Data Collection and Preprocessing

Project Name: COVID-19 Chest X-Ray Image Classification

The dataset of chest X-ray images was prepared for deep learning classification to detect COVID-19, Bacterial Pneumonia, and Normal cases using a Convolutional Neural Network (CNN) with transfer learning (VGG16)

PREPROCESSING STEPS :

SECTION	DESCRIPTION
Data Overview	Collected chest X-ray dataset organized into train and test folders, each containing subfolders for classes (Normal, COVID-19, Bacteria).
Image Resizing	Resized all images to 64×64 pixels for uniform input into the CNN model.
Rescaling	Applied pixel normalization by rescaling image values from 0–255 → 0–1.
Data Augmentation	Used ImageDataGenerator to apply transformations and split 20% of training data into validation automatically.
Train / Validation split	Training set = 80% of train folder, Validation set = 20% of train folder (via subset in ImageDataGenerator).
Test set	Loaded separately from the test directory without augmentation, only rescaling.
Label Encoding	Automatically handled by flow_from_directory, mapping class subfolders to categorical one-hot labels.
Model Preparation	Used VGG16 pretrained on ImageNet as

	the base model (without top layers). Added custom layers: Flatten → Dense → Dropout → Output.
Visualization	Generated plots comparing training accuracy/loss vs validation accuracy/loss across epochs.

Data Preprocessing Code Snapshots :

SECTION	**CODE**
Data Overview	[1]: `import os` `os.listdir("chest_xray_covid/Data")` [1]: `['test', 'train']`
Image Resizing	[2]: `Image_size = [64, 64]` `train_path = 'chest_xray_covid/Data/train'` `test_path = 'chest_xray_covid/Data/test'`
Train / Validation split	`import tensorflow as tf` `from tensorflow import keras` `from tensorflow.keras.preprocessing.image import ImageDataGenerator` `# Rescale images` `train_datagen = ImageDataGenerator(` `rescale=1./255,` `validation_split=0.2 # <-- split 20% of train into validation` `)` `test_datagen = ImageDataGenerator(rescale=1./255)` `# Training set (80% of train data)` `training_set = train_datagen.flow_from_directory(` `'chest_xray_covid/Data/train',` `target_size=(64, 64),` `batch_size=32,` `class_mode='categorical',` `subset='training', # <-- important` `seed=42` `)`

	``python # Validation set (20% of train data) val_set = train_datagen.flow_from_directory('chest_xray_covid/Data/train', target_size=(64, 64), batch_size=32, class_mode='categorical', subset='validation', # <-- important seed=42) ```
Test set	```python # Test set (uses separate test folder) test_set = test_datagen.flow_from_directory('chest_xray_covid/Data/test', target_size=(64, 64), batch_size=32, class_mode='categorical') ```
Model Preparation	```python from keras.applications import VGG16 from tensorflow.keras.models import Model from tensorflow.keras.layers import Dense, Flatten, Dropout, Input ``` ```python base_model = VGG16(input_shape = (64, 64, 3), include_top=False , weights="imagenet") ``` ```python inp = base_model.input x = base_model.output x = Flatten()(x) x = Dense(1024, activation = 'relu')(x) x = Dropout(0.5)(x) output = Dense(3, activation = 'softmax')(x) model = Model(inputs = inp, outputs = output) ``` ```python model.summary() ``` ```python model.compile(loss = 'categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy']) ``` ```python test_loss, test_acc = model.evaluate(test_set) ``` ``` C:\Users\lenovo\AppData\Local\Programs\Python\Python313\Lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapt `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multi . Do not pass these arguments to `fit()`, as they will be ignored. self._warn_if_super_not_called() 41/41 ━━━━━━━━━━━━━━━━━━━━ 71s 2s/step - accuracy: 0.0908 - loss: 1.5700 ``` ```python history = model.fit(training_set, epochs = 5, validation_data = val_set) ```
Visualization	```python import matplotlib.pyplot as plt # Assuming you already trained your model # history = model.fit(...) # Plot accuracy plt.figure(figsize=(8, 6)) plt.plot(history.history['accuracy'], label='Training Accuracy') plt.plot(history.history['val_accuracy'], label='Validation Accuracy') plt.title('Model Accuracy') plt.xlabel('Epoch') plt.ylabel('Accuracy') plt.legend(loc='lower right') plt.show() # Plot loss plt.figure(figsize=(8, 6)) plt.plot(history.history['loss'], label='Training Loss') plt.plot(history.history['val_loss'], label='Validation Loss') plt.title('Model Loss') plt.xlabel('Epoch') plt.ylabel('Loss') plt.legend(loc='upper right') plt.show() ```