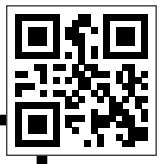

RV College of Engineering®

Autonomous institution affiliated to Visvesvaraya Technological University, Belagavi | Approved by AICTE, New Delhi

Go, change the world

Detection of Cavity from Dental Image Analysis using Machine Learning

An Internship Report

Submitted by,

Pragya Sen **1RV19EC126**

Under the guidance of

Dr. Sahana B

Professor

Dept. of ECE

RV College of Engineering

In partial fulfillment of the requirements for the degree of

Bachelor of Engineering in

Electronics and Communication Engineering

2022-23

RV College of Engineering®, Bengaluru

(Autonomous institution affiliated to VTU, Belagavi)

Department of Electronics and Communication Engineering

.

CERTIFICATE

Certified that the internship project work titled ***Detection of Cavity from Dental Image Analysis using Machine Learning*** is carried out by **Pragya Sen** (**1RV19EC126**) who is bonafide student of RV College of Engineering, Bengaluru, in partial fulfillment of the requirements for the degree of **Bachelor of Engineering** in **Electronics and Communication Engineering** of the Visvesvaraya Technological University, Belagavi during the year 2022-23. It is certified that all corrections/suggestions indicated for the Internal Assessment have been incorporated in the major project report deposited in the departmental library. The major project report has been approved as it satisfies the academic requirements in respect of major project work prescribed by the institution for the said degree.

Guide	Head of the Department	Principal
Dr. Sahana B	Dr. H. V. Ravish Aradhya	Dr. K. N. Subramanya

External Viva

Name of Examiners	Signature with Date
1.	
2.	

DECLARATION

I, **Pragya Sen** student of seventh semester B.E., Department of Electronics and Communication Engineering, RV College of Engineering, Bengaluru, hereby declare that the major project titled '**Detection of Cavity from Dental Image Analysis using Machine Learning**' has been carried out by me and submitted in partial fulfilment for the award of degree of **Bachelor of Engineering** in **Electronics and Communication Engineering** during the year 2022-23.

Further I declare that the content of the dissertation has not been submitted previously by anybody for the award of any degree or diploma to any other university.

I also declare that any Intellectual Property Rights generated out of this project carried out at RVCE will be the property of RV College of Engineering, Bengaluru and we will be one of the authors of the same.

Place: Bengaluru

Date:

Name	Signature
1. Pragya Sen(1RV19EC126)	

ACKNOWLEDGEMENT

I am indebted to my guides, **Dr. Sahana B**, Assistant Professor, Department of Electronics and Communication, **Kavitha S N** and **Veena Divya Krishnappa**, Assistant Professor, Department of Information Science and Engineering, RV College of Engineering, for the wholehearted support, suggestions and invaluable advice throughout my project work and also the help offered in the preparation of this report.

I also express our gratitude to my panel members **Dr. Sahana B**, Assistant Professor and **Dr. Chethana G**, Assistant Professor, Department of Electronics and Communication Engineering for their valuable comments and suggestions during the phase evaluations.

My sincere thanks to the project coordinators **Prof. Sindhu Rajendran R** and **Dr Veena Devi** for their timely instructions and support in coordinating the project.

My gratitude to **Prof. Narashimaraja P** for the organized latex template which made report writing easy and interesting.

My sincere thanks to **Dr. H. V. Ravish Aradhya**, Professor and Head, Department of Electronics and Communication Engineering, RVCE for the support and encouragement.

I express sincere gratitude to our beloved Principal, **Dr. K. N. Subramanya** and Vice Principal **Dr. K. S. Geetha** for the appreciation towards this project work.

I thank all the teaching staff and technical staff of Electronics and Communication Engineering department, RVCE for their help. Lastly, I take this opportunity to thank my family members and friends who provided all the backup support throughout the project work.

INTERNSHIP CERTIFICATE

Go, change the world

CoC/ISE/Internship/001/22-23

RV COLLEGE OF ENGINEERING®

(Autonomous Institution affiliated to VTU, Belagavi)

RV Vidyaniketan Post, Mysuru Road, Bengaluru - 560 059

Internship Certificate

This is to certify that ***Mr./Ms. Pragya Sen,*** *1RV19EC126, VII Semester B.E. Electronics and communication Engineering of RV College of Engineering, Bengaluru has satisfactorily completed Internship on* ***'Visual Computing - Detection of Cavity from Dental Image Analysis using Machine Learning',*** *during September-October 2022 (6 weeks).*

Mr. Sudhanva B.	Dr. Anala M.R.	Dr. K.N.Subramanya
Director, Bhargawa Info Tech Solutions (P) Ltd.	Coordinator, COC-Visual Computing	Principal, RVCE

SYNOPSIS

Half of the world's population suffers from dental and oral illnesses that are quite prevalent. These illnesses are widespread due to lack of resources or unsanitary behaviours, and it is estimated that oral disorders account for 5% of global healthcare costs. The field of medical science has greatly benefited from recent advances in machine learning and artificial intelligence. These algorithms make it possible to diagnose and treat illnesses effectively. Convolutional neural networks (CNNs) have emerged as one of machine learning's most potent tools for accurately tackling issues like image identification, segmentation, classification, etc. According to published research,CNN excels at natural picture categorisation tasks when a sizeable data set is available.

This study focuses on detecting cavities. The project has employed visual pictures of teeth and applied a deep Convolutional Neural Network (CNN) to categorise the teeth into caries or non-caries. It considers data sets from various subjects containing cavities and not containing cavities, that are to be classified. Python language provides various open source libraries like Numpy and TensorFlow etc. which are Machine Learning and Artificial Intelligence libraries. With the help of these libraries, Convolutional Neural Networks are utilized to build the model. The model is trained and tested using an open source data set.

The model makes use of an AI diagnostic system that can detect cavity presence. The data set contains in total 884 images of 2 classes, i.e., cavity and no cavity. They have been split into train and test folders in 8:2 ratio. There are 708 images belonging to class 1 and 176 images belonging to class 2. Using the MobileNet version 2 architecture, cavity detection after training the model for 100 epochs is achieved with an accuracy of 93%.

CONTENTS

LIST OF FIGURES

Chapter 1

Profile of the Organization

CHAPTER 1

PROFILE OF THE ORGANIZATION

1.1 Introduction

The internship was certified by the Centre of Competence in Visual Computing from the Information Science department of R.V. College of Engineering.The certification is done by Bhargawa Info Tech Solutions Pvt. Ltd.(BITS).Bhargawa Info Tech Solutions was founded in 2019 by Mr. Sudhanva B, a B.Tech graduate from R.V College of Engineering, Bengaluru. As an IOT and cyber security enthusiast, Sudhanva began a personal project of making his room and lab "smart". The comfort, convenience and security that was achieved from this project was truly remarkable and greatly appreciated by the elderly at his home. This motivated him to learn more about Industry 4.0 and continue his journey in the automation industry.

It is classified as Non Govt Company and it is Type 'A' enterprise.It is registered at Registrar of Companies, Bangalore. It offers services such as maintenance of websites and creation of multimedia presentation of other firms.It conducts research and development on Smart home and automation devices. BITS has following units:

1. Products
2. Services

The company offers many products of Home appliances automation devices, industrial appliances such as wireless water controller and smart home systems. It has made 14 homes smart and customized automation of 11 industries/factories as per client requirement. It also offers services such as maintenance of websites and creation of multimedia presentation of other firms. Engineering Knowledge in Industry:

1. Core Python fundamentals and Programming.
2. Python tools and libraries for Data Analysis and Visualization.
3. Deployment and Testing frameworks – Black Box Testing approach for ML models such as Model Performance, Metamorphic Testing and Dual Coding techniques

1.2 Organization Structure

Bhargawa Info Tech Solutions Private Limited's Annual General Meeting was last held on 31 December 2020 and as per records from Ministry of Corporate Affairs, its balance sheet was last filed on 31 March 2020. Mr. Sudhanva Belle and Mrs.Shailaja are the directors of BITS who lead the products and service sectors of the company. The basic structure of the organisation can be seen in Fig.1.1

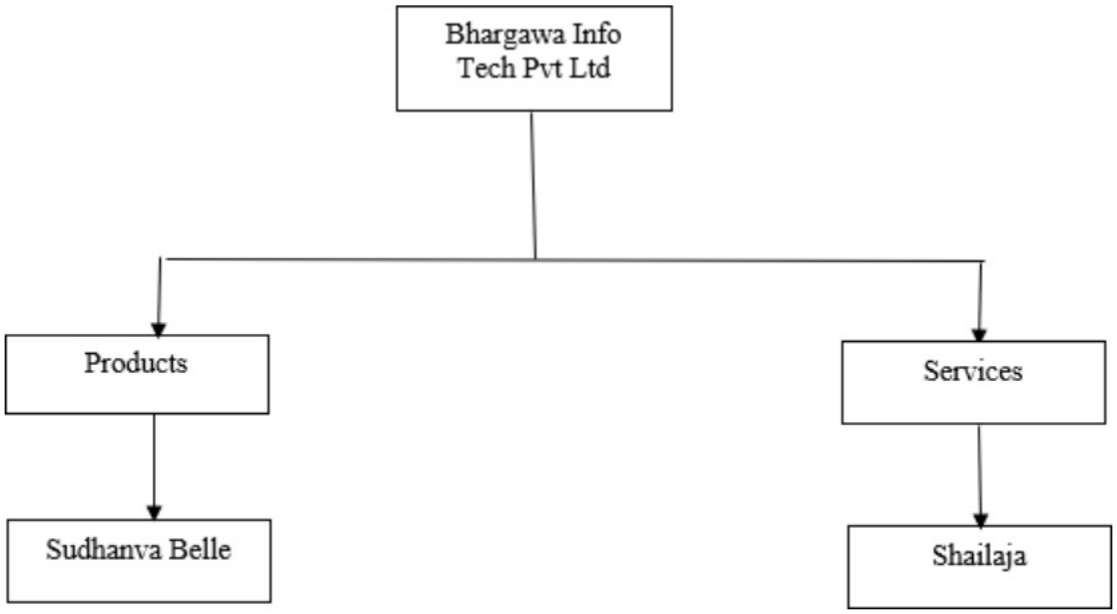

Figure 1.1: Structure Of the Organuzation

1.3 Products

R & D sector of BITS has started on smart home and industrial automation devices. Industrial automation products refer to use of control systems such as information technologies, robots and tools for handling different types of machinery and processes used in several types of industries.

Industrial systems have seen large shifts in their wired and wireless infrastructure this past decade with the shift from traditional fieldbus technologies to industrial Ethernet, with protocols such as Ethernet/IP, ProfitNET, and EtherCAT overtaking legacy networks.Industrial robot manipulators have been utilized in a wide array of industries to perform repetitive, strenuous, and precise tasks.

Many products have been launched in the sector of home appliances automation devices which includes wireless water controller for industry and home purposes.Automatic

water level controllers are a product that was created to automatically control a motor, which helps to ensure a constant reserve of water in a storage tank. These automatic water level controllers are used to automatically fill the over-head tank when it starts or has become empty as well as monitor the water level in it. The technique of water level monitoring and controlling system concentrated with some basic parts which are softly aggregated together such as Water level Indicator, Level sensors, microcontroller, pump controlling system etc. The company has successfully made 14 smart homes and automated 11 industries/factories with custom devices based on client requirements.

1.4 Services

BITS offers services for the clients which includes consultancies, maintenance of websites or creation of multimedia presentations for other firms etc. The BITS team provides products according to the client as per their requirement.It includes the services like managing and development of websites using several software tools. Web Managers are responsible for managing websites for organizations. Their duties include managing website upgrades, testing for website performance, implementing strategies to increase web traffic and subscribers, troubleshooting website issues, and responding to security breaches.

IT specialists from BITS help other organizations by managing their website. They oversee the functionality of the website, evaluate, and manage website performance, facilitate hosting and server management, and develop, maintain, and update website content. Following duties are performed by IT Specialists in BITS:

1. Plan, implement, manage, monitor, and upgrade the organization's website.
2. Respond to and troubleshoot all website issues.
3. Ensure that the website is protected by enabling the appropriate security measures.
4. Improve the User Experience of the website regularly.
5. Keep up-to-date with industry best practices and monitor competitor websites.

1.5 Financials

BITS have authorized share capital as Rs.5 lakh and its paid up capital is Rs.15k. Its balance sheet was last filed on 31 March 2020

Chapter 2

Activities of the Department

CHAPTER 2

ACTIVITIES OF THE DEPARTMENT

The Centre of Competence located in department of Information Science and engineering, RVCE provides computing facilities, offers Training and execute consultancy services. It facilitates skill enhancement programs in various domains like Computer Vision, Mobile app development, Machine learning and Deep learning with Python frameworks, etc.

2.1 About the department

Centre of Excellence in Visual Computing at RVCE in collaboration with BITS located in department of Information Science and engineering provides computing facilities for students, research scholars and faculty of RV College of Engineering.

RVCE, Bengaluru is established in 1963, offers 12 Undergraduate Engineering programs, 16 master's degree programs and Doctoral Studies. There are various Centre of Excellence(CoE) departments at RVCE for different emerging fields. The CoE is a initiative from RVCE, Bengaluru, to set up an environment that would empower young minds through knowledge and training for enhancing employability, research, and innovation in emerging fields such as Visual Computing, IoT, Quantum Computing, mobile applications etc. Internships at various CoE empower engineering students with up skilling, enrichment of practical knowledge and career building.

The Centre of Competence in Visual Computing located in department of Information Science and engineering provides computing facilities for students, research scholars and faculty of RV College of Engineering. The motive of the visual computing Centre is to bring the students and faculty of various disciplines together to execute interdisciplinary projects. The objective of CoE is to build technical capability and improve employability among students and fresh graduates.It allows students work on state-of-the-art multidisciplinary projects, sharpen experimental, analytical, and numerical skills, and learn the applicability of technologies in diverse fields.

The mission of CoE is to develop employable human resource to meet the challenges in the field of computer vision and data visualization at industries.The facility in CoE enables the execution of computationally intensive projects and research works in the area of Virtual Reality and Augmented Reality, Edge Computing, Parallel Programming,

Artificial Intelligence and Machine Learning and many more. The Centre offers internships in collaboration with "Bhargawa Info Tech Solutions Private Ltd" to facilitate skill enhancement in the domains like Computer Vision, Mobile app development, Internet of Things, Natural Language Processing, Human Computer Interaction etc.

The available infrastructure of the Centre includes high end workstations integrated with Titan X Pascal and Quadro RTX 8000 GPU cards and embedded devices like Jetson Nano, Developer and Tx2 kits.

2.2 Products

R & D sector of BITS has started on smart home and industrial automation devices. Industrial automation products refer to use of control systems such as information technologies, robots and tools for handling different types of machinery and processes used in several types of industries.

The company has successfully made 14 smart homes and automated 11 industries/-factories with custom devices based on client requirements.

Many products have been launched in the sector of home appliances automation devices which includes wireless water controller for industry and home purposes. Automatic water level controllers are a product that was created to automatically control a motor, which helps to ensure a constant reserve of water in a storage tank. These automatic water level controllers are used to automatically fill the over-head tank when it starts or has become empty as well as monitor the water level in it.

The technique of water level monitoring and controlling system concentrated with some basic parts which are softly aggregated together such as Water level Indicator, Level sensors, micro-controller, pump controlling system, etc.

Chapter 3

Tasks Performed

CHAPTER 3

TASKS PERFORMED

3.1 Dataset Description and Preprocessing

The dataset contains in total 884 images belonging to 2 classes, i.e., cavity and no cavity. They have been split into train and test folders in 8:2 ratio. There are 708 images belonging to class 1 and 176 images belonging to class 2. Images were in JPG format in the dataset. ImageDataGenerator from keras.preprocessing.image in python was used. 20% of images from the dataset were used for testing along with random horizontal flip. A zoom range of 0.15 was used for zooming purposes for our model and a rotation range of 20 for covering all rotations.

3.2 Model Architecture

The Tensorflow hub's Imagenet v3 was used to create the initial layers and a dense layer was added after that to create the neural network. The 2.0 version of Tensorflow was used as the pre-trained model. This model was used as it was giving a much higher accuracy as compared to the Tensorflow 1.x models and the keras.application models.

Reasons for using MobileNetV2:

1. MobileNet is a general architecture and can be used for multiple use cases. Depending on the use case, it can use different input layer size and different width factors.

2. MobileNets support any input size greater than 32 x 32, with larger image sizes offering better performance.

3. MobileNetV2 is very similar to the original MobileNet, except that it uses inverted residual blocks with bottlenecking features.

4. It simplifies the process of image processing.

5. It is lightweight making it have high execution speeds.

3.3 Tools and Technologies

3.3.1 Convolutional Neural Network(CNN)

A convolutional neural network, or CNN, is a deep learning neural network sketched for processing structured arrays of data such as portrayals. CNNs are very satisfactory at picking up on design in the input image, such as lines, gradients, circles, or even eyes and faces. CNN can be run directly on an underdone image and do not need any preprocessing. The strength of a convolutional neural network comes from a particular kind of layer called the convolutional layer. CNN contains many convolutional layers assembled on top of each other, each one competent of recognizing more sophisticated shapes. It is the sequential design that give permission to CNN to learn hierarchical attributes. In CNN, some of them followed by grouping layers and hidden layers are typically convolutional layers followed by activation layers. A convolutional neural network (CNN or ConvNet), is a network architecture for deeplearning which learns directly from data, eliminating the need for manual feature extraction.CNN is a special type of multilayer neural net-work or deep learning architecture inspired by the visual system of living beings. Convolutional Neural Network , is a type of Artificial Neural Net-work(ANN), which has deep feed-forward architecture and has amazing generalizing ability as compared to other networks with FC layers, it can learn highly abstracted features of objects especially spatial data and can identify them more efficiently.A deep CNN model consists of a finite set of processing layers that can learn various features of input data (e.g. image) with multiple level of abstraction . The initiatory layers learn and extract the high level features (with lower abstraction), and the deeper layers learns and ex-tracts the low level features (with higher abstraction).

A traditional convolutional neural network is made up of single or multiple blocks of convolution and pooling layers, followed by one or multiple fully connected layers and an output layer. The convolutional layer is the core building block of a CNN. This layer aims to learn feature representations of the input. The convolutional layer is composed of several learnable convolution kernels or filters which are used to compute different feature maps. Each unit of feature map is connected to a receptive field in the previous layer. The new feature map is produced by convolving the input with the kernels and applying element-wise non-linear activation function on the convolved result. The parameter sharing property of convolutional layer reduces the model complexity. Pooling

or sub-sampling layer takes a small region of the convolutional output as input and down samples it to produce a single output. There are different sub-sampling techniques as example max pooling, min pooling, average pooling, etc.

A convolutional neural network can have tens or hundreds of layers that each learn to detect different features of an image. Filters are applied to each training image at different resolutions, and the output of each convolved image is used as the input to the next layer. The filters can start as very simple features, such as brightness and edges and increase in complexity to features that uniquely define the object. Shared Weights and Biases: Like a traditional neural network, a CNN has neurons with weights and biases. The model learns these values during the training process, and it continuously updates them with each new training example. However, in the case of CNNs, the weights and bias values are the same for all hidden neurons in a given layer. This means that all hidden neurons are detecting the same feature, such as an edge or a blob, in different regions of the image. This makes the network tolerant to translation of objects in an image. For example, a network trained to recognize cars will be able to do so wherever the car is in the image.

Feature Learning, Layers, and Classification: Like other neural networks, a CNN is composed of an input layer, an output layer and many hidden layers in between. These layers perform operations that alter the data with the intent of learning features specific to the data. Three of the most common layers are: convolution, activation or ReLU, and pooling. Convolution puts the input images through a set of convolutional filters, each of which activates certain features from the images. Rectified linear unit (ReLU) allows for faster and more effective training by mapping negative values to zero and maintaining positive values. This is sometimes referred to as activation because only the activated features are carried forward into the next layer.

Pooling simplifies the output by performing nonlinear downsampling, reducing the number of parameters that the network needs to learn. These operations are repeated over tens or hundreds of layers, with each layer learning to identify different features.

Shared Weights and Biases: Like a traditional neural network, a CNN has neurons with weights and biases. The model learns these values during the training process, and it continuously updates them with each new training example. However, in the case of CNNs, the weights and bias values are the same for all hidden neurons in a given layer.

This means that all hidden neurons are detecting the same feature, such as an edge or a blob, in different regions of the image. This makes the network tolerant to translation of objects in an image. For example, a network trained to recognize cars will be able to do so wherever the car is in the image.

Classification Layers: After learning features in many layers, the architecture of a CNN shifts to classification. The next-to-last layer is a fully connected layer that outputs a vector of K dimensions where K is the number of classes that the network will be able to predict. This vector contains the probabilities for each class of any image being classified. The final layer of the CNN architecture uses a classification layer such as softmax to provide the classification output.

Applications of CNN : Using CNNs for deep learning is popular due to three important factors:

1. CNNs eliminate the need for manual feature extraction—the features are learned directly by the CNN.
2. CNNs produce highly accurate recognition results.
3. CNNs can be retrained for new recognition tasks, enabling you to build on pre existing networks.

Convolutional neural networks are most widely known for image analysis but they have also been adapted for several applications in other areas of machine learning, such as natural language processing.

Convolutional Neural Networks for Self-Driving Cars: Several companies, such as Tesla and Uber, are using convolutional neural networks as the computer vision component of a self-driving car. A self-driving car's computer vision system must be capable of localization, obstacle avoidance, and path planning. In the case of pedestrian detection, a pedestrian is a kind of obstacle which moves. A convolutional neural network must be able to identify the location of the pedestrian and extrapolate their current motion in order to calculate if a collision is imminent. A convolutional neural network for object detection is slightly more complex than a classification model, in that it must not only classify an object, but also return the four coordinates of its bounding box. Furthermore, the convolutional neural network designer must avoid unnecessary false alarms for irrelevant objects, such as litter, but also take into account the high cost of miscategorizing

a true pedestrian and causing a fatal accident. A major challenge for this kind of use is collecting labeled training data. Google's Captcha system is used for authenticating on websites, where a user is asked to categorize images as fire hydrants, traffic lights, cars, etc. This is actually a useful way to collect labeled training images for purposes such as self-driving cars and Google StreetView.

Convolutional Neural Networks for Text Classification: Although convolutional neural networks were initially conceived as a computer vision tool, they have been adapted for the field of natural language processing with great success. The principle behind their use on text is very similar to the process for images with the exception of a preprocessing stage. To use a convolutional neural network for text classification, the input sentence is tokenized and then converted into an array of word vector embeddings. It is then passed through a convolutional neural network with a final softmax layer in the usual way, as if it were an image. Consider a model which is to classify the sentence "Supreme Court to Consider Release of Mueller Grand Jury Materials to Congress" into one of two categories, 'politics' or 'sport'. Each of the 12 words in the sentence is converted to a vector, and these vectors are joined together into a matrix. Here a word vector size of 5 is used but in practice, large numbers such as 300 are often used.

Convolutional Neural Networks for Drug Discovery: The first stage of a drug development program is drug discovery, where a pharmaceutical company identifies candidate compounds which are more likely to interact with the body in a certain way. Testing candidate molecules in pre-clinical or clinical trials is expensive, and so it is advantageous to be able to screen molecules as early as possible. Proteins which play an important role in a disease are known as 'targets'. There are targets that can cause inflammation or help tumors grow. The goal of drug discovery is to identify molecules that will interact with the target for a particular disease. The drug molecule must have the appropriate shape to interact with the target and bind to it, like a key fitting in a lock. The San Francisco based startup Atomwise developed an algorithm called AtomNet, based on a convolutional neural network, which was able to analyze and predict interactions between molecules. Without being taught the rules of chemistry, AtomNet was able to learn essential organic chemical interactions. Atomwise was able to use AtomNet to identify lead candidates for drug research programs. AtomNet successfully identified a candidate treatment for the Ebola virus, which had previously not been known to have any antiviral

activity. The molecule later went on to pre-clinical trials.

Advantages of using CNN : One of the main reason for considering CNN in such case is the weight sharing feature of CNN, that reduce the number of trainable parameters in the network, which helped the model to avoid over-fitting and as well as to improved generalization. In CNN, the classification layer and the feature extraction layers learn together that makes the output of the model more organized and makes the output more dependent to the extracted features. The implementation of a large network is more difficult by using other types of neural networks rather than using Convolutional Neural Networks.

3.3.2 Jupyter Notebook

The Jupyter Notebook is a server-client application that allows editing and running notebook documents via a web browser. The Jupyter Notebook app can be executed on a local desktop requiring no internet access or can be installed on a remote server and accessed through the internet. In addition to displaying/editing/running notebook documents, the Jupyter Notebook App has a "Dashboard" (Notebook Dashboard), a "control panel" showing local files and allowing them to open notebook documents or shutting down their kernels.

It is an open-source IDE that is used to create Jupyter documents that can be created and shared with live codes. Also, it is a web-based interactive computational environment. The Jupyter notebook can support various languages that are popular in data science such as Python, Julia, Scala, R, etc.

It was formerly known as "IPython Notebooks". The "notebook" term can colloquially make reference to many different entities, mainly the Jupyter web application, Jupyter Python web server, or Jupyter document format depending on context.

According to the official website of Jupyter, Project Jupyter exists to develop open-source software, open-standards, and services for interactive computing across dozens of programming languages.

Jupyter Book is an open-source project for building books and documents from computational material. It allows the user to construct the content in a mixture of Markdown, an extended version of Markdown called MyST, Maths Equations using MathJax, Jupyter Notebooks, reStructuredText, the output of running Jupyter Notebooks at build time. Multiple output formats can be produced (currently single files, multipage HTML

web pages and PDF files).

3.3.3 Kernel

A notebook kernel is a "computational engine" that executes the code contained in a Notebook document. The python kernel executes python code. Kernels for many other languages exist. When a Notebook document is opened, the associated kernel is automatically launched. When the notebook is executed, the kernel performs the computation and produces the results. Depending on the type of computations, the kernel may consume significant CPU and RAM. RAM is not released until the kernel is shut-down. If the stride value is 2, then kernel moves by 2 columns of pixels in the input matrix. In short, the kernel is used to extract high-level features like edges from the image. Filters represent the number of output channels after convolution has been performed, while Kernel represents the size of a convolution filter being used to perform convolution on the image.

In Convolutional neural network, the kernel is nothing but a filter that is used to extract the features from the images. The kernel is a matrix that moves over the input data, performs the dot product with the sub-region of input data, and gets the output as the matrix of dot products. Kernel moves on the input data by the stride value. If the stride value is 2, then kernel moves by 2 columns of pixels in the input matrix. In short, the kernel is used to extract high-level features like edges from the image.

Filters represent the number of output channels after convolution has been performed, while Kernel represents the size of a convolution filter being used to perform convolution on the image. A simpler way to understand this is by taking into account a simple convolution operation on a 3 channel RGB Image using a Convolution layer with 32 filters and kernel size of 3. Assuming that the input image has height and width of 112 pixels, so the shape of input image becomes 112,112,3 where 112 represent the height and width, while 3 represents the channels, here RGB. Upon applying convolution on the image for single filter, a 3x3 matrix (comes from kernel size) is slid across the image that gives a single channel output. So after a successful convolution operation, the output image will consist 32 channels.

3.3.4 TensorFlow

TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

Machine learning is a complex discipline but implementing machine learning models is far less daunting than it used to be, thanks to machine learning frameworks—such as Google's TensorFlow—that ease the process of acquiring data, training models, serving predictions, and refining future results.

Created by the Google Brain team and initially released to the public in 2015, TensorFlow is an open source library for numerical computation and large-scale machine learning. TensorFlow bundles together a slew of machine learning and deep learning models and algorithms (neural networks) and makes them useful by way of common programmatic metaphors. It uses Python or JavaScript to provide a convenient front-end API for building applications, while executing those applications in high-performance C++. TensorFlow offers additional conveniences for developers who need to debug and gain introspection into TensorFlow apps. Each graph operation can be evaluated and modified separately and transparently, instead of constructing the entire graph as a single opaque object and evaluating it all at once. This so-called "eager execution mode," provided as an option in older versions of TensorFlow, is now standard. TensorFlow was developed for internal Google use in research and production.The initial version was released under the Apache License 2.0 in 2015. Google released the updated version of TensorFlow, named TensorFlow 2.0, in September 2019. TensorFlow can be used in a wide variety of programming languages, including Python, JavaScript, C++, and Java. This flexibility lends itself to a range of applications in many different sectors.

TensorFlow, which competes with frameworks such as PyTorch and Apache MXNet, can train and run deep neural networks for handwritten digit classification, image recognition, word embeddings, recurrent neural networks, sequence-to-sequence models for machine translation, natural language processing, and PDE (partial differential equation)-based simulations. Best of all, TensorFlow supports production prediction at scale, with the same models used for training. Nodes and tensors in TensorFlow are Python objects, and TensorFlow applications are themselves Python applications. The actual math operations, however, are not performed in Python. The libraries of transformations that are

available through TensorFlow are written as high-performance C++ binaries. Python just directs traffic between the pieces and provides high-level programming abstractions to hook them together.

TensorFlow allows developers to create data flow graphs—structures that describe how data moves through a graph, or a series of processing nodes. Each node in the graph represents a mathematical operation, and each connection or edge between nodes is a multidimensional data array, or tensor.

TensorFlow applications can be run on most any target that's convenient: a local machine, a cluster in the cloud, iOS and Android devices, CPUs or GPUs. By Google's own cloud,TensorFlow can be run on Google's custom TensorFlow Processing Unit silicon for further acceleration. The resulting models created by TensorFlow, though, can be deployed on most any device where they will be used to serve predictions.

TensorFlow 2.0, released in October 2019, revamped the framework in many ways based on user feedback, to make it easier to work with (as an example, by using the relatively simple Keras API for model training) and more performant. Distributed training is easier to run thanks to a new API, and support for TensorFlow Lite makes it possible to deploy models on a greater variety of platforms. However, code written for earlier versions of TensorFlow must be rewritten—sometimes only slightly, sometimes significantly—to take maximum advantage of new TensorFlow 2.0 features.

TensorFlow provides all features for the programmer by way of the Python language. Python is easy to learn and work with, and it provides convenient ways to express how high-level abstractions can be coupled together. TensorFlow is supported on Python versions 3.7 through 3.10, and while it may work on earlier versions of Python it's not guaranteed to do so.

Nodes and tensors in TensorFlow are Python objects, and TensorFlow applications are themselves Python applications. The actual math operations, however, are not performed in Python. The libraries of transformations that are available through TensorFlow are written as high-performance C++ binaries. Python just directs traffic between the pieces and provides high-level programming abstractions to hook them together.

High-level work in TensorFlow—creating nodes and layers and linking them together—uses the Keras library. The Keras API is outwardly simple; a basic model with three layers can be defined in less than 10 lines of code, and the training code for the same takes just

a few more lines of code.

The single biggest benefit TensorFlow provides for machine learning development is abstraction. Instead of dealing with the nitty-gritty details of implementing algorithms, or figuring out proper ways to hitch the output of one function to the input of another, the developer can focus on the overall application logic. TensorFlow takes care of the details behind the scenes.

TensorFlow offers additional conveniences for developers who need to debug and gain introspection into TensorFlow apps. Each graph operation can be evaluated and modified separately and transparently, instead of constructing the entire graph as a single opaque object and evaluating it all at once. This so-called "eager execution mode," provided as an option in older versions of TensorFlow, is now standard.

The TensorBoard visualization suite allows the user to inspect and profile the way graphs run by way of an interactive, web-based dashboard. A service, Tensorboard.dev (hosted by Google), allows the user to host and share machine learning experiments written in TensorFlow. It's free to use with storage for up to 100M scalars, 1GB of tensor data, and 1GB of binary object data.

TensorFlow also gains many advantages from the backing of an A-list commercial outfit in Google. Google has fueled the rapid pace of development behind the project and created many significant offerings that make TensorFlow easier to deploy and use. The above-mentioned TPU silicon for accelerated performance in Google's cloud is just one example.

3.3.5 MobileNet

MobileNet is a general architecture and can be used for multiple use cases. Depending on the use case, it can use different input layer size and different width factors. This allows different width models to reduce the number of multiply-adds and thereby reduce inference cost on mobile devices. MobileNets support any input size greater than 32 x 32, with larger image sizes offering better performance. MobileNetV2 is very similar to the original MobileNet, except that it uses inverted residual blocks with bottlenecking features. It has a drastically lower parameter count than the original MobileNet.

As the name applied, the MobileNet model is designed to be used in mobile applications, and it is TensorFlow's first mobile computer vision model. MobileNet uses depthwise separable convolutions. MobileNets are a family of mobile-first computer vision

models for TensorFlow, designed to effectively maximize accuracy while being mindful of the restricted resources for an on-device or embedded application. Mobilenet is a model which does the same convolution as done by CNN to filter images but in a different way than those done by the previous CNN. It significantly reduces the number of parameters when compared to the network with regular convolutions with the same depth in the nets. This results in lightweight deep neural networks. MobileNet is a class of CNN that was open-sourced by Google, and therefore, this gives us an excellent starting point for training our classifiers that are insanely small and insanely fast.

MobileNets are a family of mobile-first computer vision models for TensorFlow, designed to effectively maximize accuracy while being mindful of the restricted resources for an on-device or embedded application. Mobilenet is a model which does the same convolution as done by CNN to filter images but in a different way than those done by the previous CNN. It uses the idea of Depth convolution and point convolution which is different from the normal convolution as done by normal CNNs. This increases the efficiency of CNN to predict images and hence they can be able to compete in the mobile systems as well. Since these ways of convolution reduce the comparison and recognition time a lot, so it provides a better response in a very short time and hence we are using them as our image recognition model. MobileNets are small, low-latency, low-power models parameterized to meet the resource constraints of a variety of use cases. They can be built upon for classification, detection, embeddings, and segmentation.

3.4 Results Obtained

Using the above tools and technologies the algorithm is run according to the methodology, and the following outputs were obtained.

The dataset contains in total 884 images of 2 classes, i.e., cavity and no cavity. They have been split into train and test folders in 8:2 ratio.There are 708 images belonging to class 1 and 176 images belonging to class 2.Using the MobileNet V2 architecture, cavity detection after training the model for 100 epochs is achieved with an accuracy of 93%.It was stopped at 100 epochs as the model's loss was at the lowest value at this point for the training data. Finally, the model's loss and accuracy was plotted for both the training and testing data which can be seen in the Fig.3.1.

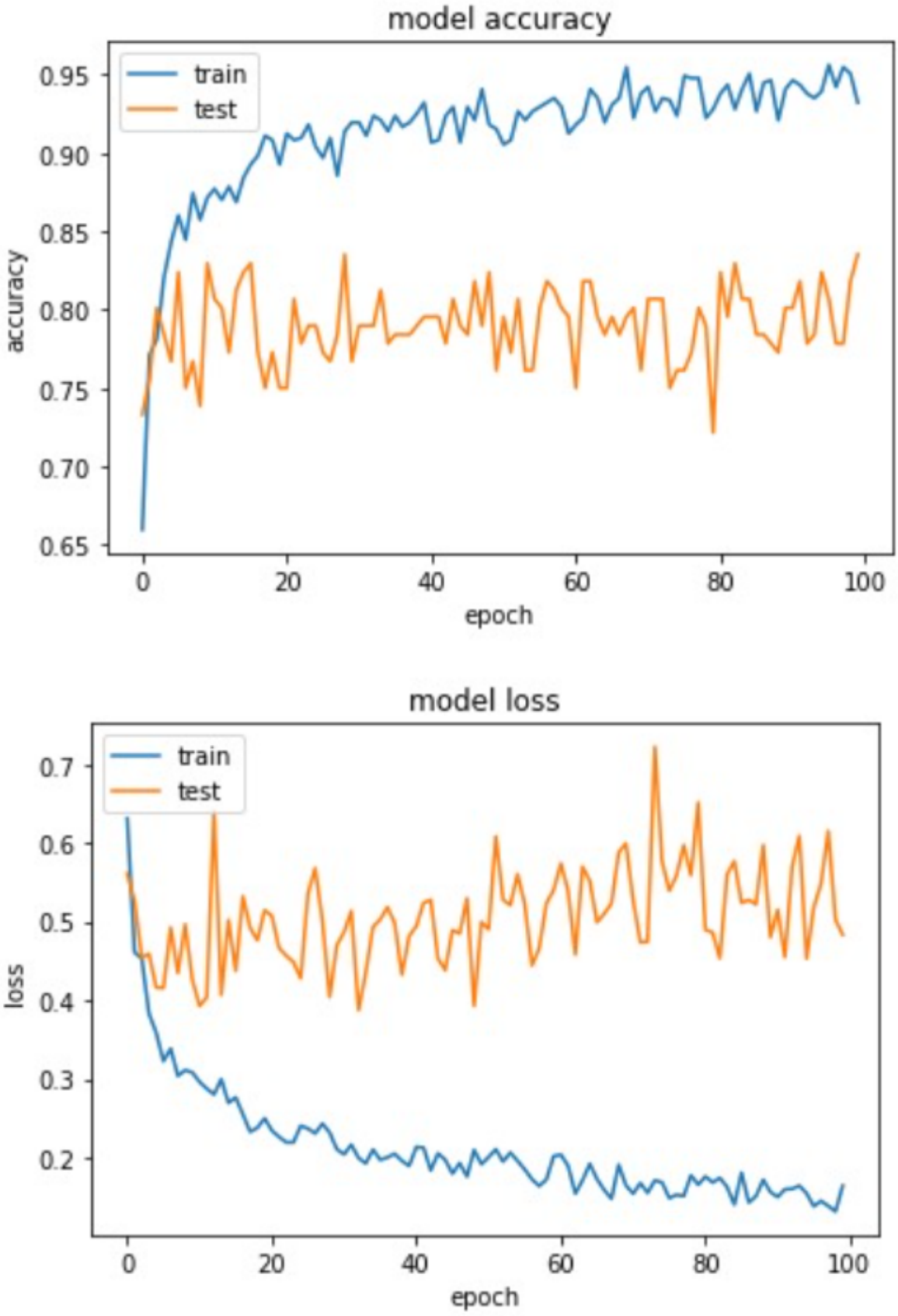

Figure 3.1: Accuracy and Loss

It can be seen that the model accuracy reaches 93% at the the 100th epoch which is the highest accuracy that can be obtained. The model's loss is also low at this point for the testing data.Hence the process is stopped at this point for the detection.

This chapter includes details about the tasks performed during the internship. It includes understanding the concepts of Python libraries, Machine Learning and various algorithms, and Image processing, developing a model for the Detection of Cavities from the analysis of Dental Images using Deep Learning, and performing Data augmentation to increase the size of the dental image dataset. It also discusses about the dataset used, data preprocessing and methodology, and the results obtained for the model detection.

Chapter 4

Reflections

CHAPTER 4

REFLECTIONS

This chapter deals with the conclusions that can be made from the project and the statistical graphs of the trainable and testable data. The parameters considered are loss and accuracy for 100 epochs. Future scope of this project based on the real world scenarios is also covered.

4.1 Learning Outcomes

Half of the world's population suffers from dental and oral illnesses that are quite prevalent.Out of these illnesses, it is estimated that chronic disorders account for 5% of global healthcare costs. Dental cavities, one of the most prevalent chronic disorders, impacts over 3.9 billion people worldwide. Typically, cavity detection requires the services of a trained dentist. However, barriers such as dentophobia, limited dentist availability, and lack of dental insurance prevent millions from receiving dental care. To address these issues, we created an AI diagnostic system that detects cavity presence. The accessible system detects cavities of all levels of decay using a Convolutional neural network (CNN). With a single photographic image, the system can provide a cavity diagnosis.

This study focused on detecting cavities. The project has employed visual pictures of teeth and applied a deep convolutional neural network (CNN) to categorise the teeth into caries or non-caries. It considers datasets from various subjects containing cavities and not containing cavities, that are to be classified. Python language provides various open source libraries like numpy and tensorflow etc. which are Machine Learning and Artificial Intelligence libraries. With the help of these libraries Convolutional Neural Networks are utilized to build the model. The model is trained and tested using an open source dataset.

The dataset contains in total 884 images of 2 classes, i.e., cavity and no cavity. They have been split into train and test folders in 8:2 ratio.There are 708 images belonging to class 1 and 176 images belonging to class 2.Using the MobileNet Version 2 architecture, cavity detection after training the model for 100 epochs is achieved with an accuracy of 93%.It was stopped at 100 epochs as the model's loss was at the lowest value at this point for the training data. Finally, the model's loss and accuracy was plotted for both the training and testing data.

Images of patients' teeth can be acquired using an intra-oral camera. Classification using Deep Learning is achievable using a computing device like a Raspberry Pi. This affordable setup could potentially be used as first level screening for dental cavities in developing nations where there is a limited dentist availability. Mobile applications can also be created which users/patients can use to snapshot the dental images and can get a result about the status of an issue. With increase in the dataset, the accuracy of the model can be improved.

Appendix A

Code

APPENDIX A

CODE

A.1 First Appendix

```
import os
import numpy as np
from zipfile import ZipFile


#extracting the zip file
zipobj=ZipFile(os.path.abspath('.')+'/dental_dataset.zip', 'r')
zipobj.extractall()


#pip install tf-nightly-gpupip install "tensorflow_hub==0.4.0"



# IMAGE AUGMENTATION #
import tensorflow as tf
from tensorflow.keras.preprocessing.image import
ImageDataGenerator

image_gen_train = ImageDataGenerator(rescale=1./255)

train_gen = image_gen_train.flow_from_directory(batch_size=20,
directory = os.path.abspath('.')+'/train',
shuffle = True,
target_size =(224, 224), class_mode = 'binary')

image_gen_test = ImageDataGenerator(rescale = 1./255)
```

```
test_gen = image_gen_test.flow_from_directory(batch_size=20,
directory=os.path.abspath('.')+'/test',
target_size=(224, 224), class_mode='binary')


# MODEL CREATION #
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Flatten,
from tensorflow.keras import layers
import tensorflow_hub as hub

URL = "https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_
vector/3"
feature_extractor = hub.KerasLayer(URL,
                                   input_shape=(224, 224,3))

feature_extractor.trainable = False #freezing the upper layer


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

model = tf.keras.Sequential([
  feature_extractor,
  layers.Dense(2, activation='softmax')
])

model.summary()

model.compile(
  optimizer='adam',
```

```
  loss='sparse_categorical_crossentropy',
  metrics=['accuracy'])


# EXECUTING THE MODEL #
EPOCHS = 100
history = model.fit_generator(train_gen,
                    epochs=EPOCHS,
                    validation_data=test_gen)
zipobj=ZipFile(os.path.abspath('.')+'/dental_dataset.zip', 'r')
zipobj.extractall()


#pip install tf-nightly-gpupip install "tensorflow_hub==0.4.0"




import tensorflow as tf
from tensorflow.keras.preprocessing.image import
ImageDataGenerator

image_gen_train = ImageDataGenerator(rescale=1./255)

train_gen = image_gen_train.flow_from_directory(batch_size=20,
directory = os.path.abspath('.')+'/train',
shuffle = True,
target_size =(224, 224), class_mode = 'binary')

model = tf.keras.Sequential([
  feature_extractor,
  layers.Dense(2, activation='softmax')
])
```

```
image_gen_test = ImageDataGenerator(rescale = 1./255)

test_gen = image_gen_test.flow_from_directory(batch_size=20,
directory=os.path.abspath('.')+'/test',
target_size=(224, 224), class_mode='binary')

train_gen = image_gen_train.flow_from_directory(batch_size=20,
directory = os.path.abspath('.')+'/train',
shuffle = True,
target_size =(224, 224), class_mode = 'binary')

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Flatten,
from tensorflow.keras import layers
import tensorflow_hub as hub

URL = "https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_
vector/3"
feature_extractor = hub.KerasLayer(URL,
                                   input_shape=(224, 224,3))

feature_extractor.trainable = False #freezing the upper layer

model = tf.keras.Sequential([
  feature_extractor,
  layers.Dense(2, activation='softmax')
])

model.summary()

model.compile(
  optimizer='adam',
```

```
  loss='sparse_categorical_crossentropy',
  metrics=['accuracy'])



# EXECUTING THE MODEL #
EPOCHS = 100
history = model.fit_generator(train_gen,
                    epochs=EPOCHS,
                    validation_data=test_gen)

# VISUALIZING THE MODEL PERFORMANCE #

import matplotlib.pyplot as plt
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model summary')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
```

```
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

BIBLIOGRAPHY

[1] Y. M, H. Y, R. Kanomi, and K. Takada, "Development of a diagnostic system that predicts the remaining period of pre-emergent eruption of lower second molars using a panoramic radiograph," Jan. 2019.

[2] P. S. M. K. V. Divya A. Jatti and R. Joshi, "Appending active contour model on digital panoramic dental x-rays images for segmentation of maxillofacial region," 2016.

[3] K. W. V. Sa-ing and S. S. Thongvigitmanee, "Automatic dental arch detection and panoramic image synthesis from ct images," in *35th Annual International Journal of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2017, pp. 733–736.

[4] R. N. S. A. Prajapati and S. Mitra, "Classification of dental diseases using cnn and transfer learning," in *5th International Symposium on Computational and Business Intelligence (ISCBI)*, vol. 3, 2017, 186–193 vol.3.

[5] S. T. A. Aberin and J. C. d. Goma, "Detecting periodontal disease using convolutional neural networks," *IEEE International Journal on Humanoid, Nanotechnology, Information Technology,Communication and Control, Environment and Management (HNICEM)*, vol. 54, no. 2, pp. 376–383, 2020.

[6] M. M. M. A. Haghanifar and S. B. Ko, "Automated teeth extraction from dental panoramic x-ray images using genetic algorithm," *IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 57, no. 8, pp. 2486–2490, 2019.

[7] S. A. AlGhamdi ASA Ragab M AlGhamdi, "Detection of dental diseases through x-ray images using neural search architecture network," *IEEE Computational Intelligence Neuroscience*, vol. 61, no. 2, pp. 606–613, 2018.

[8] K. M. Winocur E Reiter S, "Classifying degenerative joint disease by the rdc/tmd and by panoramic imaging: A retrospective analysis," *J Oral Rehabil.*, vol. 64, no. 6, pp. 2270–2282, 2021.

[9] B. Z. S. Tian N. Dai and X. Cheng, "Automatic classification and segmentation of teeth on 3d dental model using hierarchical deep learning networks," *IEEE Access*, vol. 49, no. 2, pp. 150–157, 2021.

[10] S. B. K. Bhalla D. Koundal and M. Tahir, "Fusion of infrared and visible images using fuzzy based siamese convolutional network," *Computers, Materials Continua*, vol. 53, no. 1, pp. 191–199, 2020.

[11] M. R. G. Jader J. Fontineli, M. Pithon, and L. Oliveira, "Deep instance segmentation of teeth in pano- ramic x-ray images," *Proceedings of the 31st SIBGRAPI Conference*, vol. 69, no. 3, pp. 1334–1344, 2019.

[12] P. L. H. Chen K. Zhang, "A deep learning approach to automatic teeth detection and numbering based on object detection in dental periapical films," *IEEE Scientific Reports*, vol. 67, no. 9, pp. 3673–3682, 2019.

[13] M. M. B. D. V. Tuzoff L. N. Tuzova, "Tooth detection and numbering in panoramic radiographs using convolutional neural networks," *IEEE Transactions on Dentomaxillofacial Radiology*, vol. 50, no. 3, pp. 794–805, 2019.

[14] M. J, "Segmentation of opg images in studying jawbone diseases," *Progress in Electromagnetics Research Symposium Proceedings*, vol. 63, no. 1, pp. 84–88, 2016.

[15] P. C. Kaushik Akanksha and V. Sharma, "Edge detection and level set active contour model for the segmentation of cavity present in dental x- ray images," in *nternational Journal of Computer Applications*, 2015, pp. 461–464.

[16] Y. A. Yousif Mohamed, "Study of orthopantomograph(opg) images enhancement," in *International Conference on Radiation Machine (ICRM)*, 2016, pp. 266–269.