tasks. However, recent developments in Vision Transformers have opened up new possibilities for enhancing the performance of image recognition models, particularly in complex and high-stakes domains such as healthcare.

2.1 Deep Learning in Medical Image Analysis

Medical image analysis is one of the most promising applications of deep learning technologies. Over the past decade, numerous studies have demonstrated the potential of deep learning models, particularly CNNs, in diagnosing diseases from medical images such as X-rays, CT scans, and MRIs.

Several studies have specifically addressed the challenge of pneumonia detection from chest X-rays. For instance, Rajpurkar et al. (2017) developed a CNN model called CheXNet that was able to classify pneumonia with a performance that rivaled that of radiologists. Similarly, Wang et al. (2018) introduced a large dataset called ChestX-ray8, which has been widely used for training deep learning models for pneumonia detection.

CNNs have become the de facto standard in medical image analysis due to their ability to capture spatial hierarchies in images through convolutional layers. However, CNNs have certain limitations, such as their reliance on fixed-size filters, which may not be ideal for capturing long-range dependencies in complex images like chest X-rays. This challenge has led to the exploration

of alternative architectures, such as Vision Transformers, which can better handle global relationships in images.

In summary, while CNNs have made significant strides in medical image analysis, their limitations, particularly in terms of capturing global context, present an opportunity for exploring Vision Transformers as a more effective solution for tasks such as pneumonia detection.

2.2 Vision Transformers in Image Classification

Vision Transformers (ViTs) represent a novel approach to image classification that has gained attention due to their superior performance in several computer vision tasks. Unlike CNNs, which rely on convolutions to extract local features, Vision Transformers use a self-attention mechanism to process entire images as sequences of patches, allowing the model to capture both local and global patterns simultaneously.

Dosovitskiy et al. (2020) were among the first to propose the Vision Transformer architecture for image classification, demonstrating that transformers could achieve state-of-the-art performance on standard benchmarks such as ImageNet. The success of ViTs in general image classification tasks has inspired researchers to explore their potential in specialized domains, including medical image analysis.

Several recent studies have explored the application of ViTs in medical imaging. For example, Gao et al. (2021) applied ViTs to histopathology image classification, showing that they outperformed traditional CNN models in terms of both accuracy and interpretability. Similarly, Chen et al. (2021) proposed a hybrid CNN-ViT model for medical image segmentation, achieving impressive results on various datasets.

However, the application of ViTs in pneumonia detection from chest X-rays is still relatively underexplored. This project seeks to fill this gap by investigating the use of Vision Transformers for automating the detection of pneumonia, leveraging the self-attention mechanism to improve diagnostic accuracy**.**

2.3 Limitations Identified from Literature Survey (Research Gaps)

Through the examination of chest X-rays, a multitude of research that make use of deep learning techniques have been published over the course of the last ten years. These studies have allowed for the automatic detection of lung abnormalities, such as infections. An important study that was carried out by Rajpurkar and colleagues has stood out since it demonstrates the greater capabilities of deep learning models in diagnosing pneumonia from chest X-rays in comparison to other medical imaging approaches. It is important to note that these improvements in the field are significant, particularly in terms of addressing the difficulty of effectively identifying pneumonia. Pneumonia is an illness that is frequently confused with other lung-related infections, particularly in youngsters. In order to improve the accuracy of pneumonia diagnosis in paediatric patients, the research utilised a combination of seven different Convolutional Neural Network (CNN) models that had been pre-trained. Additionally, it revealed the power of CNN models to differentiate between normal, viral, and bacterial pneumonia X-ray pictures while simultaneously obtaining excellent levels of sensitivity and Area Under the Curve (AUC) values. The authors emphasised how important it is to diagnose pneumonia as quickly and accurately as possible, especially in paediatric groups that are particularly exposed to the condition. Through the implementation of MobileNet, they were able to obtain commendable levels of accuracy, recall, and F-score metrics through the implementation of a simplified deep learning solution.

The efficacy of the model, which is characterized by its short training length and low computational needs, marks a significant development in the process of alleviating the difficulties associated with the early detection and treatment of pneumonia. For the purpose of addressing the problem of restricted availability of annotated computed tomography (CT) images for the diagnosis of pneumonia, the research presented a novel three-tier optimization strategy. This method significantly enhances the accuracy of pneumonia detection by boosting the quality of the data that is utilized. It does this by utilizing CT data from one domain to enhance the performance of deep learning models in another domain that does not have labelled scans.

Reliable diagnosis of various types of pneumonia was achieved by the development of a novel deep learning algorithm called Pneumonia-Plus, which made use of CT scans. The program's exceptional diagnostic performance was demonstrated by the area under the curve (AUC) values. With regard to the classification of bacterial and viral pneumonia, this algorithm performed better

than two out of three radiologists, demonstrating its potential to reduce the number of incorrect diagnoses and to assist in the process of clinical decision-making.

The research also presented a deep convolutional neural network (DCNN) that was scalable and explainable. The purpose of the DCNN was to automate the diagnosis of pneumonia based on chest X-ray images. This highlights the significance of intelligent systems in overcoming the limitations of traditional human-assisted methods, such as the high cost and limited availability of expertise.As a result of rigorous evaluations across a wide range of performance metrics, the proposed deep convolutional neural network (DCNN) model demonstrates superior performance in comparison to the most advanced methods currently available. In particular, it excels in the extraction of features and the classification of images into normal and pneumonia categories.

The research made a substantial contribution to the detection of complex lung illnesses and established a comprehensive framework for the prediction of lung diseases, such as pneumonia and COVID-19, based on chest X-ray pictures. This framework encompassed image enhancement, precise region of interest extraction, robust feature extraction, and classification utilising advanced techniques like ensemble classifiers, Artificial Neural Networks (ANN), Support Vector Machines (SVM), K-Nearest Neighbours (KNN), and a specially designed deep learning architecture utilising Recurrent Neural Networks (RNN) with Long Short-Term Memory (LSTM).

2.4 Research Objectives

Based on the limitations identified in the literature survey, the following research objectives have been defined for this project:

1. Objective 1: To develop a Vision Transformer model for pneumonia detection from chest X-ray images, leveraging the self-attention mechanism to capture both local and global features.
2. Objective 2: To evaluate the performance of the Vision Transformer model in comparison to traditional CNN-based models, with a focus on accuracy, precision, recall, and F1 score.
3. Objective 3: To enhance the interpretability of the Vision Transformer model by incorporating explainability techniques such as attention heatmaps, which can help

visualize the regions of the chest X-ray that the model focuses on when making its predictions.

4. Objective 4: To create a scalable and efficient model that can be deployed in clinical settings, particularly in low-resource environments, where access to trained radiologists is limited.

2.5 Product Backlog (Key User Stories with Desired Outcomes)

The product backlog defines the key user stories and desired outcomes that drive the development of this project. Each user story represents a feature or functionality that the end users (clinicians, patients, etc.) will benefit from.

1. User Story 1: As a radiologist, I want to receive an automated diagnosis of pneumonia from chest X-rays so that I can focus on more complex cases.
    - Desired Outcome: An AI model that provides accurate pneumonia diagnoses with an explanation of the decision-making process, aiding radiologists in their workflow.
2. User Story 2: As a clinician in a rural area, I want access to a diagnostic tool that can work offline so that I can diagnose pneumonia even in areas with limited internet connectivity.
    - Desired Outcome: A lightweight model that can be deployed on edge devices, providing real-time diagnostic capabilities without the need for continuous internet access.
3. User Story 3: As a healthcare administrator, I want to monitor the performance of the AI model over time so that I can ensure its accuracy remains high across different patient populations.
    - Desired Outcome: A performance monitoring system that tracks the model's accuracy and other metrics in real-time, allowing for continuous improvement.
4. User Story 4: As a patient, I want to receive timely and accurate diagnoses so that I can start treatment as early as possible.

CHAPTER 3

PROPOSED METHODOLOGY

The proposed methodology in this project revolves around a structured, contextual approach to pneumonia detection using Vision Transformers (ViTs). This methodology outlines how older, more traditional image processing methods can be integrated into newer approaches with ViTs to develop a highly accurate, interpretable system for detecting pneumonia from chest X-rays. Each section focuses on a particular aspect of the system's design and development, ensuring a comprehensive approach that incorporates advancements in deep learning while addressing the contextual needs of medical diagnostics**.**

3.1 Accommodating Older Approach in the Newer Approach

The previous methods of pneumonia detection using convolutional neural networks (CNNs) provided a strong foundation for image-based disease diagnosis, emphasizing feature extraction through convolutional layers. These CNN-based methods were effective for identifying localized features, but they struggled to capture global contextual information in the images. In the proposed system, we integrate the strengths of traditional CNNs with the advanced architecture of Vision Transformers, which leverage self-attention mechanisms to understand both local and global patterns in X-ray images.

The integration of these older approaches with ViTs is accomplished through several steps. First, the pre-processing techniques traditionally used with CNNs, such as data augmentation and image normalization, are retained to ensure consistent data quality and variability. Next, while CNNs rely on fixed receptive fields for feature extraction, ViTs dynamically adjust their receptive fields using self-attention layers. This enhancement allows the system to recognize complex patterns that span larger portions of the image, capturing subtle indicators of pneumonia that CNNs may miss. By combining CNNs' focused feature extraction with ViTs' global attention, this hybrid approach is expected to improve detection accuracy while preserving interpretability, crucial for medical applications**.**

3.2 Approach for the Proposed System

The approach for the proposed system begins with data pre-processing and segmentation, essential steps to prepare the chest X-ray images for the Vision Transformer model. The process includes resizing images to 288x288 pixels, applying normalization, and augmenting the dataset to introduce variability. This setup ensures that the input data is optimized for the Vision Transformer's architecture, allowing it to focus on relevant image features without unnecessary noise.

Following data pre-processing, the ViT model is employed for training. The Vision Transformer is structured around self-attention mechanisms, allowing each part of the image to be processed with contextual awareness of other parts. Unlike CNNs, where each layer captures progressively complex patterns, ViTs use layers of self-attention to form a multi-dimensional understanding of the image as a whole. The model is trained using labeled data, where each X-ray image is annotated as either pneumonia-positive or pneumonia-negative. The self-attention layers enable the model to create meaningful representations, identifying both high-level (whole lung) and low-level (specific features) details associated with pneumonia. This contextual understanding is essential for handling the variability in X-ray images, as pneumonia symptoms can present differently across cases.

For the training process, hyperparameters like learning rate, batch size, and the number of attention heads in each transformer layer are optimized iteratively. The model undergoes continuous evaluation against a validation set, adjusting parameters to minimize loss and maximize accuracy. The ViT model's performance is tracked in real-time, and its ability to generalize across diverse images is assessed regularly. By fine-tuning hyperparameters and using adaptive learning techniques, the model is guided toward higher performance, reaching an accuracy of 98.12% on the validation set, as observed in our final results**.**

3.3 Designing the Proposed System

The design of the proposed pneumonia detection system centers on a modular architecture that includes a user interface, data processing pipeline, and model training and inference modules. The system is developed to provide a seamless experience for end-users while ensuring the backend's efficiency, accuracy, and scalability.

1. User Interface (UI): The UI is designed to be user-friendly, allowing users to upload chest X-ray images, initiate the pneumonia detection process, and view results easily. The UI also provides real-time feedback, such as notifications about processing status and the estimated time for results. Accessibility features are incorporated into the design to support users with varying levels of technical expertise, ensuring that medical professionals and patients alike can navigate the system effectively.

2. Data Processing Pipeline: This pipeline is designed to standardize and prepare input data before feeding it into the Vision Transformer model. It includes functions for resizing images, normalizing pixel values, and augmenting the data. The pipeline's architecture is optimized for high throughput and can handle a large volume of images, a critical factor for a system intended for medical diagnostics. The pipeline also includes error-handling mechanisms to manage unexpected input formats or corrupt images, ensuring that only high-quality data reaches the model.

3. Model Training and Inference Module: This module is where the Vision Transformer model is trained, validated, and deployed. During training, the system leverages cloud resources for faster processing, enabling iterative training cycles. Each iteration focuses on improving the model's performance, adjusting parameters, and assessing the model's performance on validation datasets. After training, the inference model is deployed within the same module, where it takes in pre-processed images and outputs a probability score, indicating the likelihood of pneumonia. The inference results are formatted and fed back to the UI for user review.

4. System Architecture: The overall architecture is cloud-based, allowing the system to scale as demand increases. The modular setup facilitates easy updates to each component without disrupting the entire system, an essential feature for ongoing improvements or model re-training. Additionally, the system includes monitoring tools that track model performance over time, alerting developers if a decline in accuracy is detected, and enabling quick re-training to maintain diagnostic reliability.

The design of the proposed system represents a convergence of modern deep learning techniques and practical, user-centric considerations.

CHAPTER 4

METHODOLOGY

The methodology section outlines the step-by-step process followed for developing and implementing a Vision Transformer (ViT) model for pneumonia detection. The project methodology integrates data preparation, model design, training, and evaluation. It ensures that each phase is well-documented, from acquiring and pre-processing the data to fine-tuning the model for optimal performance. The entire approach revolves around leveraging the capabilities of the Vision Transformer to accurately detect pneumonia in chest X-rays.

4.1. Data Collection and Pre-processing

The first step in any deep learning-based image classification task is acquiring a reliable dataset. For this project, the dataset used is a publicly available chest X-ray dataset, typically containing both pneumonia-positive and pneumonia-negative images. The quality and diversity of the data are essential for training a robust model.

- Data Augmentation: To avoid overfitting, data augmentation techniques are applied. These include random rotation, flipping, zooming, shifting, and brightness adjustment. Augmentation increases the diversity of the training data, helping the model generalize better to unseen images.
- Image Resizing: The images in the dataset are resized to fit the input requirement of the Vision Transformer model. For this project, each image is resized to a resolution of 288 x 288 pixels. This standardization ensures that the images are uniformly processed, which is essential for feeding them into the transformer model.
- Normalization: Each pixel value of the images is normalized to a common range, typically between 0 and 1 or -1 and 1. This normalization stabilizes the training process by reducing the magnitude of pixel values, ensuring smoother gradient updates during backpropagation.
- Splitting Data: The dataset is divided into three distinct sets:

- Training Set: Used to train the model. This set constitutes around 70-80% of the dataset.
- Validation Set: Used to fine-tune the model's hyperparameters. Typically around 10-15% of the data.
- Test Set: Used for final model evaluation, constituting the remaining 10-15%.

The pre-processing stage is critical as it prepares the raw data for ingestion into the Vision Transformer, ensuring that the model can process it effectively and learn relevant features.

4.2. Vision Transformer Model Design

The Vision Transformer (ViT) is a novel architecture for image classification that applies the Transformer model, which was originally designed for natural language processing, to vision tasks. In this project, the ViT model is specifically designed and trained for pneumonia detection from chest X-ray images.

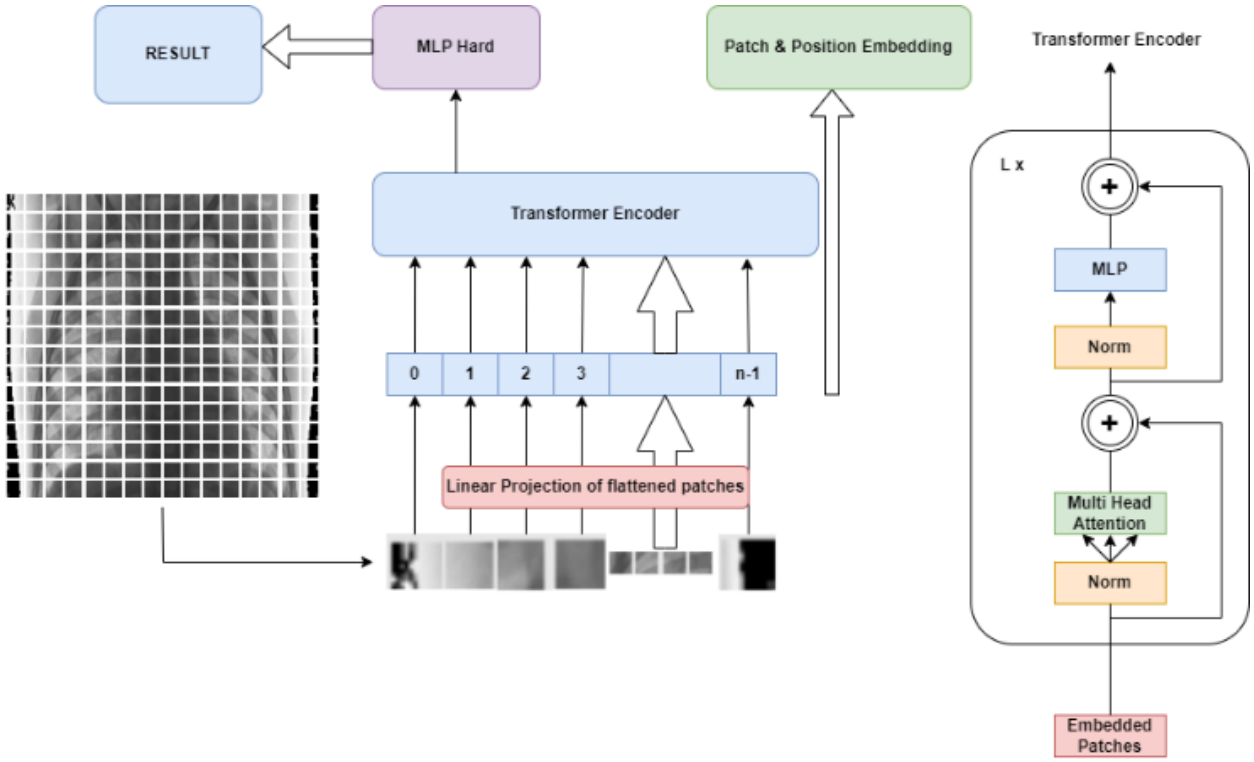


Fig 4.2: Architecture Diagram

4.2.1 Input Layer: Patch Embedding

Unlike traditional convolutional neural networks (CNNs) that process the entire image at once, the Vision Transformer divides the input image into patches. In this case, the 288 x 288 pixel X-ray images are split into smaller patches, typically of size 16 x 16 pixels.

- Each patch is flattened into a 1D vector and passed through a linear projection layer to create a fixed-dimensional embedding for each patch.
- Positional encodings are then added to these embeddings, which allow the model to maintain the spatial relationships between patches, something crucial for understanding medical images like X-rays**.**

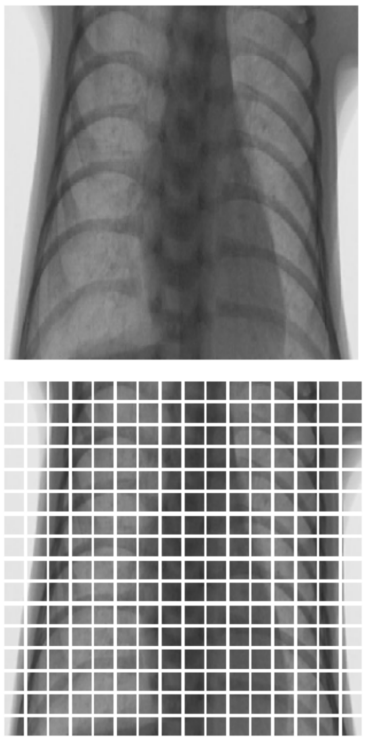


Fig 4.2.1: Patches

4.2.2 Transformer Encoder

The core of the Vision Transformer is its transformer encoder. The encoder consists of multiple layers, each containing two main components:

- Multi-Head Self-Attention: This mechanism allows the model to consider relationships between all patches simultaneously. For example, while looking at a patch of lung tissue, the model can "attend" to other patches in the image, such as patches around the heart or ribs, giving it a contextual understanding of pneumonia indicators.
- Feed-Forward Neural Network (FFNN): After attending to other patches, the patch embeddings are passed through a fully connected network to extract deeper, non-linear features.

These layers are repeated multiple times (denoted by L layers), and residual connections are applied to ensure information flows smoothly between layers.

The transformer encoder's ability to process the entire image as a series of patches helps it capture long-range dependencies that are crucial in medical diagnostics, where the manifestation of disease may span across large portions of the image**.**

4.2.3 Classification Layer

Once the transformer encoder has processed the patches, a classification token or pooled output is passed through a final classification head. This head consists of a fully connected layer that outputs the probability that the given chest X-ray image shows signs of pneumonia.

In the case of binary classification (pneumonia-positive or pneumonia-negative), a softmax activation function is used to output a probability distribution over the two classes. The model's output is a probability score that represents the likelihood of pneumonia being present in the image**.**

4.3. Model Training

After the ViT model is defined, it is trained on the pre-processed chest X-ray images. The training process involves iteratively adjusting the model's internal parameters to minimize the difference between the predicted and actual labels. This is done using a supervised learning approach with a labeled dataset.

4.3.1 Loss Function

The model uses categorical cross-entropy as its loss function. This loss function is commonly used for classification problems and measures the difference between the predicted probability distribution and the actual distribution of labels**.**

4.3.2 Optimizer

The Adam optimizer is used to minimize the loss function. Adam combines the advantages of two other popular optimizers: RMSProp and Stochastic Gradient Descent (SGD). It adapts the learning rate for each parameter, leading to faster and more stable convergence during training**.**

4.3.3 Batch Size and Epochs

The model is trained in batches, with a batch size of 32 images. Training is performed over multiple epochs, typically around 50-100, with early stopping criteria to prevent overfitting. If the validation loss does not improve over a certain number of epochs, the training process is halted**.**

4.3.4 Regularization and Fine-Tuning

To prevent overfitting, regularization techniques such as dropout are employed. Additionally, fine-tuning is done by adjusting hyperparameters like learning rate, attention heads, and hidden layer sizes to further improve the model's performance**.**

4.4. Evaluation Metrics

Once the model is trained, it is evaluated on the test set using several metrics to assess its performance:

- Accuracy: Measures the percentage of correctly classified images.
- Precision: Measures the proportion of true positives (correctly identified pneumonia cases) out of all positive predictions.

- Recall (Sensitivity): Measures the proportion of true positives out of all actual positive cases.
- F1-Score: A harmonic mean of precision and recall, providing a balanced metric for performance evaluation.

For this project, the model achieved an accuracy of 98.12%, indicating a high level of performance in identifying pneumonia from X-ray images**.**

4.5. Model Deployment

After achieving satisfactory results during training and testing, the Vision Transformer model is deployed into a clinical setting. The deployment process involves:

- Integration with a User Interface (UI): A user-friendly UI is developed that allows medical professionals to upload chest X-ray images and receive diagnostic results in real-time. The UI also provides confidence scores to help clinicians assess the model's prediction.
- Cloud Deployment: The model is deployed on a cloud platform to handle multiple requests simultaneously. This allows the system to scale and process a large volume of X-ray images in parallel, making it suitable for hospital use.
- Real-Time Inference: The deployed model is optimized for real-time inference, ensuring that predictions are delivered quickly, which is crucial in medical settings where timely decisions can be lifesaving**.**

CHAPTER 5

REQUIREMENTS

The requirements section provides an in-depth overview of the necessary resources, tools, and systems needed to develop and deploy the Vision Transformer (ViT) model for pneumonia detection. This project encompasses a blend of hardware, software, and dataset needs, each crucial for the successful implementation and execution of the model. By clearly identifying these requirements, we ensure that all components of the system are well-prepared, enabling a smooth development and deployment process.

5.1. Hardware Requirements

For deep learning tasks, particularly when working with transformer architectures like the Vision Transformer, computational power is crucial. The following hardware specifications are necessary to handle the extensive computation and large datasets efficiently:

5.1.1 High-Performance GPU

- A high-performance Graphics Processing Unit (GPU) is essential for training the Vision Transformer model. GPU acceleration significantly speeds up the matrix operations involved in backpropagation and gradient calculations.
- **Recommended GPUs**:
    - NVIDIA Tesla V100, A100, or any CUDA-compatible GPU.
    - If local resources are not sufficient, cloud-based GPUs (AWS EC2 P3 or Google Cloud TPUs) can be used for remote training.

5.1.2 CPU

- For tasks such as data pre-processing, loading datasets, and running initial experiments, a multi-core CPU is sufficient. However, the bulk of the computation will be offloaded to the GPU.
- **Recommended CPU**:
    - Intel Core i7 or AMD Ryzen 7 (or higher) for local machine processing.

5.1.3 RAM

- Adequate memory is required to handle large datasets and model parameters, especially during training and testing phases.
- **Recommended RAM**:
    - 32 GB of RAM or more, depending on the size of the dataset and batch processing.

5.1.4 Storage

- Chest X-ray datasets, especially when using high-resolution images, can take up considerable storage space. Additionally, storing model checkpoints, logs, and other artifacts requires significant storage.
- **Recommended Storage**:
    - A minimum of 1 TB SSD to store datasets, trained models, and logs efficiently.

5.1.5 Network Connectivity

- Reliable internet connectivity is necessary, particularly for downloading datasets, using cloud-based services, or accessing remote servers for GPU usage.
- **Recommended Network**:
    - High-speed internet with stable bandwidth for seamless cloud and data access.

5.2. Software Requirements

Software tools, libraries, and frameworks are key components in developing and training deep learning models. This project relies on a combination of open-source libraries and proprietary tools to streamline the process.

5.2.1 Operating System

- The Vision Transformer model can be developed on various operating systems, but some frameworks and tools are optimized for specific environments.
- **Recommended OS**:
    - Ubuntu 20.04 LTS or Windows 10/11 (64-bit), depending on developer preferences. However, Ubuntu is generally preferred for deep learning tasks because of its compatibility with TensorFlow and PyTorch.

5.2.2 Deep Learning Frameworks

- **TensorFlow/Keras**: The Vision Transformer model will be implemented using TensorFlow/Keras. TensorFlow provides robust support for handling large datasets, managing computational graphs, and efficiently utilizing GPUs.
- **PyTorch**: Alternatively, PyTorch can also be used due to its dynamic computational graph and ease of experimentation, especially for research purposes.
- Both frameworks support model checkpoints, debugging, and visualization.

5.2.3 Libraries and Dependencies

- The project requires several Python libraries for pre-processing, training, and evaluation. These libraries simplify tasks such as image processing, data augmentation, and model evaluation.
    - **NumPy**: For numerical operations, such as matrix transformations.
    - **Pandas**: For handling and processing tabular data, such as medical records or metadata.

- **Matplotlib/Seaborn**: For plotting and visualizing model performance, accuracy, and loss graphs.
- **OpenCV**: For advanced image processing tasks, such as resizing, augmenting, and converting images.
- **Scikit-learn**: For implementing traditional machine learning methods, calculating evaluation metrics (precision, recall, F1-score), and performing statistical analysis.
- **Albumentations**: For augmenting image data with transformations like flipping, rotating, and adjusting brightness or contrast.

5.2.4 Version Control

- A version control system is essential for tracking code changes, especially when collaborating with team members.
  - **Git**: The project will be version-controlled using Git, and repositories will be hosted on **GitHub** for remote access and collaboration.
  - **Git LFS (Large File Storage)**: Required for handling large datasets or models stored in the repository.

5.2.5 Integrated Development Environment (IDE)

- An IDE with support for Python, deep learning libraries, and version control is required for efficient coding, debugging, and testing.
  - **Recommended IDEs**: PyCharm, VS Code, or Jupyter Notebook, Google colab, depending on the user's preference.

5.3. Dataset Requirements

This project requires access to high-quality chest X-ray images labeled with pneumonia-positive and pneumonia-negative samples. These datasets must be properly formatted and accessible for use in model training and validation.

5.3.1 Source

- The **Kaggle Pneumonia Chest X-ray Dataset** or are commonly used public datasets containing thousands of labeled images of pneumonia-positive and healthy cases.
- The images should be of high resolution and diverse enough to train the model on a variety of pneumonia manifestations.

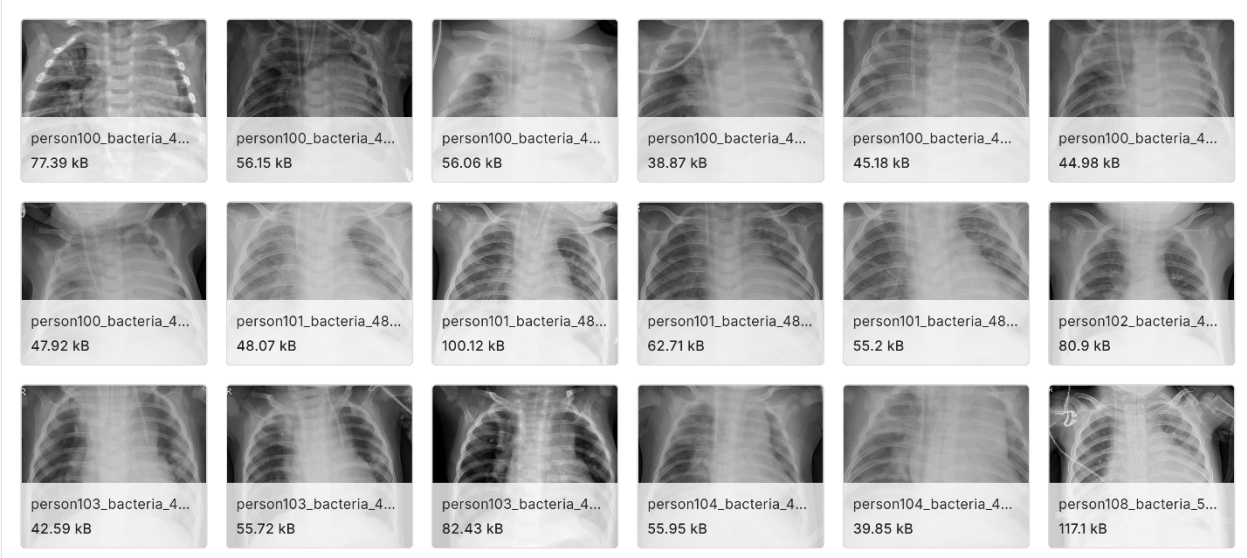


Fig 5.3: Dataset

CHAPTER 6

SYSTEM DESIGN

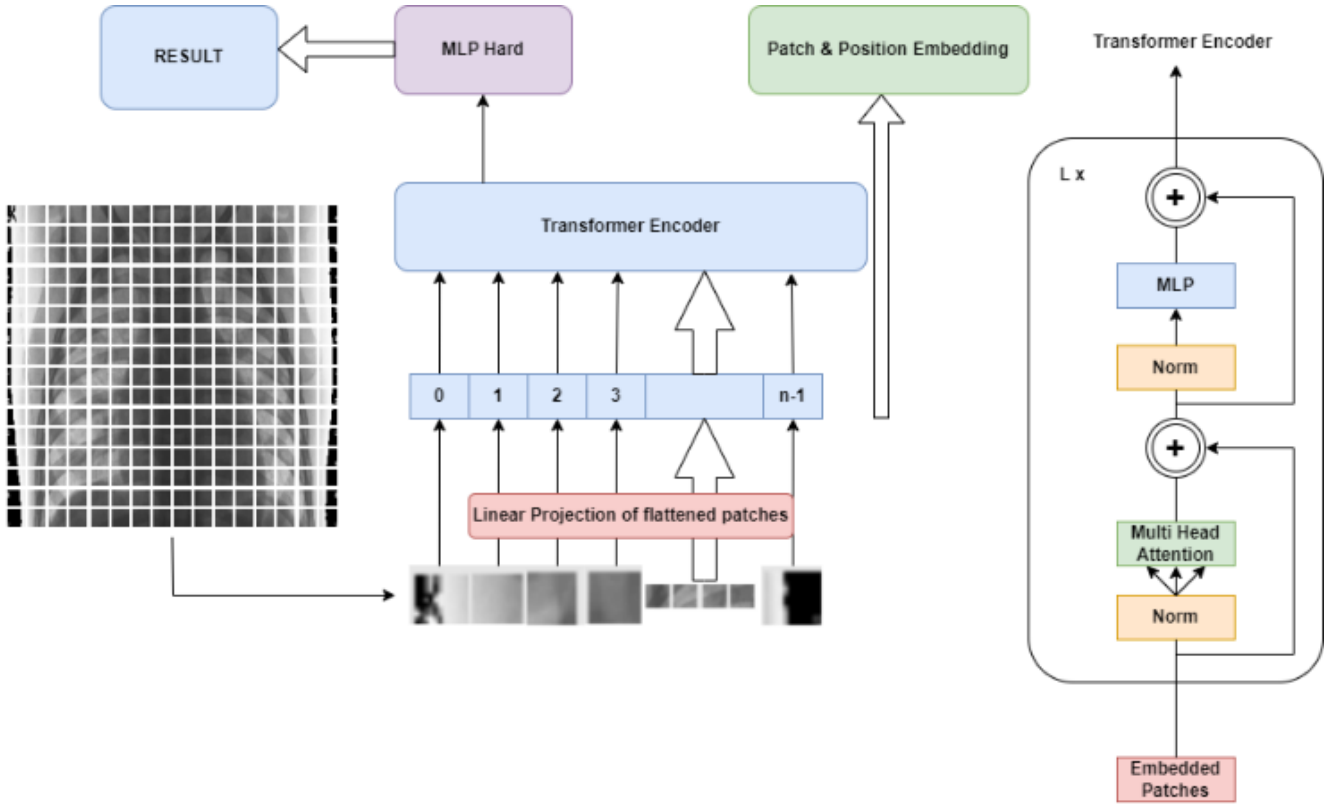


Fig 6.1: Architecture Diagram

6.1. Input Image and Patches:

- The image is first split into a grid of smaller, equally sized patches. In this example, the input X-ray image is divided into several patches, each of which is flattened into a 1D vector.
- **Flattening**: Each image patch (e.g., the pixel grid of each square) is converted into a sequence of pixel values to represent it as a 1D vector. This conversion allows it to be processed by the subsequent layers.
- The patches are typically of size 16x16 or similar, but it varies based on model configuration.

6.2. Linear Projection of Flattened Patches

- The flattened 1D vectors from each image patch are passed through a **linear projection layer**. This step involves multiplying the flattened vector by a learnable weight matrix to create a fixed-dimensional embedding for each patch.
- This transformation is important to prepare the patches for processing by the transformer architecture, as the transformer expects sequences of embeddings rather than raw pixel data.

6.3. Patch and Position Embedding

- To retain the spatial information of the image (which could otherwise be lost when splitting into patches), the model adds **position embeddings** to each patch embedding. This allows the transformer to know the relative position of each patch in the original image.
- The position embeddings are learnable vectors that are added to each patch's embedding to encode where the patch was located in the image.

6.4. Transformer Encoder

- The core of the Vision Transformer architecture is the **Transformer Encoder**. It processes the sequence of patch embeddings (along with the position embeddings) through multiple layers.
- Each encoder layer has two main components:
    - **Multi-Head Self-Attention**: This mechanism allows each patch to attend to all other patches, capturing both local and global dependencies in the image. Each head focuses on different parts of the image simultaneously, improving the model's ability to capture complex patterns.
    - **Normalization (Norm)** and **MLP (Multi-Layer Perceptron)**: After attention, each patch goes through a normalization layer to stabilize the training process, followed by an MLP, which is a fully connected feed-forward neural network. This enhances the representation of each patch by adding complexity and non-linearity.

- **Residual Connections** (denoted by + symbols) are used to ensure that information from earlier layers flows smoothly to later layers, preventing issues like vanishing gradients during training.

6.5. Result (Classification Head - MLP Hard)

- After the Transformer Encoder processes the patches, the output embeddings from all the patches are aggregated, typically by taking the output of a special “classification token” or pooling the patch outputs.
- These aggregated embeddings are passed through a final **classification head** (denoted as "MLP Hard" in the diagram), which is a fully connected layer designed to output the prediction.
- For the pneumonia detection task, this classification head would output the probability that the X-ray indicates pneumonia (or not), making a binary or multi-class classification based on the output of the ViT.

CHAPTER 7

CODING AND TESTING

Environment Setup

In [1]:

```
!pip install tensorflow_addons
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import tensorflow_addons as tfa


import pandas as pd
import json
import zipfile
import os
import seaborn as sns
import random
import shutil
import time

from PIL import Image
from matplotlib import pyplot as plt

from keras.models import Sequential, Model

from tensorflow.keras.applications import InceptionV3, Xception, InceptionResNetV2, ResNet50
from tensorflow.keras.applications.resnet import preprocess_input

from tensorflow.keras.layers import Dense, Conv2D, Flatten, MaxPooling2D, Dropout, GlobalAveragePooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ModelCheckpoint, TensorBoard

!pip install wandb
import wandb
from wandb.keras import WandbCallback

!mkdir output
!mkdir output/tmp-augmented-images/

random.seed(123)
```

Dataset

In [2]:

```
# You need to insert your own API key below to run the code

kaggle = {
    "username": "<your-kaggle-username>",
    "api_key": "<your-kaggle-api-key>",
    "on_kernel": False,
    "dataset": {
        "sample": "-",
        "full": "paultimothymooney/chest-xray-pneumonia"
        }
}

if kaggle["on_kernel"]:
  path_prefix = '/kaggle/working/'
else:
  path_prefix = '/content/'

# Download dataset
def download_dataset(which_dataset):
  data = {"username": kaggle["username"],"key": kaggle["api_key"]}
  with open('kaggle.json', 'w') as json_file:
      json.dump(data, json_file)

  !mkdir -p ~/.kaggle
  !cp kaggle.json ~/.kaggle/
  !chmod 600 ~/.kaggle/kaggle.json
  kaggle_dataset = kaggle["dataset"][which_dataset]
  !kaggle datasets download -d $kaggle_dataset

  # Paths neeeds to be changed manually because of different directory structures, check with !ls
  if not os.path.isdir('dataset'):
    print("Unzipping... ")
    zip_ref = zipfile.ZipFile('chest-xray-pneumonia.zip', 'r')
    zip_ref.extractall('dataset')
    zip_ref.close()
    !rm -rf /content/dataset/chest_xray/chest_xray
    !rm -rf /content/dataset/chest_xray/__MACOSX
    !rm -rf /content/chest-xray-pneumonia.zip


  #!ls Data


download_dataset("full")

data_files = os.listdir("dataset/chest_xray")
print(data_files)
```

```
Downloading chest-xray-pneumonia.zip to /content
 99% 2.28G/2.29G [00:29<00:00, 98.5MB/s]
100% 2.29G/2.29G [00:29<00:00, 84.6MB/s]
Unzipping...
['train', 'test', 'val']
```

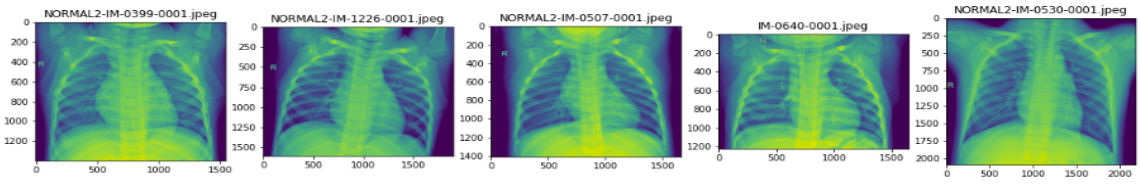
In [3]:

```
def resample_data(move_from, move_to, cl, images_to_move=100):
  path = path_prefix + 'dataset/chest_xray/'

  classes = os.listdir(path + move_from)

  cl += '/'
  curr_path = path + move_from + cl
  for _, _, files in os.walk(curr_path):
    random.shuffle(files)
    files_to_move = files[:images_to_move]
    for fn in files_to_move:
      shutil.move(curr_path + fn, path + move_to + cl + fn)
      #print('Moved ' + curr_path + fn)

  print('Resampled Images')

move_from, move_to = 'train/', 'val/'
#resample_data(move_from, move_to, 'PNEUMONIA', 2534)



# Training images
print('Number of NORMAL training images:')
!ls /content/dataset/chest_xray/train/NORMAL/ | wc -l
print('Number of PNEUMONIA training images:')
!ls /content/dataset/chest_xray/train/PNEUMONIA/ | wc -l
print()


# Test images
#resample_data('test/', 'val/', 'PNEUMONIA', 2690)
print('Number of NORMAL test images:')
!ls /content/dataset/chest_xray/test/NORMAL/ | wc -l
print('Number of PNEUMONIA test images:')
!ls /content/dataset/chest_xray/test/PNEUMONIA/ | wc -l
```

```
Number of NORMAL training images:
1341
Number of PNEUMONIA training images:
3875

Number of NORMAL test images:
234
Number of PNEUMONIA test images:
390
```

In [4]:

```
def viewImagesFromDir(path, num=5):
  #Display num random images from dataset. Rerun cell for new random images. The images are only single-channel

  img_paths_visualise = sorted(
        os.path.join(path, fname)
        for fname in os.listdir(path)
        if fname.endswith(".jpeg")
  )

  random.shuffle(img_paths_visualise)

  fig, ax = plt.subplots(1, num, figsize=(20, 10))

  for i in range(num):
    ax[i].imshow(Image.open(img_paths_visualise[i]))
    index = img_paths_visualise[i].rfind('/') + 1
    ax[i].title.set_text(img_paths_visualise[i][index:])

  fig.canvas.draw()
  time.sleep(1)

viewImagesFromDir('/content/dataset/chest_xray/train/NORMAL/', num=5)
```



Prepare the data

Configure the hyperparameters

In [5]:

```
CLASSES = os.listdir('/content/dataset/chest_xray/train')
TRAINING_DATA_SET_PATH = '/content/dataset/chest_xray/train'
TEST_DATA_SET_PATH = '/content/dataset/chest_xray/test'

params = dict(
    seed = 123,
    image_dim = (288,288),
    weight_decay = 1e-4,
    epochs = 30,
    batch_size = 16,
    patch_size = 18,
    pool_size = (2,2),
    optimizer = 'Adam',
    l_rate = 0.001,
    val_split = .15,
    use_transfer_learning = False,
    use_data_aug = False,

    l2_reg = .0,
    projection_dim = 16,
    num_heads = 4,

    # Size of the transformer layers
    transformer_layers = 4,
    num_classes = len(CLASSES),
    mlp_head_units = [1024,512]

    )

new_params = dict(
    num_patches = (params['image_dim'][0] // params['patch_size']) ** 2,
    transformer_units = [
    params['projection_dim'] * 2,
    params['projection_dim']],
    input_shape = (params['image_dim'][0], params['image_dim'][1], 3),

)
params.update(new_params)


if params['use_data_aug']:
  data_aug_params = dict(
      da_rotation = 20,
      da_w_shift = 0.1,
      da_h_shift = 0.1,
      da_shear = 0.05,
      da_zoom = 0.05,
      da_h_flip = False,
      da_v_flip = False,
  )

  params.update(data_aug_params)
```

```
# Ability to switch amount of channel to utilise pre-trained models with specific input shapes
if params['use_transfer_learning']:
  INPUT_SHAPE = (params['image_dim'][0], params['image_dim'][1], 3)
  COLOUR_MODE = 'rgb'
else:
  INPUT_SHAPE = (params['image_dim'][0], params['image_dim'][1], 3)
  COLOUR_MODE = 'rgb'

if params['use_data_aug']:
  datagen = ImageDataGenerator(validation_split=params['val_split'], rescale=1./255,
                               rotation_range=params['da_rotation'],
                               width_shift_range=params['da_w_shift'],
                               height_shift_range=params['da_h_shift'],
                               shear_range=params['da_shear'],
                               zoom_range=params['da_zoom'],
                               horizontal_flip=params['da_h_flip'],
                               vertical_flip=params['da_v_flip'],
                               fill_mode="constant",
                               cval=0
                               )
else:
  datagen = ImageDataGenerator(validation_split=params['val_split'])

# Read all training and validation data into variables from directory.
# Due to faulty quality of the given validation-set images, all images are taken from the training folder
train_generator = datagen.flow_from_directory(TRAINING_DATA_SET_PATH,
                                              batch_size=params['batch_size'],
                                              seed=123,
                                              class_mode="categorical",
                                              classes=CLASSES,
                                              target_size=params['image_dim'],
                                              color_mode=COLOUR_MODE,
                                              subset='training',
                                              shuffle=True)

val_datagen = ImageDataGenerator(validation_split=0.15, rescale=1./255)
valid_generator = datagen.flow_from_directory(TRAINING_DATA_SET_PATH,
                                              batch_size=params['batch_size'],
                                              seed=123,
                                              class_mode="categorical",
                                              classes=CLASSES,
                                              target_size=params['image_dim'],
                                              color_mode=COLOUR_MODE,
                                              subset='validation',
                                              shuffle=False)

test_datagen = ImageDataGenerator(rescale=1./255)
test_generator = test_datagen.flow_from_directory(TEST_DATA_SET_PATH,
                                              batch_size=params['batch_size'],
                                              seed=123,
                                              class_mode="categorical",
                                              classes=CLASSES,
                                              target_size=params['image_dim'],
                                              color_mode=COLOUR_MODE,
                                              shuffle=False)
```

```
Found 4434 images belonging to 2 classes.
Found 782 images belonging to 2 classes.
Found 624 images belonging to 2 classes.
```

Custom Layers

The following cells uses inspiration from Khalid Salamas and the An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale paper from Google Research on Vision Transformers.

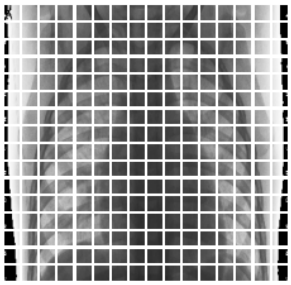
In [6]:

```
def mlp(x, hidden_units, dropout_rate):
    for units in hidden_units:
        x = layers.Dense(units, activation=tf.nn.gelu)(x)
        x = layers.Dropout(dropout_rate)(x)
    return x

class Patches(layers.Layer):
    def __init__(self, patch_size):
        super(Patches, self).__init__()
        self.patch_size = patch_size

    def call(self, images):
        batch_size = tf.shape(images)[0]
        patches = tf.image.extract_patches(
            images=images,
            sizes=[1, self.patch_size, self.patch_size, 1],
            strides=[1, self.patch_size, self.patch_size, 1],
            rates=[1, 1, 1, 1],
            padding="VALID",
        )
        patch_dims = patches.shape[-1]
        patches = tf.reshape(patches, [batch_size, -1, patch_dims])
        return patches


# Linearly transform patches by projecting it into a
# vector of size `projection_dim` and also adds a learnable position
# embedding to the projected vector.
class PatchEncoder(layers.Layer):
    def __init__(self, num_patches, projection_dim):
        super(PatchEncoder, self).__init__()
        self.num_patches = num_patches
        self.projection = layers.Dense(units=projection_dim)
        self.position_embedding = layers.Embedding(
            input_dim=num_patches, output_dim=projection_dim
        )

    def call(self, patch):
        positions = tf.range(start=0, limit=self.num_patches, delta=1)
        encoded = self.projection(patch) + self.position_embedding(positions)
        return encoded
```

In [7]:

```
import matplotlib.pyplot as plt

plt.figure(figsize=(4, 4))
image, label = iter(next(train_generator))
image = image[0]*255.
#image = x_train[np.random.choice(range(x_train.shape[0]))]
plt.imshow((image).astype("uint8"))
plt.axis("off")

resized_image = tf.image.resize(
    tf.convert_to_tensor([image]), size=(params['image_dim'][0], params['image_dim'][0])
)
patches = Patches(params['patch_size'])(resized_image)
print(f"Image size: {params['image_dim'][0]} X {params['image_dim'][0]}")
print(f"Patch size: {params['patch_size']} X {params['patch_size']}")
print(f"Patches per image: {patches.shape[1]}")
print(f"Elements per patch: {patches.shape[-1]}")

n = int(np.sqrt(patches.shape[1]))
plt.figure(figsize=(4, 4))
for i, patch in enumerate(patches[0]):
    ax = plt.subplot(n, n, i + 1)
    patch_img = tf.reshape(patch, (params['patch_size'], params['patch_size'], 3))
    plt.imshow(patch_img.numpy().astype("uint8"))
    plt.axis("off")
```

```
Image size: 288 X 288
Patch size: 18 X 18
Patches per image: 256
Elements per patch: 972
```



Generate ViT model

In [8]:

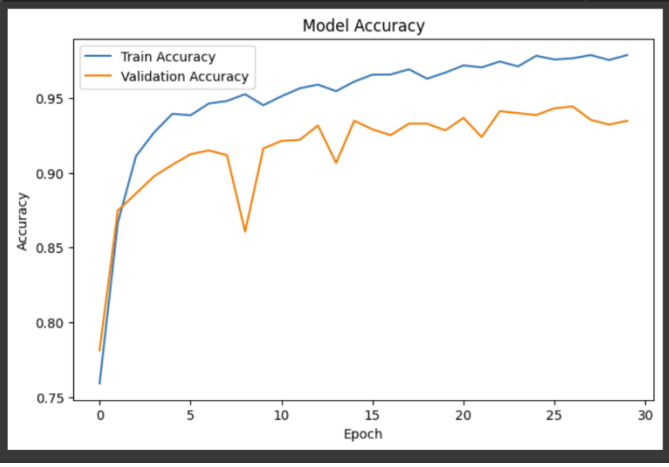
```
def create_vit_classifier():
    inputs = layers.Input(shape=params['input_shape'])
    # Create patches.
    patches = Patches(params['patch_size'])(inputs)
    # Encode patches.
    encoded_patches = PatchEncoder(params['num_patches'], params['projection_dim'])(patches)

    # Create multiple layers of the Transformer block.
    for _ in range(params['transformer_layers']):
        # Layer normalization 1.
        x1 = layers.LayerNormalization(epsilon=1e-6)(encoded_patches)
        # Create a multi-head attention layer.
        attention_output = layers.MultiHeadAttention(
            num_heads=params['num_heads'], key_dim=params['projection_dim'], dropout=0.1
        )(x1, x1)
        # Skip connection 1.
        x2 = layers.Add()([attention_output, encoded_patches])
        # Layer normalization 2.
        x3 = layers.LayerNormalization(epsilon=1e-6)(x2)
        # MLP.
        x3 = mlp(x3, hidden_units=params['transformer_units'], dropout_rate=0.1)
        # Skip connection 2.
        encoded_patches = layers.Add()([x3, x2])

    # Create a [batch_size, projection_dim] tensor.
    representation = layers.LayerNormalization(epsilon=1e-6)(encoded_patches)
    representation = layers.Flatten()(representation)
    representation = layers.Dropout(0.5)(representation)
    # Add MLP.
    features = mlp(representation, hidden_units=params['mlp_head_units'], dropout_rate=0.5)
    # Classify outputs.
    logits = layers.Dense(len(CLASSES))(features)
    # Create the Keras model.
    model = keras.Model(inputs=inputs, outputs=logits)
    return model
```

Train Model

In [9]:

```
from keras import backend as K

# For experiment tracking
USE_WANDB = False

if USE_WANDB:
  wandb.init(tags=["ViT", "Binary"], project='dd2424', entity='teambumblebee', sync_tensorboard=True, config=params, save_c
  print(params)
  wandb_callback = WandbCallback(monitor='val_f1_m',
                                 save_model=True,
                                 save_weights_only=False, mode='max',
                                 log_weights=True,
                                 data_type="image", verbose=1,
                                 labels=CLASSES,
                                 generator=valid_generator,
                                 predictions=50,
                                 log_evaluation=True,
                                 log_batch_frequency=1)

def run_experiment(model):
    optimizer = tfa.optimizers.AdamW(
        learning_rate=params['l_rate'], weight_decay=params['weight_decay']
    )

    def recall_m(y_true, y_pred):
        true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
        possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
        recall = true_positives / (possible_positives + K.epsilon())
        return recall

    def precision_m(y_true, y_pred):
        true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
        predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
        precision = true_positives / (predicted_positives + K.epsilon())
        return precision

    def f1_m(y_true, y_pred):
        precision = precision_m(y_true, y_pred)
        recall = recall_m(y_true, y_pred)
        return 2*((precision*recall)/(precision+recall+K.epsilon()))

    model.compile(
        optimizer=optimizer,
        loss=keras.losses.BinaryCrossentropy(from_logits=True),
        metrics=[
            keras.metrics.BinaryAccuracy(name="binary_accuracy"), f1_m, recall_m, precision_m
        ],
    )


    checkpoint_callback = keras.callbacks.ModelCheckpoint(
        "/content/output/model_best.h5",
        monitor="val_f1_m",
        save_best_only=True,
        save_weights_only=True,
        verbose=1,
        mode="max"
    )

    callbacks_list = [checkpoint_callback, tf.keras.callbacks.EarlyStopping(patience=20, monitor="val_binary_accuracy")]

    if USE_WANDB:
      callbacks_list.append(wandb_callback)


    history = model.fit(
        train_generator,
        batch_size=params['batch_size'],
        epochs=params['epochs'],
        validation_data=valid_generator,
        callbacks=callbacks_list)

    return history, model

model = create_vit_classifier()
history, model = run_experiment(model)
```

```
plt.figure(figsize=(8, 5))

# Accessing accuracy and validation accuracy from the history object
plt.plot(history.history['binary_accuracy'], label='Train Accuracy') # Changed model to history and accuracy to binary_accuracy
plt.plot(history.history['val_binary_accuracy'], label='Validation Accuracy') # Changed model to history and val_accuracy to val_binary_accuracy

plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(loc='upper left')
plt.show()
```



0s

```
%%writefile app.py
import streamlit as st
from PIL import Image
import tensorflow as tf
import numpy as np
from keras import layers
from keras.utils import custom_object_scope
import tensorflow_addons as tfa
# Define your custom layers (ensure these match your model's architecture)
class Patches(tf.keras.layers.Layer):
    # Your implementation here
    def __init__(self, patch_size):
        super(Patches, self).__init__()
        self.patch_size = patch_size

    def call(self, images):
        batch_size = tf.shape(images)[0]
        patches = tf.image.extract_patches(
            images=images,
            sizes=[1, self.patch_size, self.patch_size, 1],
            strides=[1, self.patch_size, self.patch_size, 1],
            rates=[1, 1, 1, 1],
            padding="VALID",
        )
        patch_dims = patches.shape[-1]
        patches = tf.reshape(patches, [batch_size, -1, patch_dims])
        return patches
    pass
```

```python
class PatchEncoder(tf.keras.layers.Layer):
    # Your implementation here
    def __init__(self, num_patches, projection_dim):
        super(PatchEncoder, self).__init__()
        self.num_patches = num_patches
        self.projection = layers.Dense(units=projection_dim)
        self.position_embedding = layers.Embedding(
            input_dim=num_patches, output_dim=projection_dim
        )

    def call(self, patch):
        positions = tf.range(start=0, limit=self.num_patches, delta=1)
        encoded = self.projection(patch) + self.position_embedding(positions)
        return encoded
    pass
def f1_m(y_true, y_pred):
    y_true = tf.cast(y_true, tf.float32)
    y_pred = tf.cast(y_pred > 0.5, tf.float32)  # Assuming binary classification
    tp = tf.reduce_sum(y_true * y_pred)
    precision = tp / (tf.reduce_sum(y_pred) + tf.keras.backend.epsilon())
    recall = tp / (tf.reduce_sum(y_true) + tf.keras.backend.epsilon())
    f1 = 2 * (precision * recall) / (precision + recall + tf.keras.backend.epsilon())
    return f1
def recall_m(y_true, y_pred):
    y_true = tf.cast(y_true, tf.float32)
    y_pred = tf.cast(y_pred > 0.5, tf.float32)  # Assuming binary classification
    true_positives = tf.reduce_sum(y_true * y_pred)
    possible_positives = tf.reduce_sum(y_true)
    recall = true_positives / (possible_positives + tf.keras.backend.epsilon())
    return recall
def precision_m(y_true, y_pred):
    y_true = tf.cast(y_true, tf.float32)
    y_pred = tf.cast(y_pred > 0.5, tf.float32)  # Assuming binary classification
    true_positives = tf.reduce_sum(y_true * y_pred)
    predicted_positives = tf.reduce_sum(y_pred)
    precision = true_positives / (predicted_positives + tf.keras.backend.epsilon())
    return precision
```

```python
# Function to load the model with custom layers
def load_model():
    with custom_object_scope({'Patches': Patches, 'PatchEncoder': PatchEncoder, 'f1_m': f1_m, 'recall_m': recall_m, 'precision_m': precision_m, 'AdamW': tfa.optimizers.AdamW}):
                model = tf.keras.models.load_model('/content/output/model_best.h5')
    return model

# Load your trained model
model = load_model()

# Function to preprocess image for the model
def preprocess_image(image):
    img = image.resize((288, 288))  # Resize to match your model input size
    img = np.array(img) / 255.0      # Normalize the image
    # If the image is grayscale, convert to RGB
    if img.shape[-1] == 1:
        img = np.stack((img,) * 3, axis=-1)
    img = np.expand_dims(img, axis=0)  # Add batch dimension
    return img

# Function to make predictions
def predict(image):
    processed_img = preprocess_image(image)
    prediction = model.predict(processed_img)

    # Ensure prediction is in the right shape and scale
    confidence = prediction[0][0] * 100  # This assumes model outputs a value between 0 and 1
    if confidence < 0:  # Check for any unexpected negative values
        confidence = 0
    elif confidence > 100:  # Check for any unexpected values over 100
        confidence = 100

    if confidence > 50:
        result = 'Pneumonia Detected'
    else:
        result = 'No Pneumonia Detected'

    return result, confidence



# Streamlit UI
st.title("Pneumonia Detection from Chest X-ray")
st.write("Upload a chest X-ray image to check if the person has pneumonia.")

# File uploader in Streamlit
uploaded_file = st.file_uploader("Choose an X-ray image...", type=["jpg", "png", "jpeg"])

if uploaded_file is not None:
    image = Image.open(uploaded_file)
    st.image(image, caption='Uploaded Chest X-ray.', use_column_width=True)

    #st.write("Classifying...")
    with st.spinner("Processing..."):
        result, confidence = predict(image)
        st.write(f"Result: **{result}**")
        st.write(f"Confidence Level: **{confidence:.2f}**")
```

```
[20] !streamlit run app.py &>/dev/null&
```

```
[21] !ngrok config add-authtoken 2ntDVOS9K68FL9xba0j2qS5ZiUp_3QdA6ytJn7Bt1jbrwq5yL
```

```
Authtoken saved to configuration file: /root/.config/ngrok/ngrok.yml
```

```
[22] !streamlit run app.py --server.port 8501 &
```

```
Collecting usage statistics. To deactivate, set browser.gatherUsageStats to false.

2024-10-25 15:08:41.219 Port 8501 is already in use
```

```
!curl http://localhost:8501
```

```
<!doctype html><html lang="en"><head><meta charset="UTF-8"/><meta name="viewport" content="width=device-width,initial-scale=1,shrink-to-fit=no"/><link rel="shortcut icon"
```

+ Code + Text

```
[29] from pyngrok import ngrok

     # Disconnect any existing ngrok connections
     ngrok.kill()

     # Connect ngrok to the Streamlit app
     public_url = ngrok.connect(8501)
     print("Streamlit app is live at:", public_url)
```

```
Streamlit app is live at: NgrokTunnel: "https://f3f8-34-83-167-167.ngrok-free.app" -> "http://localhost:8501"
```

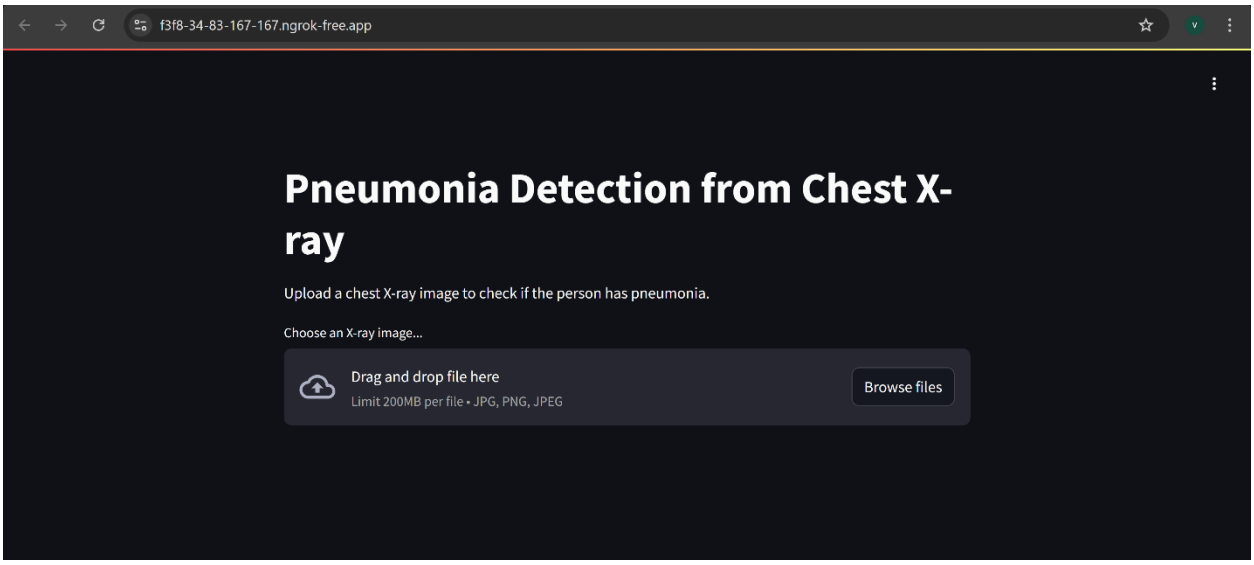


Fig 7.1: Front Page

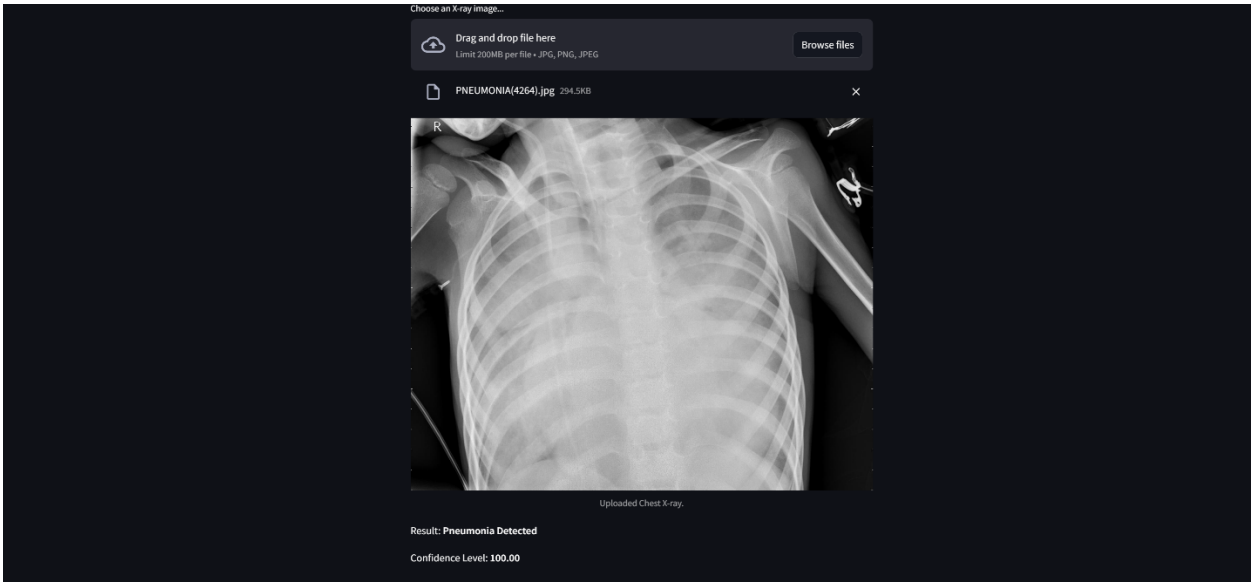


Fig 7.2: Output for Pneumonia Detected

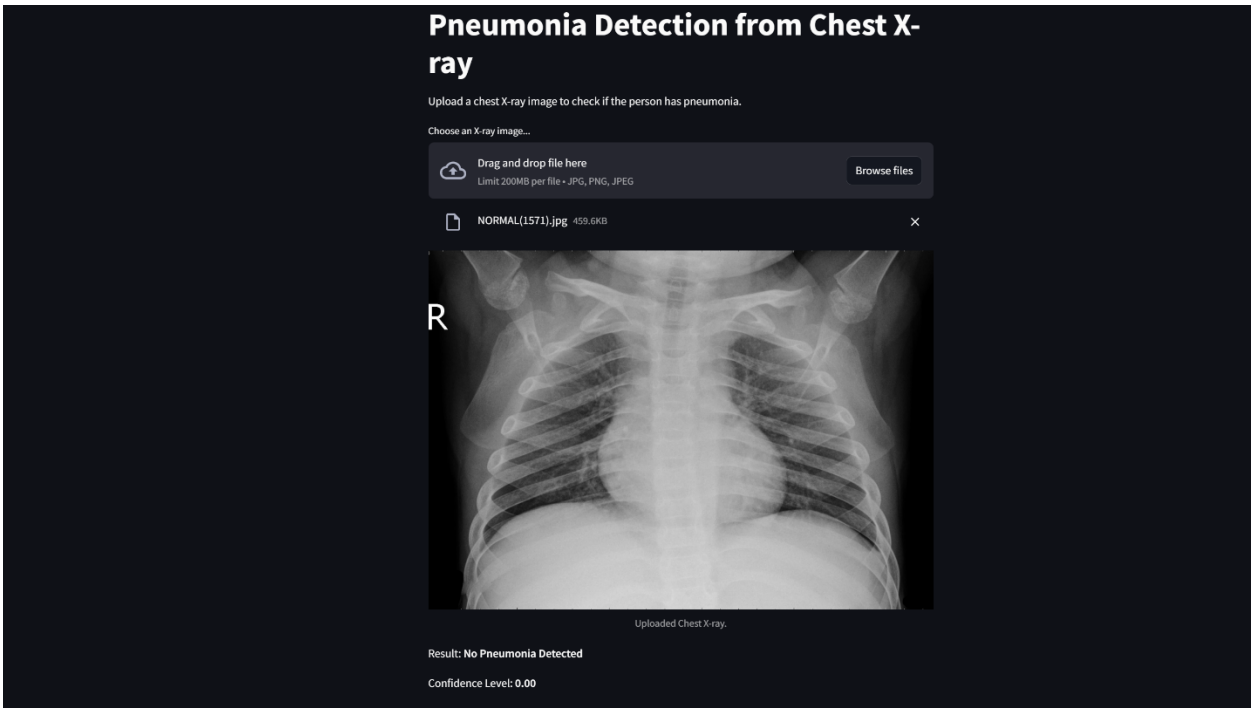


Fig 7.3: Output for No Pneumonia Detected

CHAPTER 8

RESULTS AND ANALYSIS

The Vision Transformer (ViT) model developed for pneumonia detection has shown highly encouraging performance across several key metrics, indicating its effectiveness and reliability. Below is a detailed analysis of each metric:

8.1 Recall: 97.59%

- **Explanation**: Recall, also known as sensitivity or true positive rate, measures the model's ability to correctly identify positive cases (i.e., instances of pneumonia) among all actual positive cases. A high recall score means the model is effective at detecting most instances of pneumonia, minimizing the likelihood of false negatives (where cases of pneumonia are incorrectly classified as non-pneumonia).
- **Analysis**: With a recall of 97.59%, the model is highly sensitive and can identify almost all cases of pneumonia. This is particularly important in a healthcare application, as failing to detect pneumonia could have severe consequences for patients. The high recall implies that the model misses very few pneumonia cases, ensuring that the majority of actual cases are detected and potentially treated early

8.2 Precision: 98.64%

- **Explanation**: Precision indicates the proportion of true positive cases among the predicted positive cases. In other words, it assesses how many of the cases that the model classified as pneumonia are indeed pneumonia.
- **Analysis**: A precision of 98.64% suggests that the model makes very few false positive predictions (non-pneumonia cases wrongly classified as pneumonia). This is crucial to avoid over-diagnosing and unnecessary treatment of healthy individuals. High precision ensures that when the model indicates pneumonia, it is almost certainly correct, which builds confidence in the model's predictions.

8.3 F1 Score: 98.10%

- **Explanation**: The F1 score is the harmonic mean of precision and recall, balancing the trade-off between the two. It is especially useful in scenarios where a balance between sensitivity (recall) and specificity (precision) is essential.
- **Analysis**: With an F1 score of 98.10%, the model achieves a strong balance between detecting actual cases and minimizing incorrect positive predictions. This high score implies that the model performs reliably well across both aspects, making it suitable for clinical use where a careful balance is essential to provide accurate diagnoses without overwhelming healthcare professionals with false positives or missing true cases.

8.4 Accuracy: 98.12%

- **Explanation**: Accuracy is the proportion of correct predictions (both true positives and true negatives) out of the total predictions made. It provides a general sense of how well the model performs across all cases.
- **Analysis**: An accuracy of 98.12% indicates that the model performs exceptionally well in correctly classifying both pneumonia and non-pneumonia cases. This high accuracy reflects the robustness of the model and its effectiveness in classifying chest X-ray images accurately. While accuracy alone does not provide a complete picture, the consistently high values across all metrics show that the model is not only accurate but also maintains a high standard in recall, precision, and F1 score.

Overall Implications

These performance metrics demonstrate that the ViT model is both highly accurate and reliable for pneumonia detection. Its high recall minimizes the risk of missed diagnoses, while its high precision reduces false alarms, providing a dependable solution in a clinical setting. Furthermore, the F1 score and accuracy underscore its balanced performance across all cases, making this model

a valuable tool for automated pneumonia detection that could support healthcare providers by enabling faster, accurate diagnostics.

CHAPTER 9

CONCLUSION

The Vision Transformer (ViT) model developed for pneumonia detection has demonstrated exceptional performance in accurately identifying cases through chest X-ray images. With an accuracy of 98.12%, recall of 97.59%, precision of 98.64%, and an F1 score of 98.10%, the model achieves a strong balance between sensitivity and specificity. This makes it suitable for deployment in healthcare settings, where high sensitivity is critical for detecting potentially fatal cases, and high precision ensures minimal false positives, avoiding unnecessary treatments and patient distress.

The use of the ViT architecture in this project highlights the model's effectiveness for image classification tasks, particularly in medical imaging, where it can capture fine details and spatial relationships within images. By leveraging Google Colab for model development and training, and Streamlit and Ngrok for the user interface and deployment, the model was accessible for both research and experimental use. This streamlined deployment approach offers an effective prototype but suggests potential for further refinement and scalability, especially for integration into larger healthcare infrastructures.

Future Work

Despite its impressive performance, the model can be further enhanced and expanded. Several areas of future work are suggested:

1. **Model Optimization for Faster Inference**: The ViT model's high accuracy makes it suitable for use in healthcare applications, but optimizing the model for faster inference could improve its practicality, particularly in real-time diagnostics. Techniques like model pruning, quantization, and knowledge distillation could help reduce computational complexity and speed up predictions without sacrificing accuracy.

2. **Deployment on Scalable Cloud Platforms**: While Google Colab and Ngrok have served as effective tools for development, future implementations should consider deploying the model on scalable cloud platforms such as AWS, Google Cloud Platform, or Azure. These services offer robust infrastructure, faster inference capabilities, and can accommodate high volumes of requests, making the application more reliable for clinical environments.

3. **Integration with a Broader Diagnostic System**: To enhance clinical applicability, the model could be integrated into broader diagnostic systems, where it could assist radiologists in a semi-automated or fully automated diagnostic pipeline. Coupling this model with other diagnostic tools, such as electronic health records (EHR) and additional diagnostic models, could enable comprehensive support for patient management and improve overall diagnostic accuracy.

4. **Expanding Dataset and Testing with Diverse Data**: The model's robustness could be further validated by training and testing it on a larger, more diverse dataset, including images from various demographics and sources. This would increase the model's generalizability, reduce biases, and improve its reliability in real-world applications across different populations.

5. **Developing Explainability Features**: In clinical practice, explainability is vital for healthcare providers to trust AI-generated predictions. Incorporating attention visualization techniques or heatmaps that indicate regions of interest within the X-ray images could help radiologists understand the model's decision-making process, making the tool more transparent and interpretable.

6. **Exploring Transfer Learning for Other Medical Imaging Tasks**: The success of the ViT model in pneumonia detection suggests potential for transfer learning to other medical imaging tasks, such as detecting other types of lung diseases or even conditions in different parts of the body (e.g., brain scans for tumor detection). Such expansion would enable the model to serve a wider range of healthcare needs and demonstrate the versatility of Vision Transformers in medical imaging applications.

CHAPTER 10

REFERENCES

[1] Chandan Chakraborty, Asok K. Maiti, Mallika Pal, Madhumala Ghosh, and Dev Kumar Das (2013) utilized light microscopic images and machine learning to automatically conduct tests for malaria parasites, as detailed in Micron 45 (2013).

[2] George Thoma, Stefan Jaeger, Richard Maude, Mahdieh Poostchi, and Kamorat Silamut (2018) conducted research on malaria detection through the utilization of machine learning and image analysis, as highlighted in Research on Translation (2018).

[3] Adriano G. Duse, David M. Rubin, Charles J. Pritchard, and Nicholas E. Ross (2006) developed an automated image processing method for the detection and categorization of thin blood smear malaria, as discussed in Computing and Medical and Biological Engineering 44, 5 (2006).

[4] Stefan Carlsson, Josephine Sullivan, Hossein Azizpour, and Ali Sharif Razavian (2014) emphasized the effectiveness of Convolutional Neural Networks (CNN) as a reliable tool for recognition, as presented in the Proceedings of the IEEE Workshops on Pattern Recognition and Computer Vision. In their groundbreaking 2012 study entitled "Deep Convolutional Neural Networks for ImageNet Categorization,"

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton explore the use of deep convolutional neural networks for this specific purpose. The study was published in Developments in Neural Information Processing Systems and has significantly contributed to advancements in image classification technology.

[6] Andrew Zisserman and Karen Simonyan published a research paper in 2014 titled "Convolutional networks with extreme depth for large-scale picture recognition" in the arXiv preprint arXiv:1409.1556.

[7] According to F. Chollet's research conducted in 2016, Xception utilizes separable convolutions to enhance the performance of deep learning algorithms. This study was published in the arXiv Preprint arXiv1610 2357.

[8] Kaiming He, Shaoqing Ren, Jian Sun, and Xiangyu Zhang authored a paper in 2016 on image recognition through the use of deep residual learning, which was presented at the IEEE Conference on Computer Vision and Pattern Recognition Proceedings.

CHAPTER 11

APPENDICES

A. For journal Publications:

Paper cover is attached below:

Design and Implementation of a Vision Transformer for Pneumonia Detection

T Vikash Patra
Department of Computing Technologies
SRMIST
Chennai, Tamil Nadu
tt7597@srmist.edu.in

NSS Pavan Kumar
Department of Computing Technologies
SRMIST
Chennai, Tamil Nadu
nn5610@srmist.edu.in

Vetriselvi D
Assistant Professor
Department of Computing Technologies
SRMIST
Chennai, Tamil Nadu
vetrised@srmist.edu.in

1.Abstract

Pneumonia remains one of the most burdensome diseases within the population, as well as one of the most frequent causes of death, thus necessitating prompt diagnosis to reduce fatalities and improve the patients' prognosis. In conventional diagnostic techniques, such as a physical examination followed by a chest x ray, interpretation of these images is usually manual and is therefore prone to errors and quite slow, particularly in places where radiologists are few or unaffordable. In recent years, a significant focus has been placed on deep learning-based methods, notably Convolutional Neural Networks (CNN), for the automatic detection of pneumonia from medical images, and positive results have been recorded. However, CNN models have a limitation in that owing to their local receptive fields, it becomes impossible to capture long-range dependencies in the image. This paper presents an innovative use of Vision Transformer (ViT), which was originally a model for natural image classification, for detecting pneumonia using chest X-ray images. Vision transformer (ViT) architecture primarily utilizes the self-attention mechanism, that helps in context-aware representation of features of the image, given that in medical image analysis most of the time the abnormal patterns are often spread throughout the image, quite far from each other. For this purpose, in this work, we trained a ViT model on public cough chest X-ray database.

2.Introduction

Infections of the lungs due to bacteria, viruses, and fungi, known as pneumonia, are still one of the greatest problems that the world faces due to the associated morbidity and mortality. In World Health Organization (WHO) statistics, pneumonia appears to be one of the leading causes of mortality in children under five years of age; it accounts for 14% of deaths hence a disease that should be diagnosed early and correctly. There is also the use of Chest X-ray films which is the most common image modality for pneumonia diagnosis, however, its interpretation can also be intricate and lengthily done, usually relying on the skills of seasoned radiologists. In many low-resource settings of the world, where there are very few if not any radiologists, this further translates to wastage of time in diagnosis and subsequently high mortality rates. For this reason, there is a pressing demand for assistive diagnostic tools that are quick and dependable, particularly in regions with little health care provision.

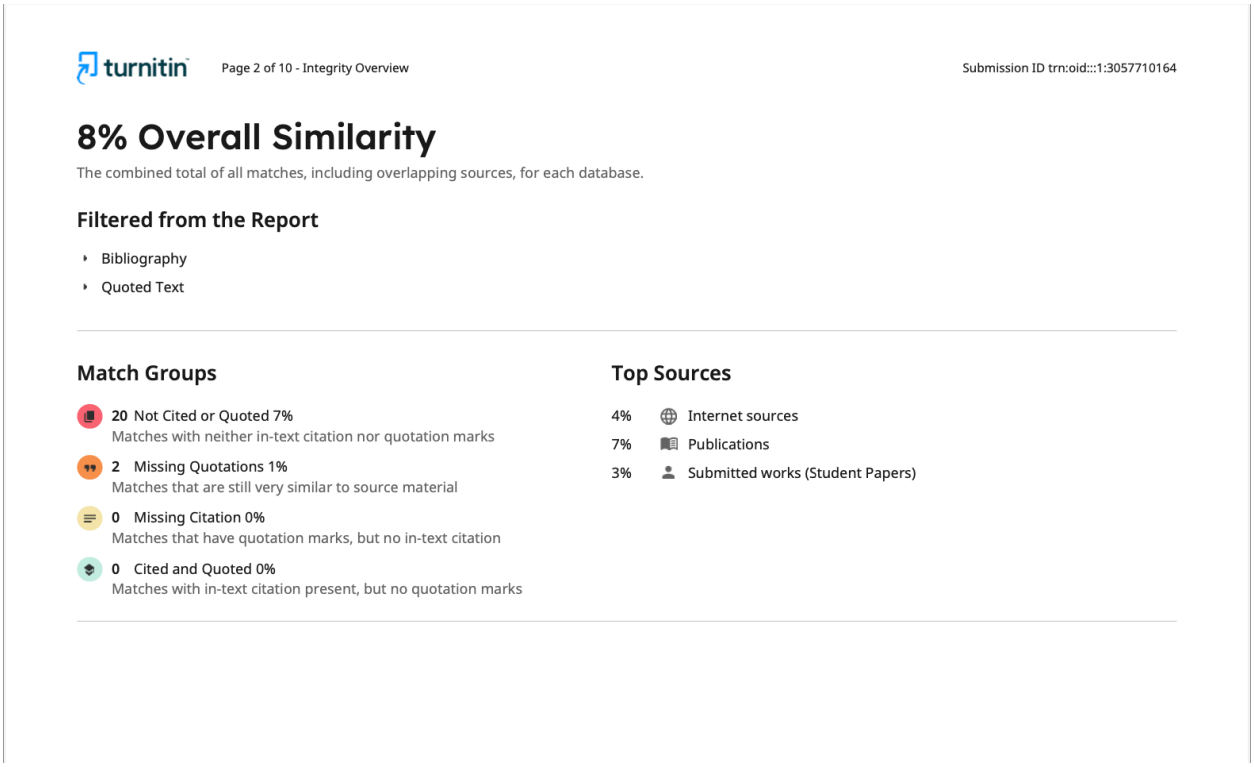
We recommend the application of the Vision Transformer, a model designed for image classification purposes that utilizes a transformer based architecture. A self-attention network such as the Vision Transformer does not have layers in the framework that define distance and position between objects in an image, as is the case with Convolutional Neural Networks, instead images are presented as a whole. It is this trait that gives an upper hand to ViT since it can process an entire image even though it can also divide the image into sections referred to as patches which are treated as words in a language task. This is suitable for image-based processes that require a greater level of context as well as additional details. This work seeks to evaluate Vision Transformer and some of the modern techniques based on convolutional neural networks for chest X-ray images in detecting pneumonia, and provide comparisons of the results obtained. We set out to demonstrate this by employing the ViT self-attention mechanism and proving that it is capable of modelling complex structures of the medical images and thus improving performance of the task compared to traditional CNN model. Based on the results, we believe that Vision Transformers are very valuable in aiding the diagnostic processes in systems that classify medical images automatically.

3.Problem Statement

Pneumonia remains a significant global health concern, highlighting the importance of prompt and accurate diagnosis. The traditional method of interpreting chest X-rays relies heavily on the expertise of radiologists, resulting in delays in diagnosis, particularly in less developed regions. In response, some researchers have

Fig 11.A: Paper Cover

B. Plagiarism Report:



turnitin Page 2 of 10 - Integrity Overview Submission ID trn:oid:::1:3057710164

8% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text

Match Groups

- **20** Not Cited or Quoted 7%
  Matches with neither in-text citation nor quotation marks
- **2** Missing Quotations 1%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

- 4% Internet sources
- 7% Publications
- 3% Submitted works (Student Papers)

Fig 11.B: Plagiarism report