**Dental Cavity Classification using Convolutional Neural Network**

Navya Verma(20BAI1178)

Pitchika Vaishnavi (20BAI1151)

Dr. Geetha S.

*School of Computer Science and Engineering*

*Vellore Institute of Technology*   
Chennai, India  
navya.varma2020@vitstudent.ac.in

pitchika.vaishnavi2020@vitstudent.ac.in

***ABSTRACT***—**This project aims to develop a dental cavity detection system using Convolutional Neural Networks (CNN) to analyze both color and x-ray images. Most existing projects deal with the detection of dental cavities in x-ray images only. The proposed system will first preprocess the images to augment the dataset to create more image samples. Then, the CNN model will be trained using a large dataset of annotated dental images to learn the features of dental cavities. The trained model will be able to accurately detect cavities in both color and x-ray images. The model for color images gives an accuracy of 95% and the model for xray images gives an accuracy of average 83%. The system will be evaluated on a test set of dental images and give prediction for cavity detection. The ultimate goal of this project is to provide dentists and healthcare professionals with a reliable tool to assist in early detection of dental cavities, thereby improving patient outcomes and reducing treatment costs.**

***Keywords—Dental, Cavities, X-ray, Dentist, Convolutional Neural Network, Image processing, Medical,Oral health.***

1. **INTRODUCTION**

Dental cavity detection is a critical task in dentistry that can help prevent and treat dental caries, one of the most common oral health problems worldwide. Traditional methods of detecting dental cavities involve visual inspection by a dental professional, which can be time-consuming, subjective, and error-prone. In recent years, there has been an increasing interest in developing computer-aided detection systems to improve the accuracy and efficiency of cavity detection.

Convolutional Neural Networks (CNNs) are a type of artificial neural network that has gained significant attention in the field of image processing and computer vision. CNNs have demonstrated remarkable performance in various image-based tasks, including object recognition, image segmentation, and medical image analysis. Dental cavity detection is a promising application of CNNs in the field of dentistry.

CNNs are designed to learn features automatically from the input images through multiple layers of convolution, pooling, and non-linear activation functions. The network architecture is typically composed of several convolutional layers followed by a series of fully connected layers, which allow the network to learn complex representations of the input images. During training, the network learns to adjust the weights of its parameters to minimize the difference between the predicted output and the ground truth labels. Once the network is trained, it can be used to predict the presence or absence of dental cavities in new images.

There are several advantages of using CNNs for dental cavity detection. First, CNNs can process large amounts of data quickly and accurately, which can help dental professionals to make timely and informed decisions. Second, CNNs can detect subtle changes in the appearance of dental cavities that may be difficult to identify with the naked eye. Third, CNNs can reduce the subjectivity and variability of cavity detection among different dental professionals.

Several studies have investigated the use of CNNs for dental cavity detection. For example, a recent study [1] developed a deep learning model based on CNNs to detect dental cavities in bitewing radiographs. The model achieved an accuracy of 94.3%, which outperformed several traditional machine learning algorithms. These studies demonstrate the potential of CNNs for dental cavity detection.

In conclusion, dental cavity detection using CNNs is a promising area of research that can improve the accuracy and efficiency of cavity detection in dentistry. CNNs have demonstrated remarkable performance in various image-based tasks and have the potential to revolutionize the way dental professionals diagnose and treat dental caries. Further research is needed to develop robust and reliable CNN-based models for dental cavity detection that can be integrated into clinical practice.

The project has been done with the joint contribution of Pitchika Vaishnavi 20BAI1151 and Navya Verma 20BAI1178. Navya Verma has contributed with topic selection, dataset procuring and model selection. Pitchika Vaishnavi has contributed with dataset augmentation, dataset sorting and model training.

1. **LITERATURE REVIEW**

In [2], the paper by Bhattacharjee describes an automated dental cavity detection system using deep learning and explainable AI. The system uses convolutional neural networks (CNNs). The dataset used to train the CNNs is composed of over 1,200 high-quality annotated dental images. The results of the study show that the automated system achieves a high level of accuracy in detecting dental cavities, with an area under the receiver operating characteristic curve (AUC-ROC) of 0.94. The authors suggest that their automated dental cavity detection system has the potential to improve the efficiency and accuracy of dental diagnoses. In [3], the paper describes a mask-based cavity detection model for dental X-ray images, which is a deep learning-based approach for automated cavity detection in dentistry. The proposed model is designed to improve the accuracy of cavity detection by using a mask-based approach to localize and segment cavities in dental X-ray images. The model uses a fully convolutional network (FCN) architecture to generate a segmentation mask for each input X-ray image. The FCN consists of an encoder-decoder structure, where the encoder extracts features from the input image and the decoder generates the segmentation mask. A dataset of 1,000 X-ray images is used, which is divided into training, validation, and testing sets. The results show that the proposed model achieves a high level of accuracy in detecting cavities, with a sensitivity of 0.89 and a specificity of 0.93. The authors suggest that the mask-based approach used in their model improves the accuracy of cavity detection by localizing and segmenting cavities in X-ray images. In [4], the authors review various studies and applications of machine learning in dentistry, including dental image analysis, periodontal disease diagnosis, caries detection, and oral cancer detection. The review highlights the potential benefits of machine learning in dentistry, including improved accuracy and efficiency in diagnosis and treatment planning. The authors also discuss some of the challenges of applying machine learning in dentistry, such as the need for large and high-quality datasets and the potential ethical concerns. Overall, the review suggests that machine learning has great potential to improve dental diagnosis. However, further research is needed for the challenges associated with applying machine learning in dentistry and to ensure that the benefits of these techniques are realized in a safe and ethical manner. In [5], the article describes a semi-supervised learning approach for dental caries detection using deep learning techniques. The proposed approach is designed to improve the accuracy of caries detection by utilizing a combination of labeled and unlabeled data in the training process. The approach uses a deep convolutional neural network (CNN) architecture, trained on a dataset of both labeled and unlabeled dental X-ray images. The CNN is first trained on the labeled images, and then fine-tuned on the unlabeled images. This approach enables the model to learn from a larger set of data and improve its accuracy in caries detection. To evaluate the performance of the proposed approach, the authors conducted experiments on a dataset of 5,880 dental X-ray images. The results show that the semi-supervised learning approach improves the accuracy of caries detection. The proposed model achieved a sensitivity of 0.91 and a specificity of 0.96 in caries detection. The authors suggest that the proposed approach has the potential to improve the efficiency and accuracy of caries detection in dentistry. In [6], the research proposes a method for early detection of dental cavities using image processing techniques. The proposed method uses Sobel edge detection and deep convolutional neural networks to predict cavities in the early stages. Preprocessing is done using histogram equalization, contrast enhancement, and feature selection. The method is compared to other segmentation techniques, including Otsu's threshold and Watershed, and achieved an accuracy of 96.08%. The proposed method is efficient for prediction and can aid in early diagnosis of dental disease. In [7], the paper addresses the problem of detecting cavities in dental and oral diseases, which are common worldwide and result in significant medical expenses. The authors propose a method that uses visual images of teeth and applies deep convolutional neural network (CNN) to classify the teeth into caries or non-caries. The dataset used in the study is from Kaggle, and the model achieves an accuracy of 71.43% after tuning. This approach can aid in the early detection of cavities and improve the efficiency of diagnosis in dental and oral diseases. In [8], the paper proposes a method to detect and number teeth in dental periapical films using the Faster R-CNN algorithm in the TensorFlow tool package. For improving precision, three post-processing techniques are proposed, including a filtering algorithm to remove overlapping boxes, a neural network model to detect missing teeth, and a rule-based module based on a teeth numbering system to modify detected results that violate certain intuitive rules. The method achieved high precisions and recalls of over 90%, with a mean IOU value of 91% between detected and ground truth boxes on a test dataset. The proposed algorithms were also compared to manual annotations by three dentists, and the results showed that the machine's performance was close to that of a junior dentist. In [9], the paper proposes a custom-made CNN, Dental-Net, to automatically detect cavities in teeth from oral photographic images. A dataset consisting of 609 diverse images of teeth with and without cavities, collected from various sources, is used to train the model. The images are resized, converted to grayscale, normalization, and on-the-fly data augmentation. The performance of four pre-trained models is also evaluated, and the results show that Dental-Net achieves the highest accuracy of 94.25% and 91.09% on the training and validation sets, respectively. The study demonstrates the potential of using CNNs for dental cavity detection and shows that the proposed model outperforms pre-trained models. The study highlights the importance of using a diverse dataset for training and testing the model's ability to perform well on images collected from varied sources. In [10], the paper proposed an automated tooth detection and dental condition classification method using deep convolutional neural networks for panoramic dental radiographs. An annotated dataset was used to train CNN and acquire semantic segmentation data. Image processing methods were applied to segment and fine-tune the bounding boxes corresponding to the teeth defections. The final step involved labeling each tooth sample inside the identified area of interest and using histogram-based majority voting to determine the issues that affected it. The method was evaluated based on criteria such as specificity, memory, and supervised classification. The proposed solution achieved accurate tooth categorization, identification of illnesses and severe gum disease like periodontitis, and estimation of cavity cleaning. In [11], the summary discusses a study on the use of deep convolutional neural networks in medicine for detection, prediction, and classification. The study proposes a novel method of automated tooth detection and dental disease categorization using panoramic X-rays. The evaluation of the approach was based on precision, recall, F1-score, and accuracy to assess bounding box detections and semantic segmentation. The results showed the superiority of the proposed solutions compared to other techniques. In [12], theis paper proposes a novel approach for automatic teeth detection and dental problem classification using panoramic X-ray images. Deep convolutional neural networks have been used to aid medical staff in making accurate diagnosis. The approach was evaluated based on several metrics such as accuracy, precision, recall, and F1-score. The results showed that the proposed solution was superior to other approaches. The study used panoramic radiographies from three dental clinics and annotated them, highlighting 14 different dental issues. A CNN was trained using annotated data, followed by image processing techniques for refining the bounding boxes corresponding to the teeth detections. Each tooth instance was labeled and the problem affecting it was identified using a histogram-based majority voting within the detected region of interest.

1. **MATERIALS AND METHODS**

This section describes all the materials used in this project as well as the methods implemented by us for achieving the results.

1. ***MODEL DESCRIPTION***

A sequential CNN model is a type of neural network architecture that is commonly used for image classification and processing tasks. It involves stacking multiple convolutional layers along with pooling and activation functions to create a deep neural network that can learn complex representations of image features. In a sequential CNN model, the input is fed through a series of convolutional layers with different filters that detect various features in the image. The output of each layer is then passed through a non-linear activation function, such as ReLU or sigmoid, to introduce non-linearity into the model. Max pooling layers are added to downsample the feature maps and reduce the number of parameters in the model. The layers are stacked in a sequential manner, with each layer building on the output of the previous layer. The final output layer of the model is usually a fully connected layer that maps the learned features to the output classes. One of the advantages of a sequential CNN model is its ability to learn hierarchical representations of the image features, allowing it to achieve high accuracy in image classification tasks. However, this comes at the cost of increased complexity and computation time, as well as a risk of overfitting if the model is too deep or the dataset is too small.

To this Sequential model, we have added the layers of Convolution, Pooling, Dropout, Flatten, Dense.

The procedure of the above is as follows:

1. *Selection of Sequential model*: Selecting a sequential model from keras models which is suitable for image training.
2. *Adding Layers to model*: We add the layers required in our model to the initial model with the model.add() function.
3. *Creating Image Data Generator and Augmenting images*: Owing to small size of the dataset we proceed to augment the dataset using image preprocessing libraries from keras.
4. *Loading the train and test datasets:* We then load the images from the labelled folders into the train image and test image generator using the flow\_from\_directory() function.
5. *Fitting the images to the model*: We proceed to fit the train and test images to the model with the fit\_generator() function with the parameters of 40 epochs and 10 steps per epoch.
6. *Verifying model*: We finally load an image from the two classes and get the prediction from the model using the auxiliary function of get\_res(), which gives us as output the image and label predicted by the model.
7. ***DATASET***

We have procured the dataset from multiple sources. We have got the dataset for colored cavity images from Kaggle [13], which had dental images divided according to classes, and the dataset for the x-ray dental images we have procured from Mendeley [14], and Zenodo [15], from which we have cropped out individual tooth x-rays and classified them into two different labels.

1. ***ARCHITECTURE***

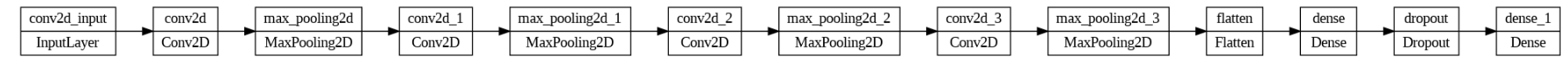
We have created a Sequential model which contains 10 layers. Architecture of the model is shown below. Training dataset was used at a learning rate of 0.001 and 40 epochs were used.

Fig 1 Sequential Model Layers

*Convolution Layer*: Convolutionary layers convolutes the inputs and transfers its result to the next layer. For its receptive region, each convolutionary neuron processes data only. Although it is possible to use completely linked feedforward neural networks to learn features as well as classify data, applying this architecture to images is not realistic. We use 3 things from the convolutionary layer: width and height of convolution kernel, number of input and output channels, depth of input filters.

*Pooling Layer*: Convolutionary networks can have local or global pooling layers, to streamline the underlying computation. By integrating the outputs of neuron clusters on one layer into a single neuron on the next layer, the pooling layers minimise the data measurements. Local pooling blends thin, usually 2 x 2 clusters. Global pooling works on all of the convolutional layer's neurons. A max or an average can also be calculated by pooling. Any neuron in one layer is bound to every neuron in another layer by completely linked layers. It is the same as the conventional multi-layer perceptron neural network (MLP) in theory. A vector of weights and a bias determine the function which is applied to the input values. Learning progresses by making iterative adjustments to these biases and weights in a neural network.

*Dropout Layer:* As provided above, we know that, CNN is prone to overfitting due to its fully connected nature. To solve this issue, we use dropout, which drops the nodes with a probability of 1-p at the training stage. We do this to validate the working of the model without some connections. After each iteration, the neurons are added back with their original weights.

*Flatten Layer:* We use the flatten layer function to flatten the images in the data and it transforms the data into a 1-dimensional sequence. To create a single long function vector, we flatten the output of the convolutional layers. And it is related to the final model of classification, called a fully connected layer. The complete architecture of our project is shown in the figure below. We repeat the Convolution, Polling and Dropout sequence two times.

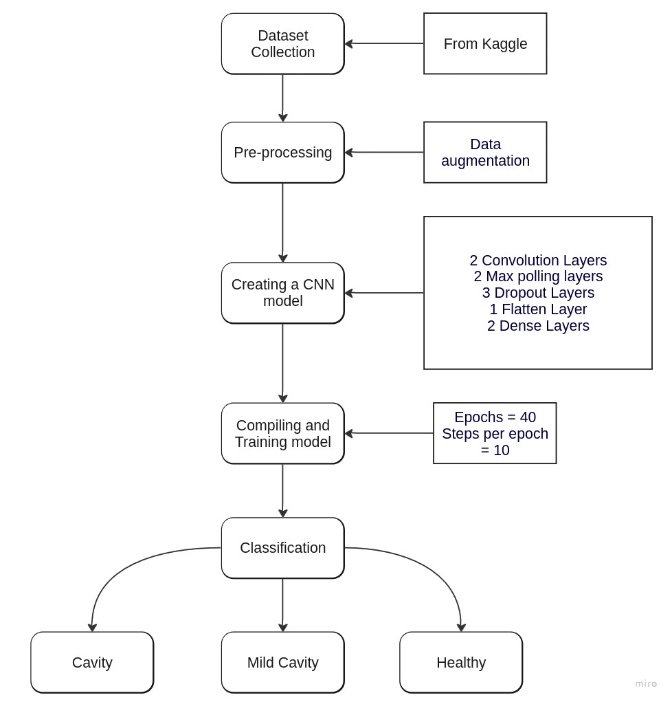


Fig 2 Complete Architecture

1. **PROPOSED WORK**
2. ***NOVELTY***

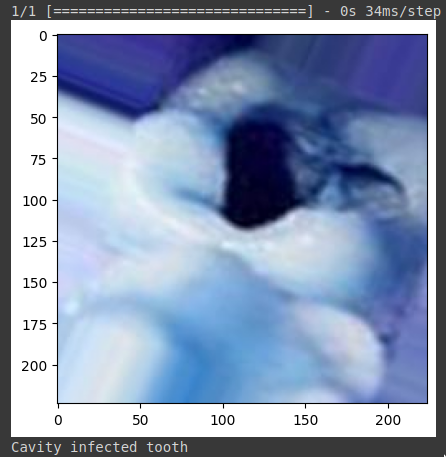
The novelty involved in our project is that we are offering a unified solution for the purpose of cavity detection rather than separate solutions that are already available. We cannot determine what data a patient may come with. Maybe they come to the dental clinic directly or maybe they did some prior x-ray and bring that to the doctor. Our solution will be able to provide assistance in both cases without using some separate tool or work. This reduces ambiguity and makes a single point of diagnosis.

1. ***CONTRIBUTIONS***

The contributions made by our project is that we have simplified oral health diagnosis in the field of cavity detection by creating an easily available model that can be used anywhere with any sort of dental image, be it a colored image or an x-ray image. This will allow patients who suspect cavity problems to easily get a preliminary diagnosis by uploading an image and consolidating whether they should visit a dental specialist or not. This helps in reducing the initial cost that is involved in going to a dental check up and having a cavity problem verified by a doctor, which involves two levels of costs, which our project has brought down to just one level, the treatment level.

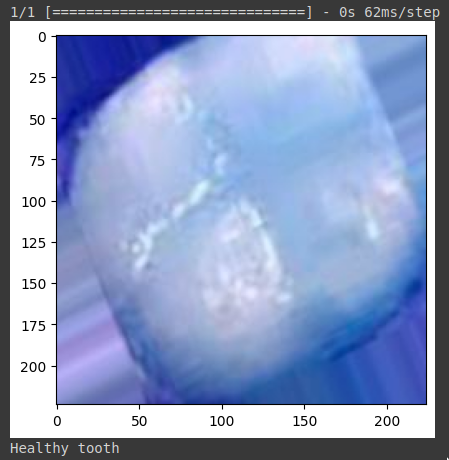
1. **RESULTS AND DISCUSSIONS**

We trained our model in 40 epochs, with 10 steps per epoch. For the dataset of colored dental imagesl, from the 40 epochs. We achieved an average accuracy of 95% from all the epochs. The model was able to predict the condition of dental cavity from the loaded images as shown in Fig 1 and 2. Fig 3 describes how the image of a cavity infected tooth is accurately classified with the text ‘Cavity infected tooth’. Fig 4 describes how the image of a healthy tooth is accurately classified with the text ‘Healthy tooth’.

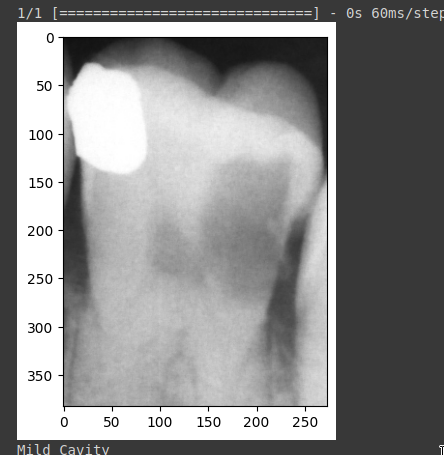


**Fig 3**

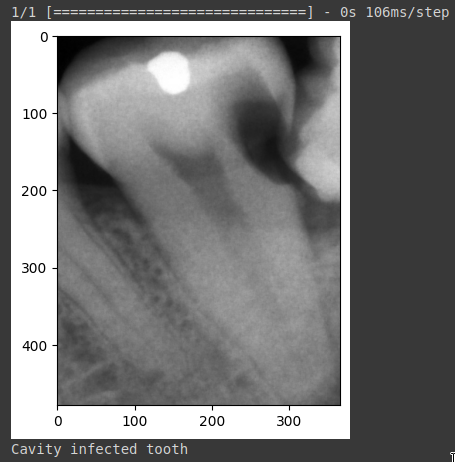
**Fig 4**



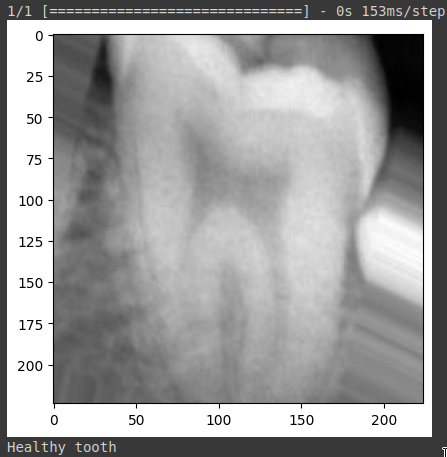
We also followed the same model approach for the dataset with x-ray images and achieved an average accuracy of 83%. The model was able to predict the condition of dental cavity from the loaded images as shown. Fig 5 shows how the image of a mildly hollow tooth is accurately classified with the text ‘Mild Cavity’. Fig 6 shows how the image of a cavity infected tooth is accurately classified with the text ‘Cavity infected tooth’. Fig 7 describes how the image of a healthy tooth is accurately classified with the text ‘Healthy tooth’.



**Fig 5**



**Fig 6**



**Fig 7**

1. **CONCLUSIONS**

In conclusion, our project aimed to develop a deep learning model for dental cavity detection using convolutional neural networks (CNNs). The project was planned to be different from existing work with the addition of both color and x-ray(greyscale) images to the prediction model compared to the mostly existing works done on just prediction from x-rays. The model was trained on both color images and X-rays to ensure the applicability of our approach in different dental imaging modalities. The proposed model achieved an average accuracy of 95% for color images and 83% for X-rays, indicating promising results for the early detection of dental cavities.

Our approach utilized a combination of image augmentation, and we added 10 layers to our Convolutional Neural Network to maximize the learned parameters and achieve fine-tuned results.

Our results demonstrate that deep learning models have the potential to be an effective tool for the early detection of dental cavities. The high accuracy achieved by our model suggests that it can be used in clinical settings to support dentists in diagnosing and treating dental cavities.

Future research can further improve the accuracy and generalizability of the model by incorporating additional data sources, such as patient history and demographics, and by exploring alternative CNN architectures. Overall, our project represents a promising step towards leveraging the power of deep learning to improve dental healthcare outcomes.

**REFERENCES**

1. Lee, S., Oh, Si., Jo, J. et al. Deep learning for early dental caries detection in bitewing radiographs. Sci Rep 11, 16807 (2021). <https://doi.org/10.1038/s41598-021-96368-7>
2. Bhattacharjee, Niharika. (2022). Automated Dental Cavity Detection System Using Deep Learning and Explainable AI. AMIA ... Annual Symposium proceedings. AMIA Symposium. 2022. 140-148.
3. Sahu, Madhusmita & Dash, Rasmita. (2020). A Mask-based Cavity Detection Model for Dental X-ray Image. 1-4. 10.1109/ICCSEA49143.2020.9132908.
4. Patil, Shashikant & Kulkarni, Vaishali & Bhise, Archana. (2019). Algorithmic analysis for dental caries detection using an adaptive neural network architecture. Heliyon. 5. e01579. 10.1016/j.heliyon.2019.e01579.
5. Qayyum, Adnan & Tahir, Ahsen & Butt, Muhammad & Luke, Alexander & Abbas, Hasan & Qadir, Junaid & Arshad, Kamran & Assaleh, Khaled & Imran, Muhammad & Abbasi, Qammer. (2023). Dental caries detection using a semi-supervised learning approach. Scientific Reports. 13. 10.1038/s41598-023-27808-9.
6. M. M. Lakshmi and P. Chitra, "Classification of Dental Cavities from X-ray images using Deep CNN algorithm," 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184), Tirunelveli, India, 2020, pp. 774-779, doi: 10.1109/ICOEI48184.2020.9143013.
7. Thamrin, Sri Astuti, et al. “Predicting Obesity in Adults Using Machine Learning Techniques: An Analysis of Indonesian Basic Health Research 2018.” *Frontiers*, 27 Apr. 2021, [www.frontiersin.org/articles/10.3389/fnut.2021.669155/full](http://www.frontiersin.org/articles/10.3389/fnut.2021.669155/full).
8. Chen, H., Zhang, K., Lyu, P. et al. A deep learning approach to automatic teeth detection and numbering based on object detection in dental periapical films. Sci Rep 9, 3840 (2019). <https://doi.org/10.1038/s41598-019-40414-y>
9. M. A. H. Khan, P. S. Giri and J. A. A. Jothi, "Detection of Cavities from Oral Images using Convolutional Neural Networks," 2022 International Conference on Electrical, Computer and Energy Technologies (ICECET), Prague, Czech Republic, 2022, pp. 1-6, doi: 10.1109/ICECET55527.2022.9872786.
10. M. A. H. Khan, P. S. Giri and J. A. A. Jothi, "Detection of Cavities from Oral Images using Convolutional Neural Networks," 2022 International Conference on Electrical, Computer and Energy Technologies (ICECET), Prague, Czech Republic, 2022, pp. 1-6, doi: 10.1109/ICECET55527.2022.9872786.
11. D. Samiappan, J. R, N. S. K and N. K. N, "Analysis of Dental X-Ray Images for the Diagnosis and Classification of Oral Conditions," 2023 7th International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2023, pp. 564-569, doi: 10.1109/ICCMC56507.2023.10083914.
12. M. P. Muresan, A. R. Barbura and S. Nedevschi, "Teeth Detection and Dental Problem Classification in Panoramic X-Ray Images using Deep Learning and Image Processing Techniques," 2020 IEEE 16th International Conference on Intelligent Computer Communication and Processing (ICCP), Cluj-Napoca, Romania, 2020, pp. 457-463, doi: 10.1109/ICCP51029.2020.9266244.
13. <https://www.kaggle.com/datasets/pushkar34/teeth-dataset>
14. <https://data.mendeley.com/datasets/hxt48yk462/1>
15. <https://zenodo.org/record/4907880>

**APPENDIX**

**Code for the project:**

# Importing necessary libraries

import matplotlib.pyplot as plt

from keras.models import load\_model

from keras.preprocessing import image

import numpy as np

import cv2

from keras.preprocessing.image import ImageDataGenerator

from PIL import Image as pil\_image

%matplotlib inline

from keras import backend as K

import tensorflow as tf

from keras.models import Sequential

from keras.layers import Activation, Dropout, Flatten, Dense, Conv2D, MaxPooling2D

from skimage import io

from mpl\_toolkits.axes\_grid1 import ImageGrid

import random

image\_gen = ImageDataGenerator(rotation\_range=30, width\_shift\_range=0.1,height\_shift\_range=0.1, rescale=1/255,shear\_range=0.2,zoom\_range=0.2, horizontal\_flip=True, fill\_mode='nearest')

image\_shape=(150,150,3)

model = Sequential()

model.add(Conv2D(filters=32, kernel\_size=(3,3), input\_shape=image\_shape, activation='relu'))

model.add(MaxPooling2D(pool\_size=(2,2)))

model.add(Conv2D(filters=64, kernel\_size=(3, 3), input\_shape=image\_shape, activation='relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Flatten())

model.add(Dense(128))

model.add(Activation('relu'))

model.add(Dropout(0.5))

model.add(Dense(1))

model.add(Activation('sigmoid'))

model.compile(loss='binary\_crossentropy',

optimizer='Adam',

metrics=['accuracy'])

model.save\_weights('model1.h5')

datagen = ImageDataGenerator(

rotation\_range = 40,

shear\_range = 0.2,

zoom\_range = 0.2,

horizontal\_flip = True,

brightness\_range = (0.5, 1.5))

import os

image\_directory = 'drive/My Drive/teeth\_dataset/Training/caries'

SIZE = 224

dataset = []

my\_images = os.listdir(image\_directory)

for i, image\_name in enumerate(my\_images):

if (image\_name.split('.')[1] == 'jpg' or image\_name.split('.')[1] == 'jpeg'):

image = io.imread(image\_directory + '/' + image\_name)

image = pil\_image.fromarray(image, 'RGB')

image = image.resize((SIZE,SIZE))

dataset.append(np.array(image))

x = np.array(dataset)

i = 0

for batch in image\_gen.flow(x, batch\_size=16,save\_to\_dir= 'drive/My Drive/teeth\_dataset/Training/caries',save\_prefix='aug',save\_format='jpg'):

i += 1

if i > 200:

break

datagen = ImageDataGenerator(

rotation\_range = 40,

shear\_range = 0.2,

zoom\_range = 0.2,

horizontal\_flip = True,

brightness\_range = (0.5, 1.5))

image\_directory = 'drive/My Drive/teeth\_dataset/Training/without\_caries'

SIZE = 224

dataset = []

my\_images = os.listdir(image\_directory)

for i, image\_name in enumerate(my\_images):

if (image\_name.split('.')[1] == 'jpg' or image\_name.split('.')[1] == 'jpeg'):

image = io.imread(image\_directory + '/' + image\_name)

image = pil\_image.fromarray(image, 'RGB')

image = image.resize((SIZE,SIZE))

dataset.append(np.array(image))

x = np.array(dataset)

i = 0

for batch in image\_gen.flow(x, batch\_size=16,save\_to\_dir= 'drive/My Drive/teeth\_dataset/Training/without\_caries',save\_prefix='aug',save\_format='jpg'):

i += 1

if i > 200:

break

datagen = ImageDataGenerator(

rotation\_range = 40,

shear\_range = 0.2,

zoom\_range = 0.2,

horizontal\_flip = True,

brightness\_range = (0.5, 1.5))

import os

image\_directory = 'drive/My Drive/teeth\_dataset/test/caries'

SIZE = 224

dataset = []

my\_images = os.listdir(image\_directory)

for i, image\_name in enumerate(my\_images):

if (image\_name.split('.')[1] == 'jpg' or image\_name.split('.')[1] == 'jpeg'):

image = io.imread(image\_directory + '/' + image\_name)

image = pil\_image.fromarray(image, 'RGB')

image = image.resize((SIZE,SIZE))

dataset.append(np.array(image))

x = np.array(dataset)

i = 0

for batch in image\_gen.flow(x, batch\_size=16,save\_to\_dir= 'drive/My Drive/teeth\_dataset/test/caries',save\_prefix='aug',save\_format='jpg'):

i += 1

if i > 100:

break

datagen = ImageDataGenerator(

rotation\_range = 40,

shear\_range = 0.2,

zoom\_range = 0.2,

horizontal\_flip = True,

brightness\_range = (0.5, 1.5))

import os

image\_directory = 'drive/My Drive/teeth\_dataset/test/no-caries'

SIZE = 224

dataset = []

my\_images = os.listdir(image\_directory)

for i, image\_name in enumerate(my\_images):

if (image\_name.split('.')[1] == 'jpg' or image\_name.split('.')[1] == 'jpeg'):

image = io.imread(image\_directory + '/' + image\_name)

image = pil\_image.fromarray(image, 'RGB')

image = image.resize((SIZE,SIZE))

dataset.append(np.array(image))

x = np.array(dataset)

i = 0

for batch in image\_gen.flow(x, batch\_size=16,save\_to\_dir= 'drive/My Drive/teeth\_dataset/test/no-caries',save\_prefix='aug',save\_format='jpg'):

i += 1

if i > 100:

break

train\_image\_gen=image\_gen.flow\_from\_directory('drive/My Drive/teeth\_dataset/Training/', target\_size=image\_shape[:2], batch\_size=batch\_size, class\_mode='binary')

test\_image\_gen=image\_gen.flow\_from\_directory('drive/My Drive/teeth\_dataset/test', target\_size=image\_shape[:2], batch\_size=batch\_size, class\_mode='binary')

import warnings

warnings.filterwarnings('ignore')

results = model.fit\_generator(train\_image\_gen,

epochs=40,

steps\_per\_epoch=10,

validation\_data=test\_image\_gen,

validation\_steps=12)

from tensorflow.keras.preprocessing import image

import numpy as np

def get\_res(path):

raw\_img = image.load\_img(path, target\_size=(150, 150))

raw\_img = image.img\_to\_array(raw\_img)

raw\_img = np.expand\_dims(raw\_img, axis=0)

raw\_img = raw\_img / 255

predict = model.predict(raw\_img)

plt.imshow(cv2.imread(path))

if predict >= 0.5:

text = "Healthy tooth"

elif 0.25 <= predict <= 0.45:

text = "Mild Cavity"

else:

text = "Cavity infected tooth"

plt.show()

print(text)

path='drive/My Drive/teeth\_dataset/test/caries/aug\_0\_1903.jpg'

get\_res(path)

datagen = ImageDataGenerator(

rotation\_range = 40,

shear\_range = 0.2,

zoom\_range = 0.2,

horizontal\_flip = True,

brightness\_range = (0.5, 1.5))

import os

image\_directory = 'drive/My Drive/teeth\_dataset\_xray/Training/xray\_cavity'

SIZE = 224

dataset = []

my\_images = os.listdir(image\_directory)

for i, image\_name in enumerate(my\_images):

if (image\_name.split('.')[1] == 'jpg' or image\_name.split('.')[1] == 'jpeg'):

image = io.imread(image\_directory + '/' + image\_name)

image = pil\_image.fromarray(image,"RGB")

image = image.resize((SIZE,SIZE))

dataset.append(np.array(image))

x = np.array(dataset)

i = 0

for batch in image\_gen.flow(x, batch\_size=16,save\_to\_dir= 'drive/My Drive/teeth\_dataset\_xray/Training/xray\_cavity',save\_prefix='aug',save\_format='jpg'):

i += 1

if i > 200:

break

datagen = ImageDataGenerator(

rotation\_range = 40,

shear\_range = 0.2,

zoom\_range = 0.2,

horizontal\_flip = True,

brightness\_range = (0.5, 1.5))

image\_directory = 'drive/My Drive/teeth\_dataset\_xray/Training/xray\_no\_cavity'

SIZE = 224

dataset = []

my\_images = os.listdir(image\_directory)

for i, image\_name in enumerate(my\_images):

if (image\_name.split('.')[1] == 'jpg' or image\_name.split('.')[1] == 'jpeg'):

image = io.imread(image\_directory + '/' + image\_name)

image = pil\_image.fromarray(image, 'RGB')

image = image.resize((SIZE,SIZE))

dataset.append(np.array(image))

x = np.array(dataset)

i = 0

for batch in image\_gen.flow(x, batch\_size=16,save\_to\_dir= 'drive/My Drive/teeth\_dataset\_xray/Training/xray\_no\_cavity',save\_prefix='aug',save\_format='jpg'):

i += 1

if i > 200:

break

datagen = ImageDataGenerator(

rotation\_range = 40,

shear\_range = 0.2,

zoom\_range = 0.2,

horizontal\_flip = True,

brightness\_range = (0.5, 1.5))

import os

image\_directory = 'drive/My Drive/teeth\_dataset\_xray/Testing/xray\_cavity'

SIZE = 224

dataset = []

my\_images = os.listdir(image\_directory)

for i, image\_name in enumerate(my\_images):

if (image\_name.split('.')[1] == 'jpg' or image\_name.split('.')[1] == 'jpeg'):

image = io.imread(image\_directory + '/' + image\_name)

image = pil\_image.fromarray(image, 'RGB')

image = image.resize((SIZE,SIZE))

dataset.append(np.array(image))

x = np.array(dataset)

i = 0

for batch in image\_gen.flow(x, batch\_size=16,save\_to\_dir= 'drive/My Drive/teeth\_dataset\_xray/Testing/xray\_cavity',save\_prefix='aug',save\_format='jpg'):

i += 1

if i > 100:

break

datagen = ImageDataGenerator(

rotation\_range = 40,

shear\_range = 0.2,

zoom\_range = 0.2,

horizontal\_flip = True,

brightness\_range = (0.5, 1.5))

import os

image\_directory = 'drive/My Drive/teeth\_dataset\_xray/Testing/xray\_no\_cavity'

SIZE = 224

dataset = []

my\_images = os.listdir(image\_directory)

for i, image\_name in enumerate(my\_images):

if (image\_name.split('.')[1] == 'jpg' or image\_name.split('.')[1] == 'jpeg'):

image = io.imread(image\_directory + '/' + image\_name)

image = pil\_image.fromarray(image, 'RGB')

image = image.resize((SIZE,SIZE))

dataset.append(np.array(image))

x = np.array(dataset)

i = 0

for batch in image\_gen.flow(x, batch\_size=16,save\_to\_dir= 'drive/My Drive/teeth\_dataset\_xray/Testing/xray\_no\_cavity',save\_prefix='aug',save\_format='jpg'):

i += 1

if i > 100:

break

train\_image\_gen=image\_gen.flow\_from\_directory('drive/My Drive/teeth\_dataset\_xray/Training/', target\_size=image\_shape[:2], batch\_size=batch\_size, class\_mode='binary')

test\_image\_gen=image\_gen.flow\_from\_directory('drive/My Drive/teeth\_dataset\_xray/Testing', target\_size=image\_shape[:2], batch\_size=batch\_size, class\_mode='binary')

results = model.fit\_generator(train\_image\_gen,

epochs=40,

steps\_per\_epoch=10,

validation\_data=test\_image\_gen,

validation\_steps=12)

path='drive/My Drive/teeth\_dataset\_xray/Testing/xray\_cavity/945.jpg'

get\_res(path)

**Output:**

