DETECTING PNEUMONIA IN A CHEST X-RAY USING DEEP LEARNING

A graduation project report submission

In partial fulfilment of the requirements for the award of the degree

Bachelor of Science

Submitted by:

Jan Youssef	89477
Mohamed Tarek	89610
Mohamed Osama	89402
Bishoy Samir	89534

Under the supervision of Professors:

Dr. Khaled Alsheshtawy

Supervisor(s):

T.A. Eng. Aya Mahmoud

Department of Computer Science - CS
Misr University for Science and Technology - MUST
College of Computer and Artificial Intelligence Technologies - CAIT
June 2023

ACKNOWLEDMENT

I would like to express my deepest gratitude and heartfelt appreciation to Dr. Rania Algohary, the esteemed Dean of the College, for her invaluable support and guidance throughout my academic journey. Her dedication to excellence and commitment to the development of her students have been truly inspiring. I am truly grateful for her wisdom, encouragement, and unwavering belief in my abilities.

I would also like to extend my sincere thanks to Prof. Dr. Khaled Alsheshtawi, my project supervisor, for his exceptional mentorship and unwavering support. His profound knowledge, expertise, and invaluable insights have been instrumental in shaping my research and academic growth. His constructive feedback and constant encouragement have pushed me to strive for excellence.

Furthermore, I would like to acknowledge the significant contributions of TAs Aya Mahmoud and Zeyad Khaled. Their dedication, patience, and commitment to their roles have been truly commendable. Their assistance and guidance throughout my academic endeavors have been invaluable, and I am grateful for their unwavering support.

I would like to express my gratitude to all of these remarkable individuals for their invaluable contributions to my academic journey. Their guidance, support, and expertise have shaped me into the person I am today. I am forever grateful for their unwavering belief in my abilities and for helping me reach new heights of academic excellence.

I extend my sincerest appreciation to Dr. Rania Algohary, Prof. Dr. Khaled Alsheshtawi, TAs Aya Mahmoud, and Zeyad Khaled for their guidance, support, and mentorship. It is because of their invaluable contributions that I have been able to achieve my goals and embark on a successful academic journey.

DECLARATION

I hereby declare that this work, which I am now submitting for evaluation in a program of study leading to a Bachelor of Science at project Depression Detection using Deep learning Algorithms is entirely my own work, and that I have exercised reasonable care to ensure that the work is original and, to the best of my knowledge, does not contravene any copyright law and publication,

and it was not taken from the work of others and to the extent that this work has been cited and acknowledged in the References section of this report.

FUNCTIONAL & NON-FUNCTIONAL REQUIREMENTS

Functional requirements:

1. System should be able to detect diseases in a medical scan
2. User should be able to upload medical scans
3. User should be able to revisit old results
4. Users can register as patients or doctors
5. Patients should be able to view their previous results
6. Doctors should be able to add, delete, and view their patients

Non-functional requirements:

Accuracy
Privacy
Security
Usability

EXPECTED OUTCOMES

A web app with which users can upload their medical scans, which are sent to AI microservices that detect different diseases present in these scans and report the aggregated results back to the user in a friendly layout.

ABSTRACT

The world faces a shortage of trained radiologists, [1] developed countries have aging populations, and developing countries have overburdened medical systems [2]. The purpose of this project is to build an AI-powered web app, that reduces the overall effort required for the diagnosis of pulmonary diseases using deep learning. Previous work has gone into using AI for medical purposes, and the World Health Organization (WHO) now recommends using AI solutions to reduce the overall costs of radiography [3], however, to the best of our knowledge no solution on the market attempts to provide detection for more than one disease at a time. In our project users will be able to upload their chest x rays (CXRs) through a simple web interface, this image is then passed to different AI models trained to detect different pulmonary diseases, initially we are targeting 4 diseases, Covid-19, Emphysema, Pneumonia, and Tuberculosis. The system will use a micro-service architecture with a publisher-subscriber design pattern to enable scalability and future additions to the system with minimal change cost.

Keywords: Artificial Intelligence; Pneumonia; X-rays; Detection; Screening

TABLE OF CONTENTS

LIST OF ACRONYMS/ABBREVIATIONS

Abbreviation	Definition
AI	Artificial Intelligence
AUC	Area Under ROC Curve
CNN	Convolutional Neural Network
COPD	Chronic Obstructive Pulmonary Disease
CT	Computed Tomography
CXR	Chest X-Ray
DDX	Differential Diagnosis
NLP	Natural Language Processing
ROC	Receiver Operating Characteristic
SSD	Single Shot Multibox Detector
TB	Tuberculosis
WHO	World Health Organization
CNNS	Convolutional Neural Networks
ANNs	Artificial Neural Networks

PROJECT SCOPE

The scope of our project will be a CXR module, which contains four diseases, selected for their prevalence, and abundance of data.

Those are:

1. Pneumonia
2. Covid-19
3. Emphysema
4. Tuberculosis

ALGORITHMS USED

Artificial Neural Networks (ANNs):

ANNs are a type of machine learning algorithm that are modeled after the structure and function of the human brain. ANNs consist of multiple interconnected layers of nodes or neurons, each of which performs a specific function in processing data.

The process of training an ANN involves feeding it a large amount of labeled data and allowing it to adjust its internal weights and biases to make accurate predictions. This process is called backpropagation, and it enables ANNs to learn complex patterns in data and make predictions with high accuracy.

There are several types of ANNs, including feedforward networks, recurrent networks, convolutional networks, and deep networks. Each of these networks has its own unique structure and is designed to perform specific tasks.

One of the most significant advantages of ANNs is their ability to learn and adapt to new data. They can be used in a wide range of applications, such as image and speech recognition, natural language processing, and predictive analytics.

To build an ANN, there are several tools and libraries available, including TensorFlow, Keras, PyTorch, and. These tools provide a range of pre-built models and functions, making it easier to build and train ANNs for specific tasks.

Overall, ANNs have become a vital tool in modern data science and have the potential to revolutionize a wide range of industries. As technology continues to advance, we can expect to see ANNs being used to solve increasingly complex problems and make predictions with even greater accuracy.

Convolutional Neural Networks (CNNs):

CNNs are a type of deep learning algorithm that are designed to process and analyze data with a grid-like structure, such as images or videos. Unlike traditional neural networks, which process data in a sequential manner, CNNs leverage a special architecture that allows them to extract meaningful features from the input data in a hierarchical manner.

CNNs consist of multiple layers, including convolutional layers, pooling layers, and fully connected layers. The convolutional layers apply a set of filters to the input data, each of which detects a specific feature or pattern in the image. The pooling layers then downsample the output of the convolutional layers, reducing the size of the feature maps and making the network more efficient. Finally, the fully connected layers process the output of the pooling layers to make a prediction.

The process of training a CNN involves feeding it a large dataset of labeled images and adjusting the internal weights and biases of the network to minimize the difference between its predictions and the true labels. This process is typically done using stochastic gradient descent or a similar optimization algorithm.

CNNs are highly effective at analyzing image data and have been used in a wide range of applications, such as object detection, face recognition, and image segmentation. One of the main advantages of

CNNs is their ability to automatically extract useful features from raw data, making them highly effective at tasks that involve visual analysis.

There are several popular tools and libraries available for building and training CNNs, such as TensorFlow, Keras, and PyTorch. These libraries provide a range of pre-built models and functions, making it easier to build and train CNNs for specific tasks.

Overall, CNNs are a powerful tool for analyzing data with a grid-like structure, such as images or videos. Their ability to extract meaningful features from raw data makes them highly effective at tasks that involve visual analysis, and they are widely used in a range of industries, from healthcare to self-driving cars. As technology continues to advance, we can expect CNNs to become even more powerful and versatile.

Real-time Parallel Processing:

In this project, we encountered the challenge of processing large image files uploaded by the user. To address this challenge, we implemented an approach that leverages several tools and techniques to begin processing the image as soon as the upload button is clicked, without waiting for the entire image to be uploaded.

To achieve this, we utilized filters from the Spring Web framework to intercept the incoming request and begin processing the image data in real-time. Specifically, we used the MultipartFilter, which allows us to handle multipart/form-data requests, such as those used to upload files, and process the request body as it is received.

To generate a unique identifier for each uploaded image, we used time-based UUIDs, which provide a reliable and unique identifier that can be used to reference the uploaded file throughout the rest of the application.

To speed up the processing of the image data, we utilized virtual threads, which are a lightweight form of thread that can be executed in parallel without incurring the overhead of traditional threads. By using virtual threads, we were able to process multiple images concurrently, improving the overall efficiency of the application.

Finally, we stored the processed image data in a database, which allowed us to easily retrieve and use the data in subsequent stages of the application.

Overall, this approach allowed us to efficiently process large image files in real-time, without waiting for the entire file to be uploaded. By leveraging the Spring Web framework, time-based UUIDs, virtual threads, and a database, we were able to create a highly efficient and scalable solution for handling image uploads in our application.

TOOLS USED

Kaggle (Data Science Platform)

Kaggle is a popular platform for data scientists and machine learning engineers, where they can participate in data s cience competitions, access datasets and kernels, and collaborate with other members of the community. With Kaggle, data scientists can showcase their skills and learn from others, while also contributing to real-world problems and challenges.

One of the key features of Kaggle is its extensive library of datasets and competitions, which cover a wide range of topics and industries. This makes it easy for data scientists to find and work on projects that align with their interests and expertise. In addition, Kaggle offers a range of tools and resources, such as kernels and notebooks, which enable data scientists to easily explore and analyze data.

Kaggle is an ideal platform for data scientists and machine learning engineers, especially those who are looking to gain experience and build their skills. Its community-driven approach and emphasis on real-world problems and challenges make it a valuable resource for both beginners and experienced professionals.

PyTorch (Machine Learning Framework)

PyTorch is a popular open-source machine learning framework that is used to build neural networks and deep learning models. With PyTorch, developers can quickly and easily build and train complex models, and deploy them in production environments.

One of the key features of PyTorch is its dynamic computation graph, which enables developers to easily define and modify neural networks on the fly. This makes it easy to experiment with different architectures and hyperparameters, and results in faster development and training times.

PyTorch is an ideal framework for building complex neural networks and deep learning models, especially those that require high flexibility and scalability. Its dynamic computation graph and intuitive API make it a popular choice among PyTorch (Machine Learning Framework)

PyTorch is a popular open-source machine learning framework that is used to build neural networks and deep learning models. With PyTorch, developers can quickly and easily build and train complex models, and deploy them in production environments.

One of the key features of PyTorch is its dynamic computation graph, which enables developers to easily define and modify neural networks on the fly. This makes it easy to experiment with different architectures and hyperparameters, and results in faster development and training times.

PyTorch is an ideal framework for building complex neural networks and deep learning models, especially those that require high flexibility and scalability. Its dynamic computation graph and intuitive API make it a popular choice among machine learning engineers, and its large and active community ensures that it is constantly improving and evolving.

Cacoo by Nulab (Diagramming Tool)

Cacoo by Nulab is a cloud-based diagramming tool that enables teams to create and collaborate on a wide variety of diagrams, such as flowcharts, wireframes, and network diagrams. With Cacoo, teams can easily create professional-looking diagrams without the need for specialized software or technical expertise.

One of the key features of Cacoo is its extensive library of pre-built templates and shapes, which allows users to quickly create diagrams that are tailored to their specific needs. In addition, Cacoo offers a range of collaboration tools, such as real-time commenting and version control, which makes it easy for teams to work together on a single diagram.

Cacoo is an ideal tool for teams that need to create and share diagrams on a regular basis, such as software development teams, project managers, and designers. Its intuitive interface and collaborative features make it an efficient and effective way to create and share visual representations of complex ideas and workflows.

Figma (Design Tool)

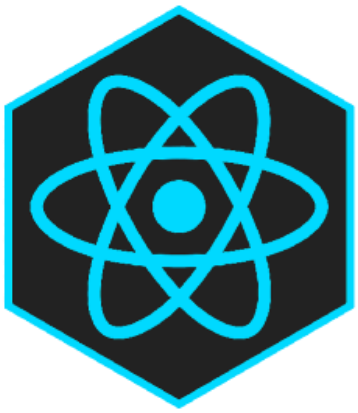
Figma is a cloud-based design tool that enables teams to create, share, and collaborate on digital designs, such as UI designs, wireframes, and prototypes. With Figma, teams can easily create designs that are tailored to their specific needs, and share them with other team members or stakeholders for feedback and review.

One of the key features of Figma is its collaborative design tools, which allow multiple users to work on a single design simultaneously. This makes it easy for teams to work together and share ideas, regardless of their location or time zone. In addition, Figma offers a range of design assets, such as icons and UI kits, which can be used to quickly create professional-looking designs.

Figma is an ideal tool for teams that need to create and share digital designs on a regular basis, such as UX designers, product managers, and developers. Its collaborative features and extensive library of design assets make it a powerful and flexible tool for creating and sharing high-quality designs.

React.js Framework

React.js is an open-source JavaScript framework that is used to build user interfaces (UIs) for web applications. With React.js, developers can create UI components that are reusable, modular, and easy to maintain, which makes it an efficient and effective way to build complex web applications.



One of the key features of React.js is its use of a virtual DOM (Document Object Model), which allows it to efficiently update the UI in response to user interactions or changes in the application state. This results in faster rendering times and improved performance, especially for applications with a large number of UI components.

React.js is an ideal framework for building modern, high-performance web applications, such as single-page applications (SPAs) and progressive web applications (PWAs). Its modular architecture and efficient rendering make it a popular choice among web developers, and its large community of users and contributors ensures that it is constantly improving and evolving.

Java Programming Language

Java is a popular programming language that is used to build a wide range of applications, from web applications to mobile apps and desktop software. Java is known for its reliability, portability, and scalability, which makes it an ideal choice for large-scale projects.

One of the key features of Java is its use of a virtual machine (VM), which allows Java applications to run on any platform that supports the Java VM, without the need for recompilation or modification. In addition, Java offers a range of powerful libraries and frameworks, such as Spring and Hibernate, which make it easy to build complex applications.

Java is an ideal programming language for building large-scale, enterprise-level applications, such as financial systems, enterprise resource planning (ERP) systems, and customer relationship management (CRM) systems. Its reliability, portability, and scalability make it a popular choice among developers, and its extensive library of third-party tools and frameworks ensures that it is constantly evolving and improving.

Spring Boot (Java Framework)

Spring Boot is a popular Java-based framework that is used to build web applications and microservices. With Spring Boot, developers can quickly and easily create production-ready applications that are scalable, reliable, and easy to maintain.

One of the key features of Spring Boot is its ability to auto-configure the application, which eliminates much of the boilerplate code that is typically

required in Java applications. This makes it easier and faster to build and deploy applications, and ensures that they follow best practices and standards.

Spring Boot is an ideal framework for building web applications and microservices, especially those that require high scalability and reliability. Its ease of use, auto-configuration, and extensive library of third-party tools and plugins make it a popular choice among developers, and its large and active community ensures that it is constantly improving and evolving.

IntelliJ IDEA (Java IDE)

IntelliJ IDEA is a powerful integrated development environment (IDE) that is specifically designed for Java development. With IntelliJ IDEA, developers can quickly and easily write, debug, and deploy Java applications, with features such as intelligent code completion, refactoring, and debugging tools.

One of the key features of IntelliJ IDEA is its integration with other development tools and technologies, such as Spring and Hibernate, which makes it easy to work with these frameworks and libraries. In addition, IntelliJ IDEA offers a range of productivity tools, such as code inspections and error highlighting, which help developers write better code faster.

IntelliJ IDEA is an ideal IDE for Java development, especially for larger and more complex projects. Its intelligent code completion, debugging tools, and integration with popular frameworks and libraries make it a popular choice among Java developers, and its constant updates and improvements ensure that it remains at the forefront of Java development tools.

Jupyter (Interactive Computing Environment)

Jupyter is an open-source interactive computing environment that is used for data science, scientific computing, and machine learning. With Jupyter, developers can easily create and share documents that contain live code, equations, visualizations, and narrative text.

One of the key features of Jupyter is its support for a wide range of programming languages, including Python, R, and Julia. This makes it easy for developers to work with the tools and libraries that they are already familiar with, and enables collaboration between different teams and individuals.

Jupyter is an ideal environment for data science and scientific computing, especially those that require interactive exploration and analysis of data. Its support for multiple programming languages, live code, and narrative text make it a powerful tool for both individuals and teams, and its active community ensures that it is constantly improving and evolving.

Azure (Cloud Computing Platform)

Azure is a popular cloud computing platform that is used to build, deploy, and manage applications and services. With Azure, developers can easily scale their applications and services to meet changing demands, while also benefiting from a range of tools and services, such as virtual machines, databases, and machine learning.

One of the key features of Azure is its support for multiple programming languages and frameworks, including Java, Python, and .NET. This makes it easy for developers to work with the tools and technologies that they are already familiar with, and ensures that they can quickly and easily deploy their applications and services to the cloud.

Azure is an ideal platform for building and deploying applications and services in the cloud, especially those that require high scalability and reliability. Its support for multiple programming languages and frameworks, as well as its extensive range of tools and services, make it a popular choice among developers and organizations of all sizes.

Chapter one

1 INTRODUCTION

1.1 ALGORITHMS USED

The following section is not final and is subject to change, as it is impossible to determine the best architecture to use without experimentation. We will use an iterative approach to build our model, starting with a simple model and slowly adding data and increasing the model's complexity, as long as the performance of our model keeps increasing then we are going in the right direction, if it starts decreasing then the cost of reverting the model to its previous state or changing the model entirely is minimized because we are working in small steps. We will take the previous approach to its conclusion, but there are other methods than building our model from scratch, we can use transfer learning with a pre-trained model, this is because our resources are limited, and we cannot afford to train a network with trillions of parameters like google net or efficient net. Using transfer learning we will only have to re-train the last few layers of the network while keeping the performance of a huge network trained on giant datasets. If our dataset is still too large, we can employ dimensionality reduction techniques, initially, we think Sparse Auto-Encoder (SAE) would be the best option, because unlike Principal Component Analysis (PCA), which will decrease the size of our data but will have a high information loss rate, SAE will not have the same amount of information loss. Note that this is an initial assessment and is subject to change. Another thing we can use is ensemble learning, specifically boosting, which is chaining multiple independent networks together to produce a more powerful model. Note that this approach cannot be used at the beginning because it requires that there be diversity in error rates, which we will only know after we use single networks first. When boosting is used, every model focuses on the previous model's mistakes and tries to produce a better result, thus it is guaranteed that at the model numbered i+1, the performance is always better than or equal to the performance at the *i*-th model.

1.2 DATASETS

Pneumonia:

- https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia
- https://www.kaggle.com/code/madz2000/pneumonia-detection-using-cnn-92-6-accuracy

COVID-19:

- https://www.kaggle.com/datasets/bachrr/covid-chest-xray
- https://github.com/ieee8023/covid-chestxray-dataset

Tuberculous:

- https://www.kaggle.com/datasets/tawsifurrahman/tuberculosis-tb-chest-xray-dataset

Emphysema:

- https://www.kaggle.com/datasets/kmader/pulmonary-chest-xray-abnormalities

Chapter two

2.1 LITERATURE REVIEW AND RELATED WORKS

2.1.1 Deep learning for detecting pulmonary tuberculosis via chest radiography [4]:

The purpose of this study was to aid doctors in interpreting CXRs, with a focus on TB. The authors gathered data from 9 countries to generalize the model and prevent overfitting, they chose to optimize their model for sensitivity (true positive rate) over specificity (true negative rate); to maximize patient benefit and minimize the risk of missing a positive case which can be life-threatening, and because CXRs are used more as a confirmatory test. Note that it is impossible to reach both high sensitivity and specificity because they are inversely correlated [5]. The authors used a 2-step process, they used mask RCNN and Resnet101 to make a bounding box around the lungs, then they used Single Shot Multibox Detector (SSD) to detect the TB in the image, using a pre-trained efficient net and attention-pooling.

2.1.2 Tuberculosis detection using deep learning and contrast-enhanced canny edge detected X-Ray images [6]:

This study also focused on TB, the authors used ensemble deep learning, which is using multiple architectures instead of just one to achieve better results [7], unlike previous work which used ensemble deep learning, the authors of this study trained their models on a diverse range of errors, instead of similar features like previous studies. Another reason for using deep learning is because previous work had found lung cancer is similar looking to TB enough to cause misclassifications in otherwise good models and human radiologists.

2.1.3 Deep learning based detection of COVID-19 from chest X-ray images [8]:

Covid-19 wreaked havoc on health systems globally, causing untold financial damage and loss of life. This study focuses on DL to mitigate the harmful effects of covid-19. They gathered public data for both covid-positive and covid-negative CXRs, then they used data augmentation techniques such as rotation, flipping, and adding noise to enlarge their dataset. The researchers experimented with 3 different DL architectures, Resnet50, Inception V3, and VGG16, and found that VGG performed the best with >98% accuracy. It is worth noting that these researchers also used transfer learning, similar to the researchers in [4], emphasizing its strength and fast training and deployment times.

2.1.4 Pneumonia Diagnosis on Chest X-Rays with Machine Learning [9]:

This study applied 20 algorithms on Pneumonia detection, the authors found that 50,000 people die of Pneumonia every year, so they decided to use a novel mechanism that uses feature extraction to improve performance, and dimensionality reduction to reduce training times. This mechanism was found to be more performant than Mobile Net, Exception Net, and Resnet. The researchers placed a heavy focus on data preprocessing, they used clustering and machine learning algorithms, just for the preprocessing, such as support vector machine, KNN,

naïve Bayes, and random forest. Then they used multiple DL architectures chained together such as VGG Net, Google Net, Res Net, Dense Net, and Inception Net, a practice known as "boosting" [10], fed them the preprocessed data and studied the results, they found that their proposed mechanism produced the highest accuracy, and that famous DL architectures are not well-suited to the task of Pneumonia detection in environments with constrained computational power.

2.1.5 Prediction of Obstructive Lung Disease from Chest Radiographs via Deep Learning Trained on Pulmonary Function Data [11]:

This study focused on Chronic Obstructive Pulmonary Disease (COPD), COPD is recommended to be diagnosed with computed tomography (CT), not CXRs which are the focus of our current work, however only 4% of eligible patients receive CT scans [12], so it would be greatly beneficial if a system that used CXRs could be developed, this is what the authors attempted to do. They used a CNN model trained on CXRs annotated with Pulmonary Function Test (PFT) data and compared it with the performance of an NLP model trained on radiologist text reports. They used a pre-trained (transfer learning) Resnet network and used AUC and ROC to compare the results, they found that the CNN had better performance with an AUC of 0.814, while the NLP model had an AUC of 0.704. The CNN model could also differentiate between the severity of the disease with an AUC of 0.837 compared to the NLP model with an AUC of 0.770.

2.2 COMPARATIVE ANALYSIS

Study	Published on	Dataset	Algorithms used	Accuracy
Hayelny	2022-23	5,863 Chest X-ray images	Channel-wise normalization Weighted loss function Weight decay Learning scheduler Data Augmentation CNN Architecture: Resnet18	88%
[6]	December 2020	Montgomery dataset (138 images) Shenzhen dataset (662 images)	Canny Algorithm [13]	93.59%
[8]	July 2021	3000 images for normal chest X-rays were selected from different public image databases. 623 chest X-ray COVID-19 images were collected from the GitHub repository, expanded to 2000 by image augmentation	VGG16 [14] ResNet50 [15] InceptionV3[16] Transfer learning [17]	>98%
[4]	May 2021	550,297 images	Mask RCNN28 with a ResNet-101-FPN. Single Shot MultiBox Detector EfficientNet-B7 with an attention pooling layer and a fully-connected layer.	Sensitivity: 88% Specificity: 79%
[9]	2021	5,863 Chest X-ray images	SIFT Feature Scale-space extreme value detection Construct scale space Keypoint positioning and feature description. Vector normalization to generate descriptors. Feature clustering algorithm: - K-means ++ - Bag of Visual Words - Support Vector Machine - K-Nearest Neighbor - Naive Bayes - Random Forest Deep Neural Networks: - VGGNet - GoogleNet - ResNet - DenseNet - Inception - MobileNet - Xception	76.4%
[11]	Jan 2021	6749 two-view chest radiographs	Resnet18 pre-trained with ImageNet CNN	74%

Chapter three

METHODOLOGY

Splitting the data:
Starting with our complete dataset, we divide it randomly into three folders, training, testing, and validation, with a ratio of 70:15:15 respectively.

Verify validity of data:
Read a sample of the training data, and visualize it with its labels to verify its reliability and provide a visual view of the data.

Batching:
Because the size of our data is too large to store in memory, we must divide our data into batches of 64 images each, so we can work in blocks.

Preprocessing:
We will use standard data augmentation techniques, such as rotation, horizontal flipping, and vertical flipping, this is done to prevent overfitting and to familiarize our model with varied data.

Training:
We will use the transfer-learning technique, we will download a pre-trained neural network, freeze all its layers except the last 3-5 layers, we will train those layers. This is done to take advantage of giant models trained on large amounts of data, while simultaneously saving on compute power, and customizing to our use case.

Evaluation and tuning:
We will evaluate and tune our model until it reaches our desired accuracy (~90%).

Deployment:
We will deploy our model using FastAPI so it will become available for consumption over the web.

Chapter four

REFERENCES

[1] "Radiology Facing a Global Shortage." https://www.rsna.org/news/2022/may/Global-Radiologist-Shortage (accessed Nov. 09, 2022).

[2] "World Bank and WHO: Half the world lacks access to essential health services, 100 million still pushed into extreme poverty because of health expenses," *World Bank*. https://www.worldbank.org/en/news/press-release/2017/12/13/world-bank-who-half-world-lacks-access-to-essential-health-services-100-million-still-pushed-into-extreme-poverty-because-of-health-expenses (accessed Nov. 09, 2022).

[3] "WHO consolidated guidelines on tuberculosis: module 2: screening: systematic screening for tuberculosis disease." https://www.who.int/publications-detail-redirect/9789240022676 (accessed Nov. 09, 2022).

[4] S. Kazemzadeh *et al.*, "Deep Learning Detection of Active Pulmonary Tuberculosis at Chest Radiography Matched the Clinical Performance of Radiologists," *Radiology*, p. 212213, Sep. 2022, doi: 10.1148/radiol.212213.

[5] J. Shreffler and M. R. Huecker, "Diagnostic Testing Accuracy: Sensitivity, Specificity, Predictive Values and Likelihood Ratios," in *StatPearls*, Treasure Island (FL): StatPearls Publishing, 2022. Accessed: Nov. 10, 2022. [Online]. Available: http://www.ncbi.nlm.nih.gov/books/NBK557491/

[6] S. Hwa, A. Bade, M. Hijazi, and M. Jeffree, "Tuberculosis detection using deep learning and contrastenhanced canny edge detected X-Ray images," *IAES Int. J. Artif. Intell. IJ-AI*, vol. 9, p. 713, Dec. 2020, doi: 10.11591/ijai.v9.i4.pp713-720.

[7] M. A. Ganaie, M. Hu, A. K. Malik, M. Tanveer, and P. N. Suganthan, "Ensemble deep learning: A review," *Eng. Appl. Artif. Intell.*, vol. 115, p. 105151, Oct. 2022, doi: 10.1016/j.engappai.2022.105151.

[8] S. Guefrechi, M. B. Jabra, A. Ammar, A. Koubaa, and H. Hamam, "Deep learning based detection of COVID-19 from chest X-ray images," *Multimed. Tools Appl.*, vol. 80, no. 21–23, pp. 31803–31820, 2021, doi: 10.1007/s11042-021-11192-5.

[9] Z. Meng, L. Meng, and H. Tomiyama, "Pneumonia Diagnosis on Chest X-Rays with Machine Learning," *Procedia Comput. Sci.*, vol. 187, pp. 42–51, Jan. 2021, doi: 10.1016/j.procs.2021.04.032.

[10] W. Gao and Z.-H. Zhou, "On the doubt about margin explanation of boosting," *Artif. Intell.*, vol. 203, pp. 1–18, Oct. 2013, doi: 10.1016/j.artint.2013.07.002.

[11] J. D. Schroeder *et al.*, "Prediction of Obstructive Lung Disease from Chest Radiographs via Deep Learning Trained on Pulmonary Function Data," *Int. J. Chron. Obstruct. Pulmon. Dis.*, vol. 15, p. 3455, 2020, doi: 10.2147/COPD.S279850.

[12] M. Va, "Screening for lung cancer: U.S. Preventive Services Task Force recommendation statement," *Ann. Intern. Med.*, vol. 160, no. 5, Mar. 2014, doi: 10.7326/M13-2771.

[13] J. Li and S. Ding, "A Research on Improved Canny Edge Detection Algorithm," in *Applied Informatics and Communication*, Berlin, Heidelberg, 2011, pp. 102–108. doi: 10.1007/978-3-642-23223-7_13.

[14] M. ul Hassan, "VGG16 - Convolutional Network for Classification and Detection," Nov. 20, 2018. https://neurohive.io/en/popular-networks/vgg16/ (accessed Nov. 09, 2022).

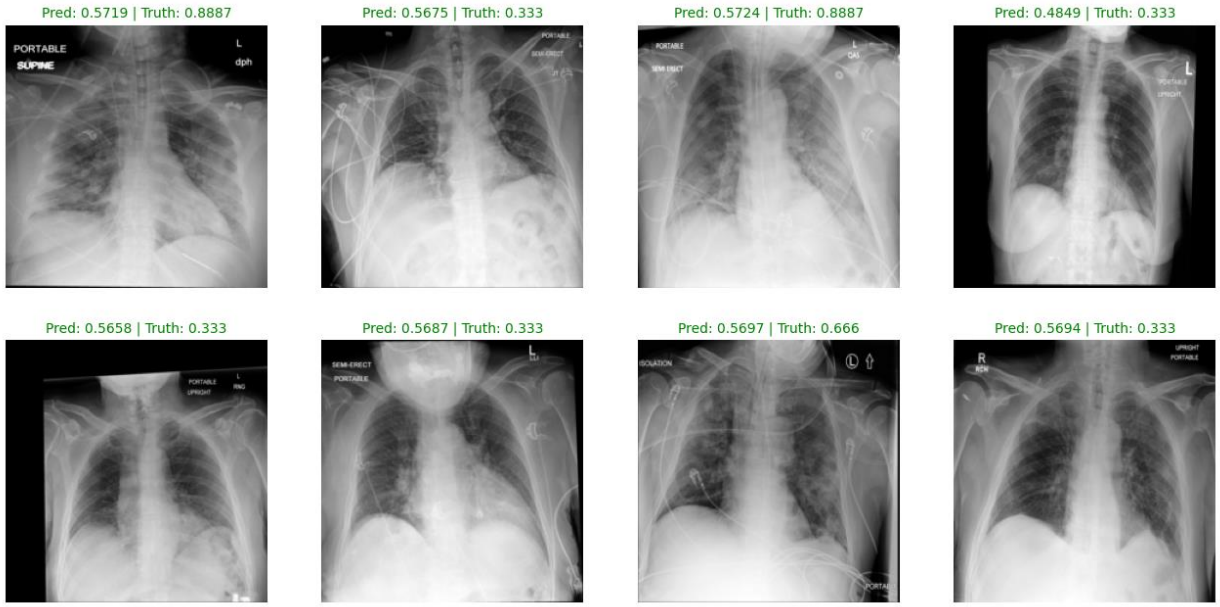
[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition." arXiv, Dec. 10, 2015. doi: 10.48550/arXiv.1512.03385.

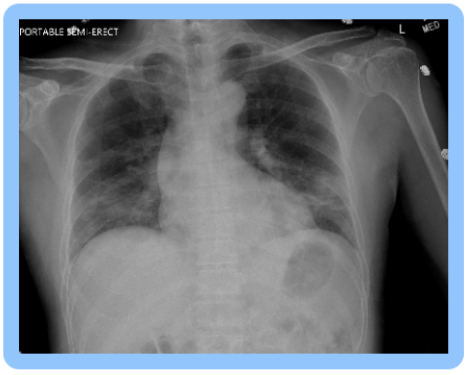
[16] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision." arXiv, Dec. 11, 2015. doi: 10.48550/arXiv.1512.00567.

[17] J. Brownlee, "A Gentle Introduction to Transfer Learning for Deep Learning," *Machine Learning Mastery*, Dec. 19, 2017. https://machinelearningmastery.com/transfer-learning-for-deep-learning/ (accessed Nov. 09, 2022).

APPENDIX

OUTPUT





Results

You have pneumonia.
You should consult a doctor immediately.

Confidence: 86.04%

Diagnosis id: 103

Diagnosis status: Completed

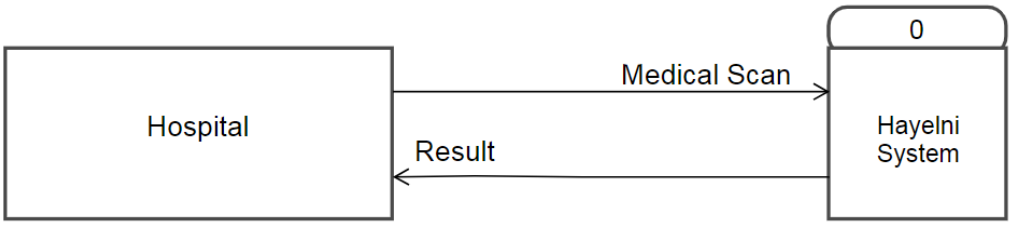
Disease: Bacterial Pneumonia

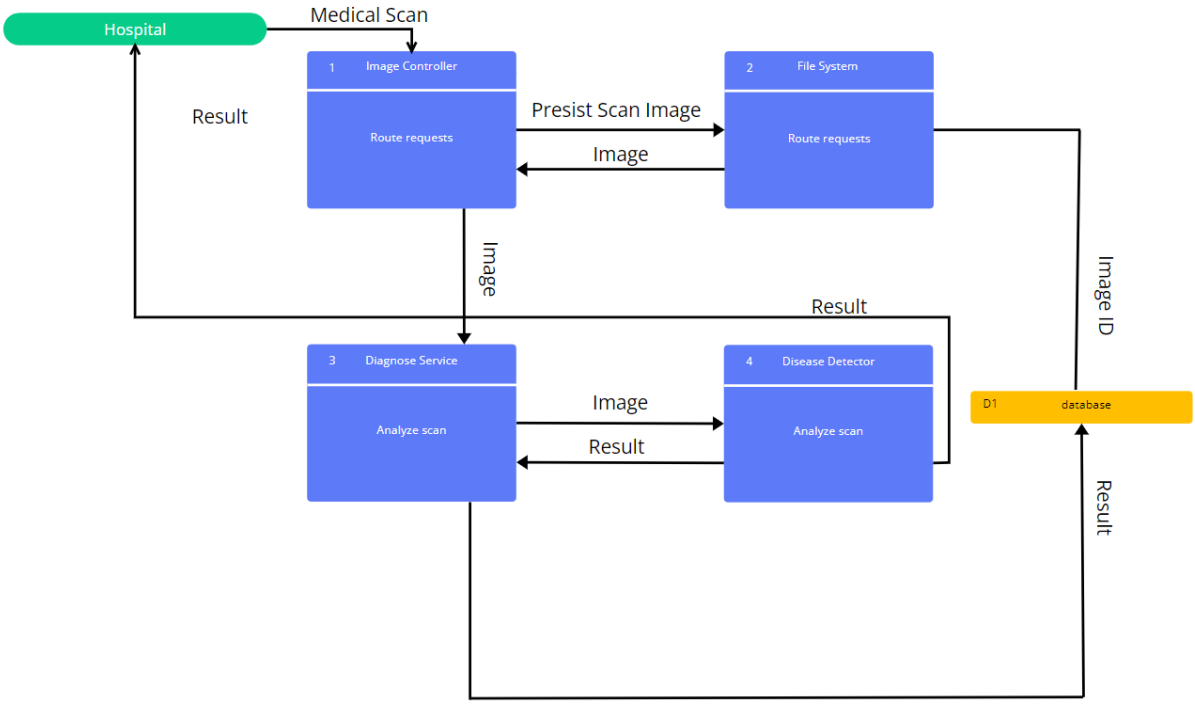
Judgement: Pneumonia Positive

Severity: 56.96%

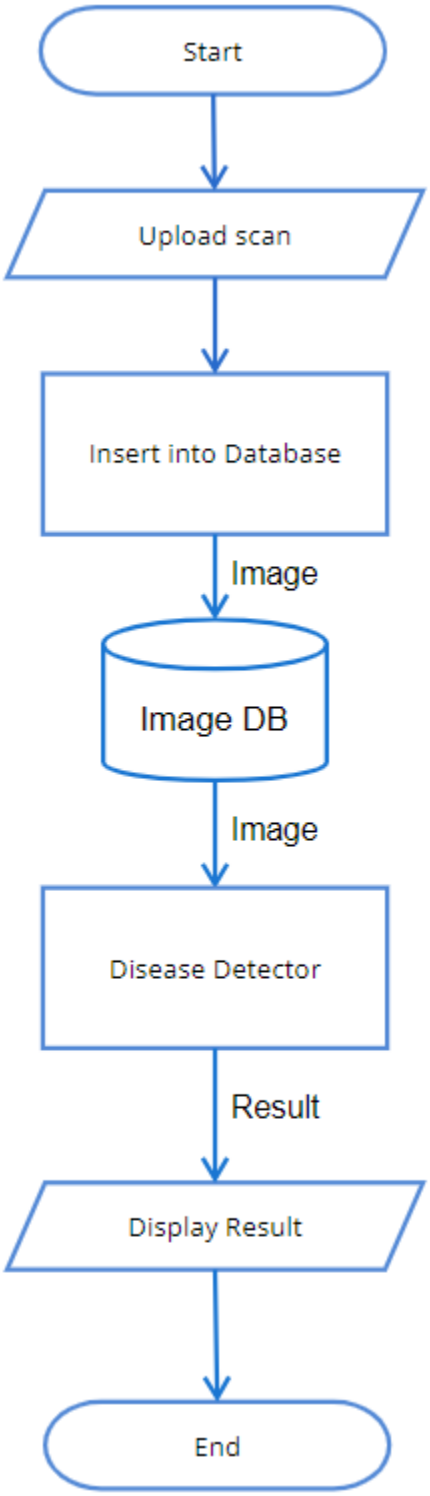
SYSTEM DESIGN

Conext and DFD

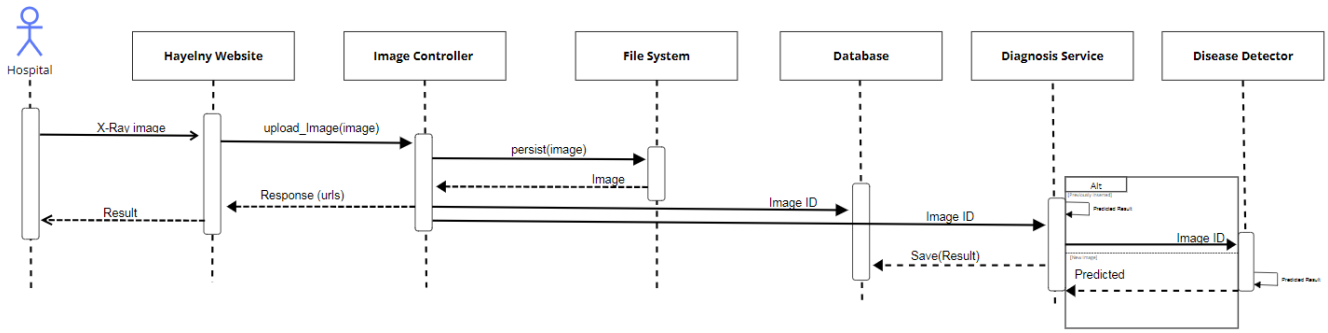




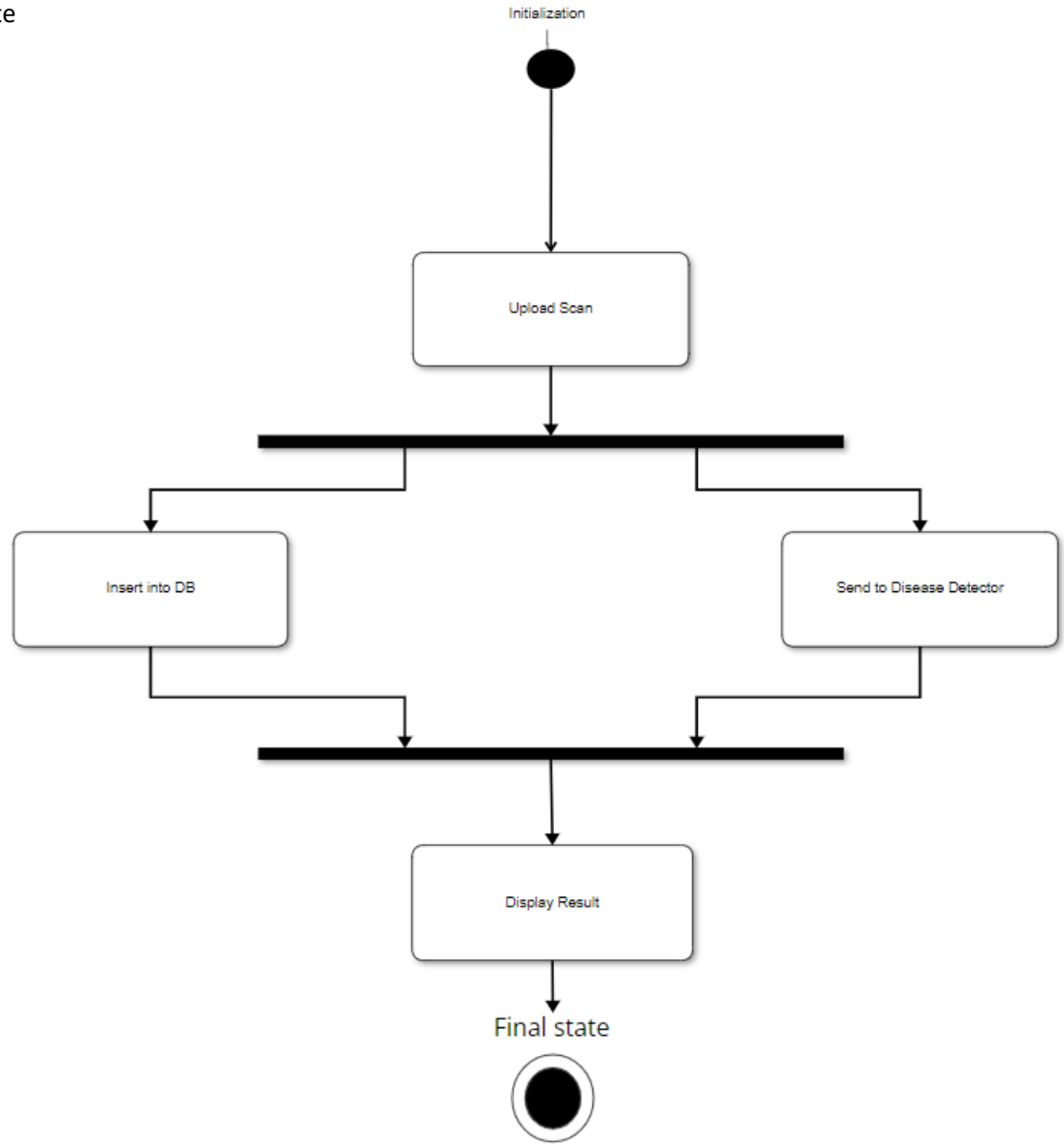
Process


Start
Upload scan
Insert into Database
Image
Image DB
Image
Disease Detector
Result
Display Result
End

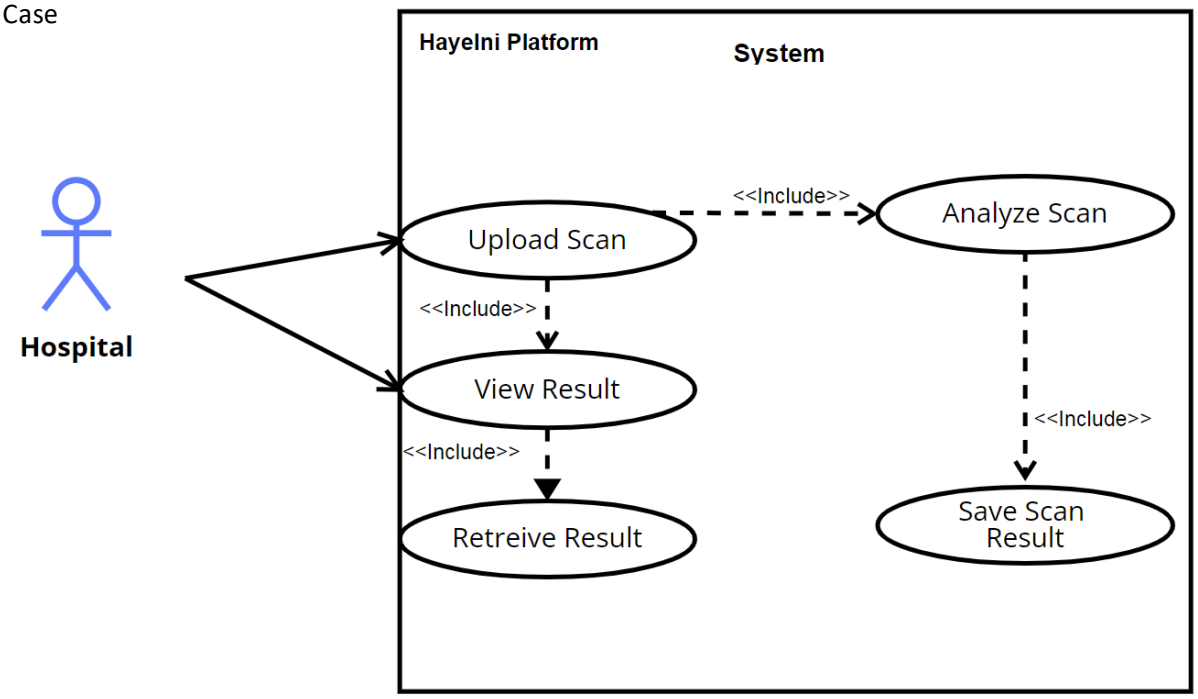
Sequence


Hospital
Hayelny Website
Image Controller
File System
Database
Diagnosis Service
Disease Detector
X-Rav image
upload_Image(image)
persist(image)
Image
Response (urls)
Result
Image ID
Image ID
Image ID
Save(Result)
Predicted

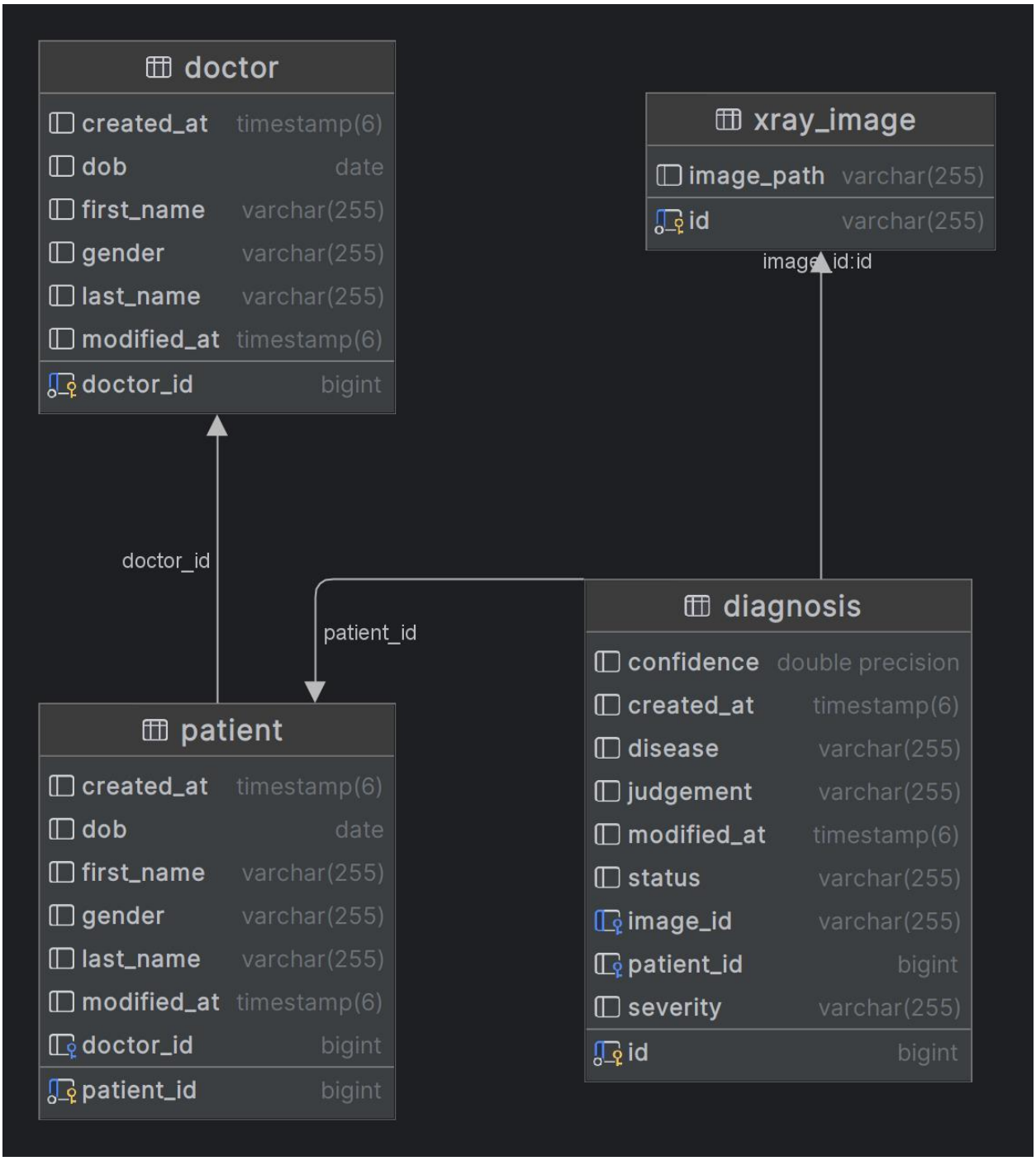
State


Initialization
Upload Scan
Insert into DB
Send to Disease Detector
Display Result
Final state

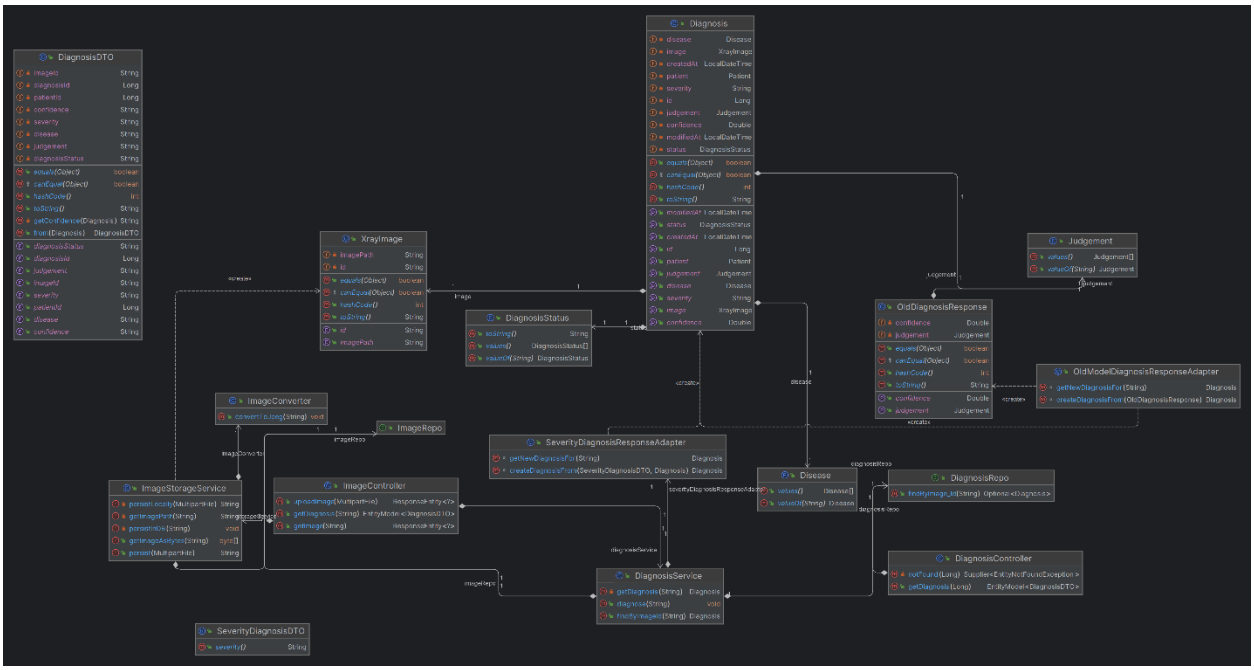
Use Case


Hayelni Platform
System
Hospital
Upload Scan
<<Include>>
Analyze Scan
<<Include>>
View Result
<<Include>>
<<Include>>
Retreive Result
Save Scan Result

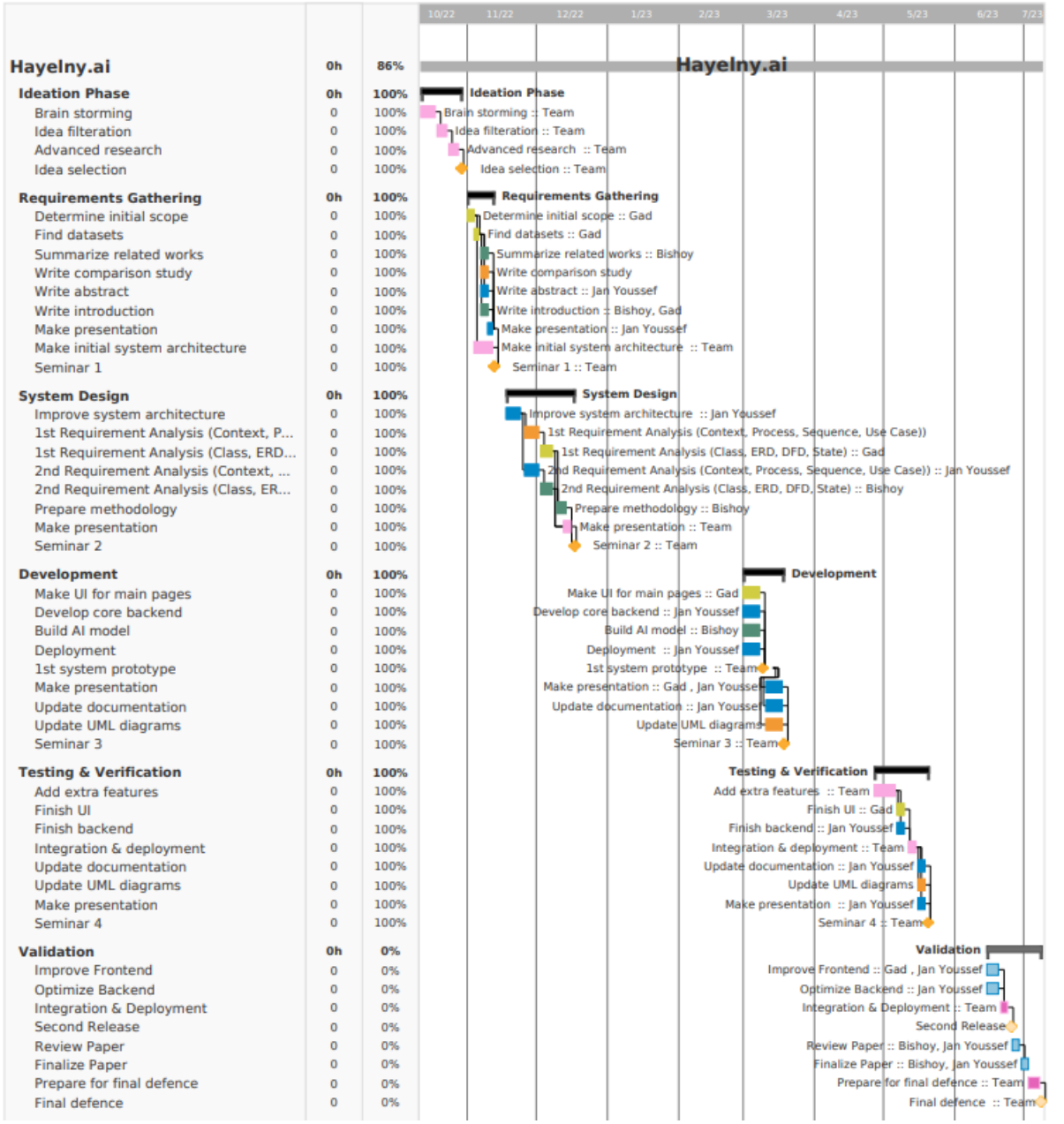
Entity Relationship



Class


DiagnosisDTO
Diagnosis
XrayImage
DiagnosisStatus
Judgement
OldDiagnosisResponse
OldModelDiagnosisResponseAdapter
ImageConverter
ImageRepo
SeverityDiagnosisResponseAdapter
Disease
DiagnosisRepo
ImageStorageService
ImageController
DiagnosisService
DiagnosisController
SeverityDiagnosisDTO

TIMELINE


10/22
11/22
12/22
1/23
2/23
3/23
4/23
5/23
6/23
7/23
Hayelny.ai 0h 86%
Ideation Phase 0h 100%
Brain storming 0 100%
Idea filteration 0 100%
Advanced research 0 100%
Idea selection 0 100%
Requirements Gathering 0h 100%
Determine initial scope 0 100%
Find datasets 0 100%
Summarize related works 0 100%
Write comparison study 0 100%
Write abstract 0 100%
Write introduction 0 100%
Make presentation 0 100%
Make initial system architecture 0 100%
Seminar 1 0 100%
System Design 0h 100%
Improve system architecture 0 100%
1st Requirement Analysis (Context, P... 0 100%
1st Requirement Analysis (Class, ERD... 0 100%
2nd Requirement Analysis (Context, ... 0 100%
2nd Requirement Analysis (Class, ER... 0 100%
Prepare methodology 0 100%
Make presentation 0 100%
Seminar 2 0 100%
Development 0h 100%
Make UI for main pages 0 100%
Develop core backend 0 100%
Build AI model 0 100%
Deployment 0 100%
1st system prototype 0 100%
Make presentation 0 100%
Update documentation 0 100%
Update UML diagrams 0 100%
Seminar 3 0 100%
Testing & Verification 0h 100%
Add extra features 0 100%
Finish UI 0 100%
Finish backend 0 100%
Integration & deployment 0 100%
Update documentation 0 100%
Update UML diagrams 0 100%
Make presentation 0 100%
Seminar 4 0 100%
Validation 0h 0%
Improve Frontend 0 0%
Optimize Backend 0 0%
Integration & Deployment 0 0%
Second Release 0 0%
Review Paper 0 0%
Finalize Paper 0 0%
Prepare for final defence 0 0%
Final defence 0 0%
Hayelny.ai
Ideation Phase
Brain storming :: Team
Idea filteration :: Team
Advanced research :: Team
Idea selection :: Team
Requirements Gathering
Determine initial scope :: Gad
Find datasets :: Gad
Summarize related works :: Bishoy
Write comparison study
Write abstract :: Jan Youssef
Write introduction :: Bishoy, Gad
Make presentation :: Jan Youssef
Make initial system architecture :: Team
Seminar 1 :: Team
System Design
Improve system architecture :: Jan Youssef
1st Requirement Analysis (Context, Process, Sequence, Use Case))
1st Requirement Analysis (Class, ERD, DFD, State) :: Gad
2nd Requirement Analysis (Context, Process, Sequence, Use Case)) :: Jan Youssef
2nd Requirement Analysis (Class, ERD, DFD, State) :: Bishoy
Prepare methodology :: Bishoy
Make presentation :: Team
Seminar 2 :: Team
Development
Make UI for main pages :: Gad
Develop core backend :: Jan Youssef
Build AI model :: Bishoy
Deployment :: Jan Youssef
1st system prototype :: Team
Make presentation :: Gad , Jan Youssef
Update documentation :: Jan Youssef
Update UML diagrams
Seminar 3 :: Team
Testing & Verification
Add extra features :: Team
Finish UI :: Gad
Finish backend :: Jan Youssef
Integration & deployment :: Team
Update documentation :: Jan Youssef
Update UML diagrams
Make presentation :: Jan Youssef
Seminar 4 :: Team
Validation
Improve Frontend :: Gad , Jan Youssef
Optimize Backend :: Jan Youssef
Integration & Deployment :: Team
Second Release
Review Paper :: Bishoy, Jan Youssef
Finalize Paper :: Bishoy, Jan Youssef
Prepare for final defence :: Team
Final defence :: Team

التعرف على الالتهاب الرئوي في في الأشعة الصدرية باستخدام التعلم العميق

مقدم من:

جان يوسف	89477
محمد أسامة	89402
محمد طارق	89610
بيشوي سمير	89534

تحت إشراف:

الدكتور: خالد الششتاوي

م. أية محمود

م. زياد خالد

MUST - جامعة مصر للعلوم والتكنولوجيا

CAIT - كلية الحاسبات وتقنيات الذكاء االصطناعي

CS - قسم علوم الحاسب