



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

J Component report

Programme : B.Tech(SCOPE)

Course Title : ARTIFICIAL INTELLIGENCE

Course Code : CSE3013

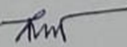
Slot : B2

Title: Pneumonia Detection using CNN and Transfer Learning

Team Members: Ayyappan K M - 19BCE1410

Bollineni Nishanth - 19BCE1805

Faculty: Dr.Padmavathy T V

Sign: 

Date: 28/4/2022

INDEX

S. No.	CONTENT	PAGE No.
1.	ACKNOWLEDGMENT	3
2.	ABSTRACT	4
3.	(i)INTRODUCTION (ii) Why AI is needed for this project? (iii) PEAS Description, Environment type, Architecture type (iv) Existing system (v) Proposed methodology	5
4.	IMPLEMENTATION	11
5.	IMPLEMENTATION DETAILS(SOFTWARES USED)	15
6.	RESULTS AND DISCUSSIONS	16
7.	CONCLUSION AND FUTURE ENHANCEMENTS	17
8.	REFERENCES	18
9.	SAMPLE CODE	19

ACKNOWLEDGMENT

Primarily, we would like to thank the almighty for all the blessings he showered over us to complete this project without any flaws.

The success and final outcome of this assignment required a lot of guidance and assistance from many people and we are extremely fortunate to have got this all along with the completion of our project. Whatever we have done is only due to such guidance and assistance by our faculty, Dr. S.L.JAYALAKSHMI and Dr. PADMAVATHY T V, to whom we are really thankful for giving us an opportunity to do this project.

Last but not the least, we are grateful to all our fellow classmates and our friends for the suggestions and support given to us throughout the completion of our project.

Abstract

Pneumonia is a respiratory infection caused by bacteria or viruses; it affects many individuals, especially in developing and underdeveloped nations, where high levels of pollution, unhygienic living conditions, and overcrowding are relatively common, together with inadequate medical infrastructure. Pneumonia causes pleural effusion, a condition in which fluids fill the lung, causing respiratory difficulty. Early diagnosis of pneumonia is crucial to ensure curative treatment and increase survival rates. Chest X-ray imaging is the most frequently used method for diagnosing pneumonia. However, the examination of chest X-rays is a challenging task and is prone to subjective variability. In this study, we developed a computer-aided diagnosis system for automatic pneumonia detection using chest X-ray images. We employed deep transfer learning to handle the scarcity of available data and designed an ensemble of three convolutional neural network models: CNN, ResNet-50, and DenseNet-121.

INTRODUCTION

Pneumonia is an acute pulmonary infection that can be caused by bacteria, viruses, or fungi and infects the lungs, causing inflammation of the air sacs and pleural effusion, a condition in which the lung is filled with fluid. It accounts for more than 15% of deaths in children under the age of five years. Pneumonia is most common in underdeveloped and developing countries, where overpopulation, pollution, and unhygienic environmental conditions exacerbate the situation, and medical resources are scanty. Therefore, early diagnosis and management can play a pivotal role in preventing the disease from becoming fatal. Radiological examination of the lungs using computed tomography (CT), magnetic resonance imaging (MRI), or radiography (X-rays) is frequently used for diagnosis. X-ray imaging constitutes a non-invasive and relatively inexpensive examination of the lungs. The white spots in the pneumonic X-ray called infiltrates, distinguish a pneumonic from a healthy condition. However, chest X-ray examinations for pneumonia detection are prone to subjective variability. Thus, an automated system for the detection of pneumonia is required.

WHY AI IS NEEDED FOR THIS PROJECT?

Artificially intelligent computer systems are used extensively in medical sciences. Common applications include diagnosing patients, end-to-end drug discovery and development, improving communication between physician and patient, transcribing medical documents, etc.

Hence the use of AI to read, analyse and determine if there is any underlying medical condition from an Chest X-Ray will be highly beneficial to the medical community and help in the advancement of medical sciences.

PEAS Description, Environment and Architecture type

The PEAS system delivers the performance measure with respect to the environment, actuators and sensors of the respective agent.

When we define an AI agent or rational agent, then we group its properties under PEAS representation model

P - Performance Measure

E - Environment

A - Actuators

S - Sensors

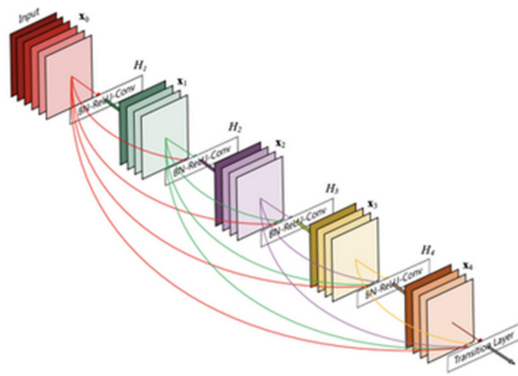
Performance Measure - minimize X-Ray analysis time, Reduce patient waiting time for results

Environment - Hospital, Patient, Customer, Doctor

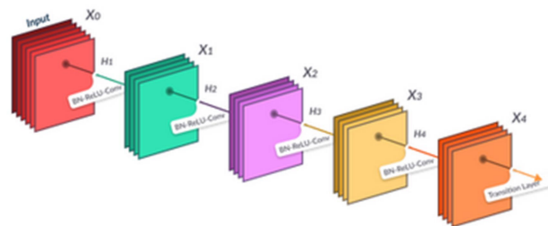
Actuators - Screen to Display Results

Sensors - Keyboard, mouse to upload X-Ray

Architecture of our Project

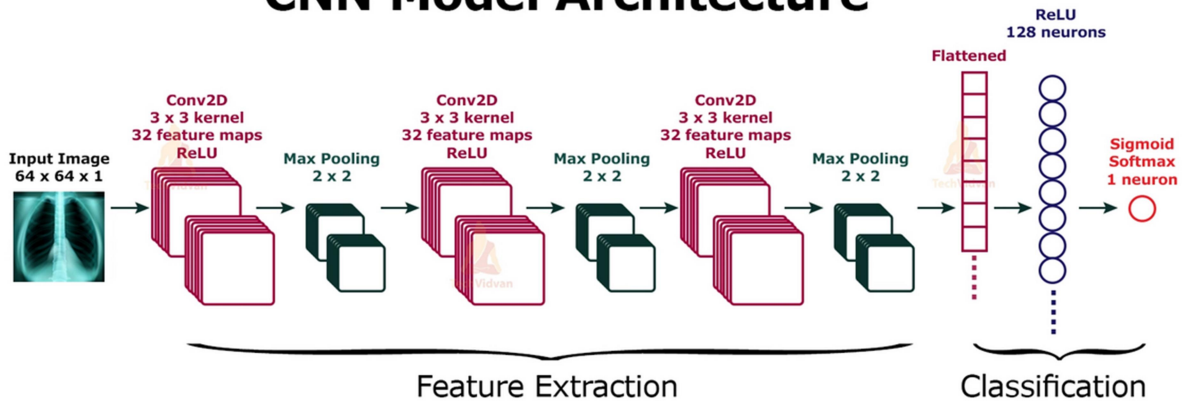


DenseNet Structure



ResNet Structure

CNN Model Architecture



Existing system

Pneumonia detection using chest X-rays has been an open problem for many years, the main limitation being the scarcity of publicly available data. Traditional machine learning methods have been explored extensively.

In contrast to machine learning algorithms, for which handcrafted features need to be extracted and selected for classification or segmentation deep learning-based methods perform end-to-end classification where the relevant and informative features are automatically extracted from the input data and classified. CNNs are preferred for image data classification because they automatically extract translationally invariant features through the convolution of the input image and filters. CNNs are translationally invariant and perform better than machine learning or traditional image processing methods in image classification tasks and thus are widely used by researchers.

Proposed Methodology

CNN:

In deep learning, a convolutional neural network (CNN/ConvNet) is a class of deep neural networks, most commonly applied to analyze visual imagery. Now when we think of a neural network we think about matrix multiplications but that is not the case with ConvNet. It uses a special technique called Convolution. Bottom line is that the role of the ConvNet is to reduce the images into a form that is easier to process, without losing features that are critical for getting a good prediction.

Transfer Learning:

Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task.

It is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks given the vast compute and time resources required to develop neural network models on these problems and from the huge jumps in skill that they provide on related problems.

We have used Densenet-121 and Resnet-50 as our transfer learning models.

IMPLEMENTATION

Data Visualization:

The dataset is divided into three sets: 1) Train set 2) Validation set and 3) Test set.

Train set:

=====

PNEUMONIA=3110

NORMAL=1082

Test set:

=====

PNEUMONIA=390

NORMAL=234

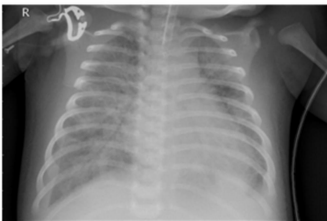
Validation set:

=====

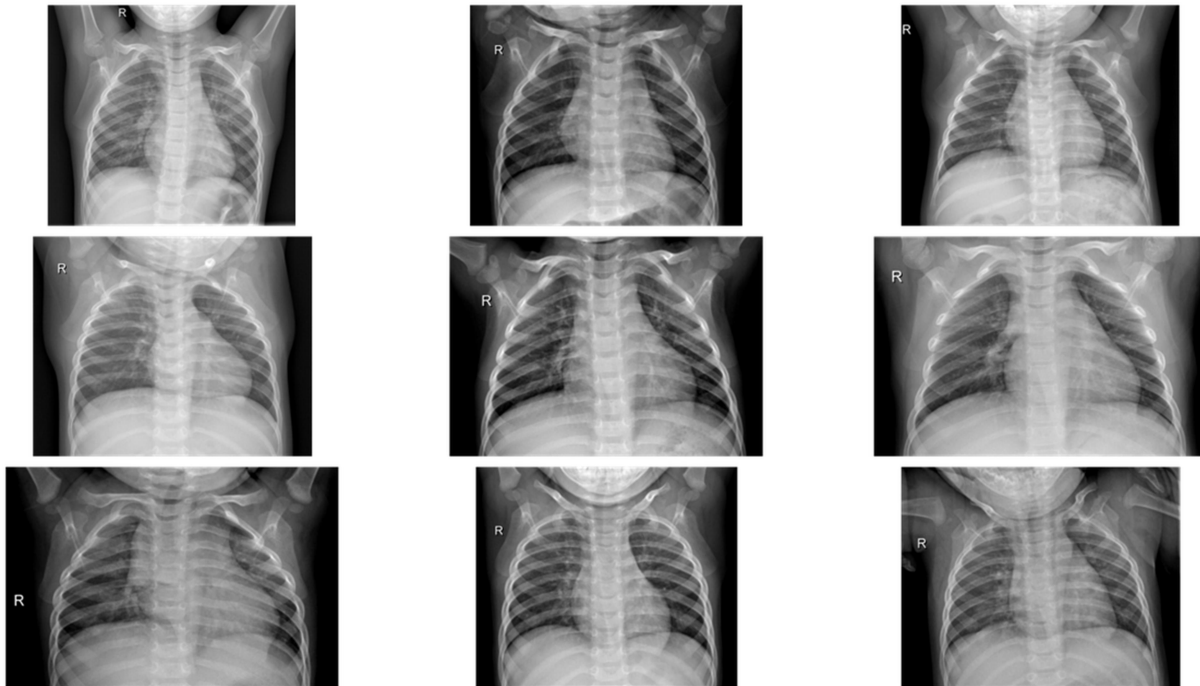
PNEUMONIA=773

NORMAL=267

Sample Images that are have opacity which means that they have pneumonia.



Sample image of normal Chest X-Rays :



Pixel Statistics:

The dimensions of the image are 1858 pixels width and 2090 pixels height, one single color channel.

The maximum pixel value is 255.0000 and the minimum is 0.0000

The mean value of the pixels is 128.9075 and the standard deviation is 62.3010



Investigate pixel value distribution:

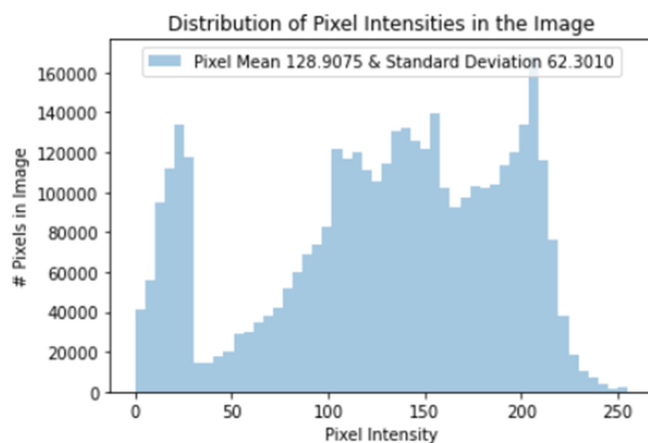


Image Preprocessing:

Before training, we'll first modify your images to be better suited for training a convolutional neural network. For this task we'll use the Keras ImageDataGenerator function to perform data preprocessing and data augmentation.

This class also provides support for basic data augmentation such as random horizontal flipping of images. We also use the generator to transform the values in each batch so that their mean is 0 and their standard deviation is 1 (this will facilitate model training by standardizing the input distribution). The generator also converts our single channel X-ray images (gray-scale) to a three-channel format by repeating the values in the image across all channels (we will want this because the pre-trained model that we'll use requires three-channel inputs).

Build a separate generator for valid and test sets

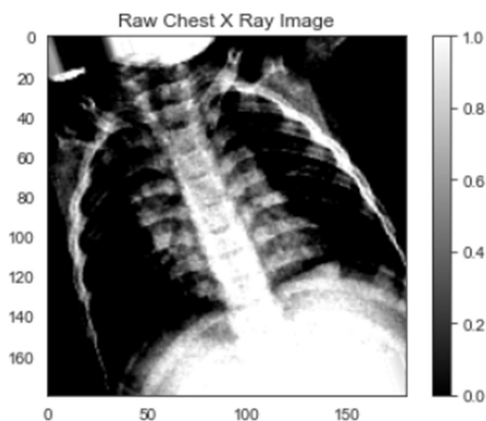
Now we need to build a new generator for validation and testing data.

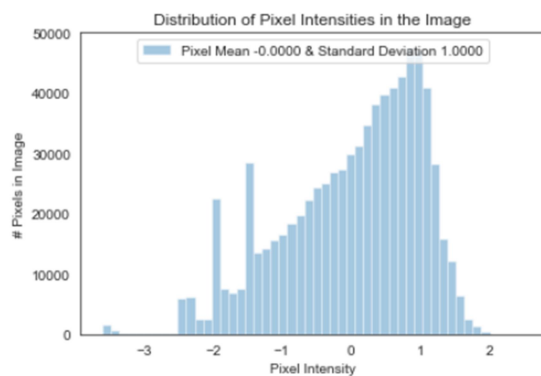
It normalizes each image per batch, meaning that it uses batch statistics. We should not do this with the test and validation data, since in a real life scenario we don't process incoming images a batch at a time (we process one image at a time). Knowing the average per batch of test data would effectively give our model an advantage (The model should not have any information about the test data). What we need to do is to normalize incoming test data using the statistics computed from the training set.

The dimensions of the image are 180 pixels width and 180 pixels height, one single color channel.

The maximum pixel value is 2.4967 and the minimum is -3.6118

The mean value of the pixels is -0.0000 and the standard deviation is 1.0000





IMPLEMENTATION DETAILS (SOFTWARES USED)

We have trained our model in jupyter notebook which is run locally but google collab can also be used.

The code requires a python version 3.8 and above to run smoothly with the prerequisite libraries such as

- Matplotlib == 3.4.3
- Numpy == 1.19.4
- Pandas == 1.3.1
- Seaborn == 0.11.2
- Tensorflow == 2.7.0

Hardware Requirements:

- Minimum 8 GB RAM
- Minimum 20GB Storage Space
- Good enough processor (at least greater than or equivalent to Intel i5 with GPU support)

RESULTS AND DISCUSSIONS

After Testing our models on the test dataset we observed the following results:
CNN :

	0	1	accuracy	macro avg	weighted avg
precision	0.763158	0.794931	0.785256	0.779044	0.783016
recall	0.619658	0.884615	0.785256	0.752137	0.785256
f1-score	0.683962	0.837379	0.785256	0.760670	0.779847
support	234.000000	390.000000	0.785256	624.000000	624.000000

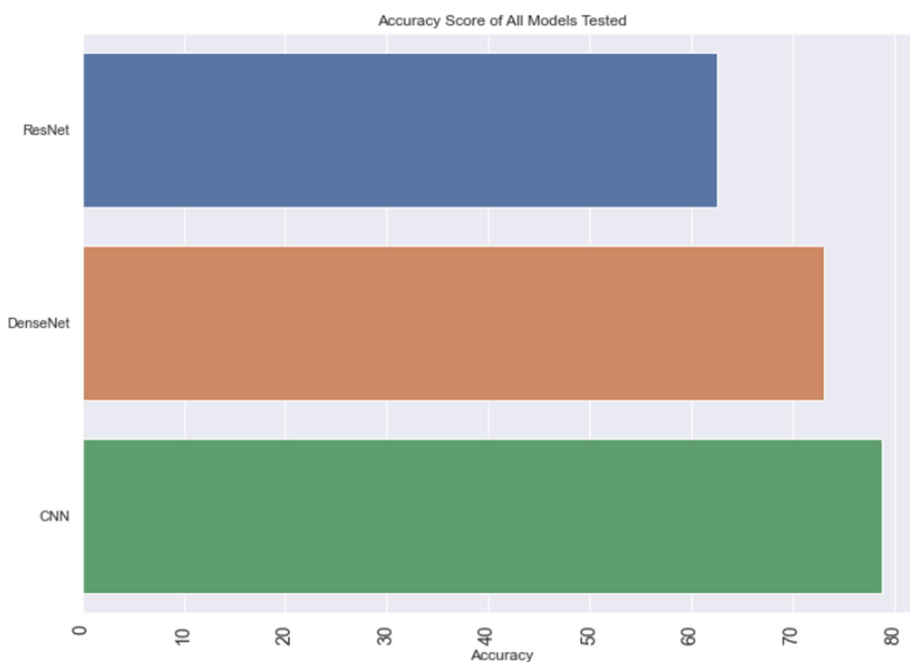
DenseNet:

	0	1	accuracy	macro avg	weighted avg
precision	0.698198	0.803483	0.766026	0.750840	0.764001
recall	0.662393	0.828205	0.766026	0.745299	0.766026
f1-score	0.679825	0.815657	0.766026	0.747741	0.764720
support	234.000000	390.000000	0.766026	624.000000	624.000000

Resnet:

```
624/624 [=====] - 81s 129ms/step - loss: 4.2737 - accuracy: 0.6250 -  
0  
Test Accuracy: 62.50%  
524/524 [=====] - 462s 882ms/step - loss: 2.8923 - accuracy: 0.7419  
00  
Train Accuracy: 74.19%
```

Comparing Different Models :



As we can see CNN performs better than DenseNet and ResNet with an accuracy score of 78% which can also be further be improved by fine tuning the parameters.

CONCLUSION AND FUTURE ENHANCEMENTS

In our modern world, technology is present everywhere from video games to performing surgeries in real life. Technology has made our lives easier. The CNN model has achieved the following metrics after the training, test accuracy of 91% and train accuracy of 78% . The Transfer net models Dense net has a train accuracy of 81.04% and test accuracy of 78.08%, whereas ResNet has a train accuracy of 74.19% and test accuracy of 62.50%. The above metrics suggest that the models is performing a decent job in classifying medical Xray images, and helping doctors in identifying the underlying illness present by analysing the chest X rays

REFERENCES

Neuman M., Lee E., Bixby S., Diperna S., Hellinger J., Markowitz R., et al. Variability in the interpretation of chest radiographs for the diagnosis of pneumonia in children. *Journal Of Hospital Medicine*. 7, 294–298 (2012) pmid:22009855

Rahman T., Chowdhury M., Khandakar A., Islam K., Islam K., Mahbub Z., et al. Transfer learning with deep convolutional neural network (CNN) for pneumonia detection using chest X-ray. *Applied Sciences*. 10, 3233 (2020)

Rajpurkar P., Irvin J., Zhu K., Yang B., Mehta H., Duan T., et al. & Others Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning.

Sharma H., Jain J., Bansal P. & Gupta S. Feature extraction and classification of chest x-ray images using cnn to detect pneumonia. *2020 10th International Conference On Cloud Computing, Data Science & Engineering (Confluence)*

Tang YX, Tang YB, Peng Y, et al. “Automated abnormality classification of chest radiographs using deep convolutional neural networks”. *NPJ Digit Med*. 2020;3:70. Published 2020 May 14.doi:10.1038/s41746-020-0273-z

Sample code

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
from tensorflow import keras

train_dir = "train"
test_dir = "test"
val_dir = "val"

print("Train
set:\n=====
==")
num_pneumonia = len(os.listdir(os.path.join(train_dir,
'opacity')))
num_normal = len(os.listdir(os.path.join(train_dir,
'normal')))
print(f"PNEUMONIA={num_pneumonia}")
print(f"NORMAL={num_normal}")

print("Test
set:\n=====
==")
print(f"PNEUMONIA={len(os.listdir(os.path.join(test_dir,
'opacity')))}")
print(f"NORMAL={len(os.listdir(os.path.join(test_dir,
'normal')))}")

print("Validation
set:\n=====
==")
print(f"PNEUMONIA={len(os.listdir(os.path.join(val_dir,
'opacity')))}")
print(f"NORMAL={len(os.listdir(os.path.join(val_dir,
'normal')))}")

pneumonia = os.listdir("train/opacity")
pneumonia_dir = "train/opacity"

plt.figure(figsize=(20, 10))

for i in range(9):
    plt.subplot(3, 3, i + 1)
    img = plt.imread(os.path.join(pneumonia_dir,
pneumonia[i]))
    plt.imshow(img, cmap='gray')
    plt.axis('off')

plt.tight_layout()
```

```

normal = os.listdir("train/normal")
normal_dir = "train/normal"

plt.figure(figsize=(20, 10))

for i in range(9):
    plt.subplot(3, 3, i + 1)
    img = plt.imread(os.path.join(normal_dir, normal[i]))
    plt.imshow(img, cmap='gray')
    plt.axis('off')

plt.tight_layout()

normal_img = os.listdir("train/normal")[0]
normal_dir = "train/normal"
sample_img = plt.imread(os.path.join(normal_dir,
normal_img))
plt.imshow(sample_img, cmap='gray')
plt.colorbar()
plt.title('Raw Chest X Ray Image')

print(f"The dimensions of the image are
{sample_img.shape[0]} pixels width and
{sample_img.shape[1]} pixels height, one single color
channel.")
print(f"The maximum pixel value is
{sample_img.max():.4f} and the minimum is
{sample_img.min():.4f}")
print(f"The mean value of the pixels is
{sample_img.mean():.4f} and the standard deviation is
{sample_img.std():.4f}")

sns.distplot(sample_img.ravel(),
              label=f"Pixel Mean {np.mean(sample_img):.4f} &
Standard Deviation {np.std(sample_img):.4f}", kde=False)
plt.legend(loc='upper center')
plt.title('Distribution of Pixel Intensities in the Image')
plt.xlabel('Pixel Intensity')
plt.ylabel('# Pixels in Image')


from keras.preprocessing.image import
ImageDataGenerator

image_generator = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.1,
    shear_range=0.1,
    zoom_range=0.1,
    samplewise_center=True,

```

```

        samplewise_std_normalization=True
    )

    train = image_generator.flow_from_directory(train_dir,
                                                batch_size=8,
                                                shuffle=True,
                                                class_mode='binary',
                                                target_size=(180, 180))

    validation = image_generator.flow_from_directory(val_dir,
                                                    batch_size=1,
                                                    shuffle=False,
                                                    class_mode='binary',
                                                    target_size=(180, 180))

    test = image_generator.flow_from_directory(test_dir,
                                              batch_size=1,
                                              shuffle=False,
                                              class_mode='binary',
                                              target_size=(180, 180))

    sns.set_style('white')
    generated_image, label = train.__getitem__(0)
    plt.imshow(generated_image[0], cmap='gray')
    plt.colorbar()
    plt.title('Raw Chest X Ray Image')

    print(f"The dimensions of the image are
    {generated_image.shape[1]} pixels width and
    {generated_image.shape[2]} pixels height, one single color
    channel.")
    print(f"The maximum pixel value is
    {generated_image.max():.4f} and the minimum is
    {generated_image.min():.4f}")
    print(f"The mean value of the pixels is
    {generated_image.mean():.4f} and the standard deviation is
    {generated_image.std():.4f}")

    # CNN Model
    # Class weights

    weight_for_0 = num_pneumonia / (num_normal +
    num_pneumonia)
    weight_for_1 = num_normal / (num_normal +
    num_pneumonia)

    class_weight = {0: weight_for_0, 1: weight_for_1}

    print(f"Weight for class 0: {weight_for_0:.2f}")
    print(f"Weight for class 1: {weight_for_1:.2f}")

    from keras.models import Sequential

```

```
from keras.layers import Dense, Conv2D, MaxPool2D,  
Dropout, Flatten, BatchNormalization
```

```
model = Sequential()
```

```
model.add(Conv2D(filters=32, kernel_size=(3, 3),  
input_shape=(180, 180, 3), activation='relu'))  
model.add(BatchNormalization())  
model.add(Conv2D(filters=32, kernel_size=(3, 3),  
input_shape=(180, 180, 3), activation='relu'))  
model.add(BatchNormalization())  
model.add(MaxPool2D(pool_size=(2, 2)))
```

```
model.add(Conv2D(filters=64, kernel_size=(3, 3),  
activation='relu'))  
model.add(BatchNormalization())  
model.add(Conv2D(filters=64, kernel_size=(3, 3),  
activation='relu'))  
model.add(BatchNormalization())  
model.add(MaxPool2D(pool_size=(2, 2)))
```

```
model.add(Conv2D(filters=128, kernel_size=(3, 3),  
activation='relu'))  
model.add(BatchNormalization())  
model.add(Conv2D(filters=128, kernel_size=(3, 3),  
activation='relu'))  
model.add(BatchNormalization())  
model.add(MaxPool2D(pool_size=(2, 2)))
```

```
model.add(Flatten())  
model.add(Dense(128, activation='relu'))  
model.add(Dropout(0.2))
```

```
model.add(Dense(1, activation='sigmoid'))
```

```
model.compile(loss='binary_crossentropy',  
              optimizer='adam',  
              metrics=['accuracy'])
```

```
model.summary()
```

```
r = model.fit(  
    train,  
    epochs=5,  
    validation_data=validation,  
    class_weight=class_weight,  
    steps_per_epoch=100,  
    validation_steps=25,  
)
```

```
plt.figure(figsize=(12, 8))
```

```
plt.subplot(2, 2, 1)
```

```
plt.plot(r.history['loss'], label='Loss')
plt.plot(r.history['val_loss'], label='Val_Loss')
plt.legend()
plt.title('Loss Evolution')

plt.subplot(2, 2, 2)
plt.plot(r.history['accuracy'], label='Accuracy')
plt.plot(r.history['val_accuracy'], label='Val_Accuracy')
plt.legend()
plt.title('Accuracy Evolution')

evaluation = model.evaluate(test)
print(f"Test Accuracy: {evaluation[1] * 100:.2f}%")

evaluation = model.evaluate(train)
print(f"Train Accuracy: {evaluation[1] * 100:.2f}%")

from sklearn.metrics import confusion_matrix,
classification_report

pred = model.predict(test)

print(confusion_matrix(test.classes, pred > 0.5))
pd.DataFrame(classification_report(test.classes, pred > 0.5,
output_dict=True))

print(confusion_matrix(test.classes, pred > 0.7))
pd.DataFrame(classification_report(test.classes, pred > 0.7,
output_dict=True))
```

Remaining code is present in the notebook.