Libraries

```
!pip install tensorflow
!pip install scikit-learn
!pip install matplotlib
!pip install Pillow
!pip install keras
!pip install opencv-python
```

Show hidden output

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import cv2
import shutil
import os
import random
import collections
import seaborn as sns
import zipfile
import os
import pandas as pd
import matplotlib.image as mpimg

from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from sklearn.model_selection import train_test_split
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, BatchNormalization
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from tensorflow.keras.optimizers import Adam
from sklearn.utils.class_weight import compute_class_weight
from sklearn.metrics import confusion_matrix
from keras.utils import load_img
from sklearn.model_selection import train_test_split
from tensorflow.keras.applications import EfficientNetB0
from tensorflow.keras.layers import GlobalAveragePooling2D
```

Load Data

```
from google.colab import files
files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving kaggle.json to kaggle.json

```
!pip install -q kaggle
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

```
!kaggle datasets download -d paultimothymooney/chest-xray-pneumonia
```

Dataset URL: https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia
License(s): other

```
!unzip -q chest-xray-pneumonia.zip -d /content/data
```

```
!ls /content/data/chest_xray
```

chest_xray __MACOSX test train val

```
print(os.listdir("/content/data/chest_xray/chest_xray"))
```

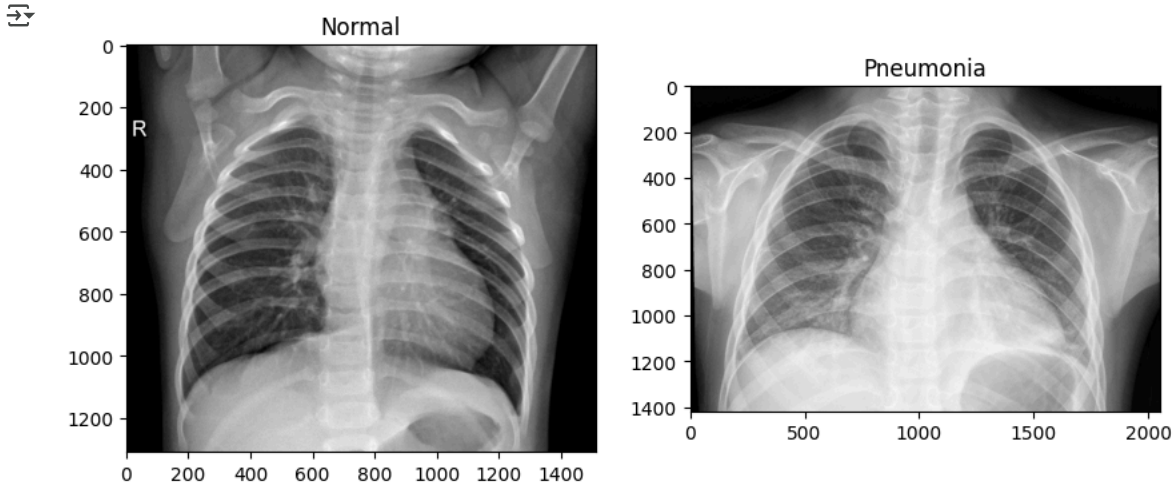
['.DS_Store', 'train', 'val', 'test']

Pre Processing

```
from sklearn.model_selection import train_test_split

base_dir = "/content/data/chest_xray/chest_xray"
train_dir = os.path.join(base_dir, 'train')
val_dir = os.path.join(base_dir, 'val')
test_dir = os.path.join(base_dir, 'test')

print("Total Numbers of Images in Train Folder:")
print("Normal:", len(os.listdir(os.path.join(train_dir, "NORMAL"))))
print("Pneumonia:", len(os.listdir(os.path.join(train_dir, "PNEUMONIA"))))

print("Total Numbers of Images in Validation Folder:")
print("Normal:", len(os.listdir(os.path.join(val_dir, "NORMAL"))))
print("Pneumonia:", len(os.listdir(os.path.join(val_dir, "PNEUMONIA"))))

print("Total Numbers of Images in Test Folder:")
print("Normal:", len(os.listdir(os.path.join(test_dir, "NORMAL"))))
print("Pneumonia:", len(os.listdir(os.path.join(test_dir, "PNEUMONIA"))))
```

```
Total Numbers of Images in Train Folder:
Normal: 1342
Pneumonia: 3876
Total Numbers of Images in Validation Folder:
Normal: 9
Pneumonia: 9
Total Numbers of Images in Test Folder:
Normal: 234
Pneumonia: 390
```

```
normal_img = os.path.join(train_dir, "NORMAL", os.listdir(os.path.join(train_dir, "NORMAL"))[0])
pneumonia_img = os.path.join(train_dir, "PNEUMONIA", os.listdir(os.path.join(train_dir, "PNEUMONIA"))[0])

fig, ax = plt.subplots(1, 2, figsize=(10,5))
ax[0].imshow(mpimg.imread(normal_img), cmap='gray')
ax[0].set_title("Normal")
ax[1].imshow(mpimg.imread(pneumonia_img), cmap='gray')
ax[1].set_title("Pneumonia")
plt.show()
```




```
!pip install split-folders
```

Show hidden output

```
import os
import shutil
import random
from sklearn.model_selection import train_test_split

original_dir = "/content/data/chest_xray/chest_xray/train"
```

```
new_base_dir = "/content/newDataSet"
os.makedirs(new_base_dir, exist_ok=True)

train_dir = os.path.join(new_base_dir, 'train')
val_dir = os.path.join(new_base_dir, 'val')
test_dir = os.path.join(new_base_dir, 'test')


for split_dir in [train_dir, val_dir, test_dir]:
    os.makedirs(os.path.join(split_dir, 'NORMAL'), exist_ok=True)
    os.makedirs(os.path.join(split_dir, 'PNEUMONIA'), exist_ok=True)


def copy_files(files, source_dir, dest_dir):
    for f in files:
        shutil.copy(os.path.join(source_dir, f), os.path.join(dest_dir, f))


for class_name in ['NORMAL', 'PNEUMONIA']:
    src_dir = os.path.join(original_dir, class_name)
    files = os.listdir(src_dir)
    random.shuffle(files)


    train_files, temp_files = train_test_split(files, test_size=0.3, random_state=42)
    val_files, test_files = train_test_split(temp_files, test_size=0.5, random_state=42)

    copy_files(train_files, src_dir, os.path.join(train_dir, class_name))
    copy_files(val_files, src_dir, os.path.join(val_dir, class_name))
    copy_files(test_files, src_dir, os.path.join(test_dir, class_name))


def print_counts(dir_path):
    print(f"\n{dir_path}:")
    print("NORMAL:", len(os.listdir(os.path.join(dir_path, "NORMAL"))))
    print("PNEUMONIA:", len(os.listdir(os.path.join(dir_path, "PNEUMONIA"))))

print_counts(train_dir)
print_counts(val_dir)
print_counts(test_dir)
```

```
/content/newDataSet/train:
NORMAL: 939
PNEUMONIA: 2713

/content/newDataSet/val:
NORMAL: 201
PNEUMONIA: 581

/content/newDataSet/test:
NORMAL: 202
PNEUMONIA: 582
```

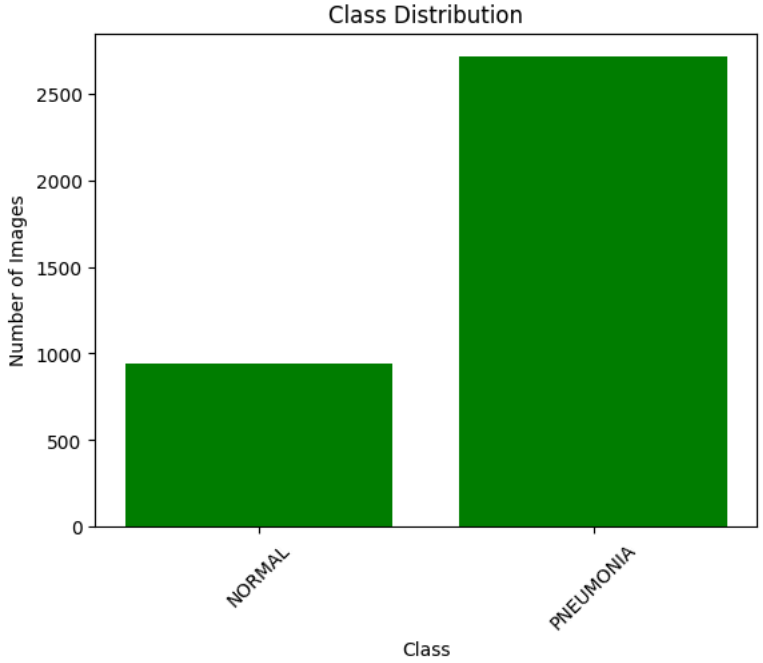
```
from sklearn.model_selection import train_test_split

new_base_dir = "/content/newDataSet"
train_dir = os.path.join(new_base_dir, 'train')
val_dir = os.path.join(new_base_dir, 'val')
test_dir = os.path.join(new_base_dir, 'test')

print("Total Numbers of Images in Train Folder:")
print("Normal:", len(os.listdir(os.path.join(train_dir, "NORMAL"))))
print("Pneumonia:", len(os.listdir(os.path.join(train_dir, "PNEUMONIA"))))

print("Total Numbers of Images in Validation Folder:")
print("Normal:", len(os.listdir(os.path.join(val_dir, "NORMAL"))))
print("Pneumonia:", len(os.listdir(os.path.join(val_dir, "PNEUMONIA"))))

print("Total Numbers of Images in Test Folder:")
print("Normal:", len(os.listdir(os.path.join(test_dir, "NORMAL"))))
print("Pneumonia:", len(os.listdir(os.path.join(test_dir, "PNEUMONIA"))))
```

```
Total Numbers of Images in Train Folder:
Normal: 939
Pneumonia: 2713
```

```
Total Numbers of Images in Validation Folder:
Normal: 201
Pneumonia: 581
Total Numbers of Images in Test Folder:
Normal: 202
Pneumonia: 582
```

```
def plot_class_distribution(directory):
    class_counts = {}
    for class_name in os.listdir(directory):
        class_path = os.path.join(directory, class_name)
        if os.path.isdir(class_path):
            class_counts[class_name] = len(os.listdir(class_path))

    plt.bar(class_counts.keys(), class_counts.values(), color='green')
    plt.title('Class Distribution')
    plt.xlabel('Class')
    plt.ylabel('Number of Images')
    plt.xticks(rotation=45)
    plt.show()

print("Train Distribution:")
plot_class_distribution('/content/newDataSet/train')
```

```
Train Distribution:
```




```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(224,224),
    class_mode='binary',
    batch_size=32
    )

validation_generator = train_datagen.flow_from_directory(
    val_dir,
    target_size=(224,224),
```

```
    class_mode='binary',
    batch_size=32
    )

test_datagen = ImageDataGenerator(rescale=1./255)

testing_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(224,224),
    class_mode='binary',
    batch_size=32,
    shuffle=False)
```

```
Found 3650 images belonging to 2 classes.
Found 782 images belonging to 2 classes.
Found 784 images belonging to 2 classes.
```

```
from sklearn.utils import class_weight

class_weights = class_weight.compute_class_weight(
    'balanced',
    classes=np.unique(train_generator.classes),
    y=train_generator.classes
)
class_weights = dict(enumerate(class_weights))

print("Class Weights:", class_weights)
```

```
Class Weights: {0: np.float64(1.9456289978678039), 1: np.float64(0.6729351032448377)}
```

Build Model

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, BatchNormalization, Activation, MaxPooling2D
from tensorflow.keras.layers import GlobalMaxPooling2D, Dense, Dropout
from tensorflow.keras.regularizers import l2
from tensorflow.keras.applications import Xception

base_model = Xception(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
base_model.trainable = False

model = Sequential([
    base_model,
    GlobalMaxPooling2D(),
    Dense(1024, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/xception/xception_weights_tf_dim_ordering_tf_kernels_
83683744/83683744 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

```
model.summary()
```

Show hidden output

```
from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, LearningRateScheduler

model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
    loss='binary_crossentropy',
     metrics=['accuracy',
                tf.keras.metrics.AUC(name='auc'),
                tf.keras.metrics.Precision(name='precision'),
                tf.keras.metrics.Recall(name='recall')])

lr_reduce = ReduceLROnPlateau(
    monitor='val_loss',
    factor=0.5,
    patience=13,
    verbose=1,
    min_lr=1e-6
```

```
 )

checkpoint_path = "best_model_Version_1.weights.h5"
checkpoint = ModelCheckpoint(
    checkpoint_path,
    monitor='val_accuracy',
    save_best_only=True,
    mode='max',
    verbose=1,
    save_weights_only=True

)

history = model.fit(
    train_generator,
    validation_data=validation_generator,
    epochs=50,
    callbacks=[checkpoint,lr_reduce],
     class_weight=class_weights
)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` clas
  self._warn_if_super_not_called()
Epoch 1/50
115/115 ━━━━━━━━━━━━━━━━━━━━ 0s 688ms/step - accuracy: 0.7373 - auc: 0.7937 - loss: 1.5600 - precision: 0.8901 - recall: 0.7344
Epoch 1: val_accuracy improved from -inf to 0.87596, saving model to best_model_Version_1.weights.h5
115/115 ━━━━━━━━━━━━━━━━━━━━ 122s 916ms/step - accuracy: 0.7379 - auc: 0.7944 - loss: 1.5531 - precision: 0.8904 - recall: 0.7350 - va
Epoch 2/50
115/115 ━━━━━━━━━━━━━━━━━━━━ 0s 658ms/step - accuracy: 0.8814 - auc: 0.9563 - loss: 0.2646 - precision: 0.9704 - recall: 0.8658
Epoch 2: val_accuracy improved from 0.87596 to 0.89258, saving model to best_model_Version_1.weights.h5
115/115 ━━━━━━━━━━━━━━━━━━━━ 93s 811ms/step - accuracy: 0.8815 - auc: 0.9564 - loss: 0.2644 - precision: 0.9704 - recall: 0.8660 - val
Epoch 3/50
115/115 ━━━━━━━━━━━━━━━━━━━━ 0s 656ms/step - accuracy: 0.8960 - auc: 0.9605 - loss: 0.2524 - precision: 0.9692 - recall: 0.8880
Epoch 3: val_accuracy improved from 0.89258 to 0.93223, saving model to best_model_Version_1.weights.h5
115/115 ━━━━━━━━━━━━━━━━━━━━ 92s 801ms/step - accuracy: 0.8960 - auc: 0.9605 - loss: 0.2523 - precision: 0.9692 - recall: 0.8880 - val
Epoch 4/50
115/115 ━━━━━━━━━━━━━━━━━━━━ 0s 663ms/step - accuracy: 0.9032 - auc: 0.9641 - loss: 0.2335 - precision: 0.9700 - recall: 0.9005
Epoch 4: val_accuracy did not improve from 0.93223
115/115 ━━━━━━━━━━━━━━━━━━━━ 142s 804ms/step - accuracy: 0.9032 - auc: 0.9641 - loss: 0.2336 - precision: 0.9700 - recall: 0.9005 - va
Epoch 5/50
115/115 ━━━━━━━━━━━━━━━━━━━━ 0s 657ms/step - accuracy: 0.9055 - auc: 0.9686 - loss: 0.2221 - precision: 0.9772 - recall: 0.8928
Epoch 5: val_accuracy improved from 0.93223 to 0.93350, saving model to best_model_Version_1.weights.h5
115/115 ━━━━━━━━━━━━━━━━━━━━ 93s 809ms/step - accuracy: 0.9055 - auc: 0.9687 - loss: 0.2220 - precision: 0.9771 - recall: 0.8929 - val
Epoch 6/50
115/115 ━━━━━━━━━━━━━━━━━━━━ 0s 655ms/step - accuracy: 0.9101 - auc: 0.9723 - loss: 0.2139 - precision: 0.9701 - recall: 0.9055
Epoch 6: val_accuracy did not improve from 0.93350
115/115 ━━━━━━━━━━━━━━━━━━━━ 92s 797ms/step - accuracy: 0.9100 - auc: 0.9722 - loss: 0.2140 - precision: 0.9701 - recall: 0.9055 - val
Epoch 7/50
115/115 ━━━━━━━━━━━━━━━━━━━━ 0s 659ms/step - accuracy: 0.9075 - auc: 0.9667 - loss: 0.2301 - precision: 0.9688 - recall: 0.9058
Epoch 7: val_accuracy improved from 0.93350 to 0.95396, saving model to best_model_Version_1.weights.h5
115/115 ━━━━━━━━━━━━━━━━━━━━ 93s 810ms/step - accuracy: 0.9075 - auc: 0.9667 - loss: 0.2300 - precision: 0.9688 - recall: 0.9057 - val
Epoch 8/50
115/115 ━━━━━━━━━━━━━━━━━━━━ 0s 657ms/step - accuracy: 0.9075 - auc: 0.9692 - loss: 0.2280 - precision: 0.9643 - recall: 0.9090
Epoch 8: val_accuracy did not improve from 0.95396
115/115 ━━━━━━━━━━━━━━━━━━━━ 92s 797ms/step - accuracy: 0.9075 - auc: 0.9692 - loss: 0.2279 - precision: 0.9643 - recall: 0.9090 - val
Epoch 9/50
115/115 ━━━━━━━━━━━━━━━━━━━━ 0s 661ms/step - accuracy: 0.9076 - auc: 0.9689 - loss: 0.2235 - precision: 0.9727 - recall: 0.8991
Epoch 9: val_accuracy did not improve from 0.95396
115/115 ━━━━━━━━━━━━━━━━━━━━ 92s 804ms/step - accuracy: 0.9076 - auc: 0.9689 - loss: 0.2235 - precision: 0.9727 - recall: 0.8991 - val
Epoch 10/50
115/115 ━━━━━━━━━━━━━━━━━━━━ 0s 657ms/step - accuracy: 0.9160 - auc: 0.9754 - loss: 0.1936 - precision: 0.9787 - recall: 0.9074
Epoch 10: val_accuracy did not improve from 0.95396
115/115 ━━━━━━━━━━━━━━━━━━━━ 91s 797ms/step - accuracy: 0.9160 - auc: 0.9753 - loss: 0.1938 - precision: 0.9786 - recall: 0.9074 - val
Epoch 11/50
115/115 ━━━━━━━━━━━━━━━━━━━━ 0s 664ms/step - accuracy: 0.9105 - auc: 0.9732 - loss: 0.2034 - precision: 0.9762 - recall: 0.9025
Epoch 11: val_accuracy did not improve from 0.95396
115/115 ━━━━━━━━━━━━━━━━━━━━ 92s 804ms/step - accuracy: 0.9105 - auc: 0.9732 - loss: 0.2034 - precision: 0.9762 - recall: 0.9025 - val
Epoch 12/50
115/115 ━━━━━━━━━━━━━━━━━━━━ 0s 655ms/step - accuracy: 0.9254 - auc: 0.9759 - loss: 0.1951 - precision: 0.9769 - recall: 0.9218
Epoch 12: val_accuracy did not improve from 0.95396
115/115 ━━━━━━━━━━━━━━━━━━━━ 91s 796ms/step - accuracy: 0.9254 - auc: 0.9759 - loss: 0.1951 - precision: 0.9769 - recall: 0.9218 - val
Epoch 13/50
115/115 ━━━━━━━━━━━━━━━━━━━━ 0s 666ms/step - accuracy: 0.9123 - auc: 0.9725 - loss: 0.2105 - precision: 0.9726 - recall: 0.9060
Epoch 13: val_accuracy did not improve from 0.95396
115/115 ━━━━━━━━━━━━━━━━━━━━ 93s 807ms/step - accuracy: 0.9123 - auc: 0.9725 - loss: 0.2105 - precision: 0.9726 - recall: 0.9060 - val
Epoch 14/50
115/115 ━━━━━━━━━━━━━━━━━━━━ 0s 657ms/step - accuracy: 0.9159 - auc: 0.9765 - loss: 0.1959 - precision: 0.9721 - recall: 0.9128
Epoch 14: val_accuracy did not improve from 0.95396
```

```
from tensorflow.keras.models import load_model
from google.colab import files
```

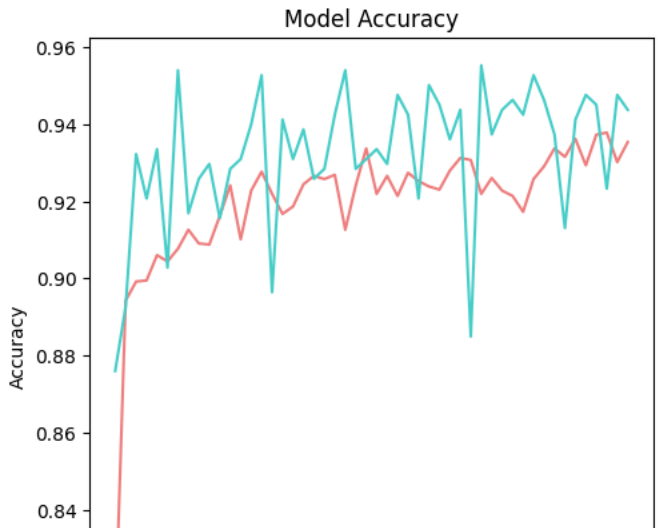
```
model.load_weights("best_model_Version_1.weights.h5")

model.save("best_model_full(x_ray).h5")
```

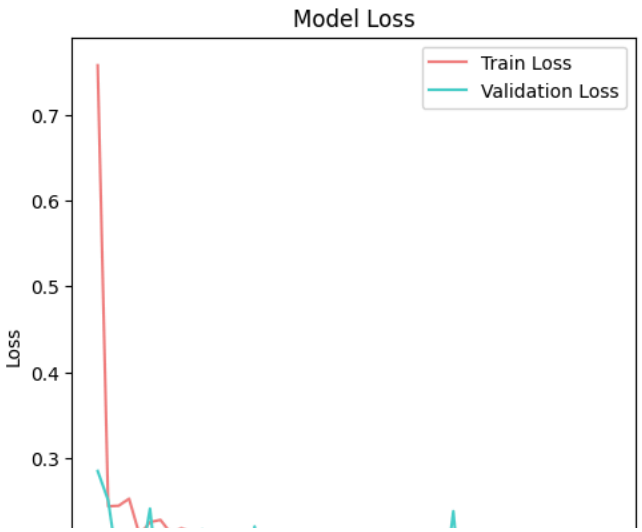
```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is consi
```

```
train_eval = model.evaluate(train_generator)
val_eval = model.evaluate(validation_generator)
test_eval = model.evaluate(testing_generator)

train_accu = train_eval[1]
val_accu = val_eval[1]
test_accu = test_eval[1]

print("Final train accuracy = {:.2f}%, validation accuracy = {:.2f}%, testing accuracy = {:.2f}%"
      .format(train_accu * 100, val_accu * 100, test_accu * 100))
```

```
115/115 ━━━━━━━━━━━━━━━━━━━━ 77s 669ms/step - accuracy: 0.9913 - auc: 0.9996 - loss: 0.0243 - precision: 0.9993 - recall: 0.9890
25/25 ━━━━━━━━━━━━━━━━━━━━ 16s 635ms/step - accuracy: 0.9822 - auc: 0.9977 - loss: 0.0575 - precision: 0.9974 - recall: 0.9787
25/25 ━━━━━━━━━━━━━━━━━━━━ 10s 401ms/step - accuracy: 0.9747 - auc: 0.7672 - loss: 0.0753 - precision: 0.7692 - recall: 0.7242
Final train accuracy = 98.90%, validation accuracy = 98.34%, testing accuracy = 95.66%
```

```
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy', color='lightcoral')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy', color='mediumturquoise')

plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss', color='lightcoral')
plt.plot(history.history['val_loss'], label='Validation Loss', color='mediumturquoise')
plt.title('Model Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()



plt.show()
```


Model Accuracy
Accuracy
0.96
0.94
0.92
0.90
0.88
0.86
0.84


Model Loss
Train Loss
Validation Loss
Loss
0.7
0.6
0.5
0.4
0.3