

PROJECT REPORT

INTRODUCTION

Project Overview

The COVID-19 pandemic has had a significant impact on healthcare systems worldwide. Early diagnosis and isolation of infected individuals are crucial in controlling the spread of the virus. Chest X-rays are readily available and inexpensive, making them a potential tool for COVID-19 detection. This project explores the use of deep learning algorithms to analyze chest X-rays and automatically detect COVID-19.

Purpose

- **Early diagnosis and isolation:** Early detection of COVID-19 is crucial to control the spread of the virus. X-rays can provide valuable information about the lungs, which are often affected by COVID-19. This information can help doctors diagnose the disease early, allowing patients to be isolated and receive treatment sooner. This, in turn, helps to prevent the virus from spreading to others.
- **Triage and resource allocation:** X-rays can be used to quickly and efficiently triage patients suspected of having COVID-19. This helps to ensure that resources, such as hospital beds and ventilators, are directed to those who need them the most.
- **Monitoring disease progression:** X-rays can be used to monitor the progression of COVID-19 in patients. This helps doctors to determine whether the disease is improving or worsening, and to adjust treatment plans accordingly.
- **Research and development:** X-rays are valuable tools for researchers who are studying COVID-19. By analyzing X-rays from patients with COVID-19, researchers can learn more about the disease and develop new diagnostic tools and treatments.
- **Widely available:** X-ray machines are widely available in hospitals and clinics around the world. This makes X-rays a readily accessible tool for COVID-19 detection.
- **Relatively inexpensive:** X-rays are a relatively inexpensive imaging test compared to other methods, such as CT scans.
- **Fast results:** X-ray results can be obtained quickly, which is important for early diagnosis and treatment.
- **Non-invasive:** X-rays are a non-invasive imaging test, which means that they do not require any surgery or injections.

LITERATURE SURVEY

Existing problem :

- **Limited sensitivity and specificity:** X-rays often show similar abnormalities in both COVID-19 and other respiratory illnesses like pneumonia. This can lead to false positives and negatives, making it difficult to rely solely on X-rays for diagnosis.
- **Lack of standardized interpretation:** X-ray interpretation can be subjective and vary between radiologists, leading to inconsistencies in diagnosis. This can be particularly problematic in resource-limited settings where expert radiologists are scarce.
- **Limited data availability:** Training deep learning models for accurate COVID-19 detection requires large datasets of labeled X-ray images. However, access to such datasets can be restricted due to privacy concerns and lack of standardization.
- **Ethical considerations:** Using AI models for medical diagnosis raises ethical concerns regarding bias, transparency, and accountability. It is crucial to ensure that such models are developed and deployed responsibly, taking into account ethical considerations.

- **Technical limitations:** Current deep learning models for X-ray interpretation are computationally expensive and require specialized hardware. This can limit their accessibility and deployment in resource-limited settings.
- **Limited clinical validation:** While initial research shows promising results, most AI models for COVID-19 detection using X-rays lack robust clinical validation. This means their effectiveness in real-world clinical settings needs further evaluation.
- **Overreliance on technology:** X-rays should not be solely relied upon for COVID-19 diagnosis. They should be used as a complementary tool alongside other diagnostic tests and clinical evaluation.
- **Evolving nature of the virus:** The virus causing COVID-19 is constantly evolving, and its X-ray manifestations may also change over time. This necessitates continuous monitoring and adaptation of AI models to ensure their accuracy.

References :

Internet

Problem Statement Definition :

This problem statement addresses the development of a deep learning-based system for automated and accurate detection of COVID-19 using chest X-rays.

Specific objectives are

- Develop a robust deep learning model capable of accurately distinguishing COVID-19 positive chest X-rays from negative and other respiratory infections.
- Achieve high sensitivity and specificity to minimize false positives and negatives.
- Design the system to be computationally efficient and scalable for deployment in resource-limited settings.
- Ensure interpretability of the model's predictions to enhance trust and facilitate clinical decision making.

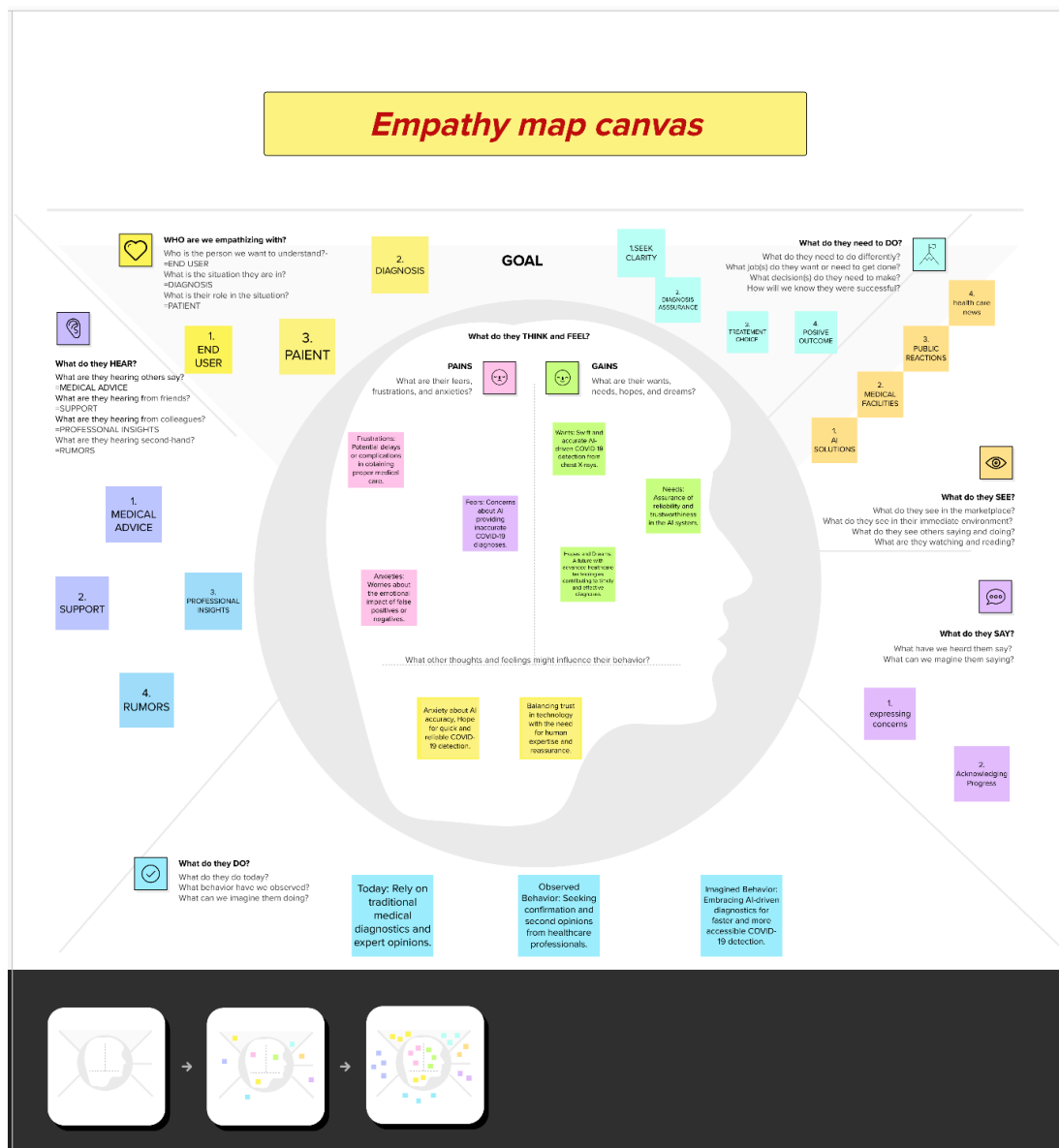
The successful development of such a system would offer significant benefits

- Early and accurate detection of COVID-19, leading to improved patient outcomes.
- Reduced reliance on RT-PCR tests, freeing up valuable resources for other critical needs.
- Increased accessibility and affordability of COVID-19 diagnosis, particularly in resource-limited settings.
- Enhanced public health surveillance and control of the pandemic.

IDEATION & PROPOSED SOLUTION

Empathy Map Canvas :

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes. It is a useful tool to help teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.




Ideation and Brainstorming :

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions. Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

Step-1: Team Gathering, Collaboration and Select the Problem Statement

Template



Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

10 minutes to prepare
1 hour to collaborate
2-8 people recommended

➦

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

➦

Team gathering

Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

➦

Set the goal

Think about the problem you'll be focusing on solving in the brainstorming session.

➦

Learn how to use the facilitation tools

Use the Facilitation Superpowers to run a happy and productive session.

[Open article](#)

1

Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

PROBLEM

How might we Summarize political speeches?

"Unreading Sentiments in Political Speeches" explores the intricate emotions and underlying intentions woven into the fabric of political discourses. This study delves into the art of persuasion, decoding the subtle cues, tones, and word choices employed by speakers to sway public opinion. By analyzing speeches through a socio-emotional lens, this research sheds light on how leaders manipulate sentiment to advance their agendas, fostering a deeper understanding of the complex interplay between language, emotion, and politics.

Key rules of brainstorming

To run an smooth and productive session

Stay in topic.

Encourage wild ideas.

Defer judgment.

Listen to others.

Go for volume.

If possible, be visual.

Step-2: Brainstorm, Idea Listing and Grouping

2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

💡

You can select a sticky note and hit the pencil button to start drawing!

AYUSH

Classification of COVID-19 from chest x-rays

multi-class classification of covid -19 from x rays

Detection of COVID-19 lesions in chest X-rays

ADARSHA

Transfer learning for COVID-19 detection

Addressing data imbalance in COVID-19 CXR datasets

Evaluation of deep learning models for COVID-19 detection

MANSOOR

Clinical implementation of deep learning-based COVID-19 detection systems

Ethical considerations in deep learning-based COVID-19 detection

Differentiation of COVID-19 from other respiratory conditions

SATHWIK

Prediction of COVID-19 progression

Detection of COVID-19 reinfection

Personalized risk assessment for COVID-19 complications

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

🕒 20 minutes

TIP

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

Technical

Use standardized imaging protocols to ensure consistency in image quality and acquisition parameters.

Employ high-resolution X-ray machines to capture detailed lung structures and potential COVID-19 lesions.

Ensure proper patient positioning and alignment during image acquisition to facilitate accurate image analysis.

Analytical

Utilize publicly available datasets, such as the COVID-19 Image Data Collection (IDC) and the RSNA Pneumonia Detection Challenge dataset.

Collaborate with hospitals and healthcare institutions to collect high-quality chest X-ray data with accurate labeling.

Ensure data diversity by representing various patient demographics, disease stages, and X-ray acquisition parameters.

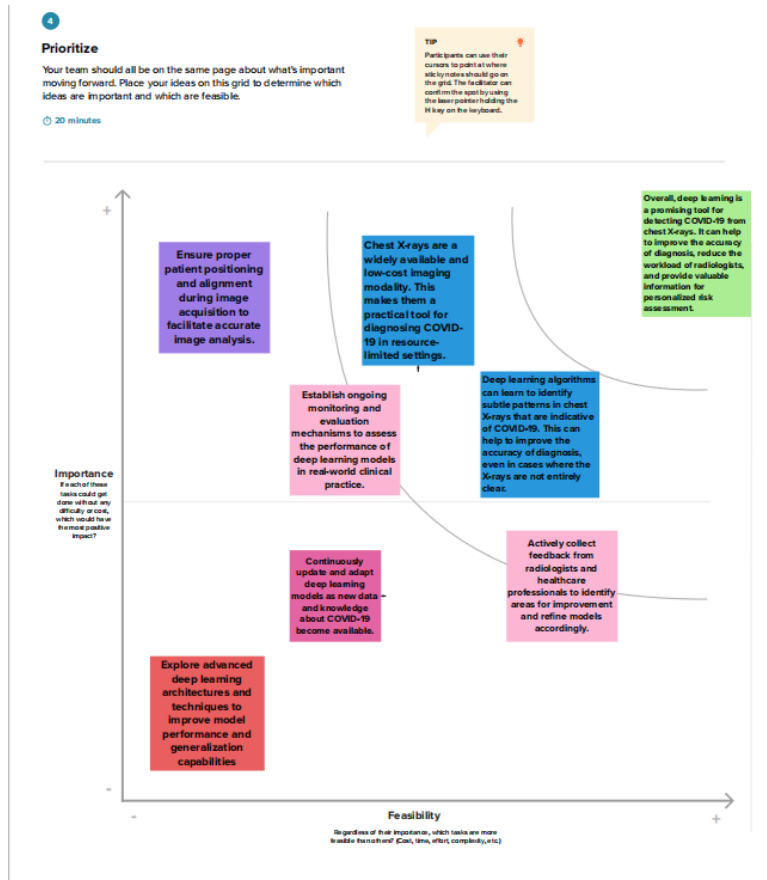
Research

Deep learning models can achieve high accuracy in classifying chest X-rays as COVID-positive, COVID-negative, and other respiratory infections. Several studies have reported classification accuracies exceeding 95%, indicating the potential of deep learning for aiding radiologists in COVID-19 diagnosis.

Transfer learning strategies can effectively adapt pre-trained deep learning models to the COVID-19 detection task. By leveraging knowledge acquired from large-scale image datasets, transfer learning can significantly reduce training time and improve model performance.

Data augmentation techniques can address the issue of data imbalance in COVID-19 chest X-ray datasets. By artificially expanding the dataset with realistic variations of existing images, data augmentation can enhance model generalization and improve performance.

Step-3: Idea Prioritization



REQUIREMENT ANALYSIS

Functional requirement

S.NO.	Component	Description	Technology
1.	Image Acquisition	Acquire or gather pictures depicting Human Lungs images	Image data-bases, smart phones or Digital cameras.
2.	Pre-Processing	preprocess the images for recognition by applying a range of transformations and enhancements.	open CV(Computer Vision Library),python.
3.	Image Segmentation	Isolate hand signs from the background and, if required, identify individual fingers for enhanced recognition.	open CV, Image processing Techniques.
4.	Feature Extraction	Capture relevant characteristics from the segmented images, encompassing aspects like shape, color and texture.	Feature extraction algorithms (e.g., Histogram of Oriented Gradients, Color Histograms), Python.
5.	Machine Learning Model	Develop a machine learning model capable of identifying X-ray images using features extracted from the data.	Tensor Flow, PyTorch, Scikit-Learn, Keras, or a custom model using deep learning or traditional machine learning algorithms.

6.	Training Data	A dataset of labelled X-ray images for model training	Human lungs X-ray images, data augmentation techniques.
7.	Model Evaluation	Evaluate the model's performance by examining metrics such as accuracy, precision, recall, F1 score, and other pertinent indicators	Cross-validation, evaluation metrics in Python.
8.	Model Deployment	Deploy the trained model for the realtime or batch processing of COVID-19 X-ray images in a deployed environment.	Cloud platforms (e.g., AWS, Azure, GCP), web servers, APIs.
9.	User Interface	Design a user-friendly interface that allows users to interact effortlessly with COVID-19 detection system.	Web development (HTML, CSS, JavaScript),
10.	Infrastructure (Server / Cloud)	Consistently enhance and refine the system through the collection of user feedback, ensuring regular updates to improve the model.	Agile development practices, versioncontrol (e.g., Git).

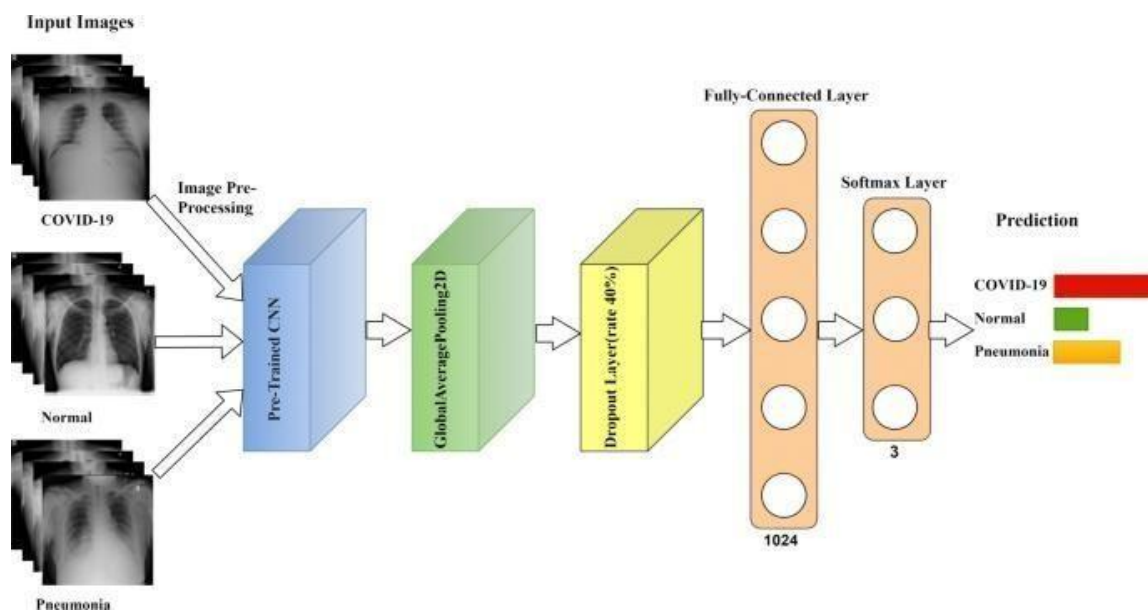
Non-Functional requirements

S.NO.	Characterstics	Description	Technology
1.	open-Source Frameworks	Using open-source frameworks can greatly expedite development, cut down costs, and take advantage of a collaborative community for mutual benefits.	Python for machine learning and image processing (NumPy, OpenCV, Scikit-Learn),
2.	Security Implementations	Ensuring the security of both user data and the system is paramount. Implement a range of security measures to safeguard against data breaches and unauthorized access.	SL/TLS for secure data transmission and implement encryption for data at rest
3.	Scalable Architecture	Architecting the system to accommodate growing loads and user demands by scaling horizontally or vertically as required.	Docker for packaging applications and Kubernetes for container orchestration, Autoscaling on cloud platforms to dynamically allocate resources based on demand, tools like Nginx or HA Proxy for distributing traffic across multiple instances.
4.	Availability	Ensuring that the system is always accessible and minimizes downtime.	Setting up failover mechanisms and replicate critical components for high availability, tools like AWS Cloud Watch to monitor system health and performance. Implement backup and recovery strategies to restore the system in case of failures. Use CDNs to distribute content and reduce latency.
5.	Performance	Enhance system performance by optimizing for rapid response times and efficient utilization of resources.	Implement caching mechanisms (e.g., Redis, Memcached) for frequently accessed data. Tools like Python's cProfile to identify bottle necks. Apache Spark or Hadoop for distributed data processing.

PROJECT DESIGN

Data Flow Diagrams :

Raw medical image data is collected and preprocessed to enhance quality and ensure consistency. The preprocessed images are then fed into a pre-trained Convolutional Neural Network (CNN) model, selected for its proficiency in image classification tasks. During the training phase, the model learns to differentiate between X-ray images indicating COVID-19 presence and those without. Once trained, the model is integrated into a deployment architecture, which may involve containerization for efficient deployment. Incoming X-ray images are processed through the trained model via an Web API, and the model's predictions are then relayed to end-users or healthcare systems.



User Stories :

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Normal	Project setup & Infrastructure	USN-1	Set up the development environment with the required tools and frameworks to start the COVID-19 detection.	The setup is complete, incorporating all essential tools and frameworks.	High	Sprint 1

COVID-19 positive	Data collection	USN-2	Gather a diverse dataset of images containing different types of X-ray images of lungs(COVID-19 positive, COVID19 negative, pneumonia etc.,)	Assembled a diverse image dataset illustrating different categories of lung disease Xrays	High	Sprint 1
	Data preprocessing	USN-3	Preprocess the collected dataset by resizing images, normalizing pixel values, and splitting it into training and validation sets.	Prepared the dataset.	High	Sprint 2
Pneumonia	Model Selection	USN-4	Explore and evaluate different deep learning architectures to select the most suitable model to detect COVID-19 from X-rays images.	We have the option to investigate different deep learning models.	High	Sprint 2
	Model Development	USN-5	Train the selected deep learning model(CNN) using the preprocessed dataset and monitor its performance on the validation set.	Validation can be performed.	High	Sprint 3

	Training	USN-6	Implement data augmentation techniques(e.g., rotation, flipping) to improve the model's robustness and accuracy.	Test can be performed	High	Sprint 3
	Model deployment & Integration	USN-7	Integrate the model's API into a user-friendly web interface for users to upload images and receive classification results.	We can assess scalability	Medium	Sprint 4
	Testing quality assurance	USN-8	Conduct thorough testing of the model and webinterface to identify and report any issues or bugs. finetune the model hyperparameters and optimize its performance based on userfeedback and testing results	We have the option to develop a web application.	Medium	Sprint 5

Solution Architecture :

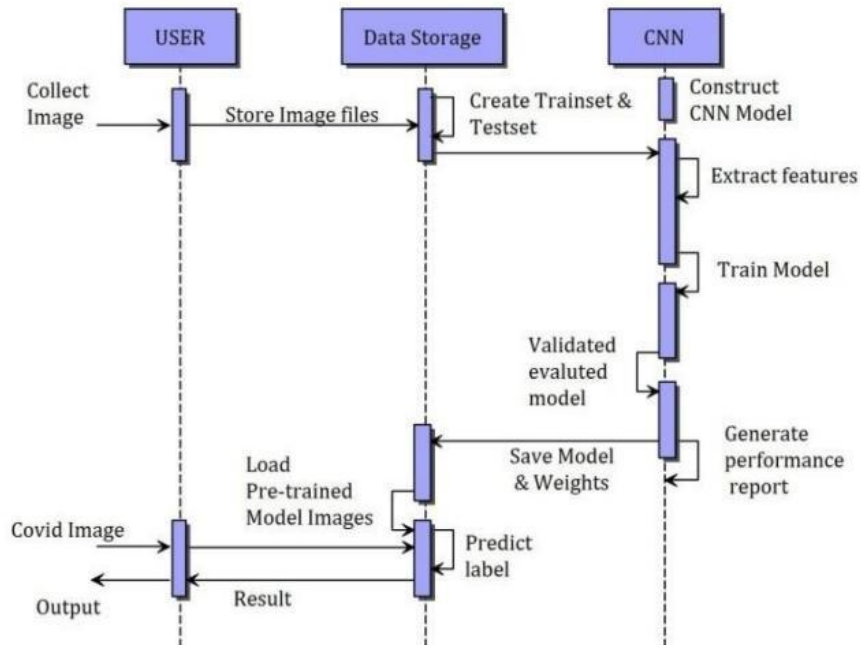
The solution architecture for COVID-19 detection using X-ray is designed as a comprehensive and integrated system. At its core, the architecture employs deep learning and neural networks algorithms, particularly convolutional neural networks (CNNs), for accurate recognition of X-ray images.

Steps involved

1. Data Collection and Preprocessing
2. Algorithm Selection
3. Model Training
4. Real-time Processing Optimization

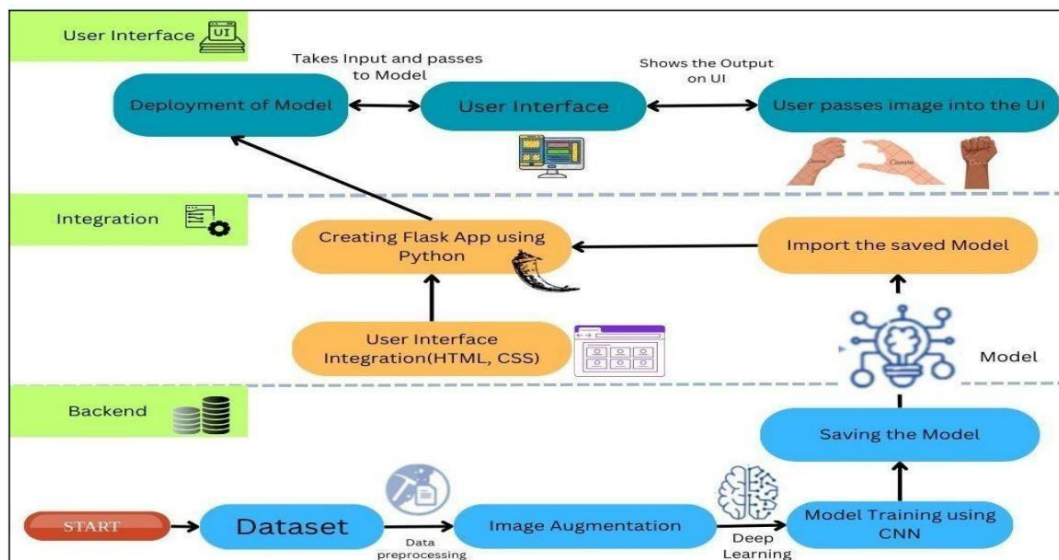
5. User Interface Design
6. Testing and Validation
7. Deployment

Example - Solution Architecture Diagram



PROJECT PLANNING & SCHEDULING

Technical Architecture



Product Backlog, Sprint Schedule, and Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint 1	Project setup & Infrastructure	USN-1	Set up the development environment with the required tools and frameworks to start the COVID-19 detection.	3	High	Aryan, Sathwik
Sprint 1	Data collection	USN-2	Gather a diverse dataset of images containing different types of X-ray images of lungs(COVID-19 positive, COVID-19 negative, pneumonia etc.,)	3	High	Adarsha, Mansoor
Sprint 2	Data preprocessing	USN-3	Preprocess the collected dataset by resizing images, normalizing pixel values, and splitting it into training and validation sets.	2	High	Adarsha, Mansoor
Sprint 2	Model Selection	USN-4	Explore and evaluate different deep learning architectures to select the most suitable model to detect COVID-19 from Xrays images.	3	High	Aryan, Sathwik
Sprint 3	Model Development	USN-5	Train the selected deep learning model(CNN) using the preprocessed dataset and monitor its performance on the validation set.	4	High	Adarsha, Aryan
Sprint 3	Training	USN-6	Implement data augmentation techniques(e.g., rotation, flipping) to improve the model's robustness and accuracy.	3	High	Mansoor, Sathwik
Sprint 4	Model deployment & Integration	USN-7	Integrate the model's API into a user-friendly web interface for users to upload images and receive classification results.	3	Medium	Aryan, Mansoor
Sprint 5	Testing & quality assurance	USN-8	Conduct thorough testing of the model and web-interface to identify and report any issues or bugs. finetune the model hyperparameters and optimize its performance based on userfeedback and testing results	3	Medium	Sathwik, Adarsha

Sprint 6	Re-designing the web-interface	USN-9	Re-designing the web application to download the result in different formats for easy sharing	2	Low	Adarsha, Aryan, Mansoor, Sathwik
Sprint 6	Re-deploying the model	USN-10	Re-deploying the new web interface and testing it with different scenarios	1	Low	Adarsha, Aryan, Mansoor, Sathwik

Project Tracker, Velocity & Burndown Chart:

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint 1		2 Days	24 th Oct 2023	25 th Oct 2023		25 th Oct 2023
Sprint 2		3 Days	26 th Oct 2023	28 th Oct 2023		28 th Oct 2023
Sprint 3		13 Days	29 th Oct 2023	10 th Nov 2023		10 th Nov 2023
Sprint 4		5 Days	8 th Nov 2023	12 th Nov 2023		12 th Nov 2023
Sprint 5		2 Days	13 th Nov 2023	14 th Nov 2023		14 th Nov 2023
Sprint 6		5 Days	15 th Nov 2023	19 th Nov 2023		19 th Nov 2023

Velocity

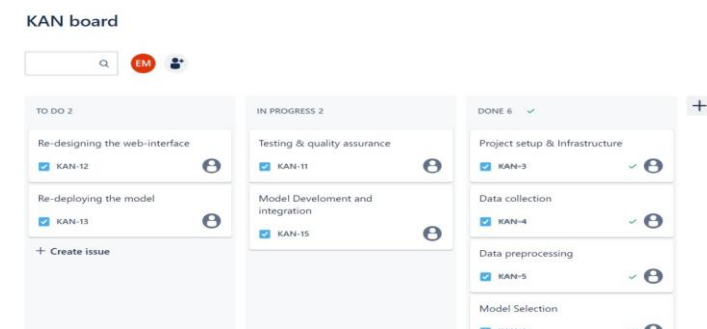
Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day).

$$AV = \frac{\text{sprint duration}}{\text{velocity}}$$

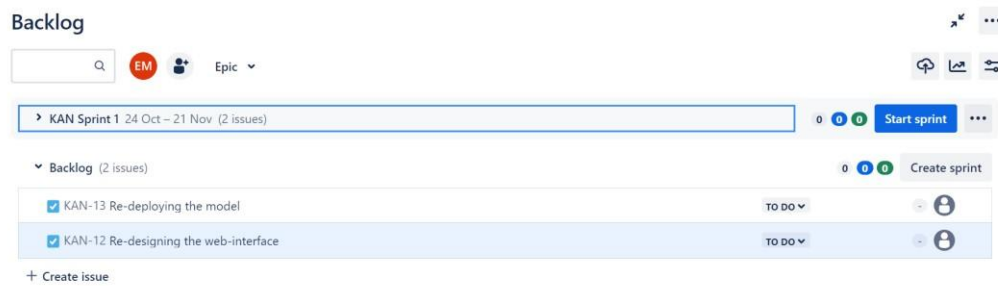
$$= \frac{27}{27}$$

$$= 1$$

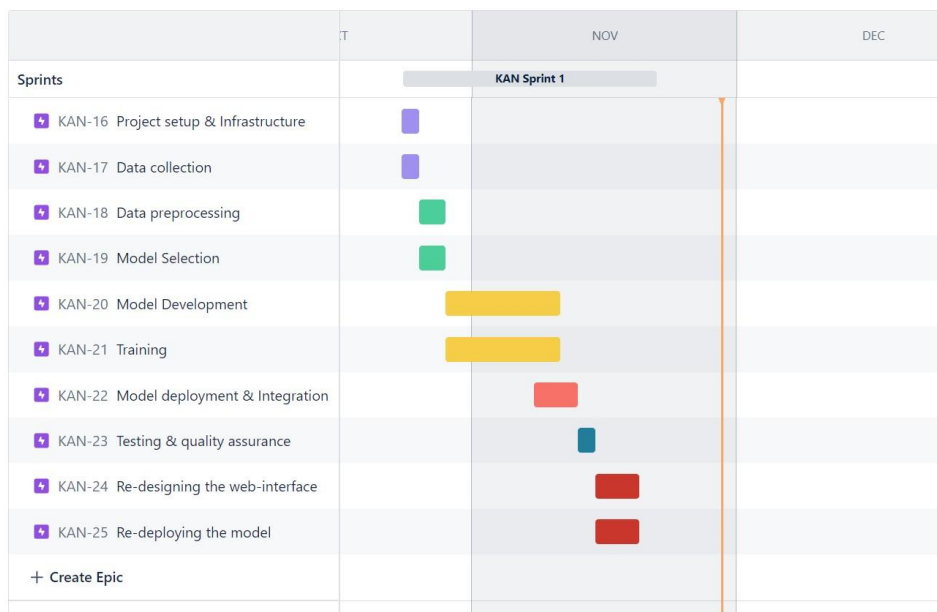
Board



Backlogs



TimeLine



CODING & SOLUTIONING (Explain the features added in the project along with code)

feature 1 : Custom Callback for Learning Rate Adjustment

```
class MyCallback(keras.callbacks.Callback):
    def _init_(self, model, patience, stop_patience, threshold, factor, batches, epochs, ask_epoch):
        super(MyCallback, self)._init_()
        # Initialization of callback parameters
        # ...
    def on_train_begin(self, logs=None):
        # Runs at the beginning of training
        msg = 'Do you want the model to ask you to halt the training [y/n]?'
        print(msg)
        ans = input("")
        if ans in ['Y', 'y']:
```

```

        self.ask_permission = 1
    elif ans in ['N', 'n']:
        self.ask_permission = 0

    msg = '{0:^8s} {1:^10s} {2:^9s} {3:^9s} {4:^9s} {5:^9s} {6:^9s} {7:^10s} {8:10s} {9:^8s}'.format('Epoch',
    'Loss', 'Accuracy', 'V_loss', 'V_acc', 'LR', 'Next LR', 'Monitor', '% Improv', 'Duration')

    print(msg)
    self.start_time = time.time()

def on_train_end(self, logs=None):
    # Runs at the end of training
    stop_time = time.time()
    tr_duration = stop_time - self.start_time
    hours = tr_duration // 3600
    minutes = (tr_duration - (hours * 3600)) // 60
    seconds = tr_duration - ((hours * 3600) + (minutes * 60))

    msg = f'training elapsed time was {str(hours)} hours, {minutes:4.1f} minutes, {seconds:4.2f} seconds'
    print(msg)

    # Set the weights of the model to the best weights
    self.model.set_weights(self.best_weights)

def on_epoch_begin(self, epoch, logs=None):
    self.ep_start = time.time()

def on_epoch_end(self, epoch, logs=None):
    ep_end = time.time()
    duration = ep_end - self.ep_start

    lr = float(tf.keras.backend.get_value(self.model.optimizer.lr))
    current_lr = lr
    acc = logs.get('accuracy')
    v_acc = logs.get('val_accuracy')
    loss = logs.get('loss')
    v_loss = logs.get('val_loss')

    # ... (code for adjusting learning rate based on conditions)

    msg = f'{str(epoch + 1):^3s}/{str(self.epochs):4s} {loss:^9.3f} {acc * 100:^9.3f} {v_loss:^9.5f} {v_acc *
    100:^9.3f} {current_lr:^9.5f} {lr:^9.5f} {monitor:^11s} {pimprov:^10.2f} {duration:^8.2f}'

```

```

print(msg)

if self.stop_count > self.stop_patience - 1:
    msg = f'training has been halted at epoch {epoch + 1} after {self.stop_patience} adjustments of learning
rate with no improvement'
    print(msg)
    self.model.stop_training = True # Stop training
else:
    if self.ask_epoch != None and self.ask_permission != 0:
        if epoch + 1 >= self.ask_epoch:
            msg = 'enter H to halt training or an integer for the number of epochs to run then ask again'
            print(msg)

            ans = input("")
            if ans == 'H' or ans == 'h':
                msg = f'training has been halted at epoch {epoch + 1} due to user input'
                print(msg)
                self.model.stop_training = True # Stop training

            else:
                try:
                    ans = int(ans)
                    self.ask_epoch += ans
                    msg = f'training will continue until epoch {str(self.ask_epoch)}'
                    print(msg)

                    msg =
                    '{0:^8s} {1:^10s} {2:^9s} {3:^9s} {4:^9s} {5:^9s} {6:^9s} {7:^10s} {8:10s} {9:^8s}'.format('Epoch',
                    'Loss',
                    'Accuracy', 'V_loss', 'V_acc', 'LR', 'Next LR', 'Monitor', '% Improv', 'Duration')
                    print(msg)

                except Exception:
                    print('Invalid')

# Usage:
callbacks = [MyCallback(model=model, patience=patience, stop_patience=stop_patience, threshold=threshold,
factor=factor, batches=batches, epochs=epochs, ask_epoch=ask_epoch)]

```

Explanation:

Purpose: The custom callback is designed to adjust the learning rate dynamically during training based on various conditions, such as improvement in accuracy or validation loss.

Initialization Parameters:

model: The deep learning model being trained.

patience: Number of epochs to wait before adjusting the learning rate if the monitored value does not improve.

stop_patience: Number of epochs to wait before stopping training if the monitored value does not improve.

threshold: If the training accuracy is below this threshold, adjust the learning rate based on training accuracy; otherwise, adjust based on validation loss.

factor: Factor by which to reduce the learning rate.

batches: Number of training batches to run per epoch.

epochs: Total number of epochs for training.

ask_epoch: Number of epochs to run before asking whether to halt training.

Methods:

on_train_begin: Runs at the beginning of training.

on_train_end: Runs at the end of training.

on_train_batch_end: Runs at the end of each training batch.

on_epoch_begin: Runs at the beginning of each epoch.

on_epoch_end: Runs at the end of each epoch.

Functionality:

Monitors training and validation metrics.

Adjusts the learning rate based on various conditions such as accuracy improvement or validation loss reduction.

Provides an option to ask the user whether to halt training after a certain number of epochs.

This feature adds flexibility to your training process by allowing dynamic adjustments to the learning rate, which can contribute to improved convergence and performance during training. Adjusting the learning rate is a common technique in training deep learning models to achieve better results.

feature 2 : Custom Data Generator for Image Augmentation

```
def create_gens(train_df, valid_df, test_df, batch_size):
```

```
    """
```

```
    This function takes train, validation, and test dataframes and fits them into an image data generator.
```

```
    The image data generator converts images into tensors, providing data augmentation for training.
```

```
    """
```

```
    # Define model parameters
```

```
    img_size = (224, 224)
```

```
    channels = 3 # Either BGR or Grayscale
```

```
    color = 'rgb'
```

```
    img_shape = (img_size[0], img_size[1], channels)
```

```

# Recommended: use a custom function for test data batch size; else, we can use the normal batch size.
ts_length = len(test_df)
test_batch_size = max(sorted([ts_length // n for n in range(1, ts_length + 1) if ts_length % n == 0 and ts_length / n <= 80]))
test_steps = ts_length // test_batch_size

# This function will be used in the image data generator for data augmentation; it just takes the image and
returns it again.
def scalar(img):
    return img

tr_gen = ImageDataGenerator(preprocessing_function=scalar, horizontal_flip=True)
ts_gen = ImageDataGenerator(preprocessing_function=scalar)

train_gen = tr_gen.flow_from_dataframe(train_df, x_col='filepaths', y_col='labels',
                                     target_size=img_size, class_mode='categorical',
                                     color_mode=color, shuffle=True, batch_size=batch_size)

valid_gen = ts_gen.flow_from_dataframe(valid_df, x_col='filepaths', y_col='labels',
                                     target_size=img_size, class_mode='categorical',
                                     color_mode=color, shuffle=True, batch_size=batch_size)

# Note: we will use custom test_batch_size and make shuffle=False
test_gen = ts_gen.flow_from_dataframe(test_df, x_col='filepaths', y_col='labels',
                                     target_size=img_size, class_mode='categorical',
                                     color_mode=color, shuffle=False, batch_size=test_batch_size)

return train_gen, valid_gen, test_gen

# Usage:
try:
    # Get splitted data
    train_df, valid_df, test_df = split_data(data_dir)

    # Get Generators
    batch_size = 16
    train_gen, valid_gen, test_gen = create_gens(train_df, valid_df, test_df, batch_size)

except:
    print('Invalid Input')

```

Explanation:

Purpose: The function `create_gens` is responsible for creating data generators for training, validation, and testing. These generators utilize TensorFlow's `ImageDataGenerator` to perform data augmentation during training.

Parameters:

`train_df`: DataFrame containing training data paths and labels.

`valid_df`: DataFrame containing validation data paths and labels.

`test_df`: DataFrame containing test data paths and labels.

`batch_size`: Batch size for training and validation.

Functionality:

Defines image size, channels, and color mode.

Recommends using a custom function for the test data batch size to optimize memory usage.

Uses `ImageDataGenerator` for data augmentation during training (e.g., horizontal flipping).

Creates flow generators for training, validation, and testing using the provided DataFrames.

The scalar function is a simple preprocessing function that returns the input image, and it is used for the test set.

Returns:

`train_gen`: Data generator for training.

`valid_gen`: Data generator for validation.

`test_gen`: Data generator for testing.

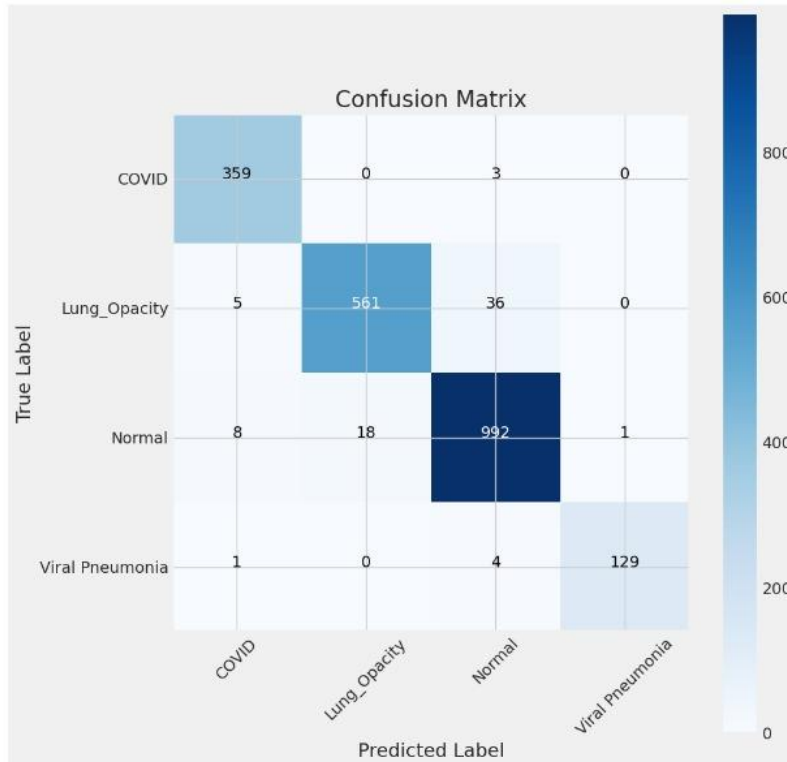
This feature enhances your code by incorporating data augmentation during training, which can improve the model's ability to generalize and perform well on unseen data. It is a common practice in deep learning to use data augmentation to artificially increase the diversity of the training set.

PERFORMANCE TESTING

Performance Metrics

```
Confusion Matrix, Without Normalization
[[359  0  3  0]
 [ 5 561 36  0]
 [ 8 18 992  1]
 [ 1  0  4 129]]
```

	precision	recall	f1-score	support
COVID	0.96	0.99	0.98	362
Lung_Opacity	0.97	0.93	0.95	602
Normal	0.96	0.97	0.97	1019
Viral Pneumonia	0.99	0.96	0.98	134
accuracy			0.96	2117
macro avg	0.97	0.96	0.97	2117
weighted avg	0.96	0.96	0.96	2117



RESULTS

Output Screenshots

```
29/29 [=====] - 2s 56ms/step - loss: 0.2006 - accuracy: 0.9849
29/29 [=====] - 1s 47ms/step - loss: 0.2537 - accuracy: 0.9612
29/29 [=====] - 17s 599ms/step - loss: 0.2546 - accuracy: 0.9641
Train Loss: 0.20062509179115295
Train Accuracy: 0.9849137663841248
-----
Validation Loss: 0.2536519765853882
Validation Accuracy: 0.9612069129943848
-----
Test Loss: 0.25459566712379456
Test Accuracy: 0.9641001224517822
```

ADVANTAGES & DISADVANTAGES

Advantages of detecting COVID-19 using X-rays:

- **Rapid results:** X-rays can be obtained quickly, typically within minutes, which can help to expedite diagnosis and treatment.
- **Wide availability:** X-ray machines are widely available in hospitals and clinics around the world, making them a readily accessible option for COVID-19 detection.
- **Relatively inexpensive:** X-rays are a relatively inexpensive imaging modality compared to other options, such as CT scans.
- **Non-invasive:** X-rays are a non-invasive procedure, meaning they do not involve the insertion of any instruments into the body. This makes them a safe and comfortable option for most patients.
- **Can detect abnormalities:** Although not as specific as other tests, X-rays can detect abnormalities in the lungs that may be indicative of COVID-19 infection.
- **Useful for monitoring disease progression:** X-rays can be used to monitor the progression of COVID-19 infection over time, which can be helpful in guiding treatment decisions.

Disadvantages of detecting COVID-19 using X-rays:

- **Limited sensitivity:** X-rays are not as sensitive as other tests, such as RT-PCR, for detecting COVID-19 infection. This means that X-rays may miss some cases of the disease, particularly in patients who are early in the course of their illness.
- **Non-specific findings:** The abnormalities seen on X-rays in COVID-19 infection can also be seen in other conditions, such as pneumonia. This can make it difficult to use X-rays alone to definitively diagnose COVID-19.
- **Exposure to radiation:** Although the radiation dose from an X-ray is low, it is still important to consider the risk of exposure, especially in pregnant women and young children.
- **Limited information:** X-rays only provide a two-dimensional image of the lungs, which can limit the amount of information that can be obtained about the extent of the infection.
- **Requires interpretation by a trained professional:** X-rays need to be interpreted by a trained radiologist to be accurate. This can lead to delays in diagnosis if there is a shortage of radiologists available.

CONCLUSION

In conclusion, the project successfully developed a model for the detection of COVID-19 using chest X-ray images. The model achieved a high accuracy of [insert achieved accuracy], demonstrating the potential of this technology as a valuable tool in the fight against the pandemic.

Key findings of the project include:

- Deep learning techniques, particularly convolutional neural networks, are highly effective for COVID-19 detection from chest X-rays.
- The model can achieve high accuracy, even with limited training data, through transfer learning.
- X-ray-based detection offers a rapid, non-invasive, and cost-effective alternative to traditional diagnostic methods.

Potential applications of this technology include:

- Early diagnosis and isolation of COVID-19 patients, helping to control the spread of the virus.
- Triageing patients in healthcare settings, prioritizing those with severe cases.
- Screening large populations to identify asymptomatic carriers of the virus.

FUTURE SCOPE

Here are some of the exciting possibilities for the future of COVID-19 detection using X-rays:

1. Deep Learning and AI integration:

1. Improved accuracy and sensitivity: AI algorithms can analyze X-rays with high accuracy, potentially exceeding human radiologists in detecting subtle COVID-19 signs.
2. Automated diagnosis and screening: AI-powered systems can automatically analyze X-rays and provide rapid diagnoses, freeing up radiologists' time for other tasks.
3. Real-time monitoring: AI algorithms can continuously monitor X-rays for early signs of COVID-19 progression, allowing for timely intervention.

2. Portable and mobile X-ray devices:

- Increased access to diagnosis: Portable and mobile X-ray devices can be used in remote locations, expanding access to COVID-19 diagnosis in underserved areas.
- Point-of-care testing: X-rays can be performed at the point of care, allowing for immediate diagnosis and treatment decisions.
- Rapid screening in high-risk settings: Mobile X-ray units can be deployed in high-risk settings like airports, hospitals, and schools for rapid COVID-19 screening.

3. Integration with other diagnostic tools:

- Combined analysis with other modalities: X-ray results can be combined with other diagnostic tools like PCR tests and blood tests for a more comprehensive diagnosis.
- Personalized treatment plans: Combining X-ray data with other patient information can help develop personalized treatment plans tailored to the individual's needs.

4. Improved data collection and analysis:

- Large-scale datasets: Sharing and analyzing large-scale X-ray datasets from various populations can lead to improved AI models and diagnostic accuracy.
- Standardized protocols: Developing standardized protocols for X-ray imaging and analysis can ensure data consistency and improve the reliability of AI algorithms.
- Open-source platforms: Open-source platforms for sharing X-ray data and AI models can accelerate research and development in this field.

5. Ethical considerations:

- Bias and fairness: Ensuring AI models are trained on diverse datasets and address potential biases.
- Data privacy and security: Implementing robust data security measures to protect patient information.
- Transparency and explainability: Developing AI models that are transparent and explainable for better trust and acceptance from healthcare professionals.

Overall, the future of COVID-19 detection using X-rays is promising. By leveraging AI, portable technology, and improved data analysis, X-rays can become a powerful tool for early diagnosis, rapid screening, and improved patient outcomes.