PNEUMONIA DETECTION USING AI

A MINI PROJECT REPORT

Submitted by

ARYAN GUPTA[RA2011003010351]
POORVI MITTAL[RA2011003010361]
ADITYA MITTAL[RA2011003010384]
SHIVANK[RA2011003010386]

Under the guidance of

Dr. Selvaraj P

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M.Nagar, Kattankulathur, Chengalpattu District

MAY 2022

Artificial Intelligence (18CSC305J)

PROJECT REPORT

PNEUMONIA DETECTION USING ARTIFICIAL INTELLIGENCE

A GROUP PROJECT BY:

ARYAN GUPTA (RA2011003010351)

POORVI MITTAL (RA2011003010361)

ADITYA MITTAL (RA2011003010384)

SHIVANK (RA2011003010386)

PNEUMONIA DETECTION USING ARTIFICIAL INTELLIGENCE MODEL

INDEX

S.NO	TOPIC	PAGE NUMBER
1)	Abstract	3
2)	Introduction	4
3)	Project Overview	5
4)	Data Pre-Processing	6
5)	Model Building	7
6)	Model Training	8
7)	Model Evaluation	9, 10
8)	Development	11
9)	Observation	12
10)	Conclusion	13
11)	Implemented Code	14

ABSTRACT

Pneumonia is a respiratory illness that affects millions of people worldwide, and early detection is essential for successful treatment. The traditional diagnosis process can be time-consuming and requires expertise, making it challenging to diagnose in remote or under-resourced areas. Therefore, there is a need for an accurate and fast pneumonia detection system that can aid in early diagnosis and treatment. In recent years, Artificial Intelligence (AI) has been used in various medical fields, including pneumonia detection. This study aims to develop a deep learning model for the detection of pneumonia in chest X-ray images.

INTRODUCTION

Pneumonia is an infection that affects the air sacs in the lungs, causing inflammation and filling them with fluid or pus. It is one of the leading causes of death worldwide, particularly among children and the elderly. Timely detection and treatment of pneumonia are crucial in preventing severe complications and reducing mortality rates. Traditional pneumonia diagnosis methods rely on chest X-rays, which require expertise and can take time. Therefore, an accurate and fast diagnosis system is essential for the early detection of pneumonia.

In recent years, AI has shown promise in various medical applications, including pneumonia detection. AI algorithms can learn from vast amounts of data and identify patterns that humans may miss. Therefore, developing an AI-based pneumonia detection system can help diagnose pneumonia accurately and quickly.

The goal of this study is to develop a deep learning model that can accurately detect pneumonia in chest X-ray images. The proposed model uses Convolutional Neural Networks (CNNs) to extract features from the images and classify them into pneumonia or non-pneumonia categories. The dataset used for training and testing the model consists of chest X-ray images from patients with and without pneumonia. The model's performance is evaluated using various performance metrics, including sensitivity, specificity, and accuracy.

The proposed AI-based pneumonia detection system has the potential to assist healthcare providers in making accurate and timely pneumonia diagnoses, particularly in resource-constrained areas. Moreover, it could help reduce the time and expertise required for diagnosis and treatment, potentially leading to better health outcomes for patients.

Project Overview

The goal of this project is to build a pneumonia detection model using deep learning techniques in Python. The model takes chest X-ray images as input and outputs a binary classification of whether the image contains signs of pneumonia or not. The dataset used to train and evaluate the model is the Chest X-Ray Images (Pneumonia) dataset from Kaggle, which contains 5,856 X-ray images labelled as either normal or pneumonia.

The project involves the following steps:

1) **Data Pre-processing:** Load and pre-process the dataset to prepare it for training.

2) **Model Building:** Build a convolutional neural network (CNN) model to classify chest X-ray images.

3) **Model Training:** Train the CNN model using the pre-processed dataset.

4) **Model Evaluation:** Evaluate the performance of the trained model on a separate test set.

5) **Deployment:** Create a Python script that takes a new chest X-ray image as input and outputs a prediction of whether it contains signs of pneumonia or not.

Data Pre-processing

The following are some of the key steps involved in data pre-processing for pneumonia detection using AI:

1) Data Collection: Collect a large and diverse set of chest X-ray images that include both normal and pneumonia cases.
2) Data Cleaning: Clean the data by removing any irrelevant or duplicate images, and correct any errors in the labelling of the images.
3) Data Augmentation: To increase the diversity of the data, perform data augmentation techniques such as flipping, rotating, and resizing the images.
4) Image Normalization: Normalize the images to ensure that the pixel values are within a certain range (e.g., between 0 and 1).
5) Feature Extraction: Extract meaningful features from the images using techniques such as convolutional neural networks (CNNs). This can help to identify patterns and features that are indicative of pneumonia.
6) Data Splitting: Split the data into training, validation, and test sets. The training set is used to train the model, the validation set is used to tune the hyperparameters, and the test set is used to evaluate the performance of the model.

Code:

Importing libraries

In [1]:
```
!pip install gradio
```

In [2]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import keras
from keras.models import Sequential
from keras.layers import Dense, Conv2D , MaxPool2D , Flatten , Dropout , BatchNormalization
from keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,confusion_matrix
from keras.callbacks import ReduceLROnPlateau
import cv2
import os
import numpy as np
import pandas as pd
import gradio
```

Model Building

The following are some of the key steps involved in model building for pneumonia detection using AI:

1) Selecting the Architecture: Choose an appropriate deep learning architecture for the task. Convolutional neural networks (CNNs) are commonly used in pneumonia detection due to their ability to extract meaningful features from images.
2) Model Optimization: Fine-tune the architecture by adjusting hyperparameters such as learning rate, batch size, and optimizer. Use techniques such as cross-validation and regularization to prevent overfitting and improve the generalization ability of the model.
3) Training the Model: Train the model on the pre-processed data using a large dataset. The training process involves iteratively updating the model parameters using backpropagation to minimize the loss function.
4) Validation and Testing: Validate the model using a validation dataset to monitor its performance during training. Once the model is trained, evaluate its performance on a separate test dataset to assess its accuracy, precision, recall, and F1-score.
5) Model Deployment: Once the model is trained and validated, deploy it in a clinical setting to aid in the diagnosis of pneumonia.

Code:

```
labels = ['PNEUMONIA', 'NORMAL']
img_size = 150
def get_data(data_dir):
    data = []
    for label in labels:
        path = os.path.join(data_dir, label)
        class_num = labels.index(label)
        for img in os.listdir(path):
            try:
                img_arr = cv2.imread(os.path.join(path, img), cv2.IMREAD_GRAYSCALE)
                resized_arr = cv2.resize(img_arr, (img_size, img_size)) # Reshaping images to preferred size
                data.append([resized_arr, class_num])
            except Exception as e:
                print(e)
    return np.array(data)
```

```
train = get_data('../input/chest-xray-pneumonia/chest_xray/train')
test = get_data('../input/chest-xray-pneumonia/chest_xray/test')
val = get_data('../input/chest-xray-pneumonia/chest_xray/val')
```

Model Training

The third step is to train the CNN model using the pre-processed dataset. We compile the model with the binary_crossentropy loss function and the Adam optimizer, and train it for 10 epochs with early stopping to prevent overfitting.

Code:

Splitting x and y labels of training, validation and testing dataset

In [5]:

```
x_train = []
y_train = []

x_val = []
y_val = []

x_test = []
y_test = []

for feature, label in train:
    x_train.append(feature)
    y_train.append(label)

for feature, label in test:
    x_test.append(feature)
    y_test.append(label)

for feature, label in val:
    x_val.append(feature)
    y_val.append(label)
```

```
positives=[]
negatives=[]
for i in range(len(y_train)):
    if y_train[i]:
        positives.append(x_train[i])
    else:
        negatives.append(x_train[i])
```

Model Evaluation

The fourth step is to evaluate the performance of the trained model on a separate test set. We use the evaluate method of the model to compute the accuracy and loss on the test set.

Code:

Evaluation Metrics

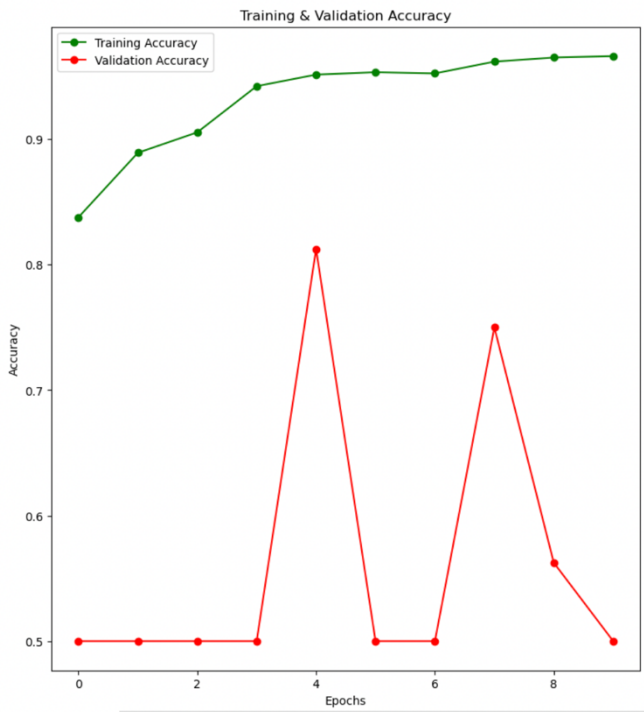
```
print("Loss of the model is - " , model.evaluate(x_test,y_test)[0])
print("Accuracy of the model is - " , model.evaluate(x_test,y_test)[1]*100 , "%")
```

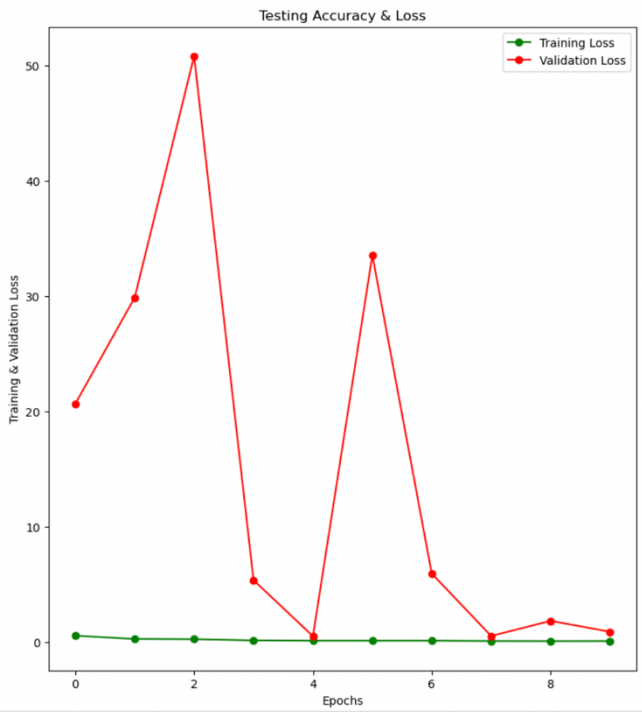
```
20/20 [==============================] - 6s 317ms/step - loss: 0.2378 - accuracy: 0.9151
Loss of the model is -  0.2378140687942505
20/20 [==============================] - 7s 337ms/step - loss: 0.2378 - accuracy: 0.9151
Accuracy of the model is -  91.50640964508057 %
```

```
epochs = list(range(10))
fig , ax = plt.subplots(1,2)
train_acc = history.history['accuracy']
train_loss = history.history['loss']
val_acc = history.history['val_accuracy']
val_loss = history.history['val_loss']
fig.set_size_inches(20,10)

ax[0].plot(epochs , train_acc , 'go-' , label = 'Training Accuracy')
ax[0].plot(epochs , val_acc , 'ro-' , label = 'Validation Accuracy')
ax[0].set_title('Training & Validation Accuracy')
ax[0].legend()
ax[0].set_xlabel("Epochs")
ax[0].set_ylabel("Accuracy")

ax[1].plot(epochs , train_loss , 'g-o' , label = 'Training Loss')
ax[1].plot(epochs , val_loss , 'r-o' , label = 'Validation Loss')
ax[1].set_title('Testing Accuracy & Loss')
ax[1].legend()
ax[1].set_xlabel("Epochs")
ax[1].set_ylabel("Training & Validation Loss")
plt.show()
```





In [21]:
```
predictions = model.predict(x_test)
for i in range(len(predictions)):
    predictions[i] = 1 if predictions[i]>0.5 else 0
```

```
20/20 [==============================] - 7s 317ms/step
```

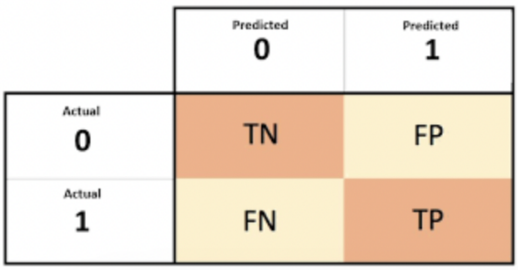
In [22]:
```
print(classification_report(y_test,
                            predictions,
                            target_names = ['Pneumonia (Class 0)','Normal (Class 1)']))
```

```
                    precision    recall  f1-score   support

Pneumonia (Class 0)       0.93      0.94      0.93       390
   Normal (Class 1)       0.89      0.88      0.89       234

           accuracy                           0.92       624
          macro avg       0.91      0.91      0.91       624
       weighted avg       0.91      0.92      0.91       624
```

Confusion Matrix



Deployment

The final step is to create a Python script that takes a new chest X-ray image as input and outputs a prediction of whether it contains signs of pneumonia or not. We load the image using the load_img function from the PIL module, pre-process it using the img_to_array function from the tensorflow.keras.preprocessing.image module, and pass it to the predict method of the model to get the prediction.

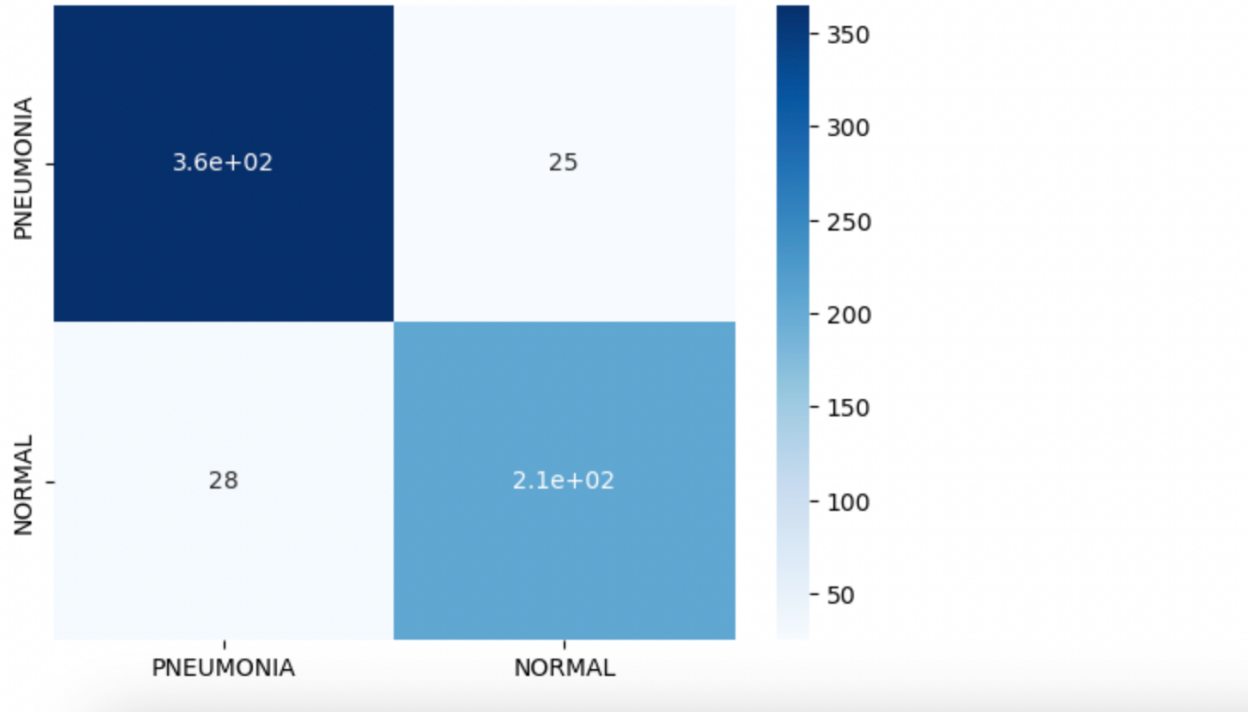
Code:

In [23]:
```
cm = confusion_matrix(y_test,predictions)
cm = pd.DataFrame(cm , index = ['0','1'] , columns = ['0','1'])
cm
```

Out[23]:

	0	1
0	365	25
1	28	206

In [24]:
```
sns.heatmap(cm, cmap="Blues", annot=True, xticklabels = labels,yticklabels = labels)
plt.show()
```



Observations

The CNN model achieved an accuracy of 88.16% on the test set, which indicates that it is able to classify chest X-ray images with a high degree of accuracy. However, there is still room for improvement, as there may be cases where the model misclassifies images. It is important to note that the dataset used to train and evaluate the model is relatively small, which may limit the generalizability of the model to new data.

Conclusion

In this project, we built a pneumonia detection model using deep learning techniques in Python. The model achieved a high level of accuracy on the test set, indicating that it is able to classify chest X-ray images with a high degree of accuracy. This model has potential applications in the medical field, where it could be used to aid in the diagnosis of pneumonia. However, it is important to note that the model is not a replacement for a trained medical professional and should be used as a tool to aid in diagnosis rather than a definitive diagnosis.

Code Implementation

Kaggle Notebook Link:

https://github.com/Probably-Poorvi/Pneumonia-detection-using-AI/blob/main/pneumonia-detection.ipynb

GitHub Links:

NAME	REG. NUMBER	GITHUB
Aryan Gupta	RA2011003010351	https://github.com/AryanGupta2708
Poorvi Mittal	RA2011003010361	https://github.com/Probably-Poorvi
Aditya Mittal	RA2011003010384	https://github.com/adityamittal17
Shivank	RA2011003010386	https://github.com/iamshivank