

Final Project Report

Team ID	SWTID1720425899
Project Name	CovidVision: Advanced COVID-19 Detection from Lung X-rays with Deep Learning

1. Introduction

- 1.1. Project overviews
- 1.2. Objectives

2. Project Initialization and Planning Phase

- 2.1. Define Problem Statement
- 2.2. Project Proposal (Proposed Solution)
- 2.3. Initial Project Planning

3. Data Collection and Preprocessing Phase

- 3.1. Data Collection Plan and Raw Data Sources Identified
- 3.2. Data Quality Report
- 3.3. Data Preprocessing

4. Model Development Phase

- 4.1. Model Selection Report
- 4.2. Initial Model Training Code, Model Validation and Evaluation Report
- 4.3. Final Model Selection Justification

5. Results

- 6.1. Output Screenshots

6. Advantages & Disadvantages

7. Conclusion

8. Future Scope

9. Appendix

- 10.1. Source Code
- 10.2. GitHub & Project Demo Link

Introduction

Project overviews

This project, "CovidVision: Advanced COVID-19 Detection from Lung X-rays with Deep Learning" utilizes deep learning algorithms to analyze lung X-ray images for signs of Covid-19 infection. By leveraging vast datasets and image recognition technology, this project aims to provide accurate and rapid diagnosis, aiding in early detection and containment of the virus.

The project also aims to address scenarios like integrating the AI system to expedite diagnosis, such that now the system swiftly analyses X-rays, aiding medical staff in triage and treatment decisions, effectively managing patient influx during peak periods; automating analysis of lung X-rays enabling identification of cases, facilitating prompt isolation and treatment, thus assisting in Covid-19 screening and containing the spread; monitoring Covid-19 trends across regions by analysing X-ray data from various healthcare facilities, identifying hotspots and allocating resources strategically, aiding in targeted interventions and control measures to curb transmission.

Objectives

- To obtain a dataset of lung X-ray images.
- Preprocessing the images.
- Create training and testing data to training and evaluation.
- Apply Transfer learning algorithms on the dataset.
- Understand how deep neural networks detect the disease.
- Know how to find the accuracy of the model and find the accuracy of the model.
- Build a web applications using the Flask framework.

Project Initialization and Planning Phase

Define Problem Statement

Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	A radiologist at a busy hospital.	Accurately diagnose COVID-19 from lung X-rays quickly and efficiently.	The current diagnostic methods are not always reliable and often result in false positives or negatives.	These methods lack advanced analytical capabilities and cannot always differentiate between COVID-19 and other respiratory conditions.	Frustrated and concerned about the accuracy of diagnoses, leading to potential delays in patient treatment and increased workload due to re-evaluations.
PS-2	A hospital administrator responsible for integrating new technologies into clinical workflows.	Implement an effective and seamless diagnostic tool for COVID-19 detection that can be used by our medical staff without extensive training.	Many advanced technologies are difficult to integrate into existing systems and require significant resources and training.	They are often complex, lack user-friendly interfaces, and demand high computational power.	Overwhelmed by the logistical challenges and worried about the cost-effectiveness and practicality of adopting new diagnostic tools.
PS - 3	A health care policy maker focused on ensuring equitable health	Ensure that COVID-19 diagnostic tools are effective	Current tools often fail to	They are trained on non-diverse datasets and	Concerned about the fairness and inclusivity of healthcare solutions, and determined to

	care and treatment.	across diverse populations and healthcare settings	generalize well, leading to biased results and unequal healthcare outcomes	are not adapted to different demographic and clinical contexts.	find technologies that provide accurate results for all population groups
--	---------------------	--	--	---	---

Project Proposal (Proposed Solution)

Early Detection of COVID-19: The primary challenge is to detect COVID-19 cases as early as possible. Early diagnosis allows for timely intervention, isolation, and appropriate medical care.

Differentiating COVID-19 from Other Lung Diseases: Distinguishing COVID-19 from other lung diseases (such as pneumonia or tuberculosis) based on chest X-rays is crucial. Accurate differentiation ensures proper treatment pathways.

Reducing False Negatives and False Positives: Avoiding false negatives (missing actual COVID-19 cases) and false positives (misclassifying non-COVID-19 cases) is essential for effective patient management.

Project Overview	
Objective	To utilize deep learning algorithms to analyze lung X-ray images for signs of Covid-19 infection. To obtain a dataset of lung X-ray images, preprocess the images, create training and testing data for training and evaluation, apply Transfer learning algorithms on the dataset, understand how deep neural networks detect the disease, find the accuracy of the model and find the accuracy of the model and build a web application using the Flask framework.
Scope	To build accurate models that can distinguish between normal lung X-rays and those showing signs of COVID-19, aiding in early diagnosis and effective patient management
Problem Statement	
Description	The primary challenge is to detect COVID-19 cases as early as possible. Early diagnosis allows for timely intervention, isolation, and appropriate medical care.

	Differentiating COVID-19 from Other Lung Diseases: Distinguishing COVID-19 from other lung diseases (such as pneumonia or tuberculosis) based on chest X-rays is crucial. Accurate differentiation ensures proper treatment pathways.
Impact	<p>Improved Patient Outcomes: Early detection enables timely treatment, reducing disease severity and mortality rates.</p> <p>Healthcare Resource Optimization: Accurate diagnosis helps allocate healthcare resources efficiently (e.g., hospital beds, ventilators, and medical staff).</p> <p>Epidemiological Insights: Identifying COVID-19 cases accurately contributes to better understanding disease spread patterns and informs public health measures.</p> <p>Reduced Transmission: Early detection and isolation prevent further transmission within communities.</p> <p>Global Health Crisis Mitigation: Solving these problems directly addresses the ongoing global health crisis caused by COVID-19.</p>
Proposed Solution	
Approach	<p>Choose a pre-trained convolutional neural network (CNN) model known for its performance in image classification tasks (e.g., ResNet, VGG, InceptionV3, Exception).</p> <p>Adapt the selected model for the COVID-19 detection task by replacing the final layers with new layers tailored to the specific problem.</p> <p>Example: Replace the last fully connected layers with a new fully connected layer with neurons matching the number of classes (e.g., COVID-19 positive, COVID-19 negative).</p>
Key Features	Utilizing state-of-the-art pre-trained convolutional neural networks (CNNs) such as ResNet, VGG, Inception, or Exception, which have demonstrated high performance in image classification tasks.

Resource Requirements

Resource Type	Description	Specification/Allocation
Hardware		
Computing Resources	CPU/GPU specifications, number of cores	Python 3 Google Compute Engine backend (Google Colab)
Memory	RAM specifications	12.67GB
Storage	Disk space for data, models, and logs	107.72GB
Software		
Frameworks	Python frameworks	Flask
Libraries	Additional libraries	tensorflow
Development Environment	IDE, version control	Jupyter Notebook, Git
Data		
Data	Source, size, format	Kaggle dataset, 10,000 images

Initial Project Planning

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members	Sprint Start Date	Sprint End Date (Planned)
Sprint-1	Obtain a dataset of lung X-ray images and create train and test dataset	USN-1	As a data scientist, I want to obtain a dataset of lung X-ray images for analysis..	2	Medium	Amritha Varshini R K Gokul S	08/07/24	10/07/24
Sprint-2	Preprocess the images (resize, normalize pixel values, etc.).	USN-2	As a machine learning engineer, I want to apply transfer learning to train a model. .	1	High	Gokul Rajan Savitha Sundaresan	11/07/24	14/07/24
Sprint-3	Select a pre-trained deep learning model (e.g., VGG, ResNet) and modify the architecture for	USN-3	As a machine learning engineer, I want to apply transfer learning algorithms on the dataset	2	Low	Gokul Rajan Savitha Sundaresan	11/07/24	14/07/24

	classification							
Sprint-4	Train the model on preprocessed dataset	USN-4	As a machine learning engineer, I want to train and evaluate the model.	2	Medium	Savitha Sundaresan Amritha Varshini R K	11/07/24	14/07/24
Sprint-5	Generate confusion matrices and classification reports, visualize model attention maps to understand key features and identify areas for further model improvement	USN-5	As a data analyst, I want to analyze the model's performance to improve accuracy.		High	Amritha Varshini R K Gokul S	11/07/24	14/07/24

Sprint-6	Design the web app user interface , implement the backend using Flask to handle image uploads		As a web developer, I want to build a web app to deploy the model		High	Amritha Varshini R K Gokul S	15/07/24	17/07/24
Sprint-7	Integrate the trained deep learning model into the web app and test the web app end to end and fix any issues	USN-5	As a developer, I want to integrated and deploy the model, test it and fix any issues that cocmes up.	1	High	Gokul S Gokul Rajan	18/07/24	19/07/24

Data Collection and Preprocessing Phase

Data Collection Plan and Raw Data Sources Identified

Section	Description
Project Overview	Develop an advanced, accurate, and real-time COVID-19 detection tool from lung X-rays using transfer learning. The tool aims to assist radiologists and healthcare professionals in diagnosing COVID-19 quickly and effectively, improving patient outcomes and supporting public health efforts.
Data Collection Plan	Data collected from Kaggle is downloaded into drive and accessed via collab using opendatasets python library. Two folders are created such that one contains the data 'Covid' images and the other the data contains 'Normal' images.
Raw Data Sources Identified	The Lung X-ray dataset is collected from Kaggle.

Raw Data Sources Template

Source Name	Description	Location/URL	Format	Size	Access Permissions
Covid -19 Detection	The dataset contains data of chest X-ray	https://www.kaggle.com/datasets/tawsifurrahman/covid19-xray-images	Image	816 MB	Public

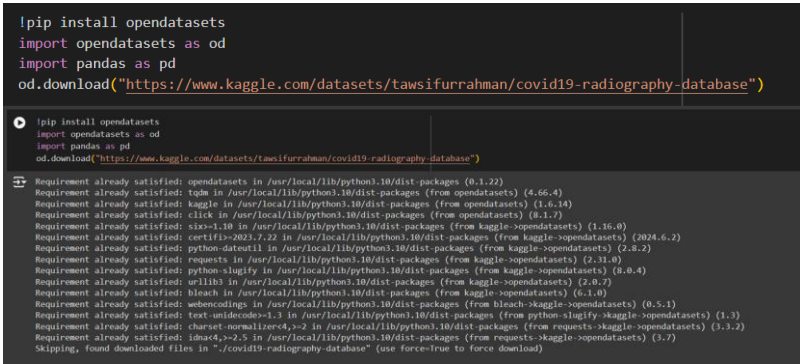


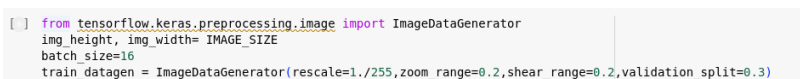
from Lung X-rays	images for Covid-19 positive cases along with Normal and Viral Pneumonia images.	id19-radiography-database			
------------------	--	---------------------------	--	--	--

Data Quality Report

Data Source	Data Quality Issue	Severity	Resolution Plan
Dataset	The instances of Normal and Covid datasets were not the same	Moderate	Taking the instance that is present in the Covid dataset in the Normal Dataset also

Data Preprocessing

Section	Description
Data Overview	The dataset contains data of chest X-ray images for Covid-19 positive cases along with Normal and Viral Pneumonia images.
Resizing	Resize images to a target size of (256, 256)

Section	Description
Normalization	rescale=1./255: This parameter is used to normalize pixel values of images. In many image datasets, pixel values range from 0 to 255.
Data Augmentation	Applied augmentation techniques such as zooming range = 0.2 , shearing range = 0.2.
Data Preprocessing Code Screenshots	
Loading Data	 <pre> !pip install opendatasets import opendatasets as od import pandas as pd od.download("https://www.kaggle.com/datasets/tawsifurrahman/covid19-radiography-database") !pip install opendatasets import opendatasets as od import pandas as pd od.download("https://www.kaggle.com/datasets/tawsifurrahman/covid19-radiography-database") Requirement already satisfied: opendatasets in /usr/local/lib/python3.10/dist-packages (0.1.22) Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from opendatasets) (4.66.4) Requirement already satisfied: kaggle in /usr/local/lib/python3.10/dist-packages (from opendatasets) (1.6.4) Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from opendatasets) (8.1.7) Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.10/dist-packages (from kaggle-sopendatasets) (1.16.0) Requirement already satisfied: certifi>=2021.7.22 in /usr/local/lib/python3.10/dist-packages (from kaggle-sopendatasets) (2024.6.2) Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-packages (from kaggle-sopendatasets) (2.8.2) Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from kaggle-sopendatasets) (2.31.0) Requirement already satisfied: python-slugify in /usr/local/lib/python3.10/dist-packages (from kaggle-sopendatasets) (8.0.4) Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from kaggle-sopendatasets) (2.0.7) Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages (from kaggle-sopendatasets) (6.1.0) Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-packages (from bleach->kaggle-sopendatasets) (0.5.1) Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.10/dist-packages (from python-slugify->kaggle-sopendatasets) (1.3) Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle-sopendatasets) (3.3.2) Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle-sopendatasets) (3.7) Skipping, found downloaded files in "/.covid19-radiography-database" (use force=True to force download) </pre>
Resizing	 <pre> IMAGE_SIZE = (256, 256) IMAGE_SHAPE = IMAGE_SIZE + (3,) </pre>
Data Augmentation	 <pre> from tensorflow.keras.preprocessing.image import ImageDataGenerator img_height, img_width= IMAGE_SIZE batch_size=16 train_datagen = ImageDataGenerator(rescale=1./255, zoom_range=0.2, shear_range=0.2, validation_split=0.3) </pre>
Normalization	 <pre> from tensorflow.keras.preprocessing.image import ImageDataGenerator img_height, img_width= IMAGE_SIZE batch_size=16 train_datagen = ImageDataGenerator(rescale=1./255, zoom_range=0.2, shear_range=0.2, validation_split=0.3) </pre>

Model Development Phase

Model Selection Report

Model	Description
Xception	<p>The Xception model, short for Extreme Inception, is a deep convolutional neural network architecture that leverages depthwise separable convolutions. Pretrained on the ImageNet dataset, Xception is known for its efficiency and performance in image classification tasks. For this project, the Xception model is fine-tuned for the task of COVID-19 detection from lung X-rays. This model's architecture allows it to capture intricate patterns and features within the images, potentially leading to high accuracy in detecting COVID-19 layers leading to a softmax activation for classification. This model leverages the spatial hierarchy of features, making it suitable for detecting patterns associated with COVID-19 in X-ray images.</p>
VGG16	<p>Utilizes the VGG16 architecture pretrained on the ImageNet dataset. The model's final layers are fine-tuned for the specific task of COVID-19 detection in lung X-rays. Transfer learning leverages the feature extraction capabilities of VGG16, reducing the need for extensive training data and computational resources while improving model performance and generalization.</p>
Inception V3	<p>The Inception V3 model, part of the Inception family of networks, is known for its efficiency and deep architecture. Pretrained on ImageNet, Inception V3 employs inception modules that allow the model to capture multi-scale features effectively. For this project, the Inception V3 model is fine-tuned for the task of COVID-19 detection from lung X-rays. The model's architecture, with its ability to handle varying spatial hierarchies and features, aims to achieve high accuracy in detecting COVID-19.</p>
ResNet50	<p>Employs the ResNet50 architecture pretrained on ImageNet, with the addition of custom dense layers tailored for the COVID-19 detection task. ResNet50's residual blocks help mitigate the vanishing gradient problem, allowing for deeper network training and improved accuracy. Fine-tuning is applied to adapt the model to the lung X-ray dataset.</p>

Initial Model Training Code, Model Validation and Evaluation Report

Model	Summary	Training and Validation Performance Metrics																																																												
VGG16	<div><div>vgg.summary()</div><div>Model: "vgg16"</div><table><thead><tr><th>Layer (type)</th><th>Output Shape</th><th>Param #</th></tr></thead><tbody><tr><td>input_1 (InputLayer)</td><td>[(None, 256, 256, 3)]</td><td>0</td></tr><tr><td>block1_conv1 (Conv2D)</td><td>(None, 256, 256, 64)</td><td>1792</td></tr><tr><td>block1_conv2 (Conv2D)</td><td>(None, 256, 256, 64)</td><td>36928</td></tr><tr><td>block1_pool (MaxPooling2D)</td><td>(None, 128, 128, 64)</td><td>0</td></tr><tr><td>block2_conv1 (Conv2D)</td><td>(None, 128, 128, 128)</td><td>73856</td></tr><tr><td>block2_conv2 (Conv2D)</td><td>(None, 128, 128, 128)</td><td>147584</td></tr><tr><td>block2_pool (MaxPooling2D)</td><td>(None, 64, 64, 128)</td><td>0</td></tr><tr><td>block3_conv1 (Conv2D)</td><td>(None, 64, 64, 256)</td><td>295168</td></tr><tr><td>block3_conv2 (Conv2D)</td><td>(None, 64, 64, 256)</td><td>590080</td></tr><tr><td>block3_conv3 (Conv2D)</td><td>(None, 64, 64, 256)</td><td>590080</td></tr><tr><td>block3_pool (MaxPooling2D)</td><td>(None, 32, 32, 256)</td><td>0</td></tr><tr><td>block4_conv1 (Conv2D)</td><td>(None, 32, 32, 512)</td><td>1180160</td></tr><tr><td>block4_conv2 (Conv2D)</td><td>(None, 32, 32, 512)</td><td>2359808</td></tr><tr><td>block4_conv3 (Conv2D)</td><td>(None, 32, 32, 512)</td><td>2359808</td></tr><tr><td>block4_pool (MaxPooling2D)</td><td>(None, 16, 16, 512)</td><td>0</td></tr><tr><td>block5_conv1 (Conv2D)</td><td>(None, 16, 16, 512)</td><td>2359808</td></tr><tr><td>block5_conv2 (Conv2D)</td><td>(None, 16, 16, 512)</td><td>2359808</td></tr><tr><td>block5_conv3 (Conv2D)</td><td>(None, 16, 16, 512)</td><td>2359808</td></tr><tr><td>block5_pool (MaxPooling2D)</td><td>(None, 8, 8, 512)</td><td>0</td></tr></tbody></table><div>Total params: 14714688 (56.13 MB) Trainable params: 14714688 (56.13 MB) Non-trainable params: 0 (0.00 Byte)</div></div>	Layer (type)	Output Shape	Param #	input_1 (InputLayer)	[(None, 256, 256, 3)]	0	block1_conv1 (Conv2D)	(None, 256, 256, 64)	1792	block1_conv2 (Conv2D)	(None, 256, 256, 64)	36928	block1_pool (MaxPooling2D)	(None, 128, 128, 64)	0	block2_conv1 (Conv2D)	(None, 128, 128, 128)	73856	block2_conv2 (Conv2D)	(None, 128, 128, 128)	147584	block2_pool (MaxPooling2D)	(None, 64, 64, 128)	0	block3_conv1 (Conv2D)	(None, 64, 64, 256)	295168	block3_conv2 (Conv2D)	(None, 64, 64, 256)	590080	block3_conv3 (Conv2D)	(None, 64, 64, 256)	590080	block3_pool (MaxPooling2D)	(None, 32, 32, 256)	0	block4_conv1 (Conv2D)	(None, 32, 32, 512)	1180160	block4_conv2 (Conv2D)	(None, 32, 32, 512)	2359808	block4_conv3 (Conv2D)	(None, 32, 32, 512)	2359808	block4_pool (MaxPooling2D)	(None, 16, 16, 512)	0	block5_conv1 (Conv2D)	(None, 16, 16, 512)	2359808	block5_conv2 (Conv2D)	(None, 16, 16, 512)	2359808	block5_conv3 (Conv2D)	(None, 16, 16, 512)	2359808	block5_pool (MaxPooling2D)	(None, 8, 8, 512)	0	<div><div>vgg16.fit(train_generator, validation_data=validation_generator, epochs=5)</div><div>Epoch 1/5 217/217 [=====] - loss: 0.8234 - accuracy: 0.7345 - val_loss: 0.3295 - val_accuracy: 0.9546 Epoch 2/5 217/217 [=====] - loss: 0.4876 - accuracy: 0.7921 - val_loss: 0.2908 - val_accuracy: 0.9442 Epoch 3/5 217/217 [=====] - loss: 0.4338 - accuracy: 0.8182 - val_loss: 0.2754 - val_accuracy: 0.9347 Epoch 4/5 217/217 [=====] - loss: 0.3895 - accuracy: 0.8376 - val_loss: 0.2657 - val_accuracy: 0.9546 Epoch 5/5 217/217 [=====] - loss: 0.3529 - accuracy: 0.8515 - val_loss: 0.2543 - val_accuracy: 0.9549 keras.src.callbacks.history at 0x7f7f012ef8db</div></div>
Layer (type)	Output Shape	Param #																																																												
input_1 (InputLayer)	[(None, 256, 256, 3)]	0																																																												
block1_conv1 (Conv2D)	(None, 256, 256, 64)	1792																																																												
block1_conv2 (Conv2D)	(None, 256, 256, 64)	36928																																																												
block1_pool (MaxPooling2D)	(None, 128, 128, 64)	0																																																												
block2_conv1 (Conv2D)	(None, 128, 128, 128)	73856																																																												
block2_conv2 (Conv2D)	(None, 128, 128, 128)	147584																																																												
block2_pool (MaxPooling2D)	(None, 64, 64, 128)	0																																																												
block3_conv1 (Conv2D)	(None, 64, 64, 256)	295168																																																												
block3_conv2 (Conv2D)	(None, 64, 64, 256)	590080																																																												
block3_conv3 (Conv2D)	(None, 64, 64, 256)	590080																																																												
block3_pool (MaxPooling2D)	(None, 32, 32, 256)	0																																																												
block4_conv1 (Conv2D)	(None, 32, 32, 512)	1180160																																																												
block4_conv2 (Conv2D)	(None, 32, 32, 512)	2359808																																																												
block4_conv3 (Conv2D)	(None, 32, 32, 512)	2359808																																																												
block4_pool (MaxPooling2D)	(None, 16, 16, 512)	0																																																												
block5_conv1 (Conv2D)	(None, 16, 16, 512)	2359808																																																												
block5_conv2 (Conv2D)	(None, 16, 16, 512)	2359808																																																												
block5_conv3 (Conv2D)	(None, 16, 16, 512)	2359808																																																												
block5_pool (MaxPooling2D)	(None, 8, 8, 512)	0																																																												
ResNet50	<div><div>resnet50.summary()</div><table><tbody><tr><td>conv5_block2_2_bn (BatchNormalizatio</td><td>2048</td><td>'conv5_block2_2_conv[0][0]'</td></tr><tr><td>conv5_block2_2_relu (Activatio</td><td>0</td><td>'conv5_block2_2_bn[0][0]'</td></tr><tr><td>conv5_block2_3_conv (Conv2</td><td>1050624</td><td>'conv5_block2_2_relu[0][0]'</td></tr><tr><td>conv5_block2_3_bn (BatchNormalizatio</td><td>8192</td><td>'conv5_block2_3_conv[0][0]'</td></tr><tr><td>conv5_block2_add (Add)</td><td>0</td><td>'conv5_block2_out[0][0]'; 'conv5_block2_3_bn[0][0]'</td></tr><tr><td>conv5_block2_out (Activatio</td><td>0</td><td>'conv5_block2_add[0][0]'</td></tr><tr><td>conv5_block3_1_conv (Conv2</td><td>1049808</td><td>'conv5_block2_out[0][0]'</td></tr><tr><td>conv5_block3_1_bn (BatchNormalizatio</td><td>2048</td><td>'conv5_block3_1_conv[0][0]'</td></tr><tr><td>conv5_block3_1_relu (Activatio</td><td>0</td><td>'conv5_block3_1_bn[0][0]'</td></tr><tr><td>conv5_block3_2_conv (Conv2</td><td>2359808</td><td>'conv5_block3_1_relu[0][0]'</td></tr><tr><td>conv5_block3_2_bn (BatchNormalizatio</td><td>2048</td><td>'conv5_block3_2_conv[0][0]'</td></tr><tr><td>conv5_block3_2_relu (Activatio</td><td>0</td><td>'conv5_block3_2_bn[0][0]'</td></tr><tr><td>conv5_block3_3_conv (Conv2</td><td>1050624</td><td>'conv5_block3_2_relu[0][0]'</td></tr><tr><td>conv5_block3_3_bn (BatchNormalizatio</td><td>8192</td><td>'conv5_block3_3_conv[0][0]'</td></tr><tr><td>conv5_block3_add (Add)</td><td>0</td><td>'conv5_block3_out[0][0]'; 'conv5_block3_3_bn[0][0]'</td></tr><tr><td>conv5_block3_out (Activatio</td><td>0</td><td>'conv5_block3_add[0][0]'</td></tr></tbody></table></div>	conv5_block2_2_bn (BatchNormalizatio	2048	'conv5_block2_2_conv[0][0]'	conv5_block2_2_relu (Activatio	0	'conv5_block2_2_bn[0][0]'	conv5_block2_3_conv (Conv2	1050624	'conv5_block2_2_relu[0][0]'	conv5_block2_3_bn (BatchNormalizatio	8192	'conv5_block2_3_conv[0][0]'	conv5_block2_add (Add)	0	'conv5_block2_out[0][0]'; 'conv5_block2_3_bn[0][0]'	conv5_block2_out (Activatio	0	'conv5_block2_add[0][0]'	conv5_block3_1_conv (Conv2	1049808	'conv5_block2_out[0][0]'	conv5_block3_1_bn (BatchNormalizatio	2048	'conv5_block3_1_conv[0][0]'	conv5_block3_1_relu (Activatio	0	'conv5_block3_1_bn[0][0]'	conv5_block3_2_conv (Conv2	2359808	'conv5_block3_1_relu[0][0]'	conv5_block3_2_bn (BatchNormalizatio	2048	'conv5_block3_2_conv[0][0]'	conv5_block3_2_relu (Activatio	0	'conv5_block3_2_bn[0][0]'	conv5_block3_3_conv (Conv2	1050624	'conv5_block3_2_relu[0][0]'	conv5_block3_3_bn (BatchNormalizatio	8192	'conv5_block3_3_conv[0][0]'	conv5_block3_add (Add)	0	'conv5_block3_out[0][0]'; 'conv5_block3_3_bn[0][0]'	conv5_block3_out (Activatio	0	'conv5_block3_add[0][0]'	<div><div>resnet50.fit(train_generator, validation_data=validation_generator, epochs=5)</div><div>Epoch 1/5 117/117 [=====] - loss: 0.8004 - accuracy: 0.6407 - val_loss: 0.4021 - val_accuracy: 0.7947 Epoch 2/5 117/117 [=====] - loss: 0.4889 - accuracy: 0.7291 - val_loss: 0.4372 - val_accuracy: 0.8178 Epoch 3/5 117/117 [=====] - loss: 0.4786 - accuracy: 0.7638 - val_loss: 0.3858 - val_accuracy: 0.8378 Epoch 4/5 117/117 [=====] - loss: 0.4786 - accuracy: 0.7238 - val_loss: 0.4083 - val_accuracy: 0.7477 Epoch 5/5 117/117 [=====] - loss: 0.4616 - accuracy: 0.7338 - val_loss: 0.4179 - val_accuracy: 0.8335 keras.src.callbacks.history at 0x7f7f012ef8db</div></div>												
conv5_block2_2_bn (BatchNormalizatio	2048	'conv5_block2_2_conv[0][0]'																																																												
conv5_block2_2_relu (Activatio	0	'conv5_block2_2_bn[0][0]'																																																												
conv5_block2_3_conv (Conv2	1050624	'conv5_block2_2_relu[0][0]'																																																												
conv5_block2_3_bn (BatchNormalizatio	8192	'conv5_block2_3_conv[0][0]'																																																												
conv5_block2_add (Add)	0	'conv5_block2_out[0][0]'; 'conv5_block2_3_bn[0][0]'																																																												
conv5_block2_out (Activatio	0	'conv5_block2_add[0][0]'																																																												
conv5_block3_1_conv (Conv2	1049808	'conv5_block2_out[0][0]'																																																												
conv5_block3_1_bn (BatchNormalizatio	2048	'conv5_block3_1_conv[0][0]'																																																												
conv5_block3_1_relu (Activatio	0	'conv5_block3_1_bn[0][0]'																																																												
conv5_block3_2_conv (Conv2	2359808	'conv5_block3_1_relu[0][0]'																																																												
conv5_block3_2_bn (BatchNormalizatio	2048	'conv5_block3_2_conv[0][0]'																																																												
conv5_block3_2_relu (Activatio	0	'conv5_block3_2_bn[0][0]'																																																												
conv5_block3_3_conv (Conv2	1050624	'conv5_block3_2_relu[0][0]'																																																												
conv5_block3_3_bn (BatchNormalizatio	8192	'conv5_block3_3_conv[0][0]'																																																												
conv5_block3_add (Add)	0	'conv5_block3_out[0][0]'; 'conv5_block3_3_bn[0][0]'																																																												
conv5_block3_out (Activatio	0	'conv5_block3_add[0][0]'																																																												

InceptionV3

```
[ ] InceptionV3.summary()
```

Model: "model"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 256, 256, 3)	0	[]
conv2d (Conv2D)	(None, 127, 127, 32)	864	['input_1[0][0]']
batch_normalization (Batch Normalization)	(None, 127, 127, 32)	96	['conv2d[0][0]']
activation (Activation)	(None, 127, 127, 32)	0	['batch_normalization[0][0]']
conv2d_1 (Conv2D)	(None, 125, 125, 32)	9216	['activation[0][0]']
batch_normalization_1 (Batch Normalization)	(None, 125, 125, 32)	96	['conv2d_1[0][0]']
activation_1 (Activation)	(None, 125, 125, 32)	0	['batch_normalization_1[0][0]']
conv2d_2 (Conv2D)	(None, 125, 125, 64)	18432	['activation_1[0][0]']
batch_normalization_2 (Batch Normalization)	(None, 125, 125, 64)	192	['conv2d_2[0][0]']
activation_2 (Activation)	(None, 125, 125, 64)	0	['batch_normalization_2[0][0]']
max_pooling2d (MaxPooling2D)	(None, 62, 62, 64)	0	['activation_2[0][0]']
conv2d_3 (Conv2D)	(None, 62, 62, 80)	5120	['max_pooling2d[0][0]']
batch_normalization_3 (Batch Normalization)	(None, 62, 62, 80)	240	['conv2d_3[0][0]']
activation_3 (Activation)	(None, 62, 62, 80)	0	['batch_normalization_3[0][0]']
conv2d_4 (Conv2D)	(None, 60, 60, 192)	138240	['activation_3[0][0]']
batch_normalization_4 (Batch Normalization)	(None, 60, 60, 192)	576	['conv2d_4[0][0]']
activation_4 (Activation)	(None, 60, 60, 192)	0	['batch_normalization_4[0][0]']

```
[ ] InceptionV3.fit(train_generator, validation_data=validation_generator, epochs=5)
```

Epoch 1/5
137/137 [=====] - 4239s 13s/step - loss: 0.5572 - accuracy: 0.8221 - val_loss: 0.3898 - val_accuracy: 0.9845
Epoch 2/5
137/137 [=====] - 3983s 12s/step - loss: 0.3359 - accuracy: 0.8841 - val_loss: 0.3879 - val_accuracy: 0.8451
Epoch 3/5
137/137 [=====] - 3986s 12s/step - loss: 0.2376 - accuracy: 0.9139 - val_loss: 0.4145 - val_accuracy: 0.8459
Epoch 4/5
137/137 [=====] - 3848s 12s/step - loss: 0.1801 - accuracy: 0.9327 - val_loss: 0.6029 - val_accuracy: 0.8252
Epoch 5/5
137/137 [=====] - 3848s 12s/step - loss: 0.7318 - accuracy: 0.7561 - val_loss: 11.8876 - val_accuracy: 0.5798
keras.src.callbacks.history at 0x709d8b7107b0

Xception

```
[ ] Xception.summary()
```

Model: "model"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 256, 256, 3)	0	[]
block1_conv1 (Conv2D)	(None, 127, 127, 32)	864	['input_1[0][0]']
block1_conv1_bn (Batch Normalization)	(None, 127, 127, 32)	128	['block1_conv1[0][0]']
block1_conv1_act (Activation)	(None, 127, 127, 32)	0	['block1_conv1_bn[0][0]']
block1_conv2 (Conv2D)	(None, 125, 125, 64)	18432	['block1_conv1_act[0][0]']
block1_conv2_bn (Batch Normalization)	(None, 125, 125, 64)	256	['block1_conv2[0][0]']
block1_conv2_act (Activation)	(None, 125, 125, 64)	0	['block1_conv2_bn[0][0]']
block2_sepconv1 (Separable Conv2D)	(None, 125, 125, 128)	8768	['block1_conv2_act[0][0]']
block2_sepconv1_bn (Batch Normalization)	(None, 125, 125, 128)	512	['block2_sepconv1[0][0]']
block2_sepconv2_act (Activation)	(None, 125, 125, 128)	0	['block2_sepconv1_bn[0][0]']
block2_sepconv2 (Separable Conv2D)	(None, 125, 125, 128)	17536	['block2_sepconv2_act[0][0]']
block2_sepconv2_bn (Batch Normalization)	(None, 125, 125, 128)	512	['block2_sepconv2[0][0]']
conv2d (Conv2D)	(None, 63, 63, 128)	8192	['block2_sepconv2_bn[0][0]']
block2_pool (MaxPooling2D)	(None, 63, 63, 128)	0	['conv2d[0][0]']
batch_normalization (Batch Normalization)	(None, 63, 63, 128)	512	['conv2d[0][0]']
add (Add)	(None, 63, 63, 128)	0	['block2_pool[0][0]', 'batch_normalization[0][0]']
block3_sepconv1_act (Activation)	(None, 63, 63, 128)	0	['add[0][0]']

```
[ ] Xception.fit(train_generator, validation_data=validation_generator, epochs=5)
```

Epoch 1/5
137/137 [=====] - 1880s 6s/step - loss: 0.7125 - accuracy: 0.7125 - val_loss: 0.3625 - val_accuracy: 0.9382
Epoch 2/5
137/137 [=====] - 1356s 61ms/step - loss: 0.4983 - accuracy: 0.7914 - val_loss: 0.3387 - val_accuracy: 0.9288
Epoch 3/5
137/137 [=====] - 1346s 61ms/step - loss: 0.4281 - accuracy: 0.8245 - val_loss: 0.3284 - val_accuracy: 0.9481
Epoch 4/5
137/137 [=====] - 1346s 61ms/step - loss: 0.3912 - accuracy: 0.8397 - val_loss: 0.3189 - val_accuracy: 0.9123
Epoch 5/5
137/137 [=====] - 1354s 608ms/step - loss: 0.3545 - accuracy: 0.8541 - val_loss: 0.2989 - val_accuracy: 0.9847
keras.src.callbacks.history at 0x777912d7f4d0
keras.src.callbacks.history at 0x709d8b7107b0

Final Model Selection Justification

The model VGG 16 provides the best accuracy of 95.4%, hence it is the selected model.

Advantages and Disadvantages

Advantages

Chest X-ray (CXR) imaging has several advantages over other imaging and detection techniques, like the ability to perform them easily using portable X-ray machines providing faster, and accurate COVID-19 diagnosis. Utilizing pretrained CNNs enables an automatic learning process that identifies important features from raw input data through convolutional layers, reducing the need for manual feature engineering. Using pre-trained models like InceptionV3 saves significant time and computational resources compared to training a CNN from scratch, enabling quick deployment.

Disadvantages

Training a Convolutional Neural Network on large datasets can require significant computational resources, including powerful GPUs and large amounts of memory, which can be expensive and time consuming.

CXR imaging exposes patients to radiation although in low doses, repeated exposures can increase the cumulative risk of radiation-induced conditions. CXRs may not detect early-stage or subtle abnormalities. More traditional methods like RT-PCR is highly sensitive and specific for detecting the genetic material of pathogens, making it ideal for diagnosing infections like COVID-19, influenza, and other viral diseases, allowing detection at earlier stages, due to its ability to amplify small amounts of viral RNA.

Conclusion

In this project, we aimed to develop a machine learning model capable of predicting COVID-19 infection from chest X-ray (CXR) images. By leveraging the power of pre-trained TensorFlow models, we sought to achieve high accuracy and robust performance in this critical diagnostic task. The pre-trained models used in this study included InceptionV3, VGG-16, ResNet-50, and Xception.

The results of our experiments demonstrated that pre-trained models are highly effective for the task of COVID-19 detection from CXR images. Each model's performance was evaluated based on its accuracy and overall robustness. Our findings revealed notable differences in performance among the models, with VGG-16 providing the highest accuracy at 95.4%.

Overall, the pre-trained models significantly reduced the training time and resources needed, while achieving high accuracy even with limited labeled data. The transfer learning approach enabled the models to generalize well to the task of COVID-19 detection, highlighting the effectiveness of utilizing pre-trained networks in medical imaging applications.

In conclusion, our comparative analysis underscores the potential of using pre-trained CNN models for rapid and accurate COVID-19 diagnosis from CXR images. This project demonstrates the feasibility and promise of leveraging state-of-the-art deep learning techniques to aid in the timely diagnosis and management of COVID-19, ultimately contributing to better patient outcomes and healthcare resource optimization.

Future Scope

Future work could focus on further fine-tuning these models with larger and more diverse datasets, exploring the integration of additional clinical data and ensemble techniques, mutlimodal integration and bias mitigation to enhance predictive performance, and evaluating the models in real-world clinical settings.

Fine-Tuning the existing models and experimenting with with different architectures (such as attention-based networks) and hyperparameters hep to achieve even better performance. Ensemble techniques can enhance robustness and generalization.

Multi-modal approaches help in providing a holistic view of the disease.Collaborating with healthcare professionals to validate the model's performance in clinical settings and conducting prospective studies help to assess its real-world impact.

Regularly auditing the model for biases related to demographics (race, gender, age) by implement techniques to mitigate bias and ensure fairness and safeguarding data during deployment are further future scopes for this project.

Appendix

Source Code

```
from google.colab import drive
drive.mount('/content/drive/')

%cd /content/drive/MyDrive

!pip install opendatasets
import opendatasets as od
import pandas as pd
od.download("https://www.kaggle.com/datasets/tawsifurrahman/covid19-radiography-database")
```

```

!unzip 'content/drive/MyDrive/covid19-radiography-database/covid19-radiography-database.zip' -d
'/content/drive/MyDrive/covid19-radiography-database'

!ls "/content/drive/MyDrive/covid19-radiography-database/COVID-19_Radiography_Dataset"

!mkdir /content/drive/MyDrive/Xray_train_data

!cp -R "/content/drive/MyDrive/covid19-radiography-database/COVID-19_Radiography_Dataset/COVID/images"
"/content/drive/MyDrive/Xray_train_data/COVID"

!ls -l "/content/drive/MyDrive/Xray_train_data/COVID"| wc -l\

!mkdir "/content/drive/MyDrive/Xray_train_data/NORMAL"

import pandas as pd
PATH_TO_METADATA = "/content/drive/MyDrive/covid19-radiography-database/COVID-
19_Radiography_Dataset/Normal.metadata.xlsx"
df = pd.read_excel(PATH_TO_METADATA)
df.head()

import os
import shutil
cnt = 0
for (i, row) in df.iterrows():
    if (cnt < 3616):
        filename = row["FILE NAME"].lower().capitalize() + "." + row["FORMAT"].lower()
        image_path = os.path.join("/content/drive/MyDrive/covid19-radiography-database/COVID-
19_Radiography_Dataset/Normal/images", filename)
        image_copy_path = os.path.join("/content/drive/MyDrive/Xray_train_data/NORMAL", filename)
        shutil.copy2(image_path, image_copy_path)
        cnt += 1

print(cnt)

!ls "/content/drive/MyDrive/Xray_train_data"

IMAGE_SIZE = (256, 256)
IMAGE_SHAPE = IMAGE_SIZE + (3,)

train_data_dir= "/content/drive/MyDrive/Xray_train_data"

test_data_dir= "/content/drive/MyDrive/Xray_train_data"

```

The rest of the code is available in the GitHub repository.

GitHub Repository

<https://github.com/Gokulselvadurai/COVID-19-Detection-from-Lung-X-rays.git>

Project Demo Link

<https://drive.google.com/file/d/1zRcDFcfjD7GJazEsfMtjHTcjK5j6-gOz/view?usp=drivesdk>