A Project report on

KIDNEY STONE DETECTION USING DEEP LEARNING

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY
In
COMPUTER SCIENCE AND ENGINEERING

Submitted by

Y HARIKA	**(18B91B0559)**
B LOKESH SAI VARMA	**(18B91B0506)**
V SAI MANOHAR	**(19B95B0506)**
S MOHAMMED YASEEN	**(18B91B0545)**

Under the Guidance of
Sri. S SURYA NARAYANA RAJU
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SRKR ENGINEERING COLLEGE (A)

Chinna Amiram, Bhimavaram, West Godavari Dist., A.P.

[2021 – 2022]

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SRKR ENGINEERING COLLEGE (A)

ChinnaAmiram, Bhimavaram, West Godavari Dist., A.P.

[2021 – 2022]

BONAFIDE CERTIFICATE

This is to certify that the project work entitled **"KIDNEY STONE DETECTION USING DEEP LEARNING"** is the bonafide work of "**Y. HARIKA (18B91B0559), B. LOKESH SAI VARMA (18B91B0506), V SAI MANOHAR (19B95B0506), S MOHAMMED YASEEN (18B91B0545)**" who carried out the project work under my supervision in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering.

SUPERVISOR
Sri. S SURYA NARAYANA RAJU
Assistant Professor

HEAD OF THE DEPARTMENT
Dr. V Chandra Sekhar
Professor

SELF DECLARATION

We hereby declare that the project work entitled "**KIDNEY STONE DETECTION USING DEEP LEARNING**" is a genuine work carried out by us in B.Tech., (Computer Science and Engineering) at SRKR Engineering College(A), Bhimavaram and has not been submitted either in part or full for the award of any other degree or diploma in any other institute or University.

1. Y. HARIKA	**18B91B0559**
2. B. LOKESH SAI VARMA	**18B91B0506**
3. V. SAI MANOHAR	**19B95B0506**
4. S. MOHAMMED YASEEN	**18B91B0545**

TABLE OF CONTENTS

ABSTRACT	i
LIST OF FIGURES	ii
1. INTRODUCTION	1
2. LITERATURE SURVEY	3
3. PROBLEM STATEMENT	6
4. METHODOLOGY	7
5. IMPLEMENTATION	13
6. RESULT ANALYSIS	16
7. CONCLUSION	20
REFERENCES	21
APPENDIX I: SAMPLE CODE	23

ABSTRACT

Kidney Stones have been a big problem in recent years, and if not found early, they can cause difficulties, necessitating surgery to remove the stone. Volumetric measurements of kidney stones are more useful and repeatable than linear measurements, according to previous research. Deep Learning-based algorithms that use non-contrast abdominal computed tomography (CT) scans may help detect stones and reduce the workload associated with manual detection. A dataset of CT scans is used to identify the stone, which includes CT scans with and without manually indicated kidney stones. To identify kidney stones, we used CNN and Random Search on the dataset.

LIST OF FIGURES

1. Median Filter Example 8
2. CNN Model Architecture 9
3. Pooling 10
4. Relu Activation Function 11
5. Sigmoid Activation Function 11
6. Median Blur 13
7. Power Law Transformation 14
8. Thresholding Method 14
9. Random Search Model 15
10. Confusion Matrix 16
11. Output1 (Kidney Stone Detected) 18
12. Output2 (Normal) 18
13. Output3 (Random Image from Web) 19
14. Accuracy Comparison 19

CHAPTER 1

INTRODUCTION

The prevalence of kidney stone illness is rising these days. Renal calculus, often known as a kidney stone, is a solid mass that forms in the kidneys. Kidney stones can affect anyone, including children, and the majority of cases go unnoticed unless there is severe abdominal pain or an odd urine color. Fever, discomfort, and nausea are some of the symptoms that persons with this issue may experience. Infection and renal failure can make an obstructing stone even worse. Small ureteral stones may usually pass on their own, but larger stones may require interventional therapy such extracorporeal shock wave lithotripsy or endoscopic lithotripsy. The early stages of many kidney stone disorders are not observed until later or are difficult to detect, causing harm to the kidney as they get larger. Diabetes, hypertension, and glomerulonephritis are the leading causes of kidney failure, with millions of people affected each year. Early stage detection is recommended and can save many lives because renal malfunction can lead to major complications and even death. Based on the location, kidney stones are classified as kidney (nephrolithiasis), ureter (ureterolithiasis) and bladder (cystolithiasis).

Imaging is becoming an important part of biological and clinical research. Medical imaging is a method of creating a visual picture of the inner organs by a clinician. They're then employed for clinical research and medical intervention. Ultrasound (US) images, Noncontrast Computed Tomography (NCCT or CT-Scan), Magnetic Resonance Imaging (MRI), and X-ray are now available options. For the diagnosis of acute flank pain, NCCT is widely used. So, detecting the stone and doing so precisely opens the door to image processing, because image processing has the potential to produce precise findings without the need for human participation.

To detect the stone from a Computed Tomography image, radiologists typically employ a manual procedure.These advantages have resulted in an increase in the use of CT for suspected urolithiasis, but they have also contributed to an increase in imaging volume, longer turnaround times, higher radiologists' workload, and longer hospital admissions. Software programming, which has discovered current and potential uses in medical technological improvements, recognises the need to contribute to CT screening progress, notably in improving kidney stone

detection in the kidney-urine-belly (KUB) region. This work used KUB CT scans to construct a semi-automatic kidney screening tool that included digital image processing and image analysis approaches.

In a variety of disciplines, deep learning algorithms have been successfully applied to medical images and physiological signals. Deep models have been successfully used in a wide range of applications, including image segmentation, classification, and detection in medicine. DL models have been built using a variety of medical imaging modalities to assist physicians in diagnosing illnesses such as Covid19, cardiac arrhythmia, and brain tumour. In the field of urology, DL techniques are used to automatically detect ureteral and kidney stones.

The CT scan information is a grayscale 3D image in which the value of each pixel is directly tied to the sort of substance that occupies that location. The value of the pixels occupied by a kidney stone can occupy a specific range since kidney stones are formed of a certain collection of chemicals. However, different types of components in the human body are constructed of this specific mix of materials. The concentrations of bones and other materials had pixels in the same range as kidney stones.

CHAPTER 2

LITERATURE SURVEY

Stalina S et al. [1], have performed Image processing techniques on CT Scan images. The author stated that image processing is performed using filtering and image enhancement. Filtering is done to smoothen the image. There are various filters such as Average filter, Weighted Average filter. Guassian filter, Median filter. In this paper the author has applied Median filter because it is the best method to remove the impulse noise or salt and pepper noise. Image Enhancement technique is used to modify the intensities of the image. CT Scan images are of low quality and hence Image Enhancement should be done. The author has performed Histogram Equilization which modifies the pixel intensity. For Image Segmentation process, the author used Thresholding technique for the partition of image into different regions to extract the desired features.

Kadir Yildirim et al. [2], have gathered 500 NCCT pictures from patients admitted to Elâz Fethi Sekin City Hospital in Turkey for urinary stone illness. The experts completed the labelling procedure by indicating whether or not there are stones. To avoid the overfitting problem, data augmentation techniques were applied to the raw photos. For the kidney stone detection training, XResNet-50 model was used. Fastai (v2) library created on Pytorch deep learning framework was used to train the XResnet-50 model. Adam Optimizer and Cross-Entropy Loss Functions were used to alter the parameters of the XResnet-50 model. The model can display the locations where the DL model focused for classification. According to this article, the tip of the lower pole of the kidney entering the cross-sectional area may have caused the model to produce incorrect results.

Anushri Parakh et al. [3], have evaluated the performance of pretrained models enriched with labeled Unenhanced abdominopelvic CT images across different scanners. All the images were first processed by several Image Processing techniques and then normalized such that all scans were oriented in upward position. Then the images were converted into grayscale. The images were then used for developing a cascading model consisting of two CNNs. The first (CNN-1) model consists of pre-trained models such as ImageNet and GrayNet. During the training process

the images were first fed to CNN-1 which helps in identifying the urinary tract and were then presented to CNN-2 for classification into presence or absence of stone.

Martin Längkvist et al. [4], have used Hyperparameter tuning to create a model. There are 465 clinically acquired unenhanced abdomen CT scans in the collection. Gaussian filter and thresholding methods were used to pre-process the scanned images. Three training strategies were described in this paper: 2D, 2.5D, and 3D input data. n = [3,6,11] and p = [1,2,5] are the filter size and Max Pooling dimension, respectively. For each of the training schemes, the number of filters k is set among k = [20,50]. Supervised backpropagation with mini-batch stochastic gradient descent is used for the training.

Nagireddi Amrutha Lakshmi et al. [5], have proposed a model for kidney stone detecting from ultrasound images. Ultrasound images are prone to noise because of its low contrast. Image pre-processing consists of smoothening, sharpening and enhancement. Gaussian Filtering was used during preprocessing phase which is used to smoothen the image. Canny Edge detection was used to extract the useful information from the image. By using the Conventional Neural Networks, the model has obtained an accuracy between 70-80%.

Jyoti Verma et al. [6], have implemented the analysis and identification of kidney stone using K^{th} Nearest Neighbor (KNN) and Support Vector Machine (SVM). In this paper, they have pre-processed image using Median filter, Gaussian filter. Entropy based Segmentation is used to find region of interest. After the Image Processing, Principal Component Analysis is used for feature extraction and reduction. PCA helps in reducing the dimensions in the dataset. They have proposed two types of classification techniques and have been applied on the different sets of kidney stone images. On the basis of this, they found that KNN worked better with an accuracy score of 89% whereas SVM has acquired an accuracy score of 84%.

Daniel C. Elton et al. [7], have used a dataset of images containing 180 images with manually marked with class labels. The dataset is divided for training and testing. CT scan images of 6185 patients from another hospital is used for validation test. U-Net model was used to segement kidneys, followed by gradient-based anisotropic denoising, thresholding,and region growing. A 13 layer CNN model was used to classify the kidney stones from normal.

Prema T. Akkasaligar et al. [8], have proposed a model for measuring the size and area of the kidney stone. CT scan images are consisting of kidney regions at the center part usually, so the images are cropped. Fuzzy C-means Clustering is used for clustering because it allows single slice of data belong to many clusters. The authors have used Level set segmentation for image segmentation and identification of CT kidney stone process depends on the segmentation results. The authors have considered region parameter extraction for renal calculi images of kidney. Region parameters used are centroid, area, etc.

Mehmet Baygin et al. [9], have created an automated system to aid clinicians in precisely detecting kidney stones. In this paper, a transfer learning algorithm (ExDark19) for detecting kidney stones is proposed. The most informative features were extracted using iterative neighbourhood component analysis (INCA), and these features were then fed into KNN to detect kidney stones using a k-fold cross validation approach. The proposed ExDark19 model has achieved good results with the hold out validation method.

A.Nithya et al. [10], have designed an approach for kidney stone detection and segmentation using a combination of clustering and classification approach. The author has used both Artificial neural networks for kidney stone detection and multi-kernel-k-means clustering algorithm for image segmentation. Initially, median filter was used to eliminate the impulse noise present in the image to extract important features. Then the image was fed to neural network model to classify the image as normal and abnormal. Finally, the abnormal images are fed to the segmentation stage to segment the stone and tumor part seperately using multi-kernel-k-means clustering algorithm.

CHAPTER 3

PROBLEM STATEMENT

Machine learning is a subset of artificial intelligence (AI) that centres around the concept of a computer program learning and adapting to new data without requiring human intervention. Kidney stones are mineral and salt deposits that occur inside the kidneys. Diet, obesity, certain medical conditions, and certain supplements and medicines can all contribute to kidney stone formation. If the stones get contaminated, the kidneys may become diseased as well. The person who is impacted by this disease will have stomach pain as well as urinary bladder issues such as urine delay and urinary bladder pain. To address all of these issues, we developed the Convolutional neural network method, which uses machine learning to detect stones early and suggest the best treatment. The majority of procedures used in the past will detect kidney stones when they are severely impacted. This algorithm is far more precise than previous algorithms. This method will produce the best photos and a good view of the stones, assisting in the early detection of the stones.

CHAPTER 4

METHODOLOGY

The Major Steps involved in the Detection of kidney stone using Deep Neural Networks are as follows:

1. Gathering data
2. Data Pre-Processing
3. Image Processing
4. Choosing the Deep Learning Model

1. GATHERING DATA:

The process of gathering the dataset depends upon the type of problem we are trying to solve. As this project is mainly focused on Image Classification, we need to acquire the required resources from open-source websites such as Kaggle, Github etc.

The dataset was uploaded to the Github repository and available in the following link: https://github.com/yildirimozal/Kidney_stone_detection/tree/main/Dataset

2. DATA PRE-PROCESSING:

Data Preprocessing is a crucial step when you're dealing with Image Datasets. As the images are of variable size, we need to convert them into fixed size. The dataset contains two folders "Kidney_Stone" and "Normal". As the images doesn't contain any label, we use these folder names to classify the images for training the Deep Learning Model. We resize all the images into 128 x 128 pixels. We then convert the image into Numpy array with the class label. We split the data into 90% for the training and 10% for testing the model.

3. IMAGE PROCESSING:

Image Processing is divided into 2 modules:

3.1. Image Pre-processing:

It is basically one of the critical tasks because CT Scan images may have noise. In this operation we apply methods to enhance and filter the image using the Median filter and histogram equalization or Power Law Transformation.

3.1.1. Median Filter:

Python OpenCV provides the cv2.medianBlur() function to blur the image with a median kernel. This is a non-linear filtering technique. It is highly effective in removing impulse noise and salt-and-pepper noise. This takes a median of all the pixels under the kernel area and replaces the central component with this median value. Fig 4.1 illustrates the median filter example.

Syntax: cv2.medianBlur(image,ksize)

Fig: 4.1: Median Filter Example

3.1.2. Histogram Equalization:

Histogram Equalization technique is used for modifying the intensities of the image. The image we get is of lower quality therefore the image enhancing is done to improve the quality of the image. In this, each pixel intensity is modified so if the image is towards the darker side, then it gets stretched towards more white side and hence, we can say that the image is enhanced.

3.1.3. Power Law Transformation:

This method is better option for image enhancement. Here, the value of constant should be assumed on the basis of trial-and-error method. Gamma correction is useful when you want to change the contrast and brightness of an image.

3.2. Image Segmentation:

Image Segmentation means to partition the image into different regions to extract relevant information. Segmentation is a vital aspect of medical imaging. It aids in the visualization of medical data and diagnostics of various diseases. Thresholding Method is applied on the image resulting from gamma adjustment to allow segmentation of image the foreground (stone and bones) and background. Thresholding value is based on the intensity of the pixel and intensities below this level becomes zero.

4. APPLIED MODELS:

4.1. Convolutional Neural Networks (CNN): Building a model to process the data collected is the most important step in the entire project. The algorithm we are using is Convolutional Neural Networks (CNN). CNNs are a class of Deep Neural Networks inspired by the visual cortex of human brain. A Convolutional Neural Network, or CNN, is a deep learning neural network designed for processing structured arrays of data such as images. CNNs are widely used in Computer Vision and have become the state of the art for many visual applications such as Image Classification and have also found success in Natural Language Processing for Text Classification. The Basic architecture of CNN model is shown in below figure 4.2.

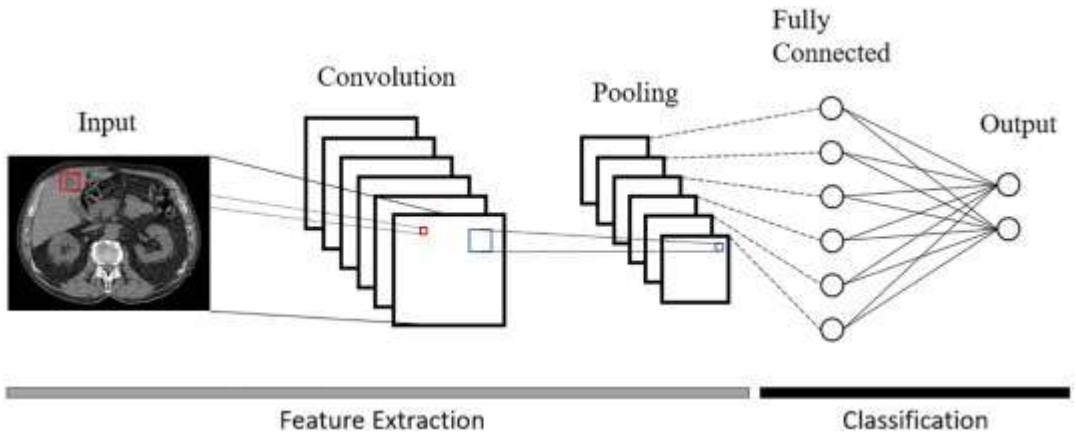

Fig. 4.2 CNN Model Architecture Example

- Here in our project the architecture we are using is 2-D CNN layer along with RELU (Rectified Linear Unit) activation function. The input frame to this layer is of dimensions 128x128

- The above figure is just a proposed architecture. Actual parameters and layers are yet to be tested using hyperparameter tuning.
- In the above proposed architecture, we are also performing pooling to decrease the size of the feature maps, so that the number of parameters that a model must learn is decreased.
- Pooling is used in reduction of the size of an image without losing patterns or details, because the oversized image can become a complex task for the machine to handle. There are two types of pooling can be performed:

  Max Pooling – This identifies only important features of the previous feature map.

  Average Pooling – This computes the average of the elements presents in the region of the feature map covered by filter. The difference between the 2 types is illustrated in fig 4.3.

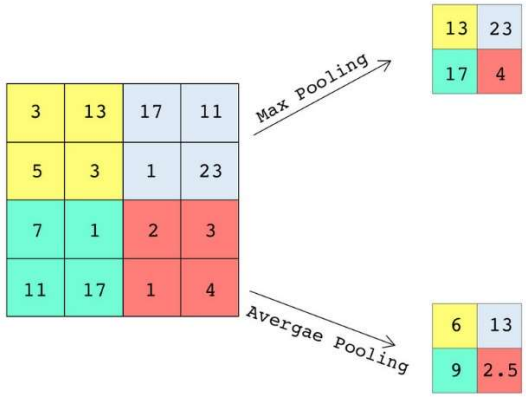

Fig. 4.3 Pooling

- We are going to use two dense layers with RELU and Sigmoid activation functions.

  ReLU Activation Function: The rectified linear activation function, or ReLU, is a linear function that, if the input is positive, outputs the input directly; else, it outputs zero. Because a model that uses this is quicker to train and generally achieves higher performance, it has become the default activation function for many types of neural networks. Fig 4.4 illustrates the ReLU Activation function.

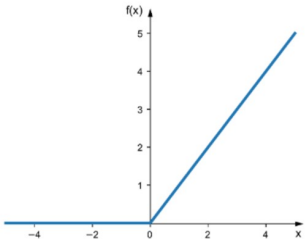

Fig. 4.4 ReLU Activation Function

Sigmoid Activation Function: This function takes any real value as input and outputs values in the range of 0 to 1. A weighted sum of inputs is passed through an activation function and this output serves as an input to the next layer. When the activation function for a neuron is a sigmoid function, it is a guarantee that the output of this unit will always be between 0 and 1. The larger the input (more positive), the closer the output value will be to 1, whereas the smaller the input (more negative), the closer the output will be to 0, as shown below. We use this Activation function in output layer. Fig 4.5 represents the Sigmoid activation function.

$$S(x) = \frac{1}{1+e^{-x}}$$

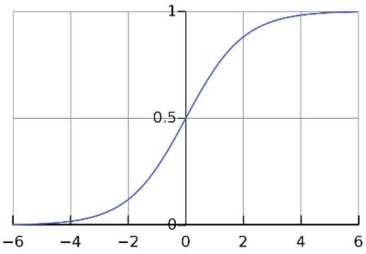

Fig. 4.5 Sigmoid Activation Function

- Here, we are using the kernel size of 3x3 in the proposed architecture.

Convolution: Convolution is a process of identifying certain features in a feature map. Convolution helps the model or machine learn some important qualities of a picture through edge detection, noise reduction, blurring, sharpening and more.

Flattening: A two-dimensional matrix of features is flattened into vector of features that can be input to the Dense layer.

4.2. Hyperparameter Tuning: Hyperparameter Tuning is the process of modifying the model architecture to fit the available space. This is nothing more than looking for the suitable hyperparameter to achieve great precision and accuracy. There are various parameter tuning methods, the most popular ones are:

- Grid Search
- Random Search

Grid Search: Grid search is a method for determining the optimal collection of hyperparameters for a given model. Hyperparameters are not model parameters, and finding the optimal set from training data is impossible. When we use something like Adam, RMSprop etc. to optimize a loss function, we learn model parameters during training. We just generate a model for each combination of hyperparameters and test each model in this tuning process.

Random Search: Random search is a method for finding the optimum solution for the model by using random combinations of hyper-parameters. It's similar to grid search, but it is better in terms of performance. The only limitation of random search is that it generates a lot of variance during computation. Because the parameters are chosen at random and no intelligence is applied to sample these combinations, we cannot guarantee for optimal parameters.

CHAPTER 5

IMPLEMENTATION

This project consists of the following modules:

5.1. Read Images using OpenCV: We use OpenCV to read the images from the Hard Disk. Images Dataset is collected from GitHub repository. It contains around 1700 images in two different folders with class labels.

5.2. Data Splitting: After reading them, we Split the dataset into Train and Test using train_test_split method from sklearn.model_selection module.

5.3. Image Processing: We used Median Blur, Power Law Transformation and Thresholding methods in the Image processing stage. Image processing is used to remove the noise.

5.3.1. Median Blur: Python OpenCV provides the cv2.medianBlur() function to blur the image with a median kernel. Fig 5.1 represents the median filter.

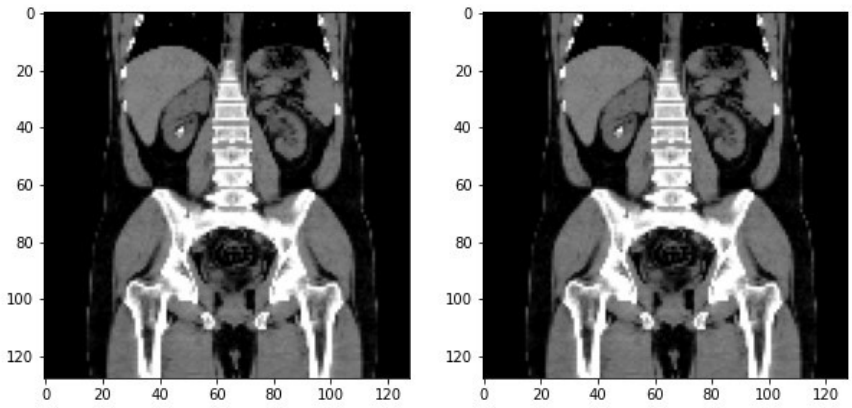

Fig. 5.1 Median Blur

5.3.2. Power Law Transformation: This method is better option for image enhancement. Fig 5.2 represents the power law transformation.

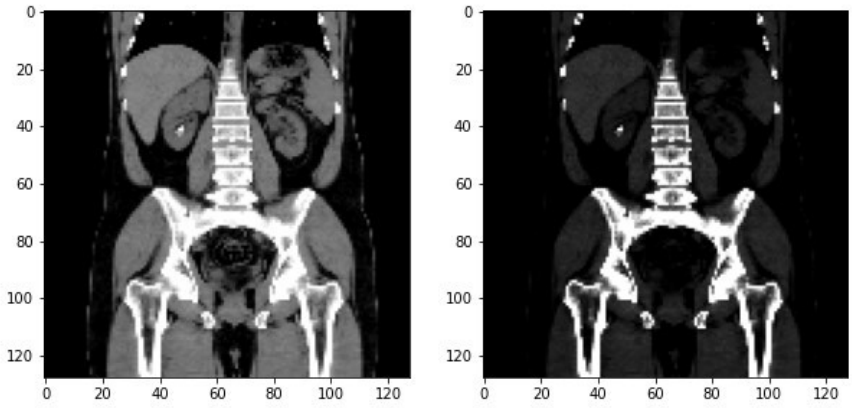

Fig. 5.2 Power Law Transformation

5.3.3. Thresholding Method: Thresholding Method is applied on the image resulting from gamma adjustment to allow segmentation of image the foreground (stone and bones) and background. Fig 5.3 represents the thresholding method.

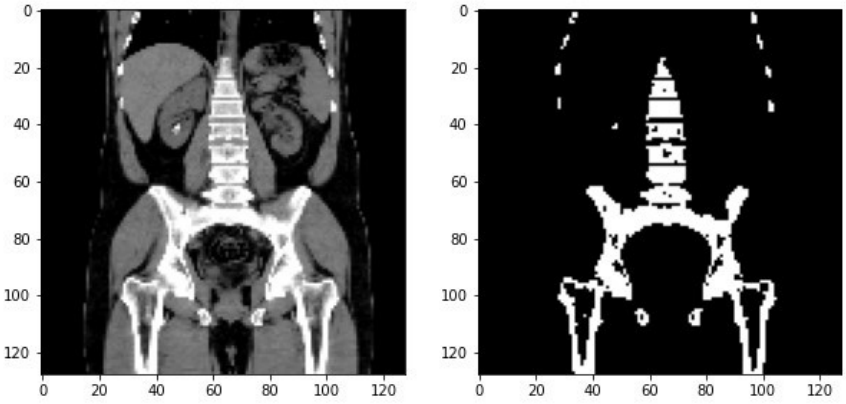

Fig. 5.3 Thresholding Method

5.4. Building a Model: Convolutional Neural Networks (CNN) model is used for training a model. We use Random Search method to choose the model parameters. The following figure **Fig. 5.4** shows the model summary after performing Random Search Algorithm. We are running 20 epochs to fit the model with our training data.

```
In [355]: model.summary()
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 126, 126, 64)      640

 max_pooling2d (MaxPooling2D  (None, 42, 42, 64)       0
 )

 conv2d_1 (Conv2D)           (None, 38, 38, 112)       179312

 max_pooling2d_1 (MaxPooling  (None, 12, 12, 112)      0
 2D)

 flatten (Flatten)           (None, 16128)             0

 dense (Dense)               (None, 80)                1290320

 dropout (Dropout)           (None, 80)                0

 dense_1 (Dense)             (None, 1)                 81

=================================================================
Total params: 1,470,353
Trainable params: 1,470,353
Non-trainable params: 0
_________________________________________________________________
```

Fig. 5.4 Random Search Model Summary

5.5. Saving the Model: Random Search algorithm randomly chooses different parameters every time we run the code. After Identifying the best parameters using Random Search, we need to save the model using keras.models module.

Syntax: model.save(‘/path’)

Best parameters will be saved in file and can be used for further testing. We don’t need to train every time in order to test the data. We can just load the model and use predict method to get the desired output.

Syntax: keras.models.load_model(‘/path’)

CHAPTER 6

RESULT ANALYSIS

Metrics helps in analyzing, measuring the performance quality of machine learning models in different areas such as efficiency and error proneness by using Accuracy, Precision, Recall, F1 score and specificity values.

Firstly, before learning about these metrics we classify the results into four different labels:

True Positive (TP): Predicted value is True and True is actual value.

True Negative (TN): Predicted value is False and False is actual value.

False Positive (FP): Predicted value is True but False is actual value.

False Negative (FN): Predicted value is False but True is actual value.

6.1. Confusion Matrix: Confusion matrix is a NxN matrix which is used to describe the performance of the model while solving classification problems. It is used for both binary classification and multiclass classification. Fig 6.1 shows the basic illustration of confusion matrix.

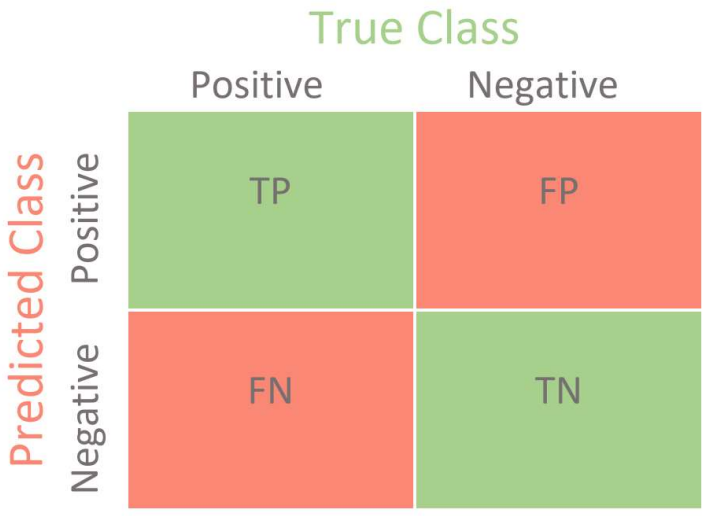

Fig. 6.1 Confusion Matrix

In our project, after splitting the dataset into train and test, we performed testing on 161 unseen images and the results are shown in the below table

Predicted \ Actual	Positive	Negative
Positive	83	1
Negative	2	75

True Positive (TP) : 83
True Negative (TN) : 75
False Positive (FP) : 1
False Negative (FN) : 2

6.2. Accuracy Value: Accuracy is a common evaluation metric for classification problems. It's the number of correct predictions made as a ratio of all predictions made. The percentage of correct predictions for the test data is known as accuracy.

$$Accuracy = \frac{(TP + TN)}{TP + FP + FN + TN}$$

Accuracy value for our project after testing it on unseen data is as follows:

$$Accuracy = \frac{(83+7\ )}{(83 + 1 + 2 + 75)} = 0.981$$

6.3. Testing on Unseen Data: We separated 18 images from the original dataset before feeding them to the model for training. These images will be used for testing. Fig 6.2 and Fig 6.3 shows the outputs we got when we tested them on unseen data.

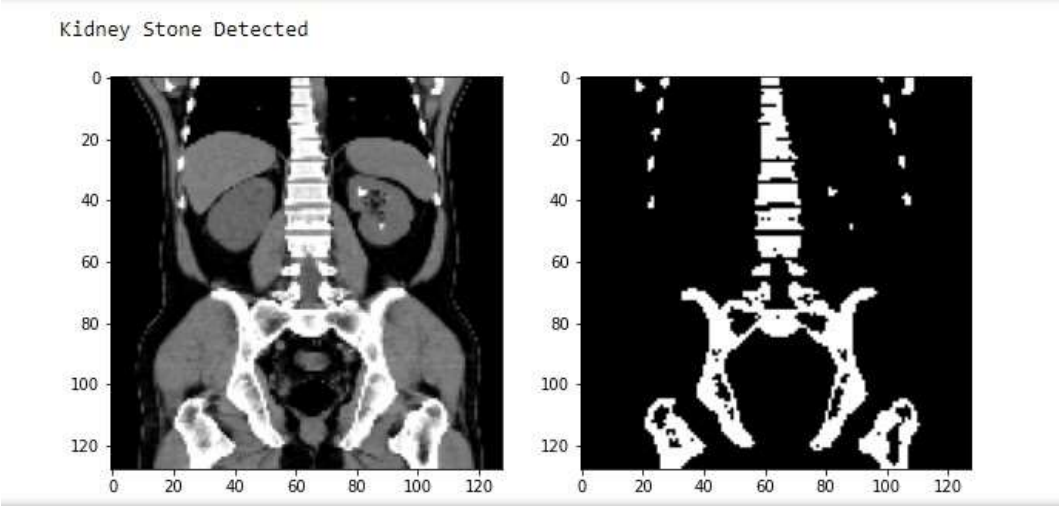

Fig. 6.2 Output 1 (Kidney Stone Detected)

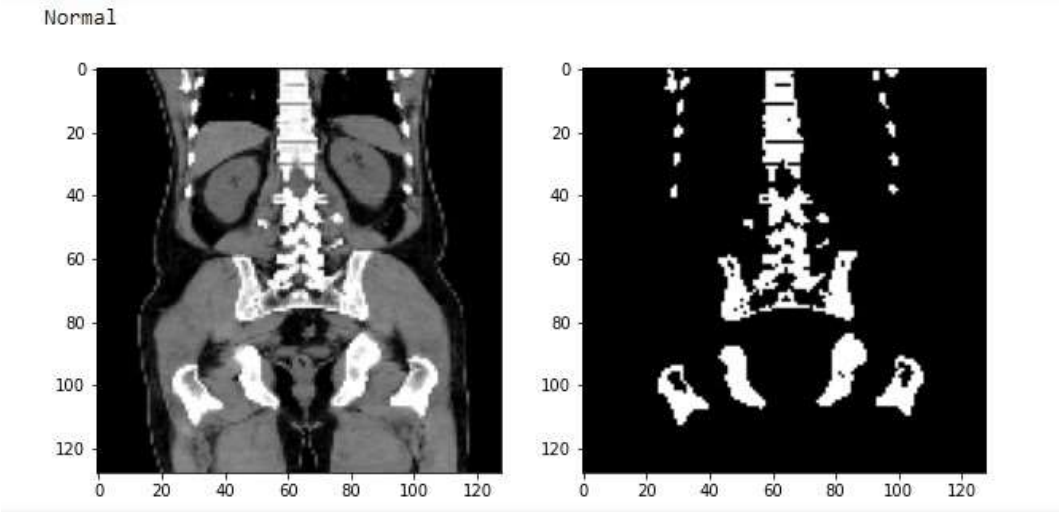

Fig. 6.3 Output 2 (Normal)

6.4. Testing on Radom Image from Internet: We tested the model using the image found in the web and it showed the accurate results. Fig 6.4 shows the output (Abnormal) when we fed random image from web to the model.

URL for the image:

https://www.renalandurologynews.com/wp-content/uploads/sites/22/2019/01/6cmlowerpolerrenalmassqu_1088120.png

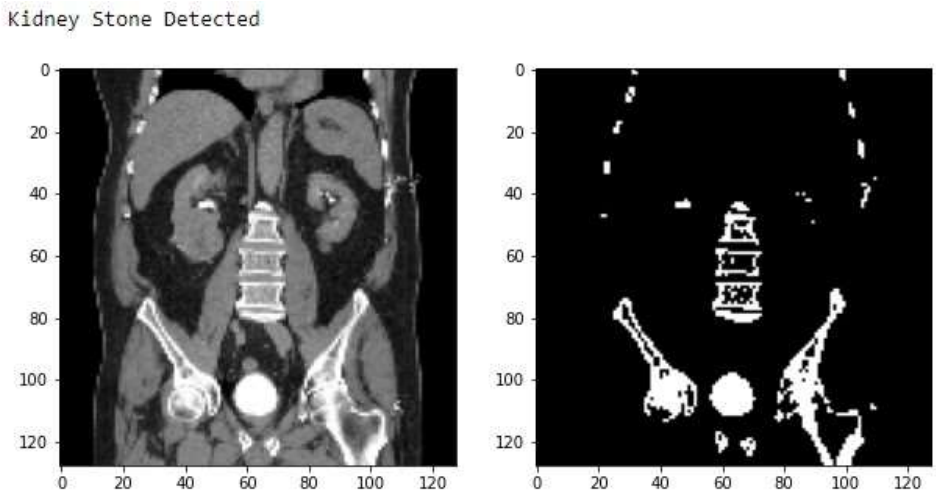

Fig. 6.4 Output 3 (Random Image from Web)

6.5. Accuracy Comparison: The XResNet50 has acquired an accuracy around 96 percent and the CNN has achieved an accuracy around 98 percent. The CNN is trained with the images after applying the image processing techniques on them. Random Search was used to find the model parameters. The below Fig. 6.5 shows the comparison between accuracy of the existing and proposed models.

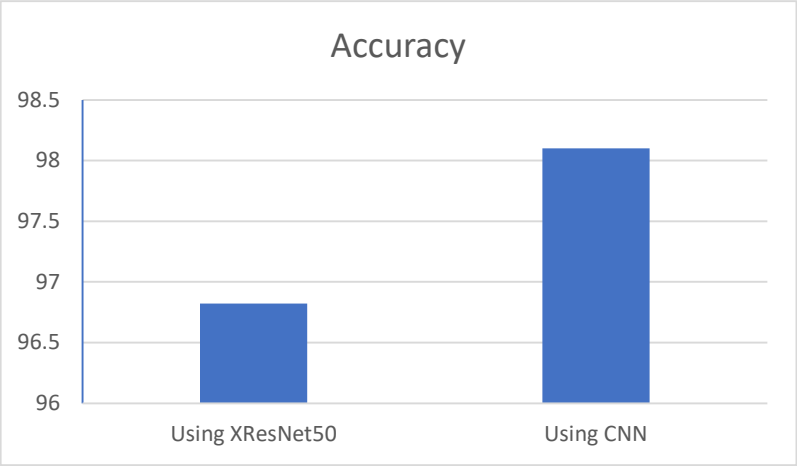

Fig. 6.5 Accuracy Comparison

CHAPTER 7

CONCLUSION

An efficient machine learning based detection system has been developed for identification of Kidney Stones. Convolutional Neural Network is used in designing of the system. Thus, another innovative touch of our project is that with help of image processing important features are extracted for classification thus it helped to reduce the processing time of the detection system. The model has acquired an accuracy of around 90% for test data. Thus, with the help of our project the detection of Kidney Stone becomes easy.

References:

[1] Stalina S, Aditi S, Anuja R, Prof. Pooja L Gohel, “Kidney Stone Detection Using Image Processing on CT Images”, ISSN NO: 2249-7455

[2] Kadir Yildirim, Pinar Gundogan Bozdag, Muhammed Talo, Ozal Yildirim, Murat Karabatak, U.Rajendra Acharya, “Deep learning model for automated kidney stone detection using coronal CT images” , 2021.

[3] Parakh, A., Lee, H., Lee, J. H., Eisner, B. H., Sahani, D. V., & Do, S. (2019). Urinary stone detection on CT images using deep convolutional neural networks: Evaluation of Model Performance and generalization. *Radiology: Artificial Intelligence*, *1*(4).

[4] Längkvist, Martin & Jendeberg, Johan & Thunberg, Per & Loutfi, Amy & Lidén, Mats. (2018). Computer aided detection of ureteral stones in thin slice computed tomography volumes using Convolutional Neural Networks. Computers in Biology and Medicine.

[5] Nagireddi Amrutha Lakshmi, Bodasakurti Jyithirmayi, Manjeti Sushma, Attaluri Sri Manoj, G. Santoshi, “Kidney Stone Detection from Ultrasound Images by Using Canny Edge Detection and CNN Classification”, 2018.

[6] Verma, J., Nath, M., Tripathi, P. et al. Analysis and identification of kidney stone using Kth nearest neighbour (KNN) and support vector machine (SVM) classification techniques. Pattern Recognit. Image Anal. 27, 574–580 (2017).

[7] Daniel C. Elton, Evrim B. Turkbey, Perry J. Pickhardt, Ronald M. Summers “A deep learning system for automated kidney stone detection and volumetric segmentation on noncontrast CT scans” 2021

[8] Prema T. Akkasaligar, Sunanda Biradar, Veena Kumbar “Kidney stone detection in computed tomography images” IEEE 2017

[9] Mehmet Baygin, Orhan Yaman, Prabal Datta Barua, Sengul Dogan, Turker Tuncer, U. Rajendra Acharya “Exemplar Darknet19 feature generation technique for automated kidney stone detection with coronal CT images” 2021

[10] A.Nithya, Ahilan Appathurai, N.Venkatadri,D.R.Ramji, C.Anna Palagan " Kidney disease detection and segmentation using artificial neural network and multi-kernel k-means clustering for ultrasound images" 2020

APPENDIX I: SAMPLE CODE

Project_K.ipynb:

```
import numpy as np
import matplotlib.pyplot as plt
import os
import cv2
import random
DIR='C:/Users/Manohar Vemuri/Desktop/M/project/CT_SCAN'
CATEGORIES=["Normal","Kidney_stone"]
training_data=[]
def train_data():
    for categories in CATEGORIES:
        path=os.path.join(DIR,categories)
        class_num=CATEGORIES.index(categories)
        for img in os.listdir(path):
            try:
                img_array=cv2.imread(os.path.join(path,img))
                new_array=cv2.resize(img_array,(128,128))
                training_data.append([new_array,class_num])
            except Exception as e:
                pass
train_data()
random.shuffle(training_data)
X=[]
y=[]
for features,labels in training_data:
    X.append(features)
    y.append(labels)
X=np.array(X)
y=np.array(y)
```

```
from sklearn.model_selection import train_test_split
Xtrain,Xtest,ytrain,ytest=train_test_split(X,y,test_size=0.1,random_state=1)
print(len(Xtrain),len(Xtest),len(ytrain),len(ytest))
Xtrain=np.array(Xtrain)
Xtest=np.array(Xtest)
import numpy as np
import cv2 as cv
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.optimizers import SGD
from keras.layers import Dense, Conv2D, Flatten, Convolution2D, Activation
from keras.layers import Dropout, MaxPooling2D
from tensorflow.keras.applications import ResNet50,Xception
import pickle
def median_blur(img):
    mb = cv2.medianBlur(img, 1)
    res=hist(mb)
    return res
def hist(img):
    res = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    gamma_corrected = np.array(255*(res / 255) ** 2.5, dtype = 'uint8')
    thres=thresholding_img(gamma_corrected)
    return thres
def thresholding_img(img):
    ret, thresh1 = cv2.threshold(img, 100, 255, cv2.THRESH_BINARY)
    return thresh1
```

```
processed_img=[]
for img in Xtrain:
    res=median_blur(img)
    processed_img.append(res)
processed_img=np.array(processed_img)
def build_model(hp):
    model=keras.Sequential([
        keras.layers.Convolution2D(
            filters=hp.Int('conv_1_filter',min_value=32,max_value=128,step=16),
            kernel_size=hp.Choice('conv_1_kernel',values=[3,5]),
            activation='relu',
            input_shape=(128, 128, 1)
        ),
        keras.layers.MaxPooling2D(pool_size=(3, 3)),
        keras.layers.Conv2D(
            filters=hp.Int('conv_2_filter',min_value=32,max_value=128,step=16),
            kernel_size=hp.Choice('conv_2_kernel',values=[3,5]),
            activation='relu'
        ),
        keras.layers.MaxPooling2D(pool_size=(3, 3)),

        keras.layers.Flatten(),
        keras.layers.Dense(
            units=hp.Int('dense_1_units',min_value=32,max_value=128,step=16),
            activation='relu'
        ),
        keras.layers.Dropout(0.5),
        keras.layers.Dense(1,activation='sigmoid')
    ]
    )
```

```
model.compile(optimizer=keras.optimizers.RMSprop(hp.Choice('learning_rate',values=[1e-2,1e-3])),
              loss='binary_crossentropy',
              metrics=['accuracy'])
    return model
from kerastuner import RandomSearch
from kerastuner.engine.hyperparameters import HyperParameters
tuner_search=RandomSearch(build_model,objective='val_accuracy',max_trials=5,overwrite=True)
tuner_search.search(processed_img,ytrain,epochs=5,validation_split=0.1)
model=tuner_search.get_best_models(num_models=1)[0]
model.summary()
model.fit(processed_img,ytrain,epochs=20,batch_size=100,verbose=1,validation_split=0.1)
model.save('C:/Users/Manohar Vemuri/Desktop/M/project')
model = keras.models.load_model('C:/Users/Manohar Vemuri/Desktop/M/project')
Xtest_processed_img=[]
for img in Xtest:
    res=median_blur(img)
    Xtest_processed_img.append(res)
    Xtest_processed_img=np.array(Xtest_processed_img)
ypred=model.predict(Xtest_processed_img)
#Confusion Matrix
from sklearn.metrics import confusion_matrix
cc=confusion_matrix(ytest,(ypred>0.75)*1)
cc
#Accuracy
from sklearn.metrics import accuracy_score
accuracy_score(ytest,(ypred>0.8)*1)*100
```

Test.ipynb:

```
import numpy as np
import cv2 as cv
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.optimizers import SGD
from keras.layers import Dense, Conv2D, Flatten, Convolution2D, Activation
from keras.layers import Dropout, MaxPooling2D
from tensorflow.keras.applications import ResNet50,Xception
import pickle
import os
training_data1=[]
path='C:/Users/Manohar Vemuri/Desktop/M/project/K_Test'
for img in os.listdir(path):
        try:
            img_array=cv.imread(os.path.join(path,img))
            new_array=cv.resize(img_array,(128,128))
            training_data1.append(new_array)
        except Exception as e:
            pass
def median_blur(img):
   mb = cv.medianBlur(img, 1)
   res=hist(mb)
   return res
def hist(img):
   res = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
```

```
    gamma_corrected = np.array(255*(res / 255) ** 2.5, dtype = 'uint8')
    thres=thresholding_img(gamma_corrected)
    return thres
def thresholding_img(img):
    ret, thresh1 = cv.threshold(img, 100, 255, cv.THRESH_BINARY)
    return thresh1
processed_img=[]
for img in training_data1:
    res=median_blur(img)
    processed_img.append(res)
processed_img=np.array(processed_img)
from keras.models import load_model
#model = keras.models.load_model('C:/Users/Manohar Vemuri/Desktop/M/project')
model=load_model('my_model.h5')
ypred=model.predict(processed_img)
x = 1
for i in range(len(processed_img)):
    if((ypred[[i]]>0.75)*1):
        print('Kidney Stone Detected')
    else:
        print('Normal')
    plt.figure(figsize=(10,128))
    plt.subplot(len(processed_img),2,x)
    x+=1
    plt.imshow(training_data1[i],cmap='gray')
    plt.subplot(len(processed_img),2,x)
    x+=1
    plt.imshow(processed_img[i],cmap='gray')
    plt.show()
```