ABSTRACT

Deep learning has the potential to revolutionize medical imaging by significantly enhancing diagnostic accuracy, particularly in dentistry, where X-ray analysis is crucial but often subjective. This proposal outlines the development of a NASNet (Neural Architecture Search Network) deep learning model aimed at improving the accuracy of dental pathology detection from radiographic images. The current challenges of manual X-ray interpretation, such as human error, variability in image quality, and subjective diagnosis, underline the need for an automated, scalable solution.

Traditional methods of dental X-ray analysis suffer from inconsistency and are time-consuming, especially in resource-constrained environments with limited access to specialized dental professionals. Deep learning models, particularly Convolutional Neural Networks (CNNs), offer a more consistent and accurate approach to diagnostics. NASNet, by automating the configuration of its network architecture, optimizes both accuracy and computational efficiency, addressing many of the shortcomings of manual interpretation and conventional automated systems.

This project proposes the development of a NASNet-based model, trained on a large and diverse dataset of over 8000 radiographic images annotated for four dental pathologies: cavities, implants, fillings, and impacted teeth. Through data augmentation and fine-tuning of the NASNet architecture, this project seeks to overcome challenges related to image quality and classification error rates. The proposed model aims to be scalable and adaptable to diverse clinical environments, offering robust performance in real-world scenarios.

Upon successful implementation, this model has the potential to improve diagnostic support and streamline clinical workflows, providing faster, more accurate diagnoses in high-patient-volume settings and areas with limited specialist access. By addressing key limitations of current automated systems, this research proposal presents a forward-looking AI-driven solution that could significantly impact the future of dental diagnostics.

LIST OF TABLES

LIST OF FIGURES

CONTENTS

CONTENTS

1. INTRODUCTION

In the evolving landscape of healthcare, the integration of advanced technologies such as deep learning has emerged as a game-changer in enhancing diagnostic accuracy and efficiency. Nowhere is this transformation more impactful than in the field of dentistry, where the interpretation of radiographic images is crucial for identifying dental pathologies. The complexity of these images, coupled with the subjectivity inherent in manual analysis, highlights the urgent need for robust and reliable automated solutions.

As the volume of dental imaging data increases, traditional methods of diagnosis often struggle with inconsistencies and errors, leading to potential misdiagnoses and delayed treatments. This challenge calls for the development of intelligent systems that can accurately analyse and classify dental conditions, ultimately improving patient outcomes. Deep learning models, particularly Convolutional Neural Networks (CNNs), offer a powerful alternative, leveraging large datasets and advanced algorithms to deliver more consistent and objective results.

At the heart of this endeavour is the ambition to enhance diagnostic support through a dedicated deep learning-based classification system. By employing a Neural Architecture Search Network (NASNet), this project aims to address the limitations of current automated systems, focusing specifically on the detection of common dental pathologies such as cavities, fillings, implants, and impacted teeth. The proposed model will utilize a diverse dataset of over 8000 annotated radiographic images, ensuring that the system is not only accurate but also scalable and adaptable to various clinical environments.

Through this project, we seek to contribute to the ongoing advancements in dental diagnostics, providing a sophisticated tool that simplifies the interpretation of radiographic images. By enhancing user experience and improving diagnostic precision, this research aims to forge a meaningful connection between healthcare professionals and their patients, ultimately transforming the future of dental care.

2. SUPPORTING LITERATURE

2.1 Literature Review

Paper 1: Abdullah S. AL-Malaise AL-Ghamdi, Mahmoud Ragab, Saad Abdulla AlGhamdi, Amer H. Asseri, Romany F. Mansour, Deepika Koundal "*Detection of Dental Diseases through X-Ray Images Using Neural Search Architecture Network*" Proceedings of the Fifth International Conference on Communication and ElectronicsS Systems (ICCES 2020)

In the quest to advance diagnostic capabilities within the field of dental health, researchers Abdullah S. AL-Malaise AL-Ghamdi, Mahmoud Ragab, Saad Abdulla AlGhamdi, Amer H. Asseri, Romany F. Mansour, and Deepika Koundal have explored the potential of deep learning technologies. Their study, titled "Detection of Dental Diseases through X-Ray Images Using Neural Search Architecture Network," was presented at the Fifth International Conference on Communication and Electronics Systems (ICCES 2020). This research investigates the efficacy of Convolutional Neural Networks (CNNs), specifically employing the NAS Net architecture, to enhance the diagnostic accuracy of dental diseases through the analysis of X-ray images. The following table provides a concise overview of the key aspects of their study, including the area of work, dataset specifics, methodology, algorithm, results, advantages, limitations, and future proposals.

The study is grounded in the necessity for improved diagnostic tools in dentistry, where traditional methods often fall short due to human error and variability in image interpretation. By employing a robust dataset composed of diverse radiographic images, the researchers implemented the NASNet architecture to automate and refine the classification process. The methodology focused on training the model to recognize patterns indicative of specific dental conditions, effectively reducing the diagnostic burden on dental professionals and enabling faster and more accurate assessments.

The results of their study demonstrated promising improvements in diagnostic performance, showcasing the advantages of using deep learning frameworks over conventional approaches. Key findings highlighted the potential for increased accuracy and efficiency in diagnosing dental diseases through automated systems. However, the research also acknowledged limitations such as the need for larger and more varied datasets to enhance model robustness. Looking ahead, the researchers propose further exploration into hybrid models and the integration of additional imaging modalities to augment diagnostic capabilities, ultimately contributing to the future of dental healthcare technology.

Paper 2: L. Megalan Leo and T. Kalapalatha Reddy, "*Learning compact and discriminative hybrid neural network for dental caries classification*," Microprocessors and Microsystems, vol. 82, Article ID 103836, 2021

The paper by L. Megalan Leo and T. Kalapalatha Reddy, titled "Learning compact and discriminative hybrid neural network for dental caries classification," explores an advanced technique for early detection and classification of dental caries. The study aims to improve diagnostic accuracy and efficiency in dentistry using innovative neural network methods. It utilizes a dataset of 80 images, focusing on preprocessing, segmentation, and feature extraction to enhance detection. The proposed Hybrid Neural Network (HNN) demonstrates a high accuracy of 96.0%, offering advantages over traditional methods but is limited by the small dataset size.

The methodology of the study involves a comprehensive analysis of a dataset comprising 80 images, which serves as the foundation for training and validating the proposed model. The researchers focus on several critical steps, including preprocessing, segmentation, and feature extraction, to optimize the detection process. By refining these elements, the HNN is able to discern subtle patterns indicative of dental caries, thus enhancing its overall performance. The implementation of this approach highlights the potential for neural networks to streamline diagnostic processes and reduce the reliance on traditional, manual methods.

This research not only demonstrates the efficacy of the Hybrid Neural Network (HNN) in detecting dental caries but also opens the door to further innovations in the realm of dental diagnostics. By leveraging the strengths of hybrid models, future studies could explore the integration of additional features, such as patient demographics and clinical history, to create a more comprehensive diagnostic tool. Furthermore, the findings suggest the potential for real-time applications in clinical settings, where practitioners could utilize such advanced systems to assist in making informed decisions during patient consultations. The successful implementation of the HNN model could lead to the development of user-friendly software that integrates seamlessly into existing dental practices, ultimately enhancing the quality of care provided to patients. As the field continues to evolve, ongoing research will be essential in refining these models and ensuring that they can adapt to the dynamic challenges faced in dental health.

The results of the study reveal that the proposed HNN achieves an impressive accuracy of 96.0% in classifying dental caries, demonstrating significant advantages over conventional diagnostic techniques. However, the authors acknowledge the limitation posed by the small size of the dataset, which may affect the model's generalizability and robustness in real-world applications. Moving forward, the study suggests the need for larger and more diverse datasets to further validate the effectiveness of the proposed methodology, paving the way for future advancements in dental diagnostic technologies.

Paper 3: JH Lee, DH Kim, SN Jeong, SH Choi, "*Detection and diagnosis of dental caries using a deep learning-based convolutional neural network algorithm*" Journal of dentistry, 2018.

The paper by JH Lee et al., titled "Detection and Diagnosis of Dental Caries Using a Deep Learning-Based Convolutional Neural Network Algorithm," presents a comprehensive investigation into the application of Convolutional Neural Networks (CNNs) for the detection and diagnosis of dental caries. The study emphasizes the critical role that advanced machine learning techniques play in enhancing diagnostic capabilities in dentistry, particularly given the complexities associated with visual interpretation of dental radiographs. By leveraging CNNs, this research aims to automate and improve the accuracy of caries detection, addressing the limitations of traditional diagnostic methods.

Utilizing an extensive dataset of 3,000 periapical radiographic images, the researchers adopt a robust methodology that incorporates a pre-trained GoogleNet Inception v3 CNN. This approach allows for effective preprocessing and transfer learning, maximizing the performance of the model on dental imagery. The utilization of transfer learning is particularly advantageous as it enables the model to harness pre-existing knowledge from related tasks, thereby improving its ability to accurately classify images even with a limited number of dental-specific training examples.

The results of the study indicate a high level of accuracy in detecting dental caries, showcasing the CNN's efficiency and effectiveness in handling complex radiographic data. By demonstrating the model's capability to identify subtle features indicative of dental caries, the research contributes to the growing body of evidence supporting the integration of deep learning technologies in clinical dental practice. This progress represents a significant step forward in automating diagnostic processes, potentially leading to quicker and more reliable assessments for dental professionals.

However, the authors also acknowledge several challenges that must be addressed to enhance the model's utility in real-world settings. Factors such as anatomical variations among patients can impact diagnostic accuracy, highlighting the need for comprehensive validation with more diverse datasets. As the study suggests, future research should focus on expanding the dataset to include a wider range of anatomical variations and conditions, ultimately leading to more robust and generalizable models for dental caries detection.

2.1.1 Summary Table

	TITLE	DATASET	ALGORITHM	ACCURACY
PAPER 1	Abdullah S. AL-Malaise AL-Ghamdi, Mahmoud Ragab, Saad Abdulla AlGhamdi, Amer H. Asseri, Romany F. Mansour, Deepika Koundal ”Detection of Dental Diseases through X-Ray Images Using Neural Search Architecture Network” Proceedings of the Fifth International Conference on Communication and ElectronicsS Systems (ICCES 2020)	The dataset, from 116 patients at Noor Medical Imaging Centre, Qom, Iran, includes various dental conditions, from healthy to edentulous. Data augmentation increased the images from 83 to 245. The classification focused on three classes: cavity, filling, and implant.	The primary algorithm used is a NAS Net- based CNN.	96.0%
PAPER 2	L. Megalan Leo and T. Kalapalatha Reddy, “Learning compact and discriminative hybrid neural network for dental caries classification,” Microprocessors and Microsystems, vol. 82, Article ID 103836, 2021	The dataset, collected from the work by Ching et al. [18] and available on the website wwwo.ntust.edu.tw/~cweiwang, comprises 80 images	The core algorithm employed in this study is the Hybrid Neural Network (HNN), which combines the strengths of both ANN and DNN.	96.0%
PAPER 3	JH Lee, DH Kim, SN Jeong, SH Choi, “Detection and diagnosis of dental caries using a deep learning-based convolutional neural network algorithm” Journal of dentistry, 2018	A total of 3000 periapical radiographic images were used, divided into training and validation datasets (2400 images, 80%) and a test dataset (600 images, 20%).	The paper employs the Google Net Inception v3 convolutional neural network (CNN) architecture, known for its performance in image detection, classification, and segmentation.	82.0%

Table 2.1 Reference papers summary

2.2. Findings and Proposals

After conducting a thorough review of three significant studies in the field of deep learning applications for dental diagnostics, it becomes clear that each contributes valuable insights into the potential of advanced neural network techniques. Among these, the development of a NASNet-based model for detecting dental pathologies stands out as a promising approach. The choice of utilizing NASNet, a model optimized through Neural Architecture Search, aligns well with the need for high accuracy in diagnosing conditions such as cavities, fillings, and impacted teeth. This advanced architecture not only enhances diagnostic precision but also addresses the variability and inconsistencies often encountered in manual interpretations of radiographic images.

The first paper by Abdullah S. AL-Malaise AL-Ghamdi et al. presents a NASNet-based framework that effectively enhances diagnostic accuracy through the analysis of X-ray images. Their research demonstrates a high accuracy rate of 96.0%, highlighting the effectiveness of deep learning models in clinical settings. The second study by L. Megalan Leo and T. Kalapalatha Reddy introduces a Hybrid Neural Network for early dental caries classification, achieving similar accuracy while emphasizing the importance of preprocessing and feature extraction. Lastly, the work of JH Lee et al. showcases the application of a pre-trained GoogleNet Inception v3 model, achieving impressive results with a larger dataset of 3,000 images, although it acknowledges challenges related to anatomical diversity in dental radiographs.

The collective findings from these studies underscore the importance of utilizing deep learning technologies to enhance the reliability and efficiency of dental diagnostics. By focusing on the development of models that adapt to individual patient characteristics and incorporate diverse datasets, these approaches not only improve diagnostic outcomes but also facilitate better clinical decision-making. Ultimately, the integration of these advanced methodologies into everyday dental practice has the potential to revolutionize the field, providing a pathway for quicker and more accurate diagnoses that can significantly enhance patient care.

In conclusion, the exploration of deep learning models, particularly NASNet and other neural network architectures, reveals a transformative potential in the field of dental diagnostics. The reviewed studies collectively demonstrate that these advanced techniques can significantly improve the accuracy and efficiency of detecting various dental pathologies from radiographic images. By leveraging large and diverse datasets, these models address the limitations of traditional diagnostic methods, such as subjectivity and variability in interpretation. As we advance toward implementing these systems in clinical settings, the emphasis on scalability and adaptability will be crucial for ensuring that these technological innovations can be effectively integrated into existing workflows. Ultimately, this research not only contributes to the body of knowledge in dental health diagnostics but also paves the way for future innovations that can enhance patient outcomes and streamline clinical processes. The continued development and validation of these models hold the promise of elevating the standard of care in dentistry, making accurate and timely diagnoses more accessible than ever before.

.

3. SYSTEM ANALYSIS

3.1. Analysis of Dataset

3.1.1. About the Dataset

The Dental Radiograph Dataset is an essential resource for advancing research in deep learning applications for dental diagnostics. This dataset consists of annotated radiographic images that serve as the foundation for training and evaluating deep learning models aimed at classifying dental pathologies.

Dataset Link: https://www.kaggle.com/datasets/imtkaggleteam/dental-radiography

Size: The dataset comprises over 8,000 radiographic images, meticulously labeled for four distinct dental pathologies: cavities, implants, fillings, and impacted teeth. This substantial size provides a diverse range of examples for training robust models.

Data Structure: The images are organized into three main folders: train, test, and valid, with each folder containing images along with corresponding annotation files that list image names and their respective class labels. This structured approach facilitates effective data handling and model training.

Sparsity: While the dataset is extensive, it is important to note that the presence of varying image quality and the complexity of dental conditions can introduce challenges in classification. The dataset's diversity in terms of image conditions and dental pathologies allows for the development of a model capable of generalizing across different scenarios.

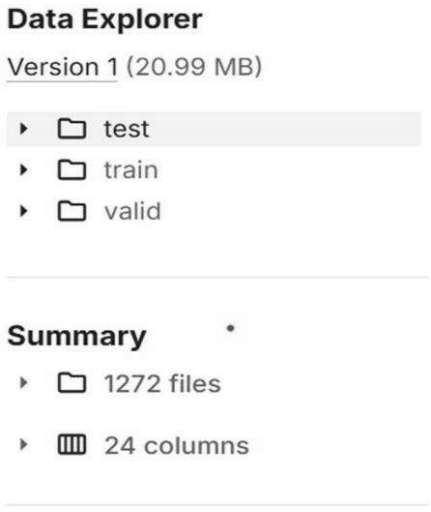


Fig 3.1 Snapshot of folder structure

3.1.2. Explore the Dataset

The chart illustrates the class distribution in a dental image dataset, revealing a significant imbalance among the categories. The *Fillings* class is overwhelmingly dominant, containing over 5000 images, while the other classes—*Implant*, *Cavity*, and *Impacted Tooth*—have considerably fewer samples. This discrepancy indicates that the dataset is heavily skewed towards *Fillings*, which could lead to a model that is biased toward predicting this class over others. In particular, *Cavity* and *Impacted Tooth* are the least represented, posing a challenge for the model to learn effectively from these limited samples. To address this, data balancing techniques, such as data augmentation or resampling, may be necessary to ensure better performance and fairness across all categories in the dataset.

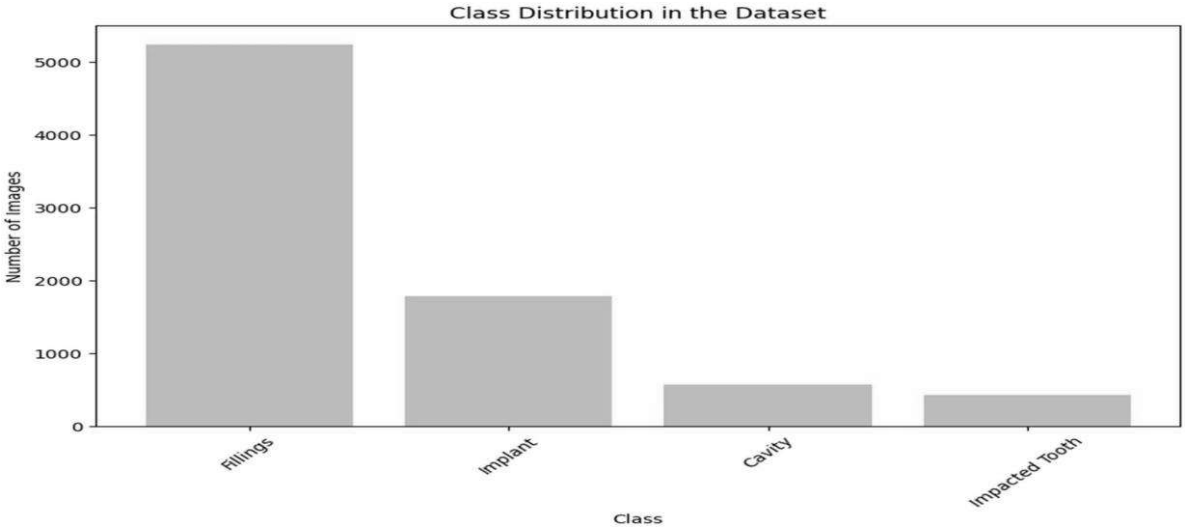


Fig 3.2 Bar plot shows the class imbalance

The histogram reveals that the grayscale image is heavily skewed towards high pixel intensities, indicating a predominantly bright image with limited contrast and few dark areas.

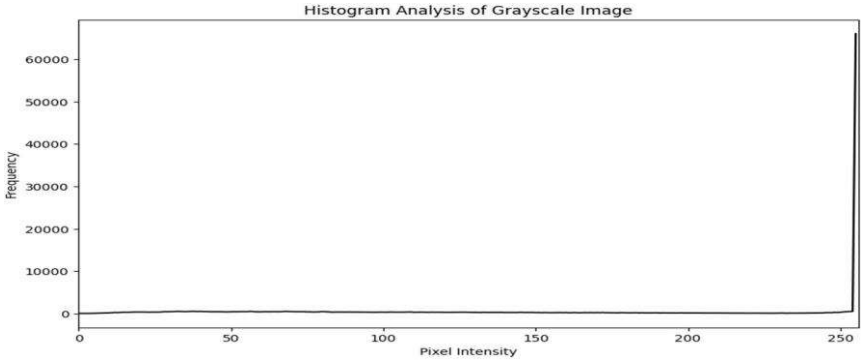


Fig 3.3 Histogram analysis of image

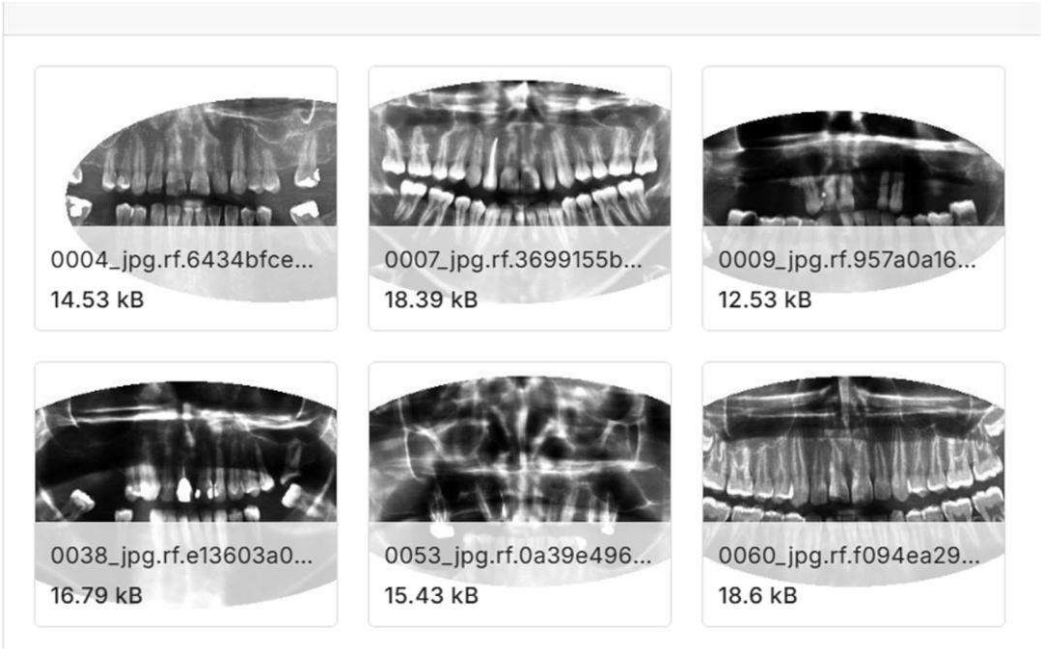


Fig 3.4 Snap shot of sample image

filename	width	height	class	xmin	ymin	xmax	ymax
0136_jpg.r	512	256	Implant	195	169	209	212
0136_jpg.r	512	256	Implant	288	171	301	205
0136_jpg.r	512	256	Implant	203	107	217	150
0136_jpg.r	512	256	Implant	287	104	302	150
0136_jpg.r	512	256	Implant	301	103	312	150

Fig 3.5 Snapshot of _anotation.csv

3.2 Data Preprocessing

For the dental pathology classification project, the dataset underwent thorough preprocessing and augmentation to prepare it for effective training using the NASNet Large model. The dataset comprised over 8000 dental radiographic images, which were initially labeled into four categories: cavity, fillings, impacted tooth, and implant. Preprocessing steps included standardizing the image resolution to 331x331 pixels, ensuring uniformity across the dataset and compatibility with the model architecture. To maintain aspect ratios and reduce image distortion, a "fit" method was employed, adding black padding as needed to meet the specified dimensions.

Augmentation techniques were applied to increase dataset diversity and simulate real-world conditions. Each image was augmented to create additional variations by performing random transformations such as cropping between 0% and 20% of the image area, rotating the image within a range of -10° to +10°, and adjusting brightness by ±10%. Additional augmentations included horizontal and vertical shearing by up to 5° and applying a random Gaussian blur with a range of 0 to 0.5 pixels.

The balanced representation of all four classes was maintained throughout preprocessing and augmentation to avoid any bias towards a particular category. This was achieved by applying an equal number of augmentations to images from each class and validating that the augmented dataset maintained a similar distribution as the original. By preparing the dataset with these methods, the goal was to create a robust training environment that could capture subtle differences between dental pathologies, ultimately leading to higher classification accuracy and a more reliable model for clinical applications.

```
import os
import cv2
import random
import imgaug.augmenters as iaa

# Define the path to the folder containing your X-ray images
folder_path = r"C:\Users\ASUS\OneDrive\Documents\Desktop\newdataset\segmented_image_test\Implant"

# Define augmentation sequence
augmentation_sequence = iaa.Sequential([
    iaa.Fliplr(0.5),  # Flip horizontally with a probability of 50%
    iaa.Rotate((-10, 10)),  # Rotate between -10 and 10 degrees
    iaa.Multiply((0.8, 1.2)),  # Adjust brightness between 80% and 120%
    iaa.GaussianBlur(sigma=(0.0, 1.0)),  # Apply Gaussian blur with random sigma
    iaa.AdditiveGaussianNoise(scale=(0, 0.05*255))  # Add Gaussian noise
])

# Get all image file names in the folder
image_files = [f for f in os.listdir(folder_path) if f.lower().endswith(('.png', '.jpg', '.jpeg'))]
num_images = len(image_files)

# Augment the images to reach a total of 1,000
current_count = num_images

# Augment only if the current count is less than 1,000
if current_count < 1000:
    # Keep augmenting until there are exactly 1,000 images
    while current_count < 100:
        # Randomly select an existing image
        image_file = random.choice(image_files)
        image_path = os.path.join(folder_path, image_file)

        # Read the image
        image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)


        # Apply augmentation
        augmented_image = augmentation_sequence(image=image)

        # Save the augmented image
        new_image_name = f"aug_{current_count}_{image_file}"
        new_image_path = os.path.join(folder_path, new_image_name)
        cv2.imwrite(new_image_path, augmented_image)


        # Update the count
        current_count += 1

    print(f"Augmentation completed. The folder now contains exactly {current_count} images.")
else:
    print("No augmentation needed. The folder already contains 1,000 or more images.")
```

```
Augmentation completed. The folder now contains exactly 1000 images.
```

Fig 3.6 Image of augmentation applied on dataset

```
import os
import pandas as pd
import cv2
import numpy as np
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Paths to the dataset
csv_path = r'C:\Users\ASUS\Documents\archive (5)\valid\_annotations.csv'  # Correct CSV file path
image_folder = r'C:\Users\ASUS\Documents\archive (5)\valid'
output_folder = r'C:\Users\ASUS\Documents\archive (5)\valid\segmented_image'

# Load the CSV file
df = pd.read_csv(csv_path)

# Create output directories for each class if not already present
classes = ['Implant', 'Fillings', 'Impacted Tooth', 'Cavity']
for cls in classes:
    os.makedirs(os.path.join(output_folder, cls), exist_ok=True)

# Preprocessing: Resize, Normalize, and Enhance clarity
image_size = (331, 331)  # NASNet Large input size
mean_pixel = [123.68, 116.779, 103.939]  # Mean pixel values for ImageNet

# Function to increase image clarity by sharpening and enhancing contrast
def enhance_image(img):
    # Sharpening Kernel (Laplacian filter)
    kernel = np.array([[0, -1, 0], [-1, 5,-1], [0, -1, 0]])  # Simple sharpening kernel
    img_sharpened = cv2.filter2D(img, -1, kernel)  # Apply the sharpening filter

    # Convert to YUV color space for contrast adjustment
    img_yuv = cv2.cvtColor(img_sharpened, cv2.COLOR_BGR2YUV)

    # Histogram equalization on the Y channel to improve contrast
    img_yuv[:, :, 0] = cv2.equalizeHist(img_yuv[:, :, 0])  # Equalize the brightness (Y channel)
```

```
    # Convert to YUV color space for contrast adjustment
    img_yuv = cv2.cvtColor(img_sharpened, cv2.COLOR_BGR2YUV)

    # Histogram equalization on the Y channel to improve contrast
    img_yuv[:, :, 0] = cv2.equalizeHist(img_yuv[:, :, 0])  # Equalize the brightness (Y channel)

    # Convert back to BGR color space
    img_contrast_enhanced = cv2.cvtColor(img_yuv, cv2.COLOR_YUV2BGR)

    return img_contrast_enhanced

# Loop over the CSV file and extract image segments
for index, row in df.iterrows():
    img_path = os.path.join(image_folder, row['filename'])
```

```
    # Ensure the image path exists
    if not os.path.exists(img_path):
        print(f"Error: Image {row['filename']} does not exist.")
        continue

    img = cv2.imread(img_path)

    if img is None:
        print(f"Error: Could not load image {row['filename']}")
        continue

    # Get the bounding box coordinates
    xmin, ymin, xmax, ymax = int(row['xmin']), int(row['ymin']), int(row['xmax']), int(row['ymax'])
```

```
    # Enhance the image clarity (sharpen and adjust contrast)
    img_enhanced = enhance_image(img_cropped)

    # Resize the image to the input size required by NASNet Large (331x331)
    img_resized = cv2.resize(img_enhanced, image_size)

    # Normalize the image (subtract the mean pixel values)
    img_normalized = img_resized.astype(np.float32)
    img_normalized -= mean_pixel  # Normalize to ImageNet standard

    # Get the class name and clean any extra spaces
    class_name = row['class'].strip()
```

```
    # Ensure the class name is one of the expected classes
    if class_name not in classes:
        print(f"Warning: Invalid or unexpected class name '{class_name}' in row {index}. Skipping this row.")
        continue

    # Safe filename creation, ensuring no problematic characters
    safe_filename = f"{index}_{os.path.basename(row['filename'])}"
    output_path = os.path.join(output_folder, class_name, safe_filename)

    # Save the preprocessed image to the class folder
    cv2.imwrite(output_path, img_normalized)

    print(f"Saved preprocessed image to {output_path}")

# Use ImageDataGenerator for further augmentation and preparing data for NASNet
train_datagen = ImageDataGenerator(
    rescale=1.0 / 255.0,  # Normalize images to [0,1]
    rotation_range=20,    # Random rotation
    width_shift_range=0.2, # Horizontal shift
```

```
    # Ensure the bounding box coordinates are valid (within image bounds)
    h, w = img.shape[:2]
    xmin = max(0, xmin)
    ymin = max(0, ymin)
    xmax = min(w, xmax)
    ymax = min(h, ymax)

    # Crop the image based on bounding box
    img_cropped = img[ymin:ymax, xmin:xmax]

    # Ensure the cropped image has valid dimensions (non-zero size)
    if img_cropped.shape[0] == 0 or img_cropped.shape[1] == 0:
        print(f"Error: Invalid bounding box for {row['filename']} (row {index}). Skipping this row.")
        continue

    # Enhance the image clarity (sharpen and adjust contrast)
    img_enhanced = enhance_image(img_cropped)

    # Resize the image to the input size required by NASNet Large (331x331)
    img_resized = cv2.resize(img_enhanced, image_size)

    # Normalize the image (subtract the mean pixel values)
    img_normalized = img_resized.astype(np.float32)
    img_normalized -= mean_pixel  # Normalize to ImageNet standard

    # Get the class name and clean any extra spaces
    class_name = row['class'].strip()
```

```
    # Ensure the class name is one of the expected classes
    if class_name not in classes:
        print(f"Warning: Invalid or unexpected class name '{class_name}' in row {index}. Skipping this row.")
        continue
```

```
    height_shift_range=0.2, # Vertical shift
    shear_range=0.2,      # Shear transformation
    zoom_range=0.2,       # Zoom
    horizontal_flip=True, # Flip images horizontally
    fill_mode='nearest'  # Fill missing pixels after transformations
)

# Set up flow from directory for training
train_generator = train_datagen.flow_from_directory(
    os.path.join(output_folder, 'train'),  # Directory path for training data
    target_size=image_size,               # Resize images to 331x331
    batch_size=32,                        # Batch size for training
    class_mode='categorical',             # Multi-class classification
    shuffle=True                          # Shuffle the data
)

# Optionally, set up validation and test generators similarly
validation_datagen = ImageDataGenerator(rescale=1.0 / 255.0)

validation_generator = validation_datagen.flow_from_directory(
    os.path.join(output_folder, 'validation'),  # Directory path for validation data
    target_size=image_size,                    # Resize images to 331x331
    batch_size=32,                             # Batch size for validation
    class_mode='categorical',                  # Multi-class classification
```

```
    shuffle=False                              # No shuffling for validation
)
```

Fig 3.7 Image shows the exact code used for data enhancing and duplication

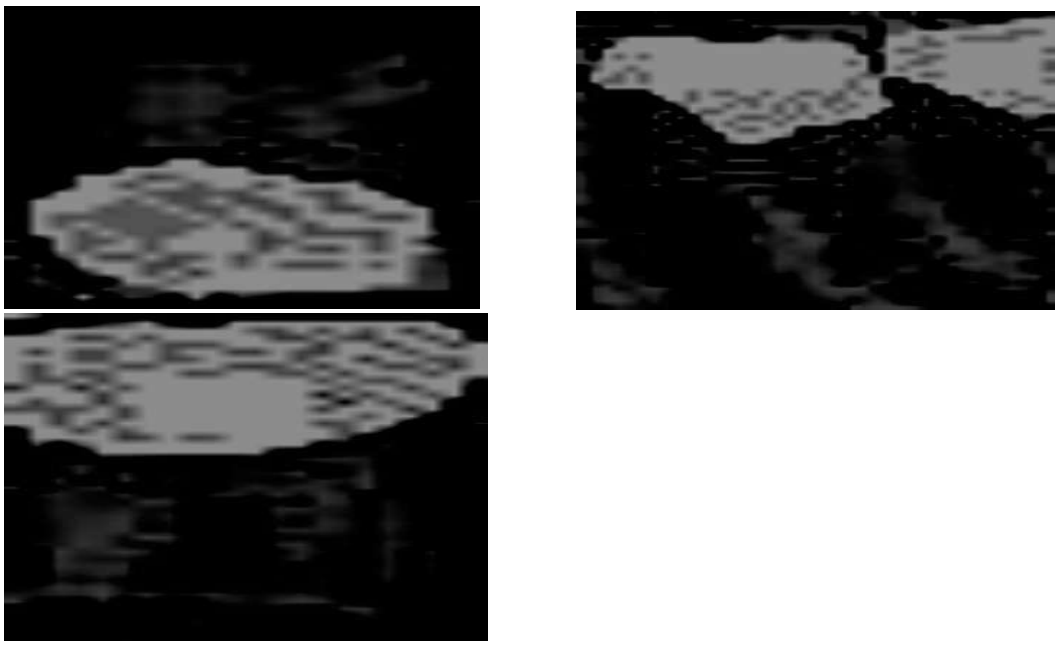


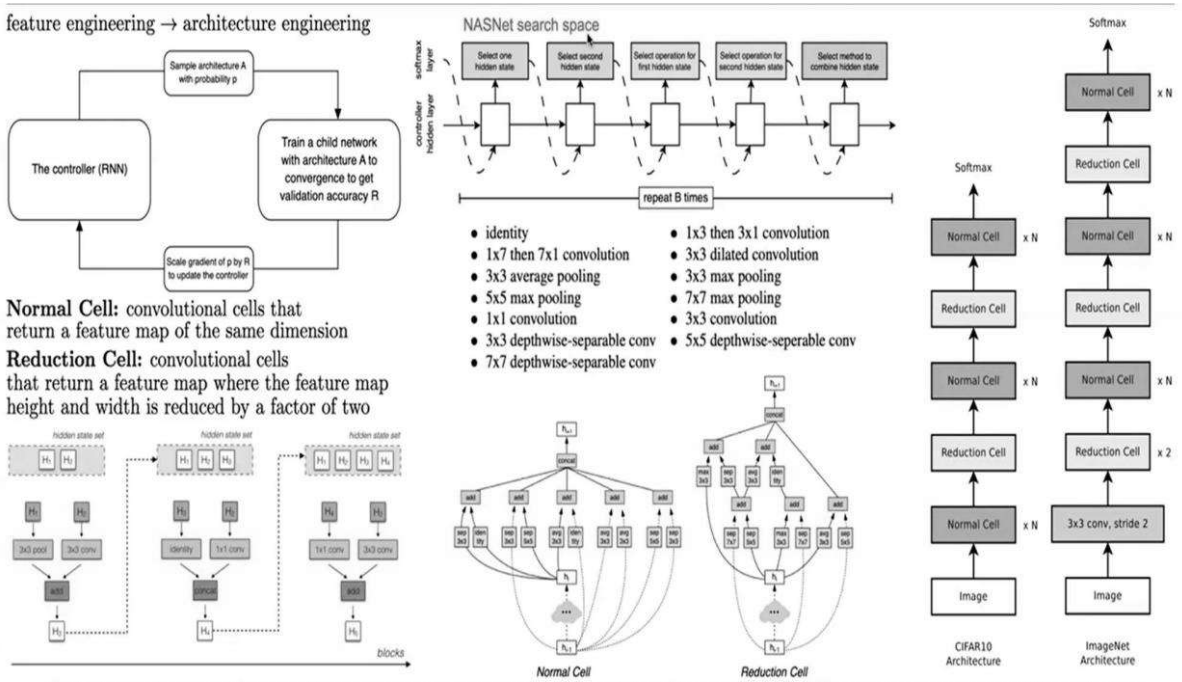
Fig 3.8 Sample images after all preprocessing tasks

3.3 Analysis of Architecture

NASNet, or Neural Architecture Search Network, is a state-of-the-art deep learning architecture specifically designed for image classification tasks. It employs a unique search strategy to automatically identify optimal architectures, thereby enhancing the model's performance. The architecture is composed of multiple blocks, including normal cells and reduction cells, which allow for adaptive scaling of the network's capacity based on the complexity of the task. The architecture is optimized for various input sizes, making it versatile for different imaging datasets, including dental radiographs.

Stage	Layer Type	Output Dimension	Layers in Each Cell	Description
Input	Input Layer	(331, 331, 3)	-	Input image with size 331x331 and 3 color channels (RGB).
Stem Cell	Convolution + Pooling	(165, 165, 96)	18	Initial convolutional layer followed by pooling.
Cell 1 (Reduction Cell)	NASNet Reduction Cell	(83, 83, 168)	12	First reduction cell, reducing spatial dimensions by half.
Cells 2-6	NASNet Normal Cells	(83, 83, 168)	5 cells × 12 layers = 60	Five normal cells, keeping the same spatial dimensions.
Cell 7 (Reduction Cell)	NASNet Reduction Cell	(42, 42, 336)	12	Second reduction cell, further reducing spatial dimensions.
Cells 8-12	NASNet Normal Cells	(42, 42, 336)	5 cells × 12 layers = 60	Five normal cells at this resolution.
Cell 13 (Reduction Cell)	NASNet Reduction Cell	(21, 21, 672)	12	Third reduction cell, reducing the spatial resolution.
Cells 14-18	NASNet Normal Cells	(21, 21, 672)	5 cells × 12 layers = 60	Five normal cells maintaining the same dimensions.
Cell 19 (Reduction Cell)	NASNet Reduction Cell	(11, 11, 1344)	12	Fourth reduction cell, reducing the spatial resolution.
Cells 20-24	NASNet Normal Cells	(11, 11, 1344)	5 cells × 12 layers = 60	Five normal cells maintaining the same dimensions.
Global Pooling	Global Average Pooling	(1344)	1	Global average pooling across spatial dimensions.
Output	Fully Connected (Dense)	(Number of classes)	1	Produces the final classification output probabilities.

Fig 3.9 Dimension table of NASNet Large



Zoph, Barret, et al. "Learning transferable architectures for scalable image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.

Fig 3.10 General figure of NASnet architecture

This image illustrates the NASNet architecture search space, the process of neural architecture search using a reinforcement learning (RL) approach, and the design of Normal and Reduction cells.

1. **NASNet Search Process:**
    - The controller (an RNN) generates neural network architectures by sampling different configurations with probability p.
    - It trains a "child network" with the sampled architecture on a dataset to convergence, measuring the validation accuracy R.
    - The gradient of p is scaled by R to update the controller, optimizing the search.
2. **NASNet Search Space:**
    - The architecture is defined as a sequence of steps to choose hidden states and operations, with various convolution and pooling options.
    - The operations include identity, convolutions (1x1, 3x3, depthwise-separable), dilated convolutions, max pooling, and average pooling.
    - The search repeats B times to design a cell.

3. **Cells in NASNet:**
    - **Normal Cell:** Outputs a feature map of the same dimensions as the input.
    - **Reduction Cell:** Reduces the feature map dimensions by half, effectively performing down sampling.
    - Both cells consist of a series of building blocks, each combining different hidden states using operations.
4. **NASNet Architecture for Datasets:**
    - For CIFAR-10: Uses Normal and Reduction cells in multiple stages, with stacked Normal cells followed by Reduction cells.
    - For ImageNet: Begins with a 3x3 convolution, followed by several stacked Normal cells, with Reduction cells used to down sample.
5. **The Block Design:**
    - In the cells, different operations (convolutions, pooling) are combined, sometimes followed by concatenation or addition of outputs, allowing for diverse and complex feature extraction.

This architecture enables efficient and scalable image recognition by searching for optimal network structures within this defined space.

Steps for Image Classification Using NASNet

1. **Input Preprocessing**: Radiographic images are resized to a standard input dimension of 224x224 pixels before being fed into the NASNet model. This resizing ensures that the input conforms to the expected format and reduces computational overhead.
2. **Model Inference**: The preprocessed images are then passed through the NASNet architecture, where the network's convolutional layers extract hierarchical features from the images. The model processes the images in batches, allowing for efficient utilization of hardware resources.
3. **Output Predictions**: The model generates a list of predictions, each representing the probability of the image belonging to a particular dental pathology class, such as cavity, filling, implant, or impacted tooth. Each output prediction is accompanied by a confidence score, indicating the likelihood that the prediction is correct.

4. **Post-Processing**: After obtaining the raw predictions, a threshold is applied to filter out low-confidence predictions. The results are further processed to map the predicted classes back to the original images, allowing for visual verification of the model's classifications.

Architecture and Hyperparameters

- **Base Model**: NASNet employs an intricate arrangement of convolutional layers with various kernel sizes, batch normalization, and activation functions to enhance feature extraction capabilities. The architecture consists of multiple convolutional blocks that include skip connections, similar to residual networks, which help preserve information and combat the vanishing gradient problem.

- **Hyperparameters**: Key hyperparameters include learning rate, batch size, and the number of epochs. A learning rate of 0.0001 is typically used to ensure stable convergence during training. The batch size can be adjusted based on available memory resources, and the number of epochs is chosen based on validation performance to prevent overfitting.

Speed and Accuracy

The performance of NASNet in the context of dental pathology classification is characterized by a balance between speed and accuracy. While NASNet models can achieve high accuracy rates, their computational demands can vary significantly based on the complexity of the architecture and the size of the input images. Generally, NASNet can process images at a rate of several frames per second (FPS) on modern GPUs, facilitating real-time analysis in clinical settings. This efficiency is critical in dental diagnostics, where timely and accurate classifications can significantly enhance clinical workflows and patient outcomes. The deployment of NASNet for classifying dental pathologies not only optimizes diagnostic processes but also aids in the early detection of conditions that may require immediate intervention. By leveraging the capabilities of NASNet, this project aims to improve the accuracy and efficiency of dental pathology detection from radiographic images, contributing to enhanced clinical decision-making and patient care.

3.4 PROJECT PLAN

3.4.1 Project Pipeline

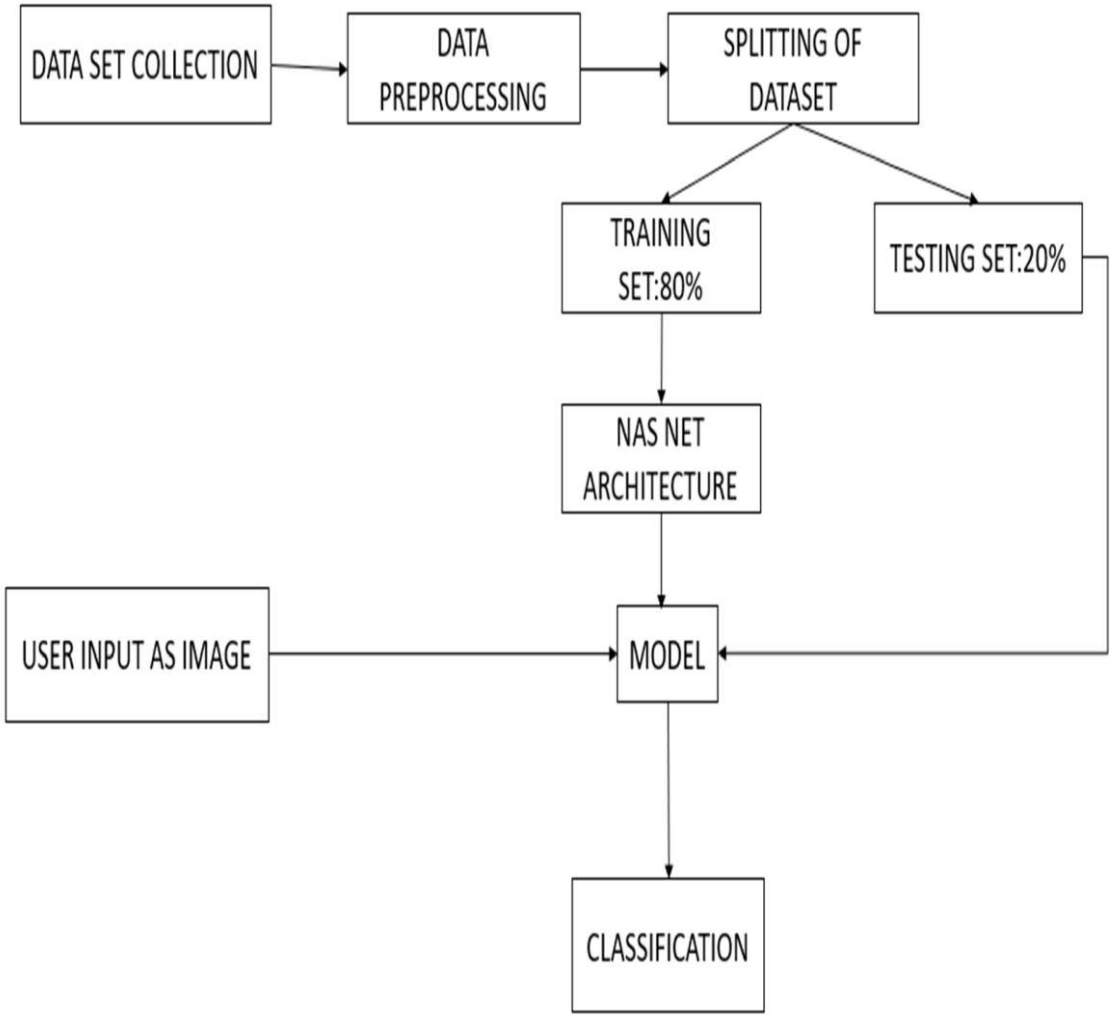


Fig 3.11 Pipeline diagram of the project

The flowchart represents the workflow for building a deep learning model using NASNet for image classification. Here's a step-by-step explanation of the process:

1. **Data Collection**: The first step involves gathering a dataset relevant to the problem being solved. In this case, the dataset consists of images that need to be classified into various categories. Proper data collection ensures that there are enough images representing each class, which is essential for training a robust model.

2. **Data Preprocessing**: Once the dataset is collected, preprocessing is performed to prepare the data for training. This involves operations such as resizing images to a uniform dimension, normalizing pixel values, and augmenting the images to increase the dataset's diversity. Augmentation techniques like rotation, scaling, and flipping help improve the model's ability to generalize.

3. **Splitting of Dataset**: After preprocessing, the dataset is split into two subsets: a **training set** (80%) and a **testing set** (20%). The training set is used to train the model, while the testing set is used to evaluate the model's performance. The split ensures that the model is evaluated on data it hasn't seen before, providing an unbiased estimate of its accuracy.

4. **NASNet Architecture**: The model architecture used here is NASNet (Neural Architecture Search Network), which automates the design of deep learning models. NASNet uses reinforcement learning to explore various architectures and selects the one that optimizes performance. It consists of two types of cells: **normal cells**, which maintain the feature map dimensions, and **reduction cells**, which downsample the feature maps by reducing the height and width by a factor of two.

5. **Model Training**: The training process involves feeding the training set to the NASNet model. The model learns to recognize patterns in the images by minimizing the error through backpropagation and adjusting its weights. The architecture is optimized iteratively based on feedback from the controller (RNN in NASNet's design), which selects the optimal structure for different layers.

6. **User Input as Image**: Once the model is trained, it can be used to classify new images provided by users. The user's input image undergoes the same preprocessing steps before being fed into the model.

7. **Classification**: Finally, the trained model predicts the class of the input image. The output may include the predicted class label and associated probabilities, indicating the model's confidence in its prediction.

This structured workflow is aimed at improving the classification accuracy while ensuring scalability and adaptability of the model to different image classification tasks.

3.5 Feasibility Analysis

This feasibility analysis explores the viability of implementing a NASNet-based deep learning system for the classification of dental pathologies from radiographic images. The analysis considers technical, economic, and operational factors.

3.5.1 Technical Feasibility

System Requirements and Model Suitability:

- NASNet Architecture: The NASNet (Neural Architecture Search Network) architecture is well-suited for the proposed project, given its strong performance in image classification tasks. It is designed to optimize both accuracy and computational cost by using a search algorithm to find the best neural network architecture. The model's adaptability for transfer learning allows for fine-tuning on a dental radiographic dataset, making it effective in classifying different pathologies such as cavities, implants, impacted teeth, and fillings.

- Image Processing Capabilities:

    - Python Libraries: The availability of open-source Python libraries like TensorFlow and Keras for building deep learning models provides a robust framework for model development. Other libraries, such as OpenCV for image processing, NumPy for numerical operations, and Pandas for data handling, support different stages of the data preparation and model training process.

    - GPU Acceleration: The training of deep learning models can be accelerated using GPUs. Google Colab provides access to free GPU resources, such as NVIDIA Tesla K80 or T4, which are suitable for training deep learning models without needing to invest in high-end hardware.

- Data Preparation: The dataset consists of more than 8,000 dental radiographic images, which have been annotated with four classes. Preprocessing techniques, such as image normalization and augmentation, will be applied to enhance model generalization. The structure of the dataset (split into training, testing, and validation sets) ensures that the model is trained effectively while avoiding overfitting.

Development Tools and Environment:

- Integrated Development Environment (IDE): Tools such as Visual Studio Code and Jupyter Notebooks on Google Colab facilitate code development and experimentation.

- Libraries and Frameworks: TensorFlow and Keras for deep learning, along with Flask for building a web-based application, offer a complete toolset for implementing the proposed solution.

- Model Training Considerations: Training a NASNet model can be computationally intensive. However, using transfer learning with a pre-trained NASNet model significantly reduces the training time by initializing the weights with learned features from a large dataset like ImageNet, making it feasible on hardware like a laptop with a 16 GB RAM and an i5 processor.

3.5.2 Economic Feasibility

Cost Analysis:

- Software Costs: The project primarily utilizes open-source libraries and tools (TensorFlow, Keras, Flask), which do not require any licensing fees. This minimizes software-related expenses.

- Hardware Costs:

    - Current Hardware Sufficiency: The existing laptop configuration (16 GB RAM, 12th Gen Intel i5 processor) is adequate for initial development and smaller-scale training. More computationally intensive tasks can be offloaded to cloud-based platforms like Google Colab, which offer free and paid plans for GPU usage.

    - Potential Upgrade Costs: If local hardware limitations are encountered, investment in an external GPU (e.g., NVIDIA RTX series) or a high-performance desktop/server setup may be considered. However, using cloud resources would likely be more cost-effective for this project.

- Cloud Services: For scaling up, cloud services like Google Cloud Platform or Amazon Web Services (AWS) could be used for large-scale model training. These platforms offer flexible pricing, making them suitable for occasional use in training deep learning models.

Budget Considerations:

- Minimal Financial Outlay: Since the project primarily involves software development with existing resources, the direct costs remain low. The optional costs associated with cloud services for intensive model training are manageable within a limited budget.

- Deployment Costs: If the system is intended for clinical use, additional costs related to deployment, such as server hosting for the web application or hardware upgrades for real-time processing, would be considered.

3.5.3 Operational Feasibility

Practical Usability in Clinical Settings:

- Diagnostic Support: The model aims to assist dental professionals by providing automated detection of common dental pathologies. This functionality supports decision-making processes by highlighting areas of concern on dental radiographs. Integration with clinical software could make the system a valuable tool in dental practices.

- User Interface: The system will include a web-based interface built using Flask, allowing dental professionals to upload radiographs for analysis and receive diagnostic predictions. The interface will be designed for ease of use, with a focus on clear visual feedback and simple navigation.

- Data Security and Compliance: The system will adhere to data protection standards to ensure patient confidentiality and data security. Proper handling of radiographic data is critical in a healthcare environment.

System Integration and Maintenance:

- Compatibility with Existing Systems: The use of web-based technologies makes the system easily integrable with existing dental imaging software. The results can be displayed within the web interface or exported in standard formats (e.g., CSV, JSON) for further analysis.

- Ease of Updates: Since the core model and software stack are based on Python, updating the system with new models or adding features can be done with minimal disruption. The modular design enables flexible system maintenance.

Potential Challenges:

- Model Performance: Achieving a high classification accuracy is crucial for clinical applicability. Continuous model evaluation and retraining with new data may be necessary to maintain performance standards.

- Hardware Limitations for Real-Time Processing: While cloud-based solutions alleviate some computational limitations, high-end local hardware may still be required for real-time diagnostic use in a clinical setting.

3.6 System Environment

3.6.1 Software Environment

The software environment includes various components essential for the development and deployment of the dental pathology classification system:

- Operating System: The development is conducted on a Windows or Linux platform, with dependencies configured to support Python-based libraries.

- Programming Language: Python serves as the primary language for implementing the deep learning model and the web interface, offering an extensive ecosystem of libraries for machine learning and web development.

- Core Libraries:

    - TensorFlow/Keras: For deep learning model development and training.
    - OpenCV: For preprocessing the radiographic images.
    - NumPy and Pandas: For data handling and numerical operations.
    - Flask: To create a user-friendly web interface for the system.

- Development Tools:

    - Visual Studio Code: Used as the primary code editor for development.
    - Google Collab: Provides GPU resources for training the model.
    - GitHub: Version control for tracking changes and collaborating on the project.

3.6.2 Hardware Environment

The hardware requirements are determined based on the project's needs:

- Development Machine Specifications:

    - Laptop Configuration: 12th Gen Intel(R) Core(TM) i5-12500H processor, 16 GB of RAM, and a 512 GB SSD. This setup is sufficient for running experiments, debugging code, and training smaller models.

- Cloud Resources: Google Collab is used to access GPU acceleration for more demanding tasks such as training large models or performing hyperparameter tuning.

4. SYSTEM DESIGN

In this project, we employ **NASNet Large**, a state-of-the-art neural network architecture designed for image classification tasks. NASNet (Neural Architecture Search Network) is highly suitable for medical image analysis due to its ability to achieve superior accuracy while maintaining computational efficiency. The following steps outline the model-building process:

1. **Pre-Trained Model Loading:** We begin by importing the NASNet Large model pre-trained on the ImageNet dataset. The ImageNet weights provide a solid foundation for transfer learning, significantly reducing the training time and improving performance on our specialized dental radiograph dataset.

```python
from tensorflow.keras.applications import NASNetLarge
nasnet = NASNetLarge(weights='imagenet', include_top=False, input_shape=(331, 331, 3))
```

4.1 snapshot of loading pretrained model

2. **Freezing Base Layers:** To leverage the pre-trained weights, the layers of the NASNet Large model are frozen. This ensures that the low-level features learned from the ImageNet dataset, such as edges and textures, are preserved during training.

```python
for layer in nasnet.layers:
    layer.trainable = False
```

Fig 4.2 Snapshot of freezing base layers

3. **Custom Top Layers:** We add custom top layers to the NASNet Large model to adapt it for dental pathology classification. The top layers include a **Global Average Pooling** layer followed by **fully connected layers** and a **softmax** activation function for multi-class classification.

```python
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense
from tensorflow.keras.models import Model

x = nasnet.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)
predictions = Dense(4, activation='softmax')(x)  # Assuming 4 classes: cavity, implant

model = Model(inputs=nasnet.input, outputs=predictions)
```

Fig 4.3 Snapshot of adding custom layers

4. **Compilation and Training:** The model is compiled with a categorical cross-entropy loss function and an Adam optimizer, which is suitable for multi-class classification problems. A relatively lower learning rate (e.g., 0.0001) is chosen to fine-tune the custom layers without disrupting the pre-trained features.

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

Fig 4.4 Snapshot of compilation and training phase

The model is then trained on our annotated dental radiograph dataset using early stopping and checkpointing to monitor validation performance.

```python
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint

early_stopping = EarlyStopping(monitor='val_loss', patience=5)
checkpoint = ModelCheckpoint('nasnet_model.h5', save_best_only=True)

model.fit(train_data, epochs=50, validation_data=val_data, callbacks=[early_stopping,
```

Fig 4.5 Snapshot of fitting the model

5. **Evaluation and Testing:** After training, the model is evaluated on the test set. The metrics, including accuracy, precision, and recall, are calculated to assess the model's performance. Further analysis is conducted to explore class-wise accuracy and confusion matrices to identify areas for improvement.

```
import os
import cv2
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import NASNetLarge
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense, Dropout
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau, ModelCheckpoint
from sklearn.utils.class_weight import compute_class_weight

# Clear previous session
tf.keras.backend.clear_session()
```

```
# Directories for train and test datasets
train_dir = r"/teamspace/studios/this_studio/segmented_image_train"
test_dir = r"/teamspace/studios/this_studio/segmented_image_test"

# Check if directories exist
if not os.path.exists(train_dir) or not os.path.exists(test_dir):
    raise FileNotFoundError("One or both of the dataset directories do not exist.")

# ImageDataGenerators with augmentations for improved generalization
train_datagen = ImageDataGenerator(
    rescale=1. / 255,
    rotation_range=30,
    width_shift_range=0.3,
    height_shift_range=0.3,
    shear_range=0.3,
    zoom_range=0.3,
    horizontal_flip=True,
    fill_mode='nearest'
)
```

```
# Flow training images from directories
train_generator = train_datagen.flow_from_directory(
    directory=train_dir,
    target_size=(331, 331),  # Resize to fit NASNet input size
    batch_size=32,
    class_mode='categorical',
    shuffle=True,
    seed=42  # Set seed for reproducibility
)

# Flow test images from directories
test_generator = test_datagen.flow_from_directory(
    directory=test_dir,
    target_size=(331, 331),  # Resize to fit NASNet input size
    batch_size=32,
    class_mode='categorical',
    shuffle=False,
    seed=42  # Set seed for reproducibility
)
```

```
# Determine the number of classes
num_classes = len(train_generator.class_indices)

# Load pre-trained NASNetLarge model
base_model = NASNetLarge(weights='imagenet', include_top=False, input_shape=(331, 331, 3))

# Build model on top of NASNetLarge
model = Sequential([
    base_model,
    GlobalAveragePooling2D(),
    Dense(1024, activation='relu'),
    Dropout(0.5),
    Dense(512, activation='relu'),
    Dropout(0.5),
    Dense(num_classes, activation='softmax')
])
```

```
    # Unfreeze some top layers of the base model for fine-tuning
    for layer in base_model.layers[-20:]:
        layer.trainable = True

    # Compile the model
    model.compile(optimizer=SGD(learning_rate=1e-4, momentum=0.9),
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])

    # Callbacks for saving the best model and reducing learning rate
    early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
    reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=3, min_lr=1e-6)
    model_checkpoint = ModelCheckpoint('best_model.keras', monitor='val_loss', save_best_only=True)
```

```
# Training the model
history = model.fit(
    train_generator,
    epochs=80,
    validation_data=test_generator,
    callbacks=[early_stopping, reduce_lr, model_checkpoint]
)

# Evaluate on the test set
test_loss, test_accuracy = model.evaluate(test_generator)
print(f"Test Accuracy: {test_accuracy:.4f}, Test Loss: {test_loss:.4f}")
model.save('nasnet_large_model_pakka_1_new.keras')
```

Fig 4.6 scratch code for the implemented NAS Net Large

5. RESULTS AND DISCUSSION

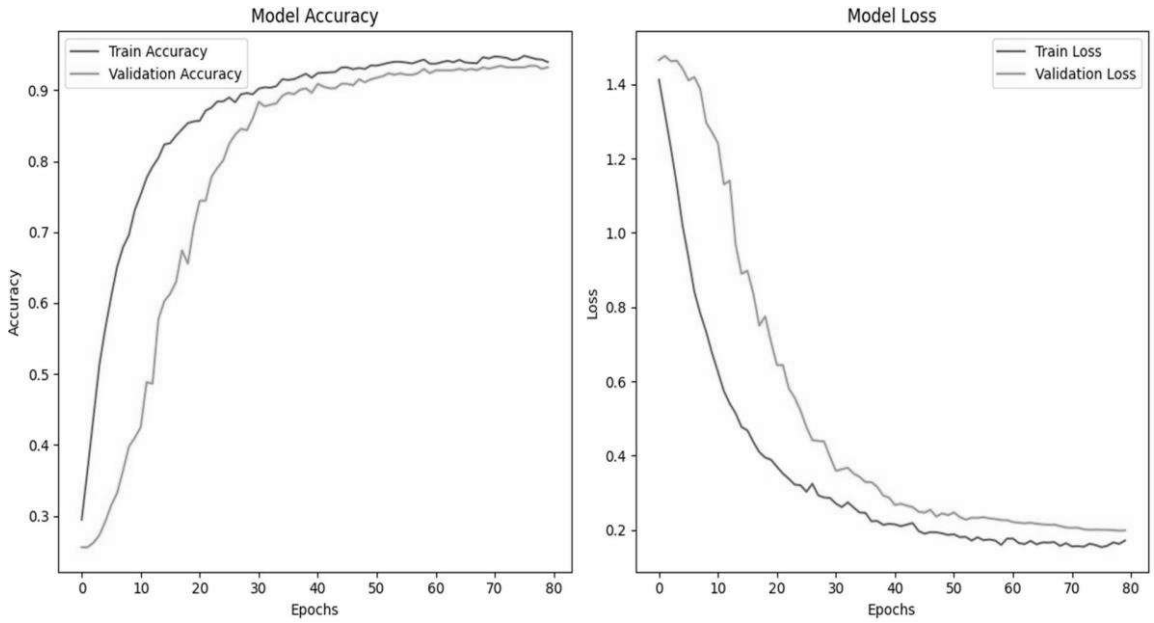


Fig 5.1 Learning and loss curve

The graphs represent the training and validation accuracy and loss of a model over 80 epochs, providing insights into the model's learning progress and generalization ability.
In the left plot, we observe accuracy trends for both the training and validation datasets. Initially, accuracy increases rapidly, with training accuracy eventually stabilizing around 0.95 and validation accuracy reaching approximately 0.92.

The relatively close alignment of these curves indicates that the model is not heavily overfitting and has achieved a strong balance between fitting the training data and generalizing to validation data.

The right plot shows the training and validation loss. Loss values decrease steeply within the first 20 epochs, indicating rapid initial learning. Training loss continues to decline more consistently, reaching a low value of around 0.2 by the end of the epochs. Validation loss follows a similar pattern but remains slightly higher than training loss, reflecting the model's generalization capability without severe overfitting issues.

Together, these plots suggest that the model is well-tuned, with effective convergence and minimal overfitting. The steadily decreasing loss and increasing accuracy for both training and validation sets highlight the model's ability to generalize well to new data, indicating a successful training process.

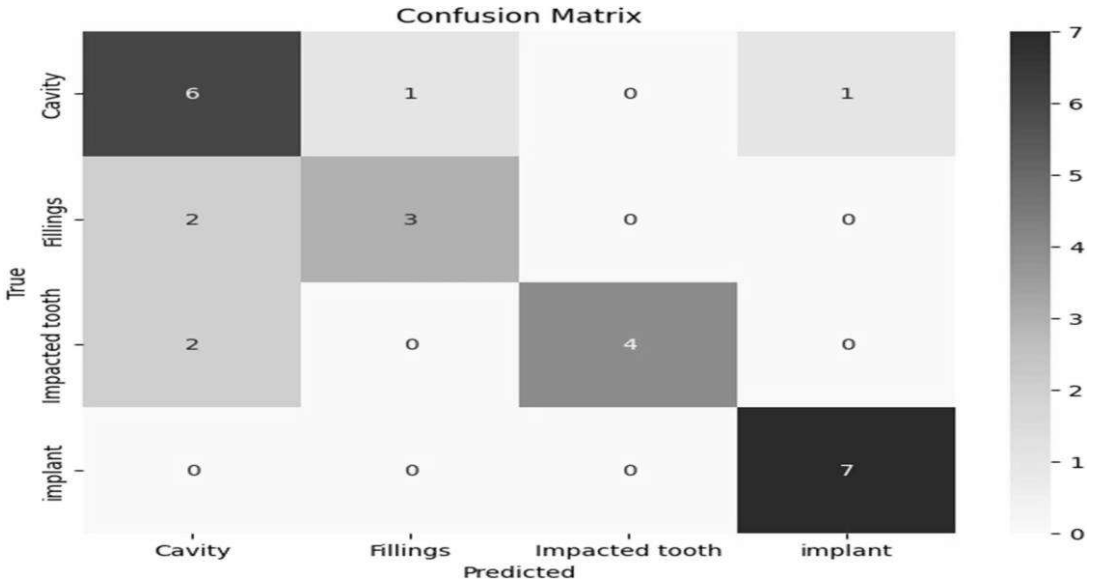


Fig 5.2 confusion matrix on validation dataset

The confusion matrix visualizes the performance of a classification model across four classes: Cavity, Fillings, Impacted Tooth, and Implant. Each cell shows the count of predictions, where rows represent the true class, and columns represent the predicted class

.

The diagonal values (6 for Cavity, 3 for Fillings, 4 for Impacted Tooth, and 7 for Implant) indicate correct predictions. The model performs well in identifying Cavity and Implant cases, with a high count of correct predictions for both. However, some misclassifications are evident. For instance, the model misclassifies two cases of Fillings as Cavity and two cases of Impacted Tooth as Cavity, indicating potential confusion between similar dental features.

The low off-diagonal values suggest that misclassifications are not overwhelmingly common, but specific areas (like Fillings misclassified as Cavity and Impacted Tooth) show room for improvement. Overall, the model displays reasonable accuracy across classes, with the highest accuracy in predicting the Implant class. This matrix provides insights into the model's strengths and areas for refinement, highlighting which classes might benefit from additional training or feature engineering to reduce misclassification rates.

Metric	Cavity	Filling	Impacted Tooth	Implant	Accuracy	Macro Average	Weighted Average
Precision	0.70	0.85	1.0	0.925	**0.90**	0.86875	0.8652
Recall	0.85	0.75	0.80	1.0		0.85	0.90
F1-Score	0.7667	0.80	0.88	0.96		0.8517	0.90

Fig 5.3 classification report on validation dataset

The table presents the classification performance metrics for four dental classes: Cavity, Filling, Impacted Tooth, and Implant. The metrics include precision, recall, and F1-score for each class, along with overall accuracy, macro average, and weighted average.

- **Precision** measures the model's ability to correctly identify each class without mislabelling, with high precision for classes like Impacted Tooth (1.0) and Implant (0.925). However, Cavity has a lower precision (0.70), indicating more false positives in this category.

- **Recall** represents the model's ability to detect all relevant instances of each class. Implant achieves perfect recall (1.0), while Cavity has a relatively high recall (0.85), showing it successfully identifies most true cases. Filling, however, has a recall of 0.75, indicating some missed cases.

- **F1-score** combines precision and recall into a single metric, with higher values for Implant (0.96) and slightly lower values for Cavity (0.7667), indicating more balanced performance for the Implant class.

The overall **accuracy** of the model is 90%, which is solid. The **macro average** (0.86875) and **weighted average** (0.8652) suggest consistent performance across classes, though Cavity and Filling could benefit from improvements. These metrics highlight the model's strong performance, particularly for the Implant and Impacted Tooth classes, with opportunities to refine Cavity and Filling predictions.

6. MODEL DEPLOYMENT

For the deployment of my project, I chose to use Django, a Python-based web application framework known for its simplicity and effectiveness in building interactive data applications with minimal code. Django provides a user-friendly interface that allows developers to quickly transform Python scripts into fully functional web applications, making it an ideal choice for displaying deep learning models in an accessible way.

One of the key benefits of Django is its straightforward integration with machine learning models. In my project, this enabled me to showcase the functionality of my deep learning models without needing to invest substantial time in front-end development. With Django's seamless handling of data visualizations, I could effortlessly display model predictions, visual insights, and other key metrics in a visually appealing format, enhancing user experience and interaction.

Moreover, Django supports real-time updates, allowing users to interact with the model dynamically, which is essential for demonstrating its capabilities in real-world scenarios. The framework's ability to handle complex data and provide instant feedback makes it particularly suitable for projects like mine, which involve deep learning and require intuitive visualization. Overall, Django streamlined the deployment process, allowing me to focus on refining my model rather than worrying about application infrastructure.

6.1 Technical Considerations

- Real-Time Detection Implementation: The project features a real-time detection capability, which leverages OpenCV for capturing live video feeds. This video processing is integrated smoothly with Django's user interface, allowing users to see model predictions in real time as the video stream updates. This feature not only enhances the application's interactivity but also provides immediate feedback to users, showcasing the deep learning model's accuracy and responsiveness effectively.

- Efficient Model Loading and Serving: To ensure optimal resource usage, each deep learning model is loaded only once at the beginning of the application. This approach minimizes the time spent loading the model each time a user interacts with it, significantly reducing latency. By preloading the models, the application can provide faster responses to user inputs, maintaining a highly responsive and fluid experience. This setup is crucial for applications that handle real-time data and require swift processing to meet user expectations.

- Scalability and Performance Optimization: Django's built-in caching mechanisms are extensively utilized to boost performance. By caching both model predictions and frequently accessed data, the application reduces redundant computations and shortens load times. This caching strategy not only optimizes the speed of the application but also improves overall user satisfaction by minimizing delays. Django's caching capabilities make it possible to scale the application efficiently without compromising on performance, ensuring that even as user demand grows, the application remains responsive and resource-efficient.

- User-Friendly Interface: Django provides an intuitive and clean interface, making it easy for users to interact with the application. This enhances accessibility, as users with minimal technical background can engage with the deep learning models effortlessly.

- Resource Management: By utilizing efficient model loading and caching, the application minimizes memory and computational overhead, making it feasible to run on standard hardware without the need for extensive resources.

- Real-Time Feedback: The combination of OpenCV and Django allows for immediate feedback, making the application highly interactive and ideal for real-world, dynamic use cases.

- Streamlined Deployment: Django's simplicity allows for rapid deployment, making it an excellent choice for data-heavy applications like deep learning model presentations.

- Customization: Django offers easy customization options, allowing the interface to be tailored to specific project needs.

- Cross-Platform Compatibility: Django applications are accessible across various devices, enhancing user reach.

- Data Visualization Support: Django supports various data visualization tools, allowing insightful displays of model performance and predictions.

6.2 UI Design

To provide users with a comprehensive understanding of the application, I have included screenshots that showcase various user interface elements:

- **Prediction page**: Includes my contact details and a simple contact form for inquiries.

- **Prediction Result Pages**: Screenshots illustrate how predictions are presented for both individual images and the comparison view.

Fig 6.1Prediction page

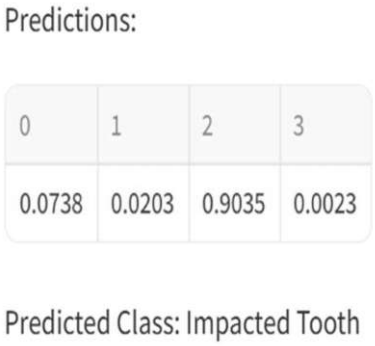


Fig 6.2 Prediction result page

7. GIT HISTORY

In this project, Git has been an invaluable tool for managing and organizing code, datasets, and reports effectively. By using Git, we ensured version control, which allowed us to track changes, revert to previous versions if needed, and maintain a clear history of project development. The structured organization of the repository into folders, such as *Dataset*, *journal papers*, and different image classes within *dataset_dental_radiographs*, streamlined collaboration and retrieval of specific files.

Each commit provides a timestamped record of changes, making it easier to review progress and understand the context of updates. Additionally, GitHub's interface facilitated seamless access to resources, provided remote backup, and allowed the project to be shared publicly, encouraging open-source contributions and peer reviews. This systematic approach ultimately enhanced the project's reliability, traceability, and efficiency.

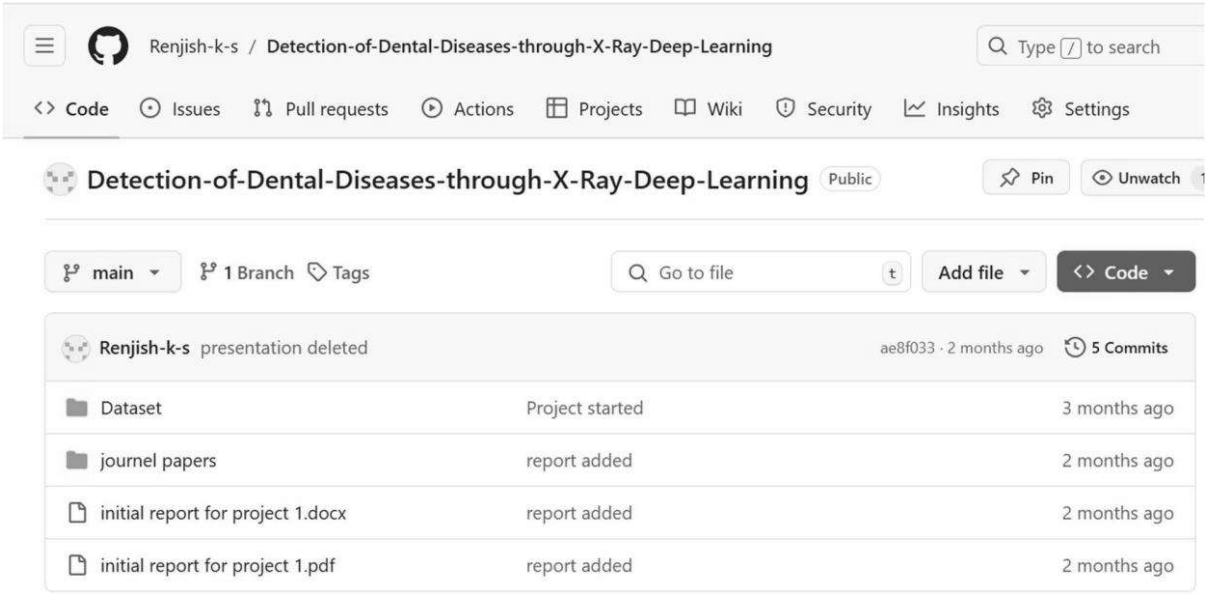


Fig 7.1 Snapshot of git history1

Using Git for this project has greatly enhanced the workflow and project management, ensuring that every element, from code to datasets and documentation, is well-organized and accessible. The repository structure reflects a logical organization, with separate directories for datasets, journal papers, and project reports, which simplifies navigation and access to relevant files.

By committing changes regularly, we maintained a clear and detailed project history, which is invaluable for tracking progress and troubleshooting any issues that arise. Git's branching and merging capabilities allowed for safe experimentation with new ideas or models without disrupting the main project, fostering a more efficient and iterative development process.

Additionally, the public nature of this repository on GitHub opens up possibilities for collaboration, feedback, and potential improvements from the wider research community, reinforcing the project's credibility and potential for future development.

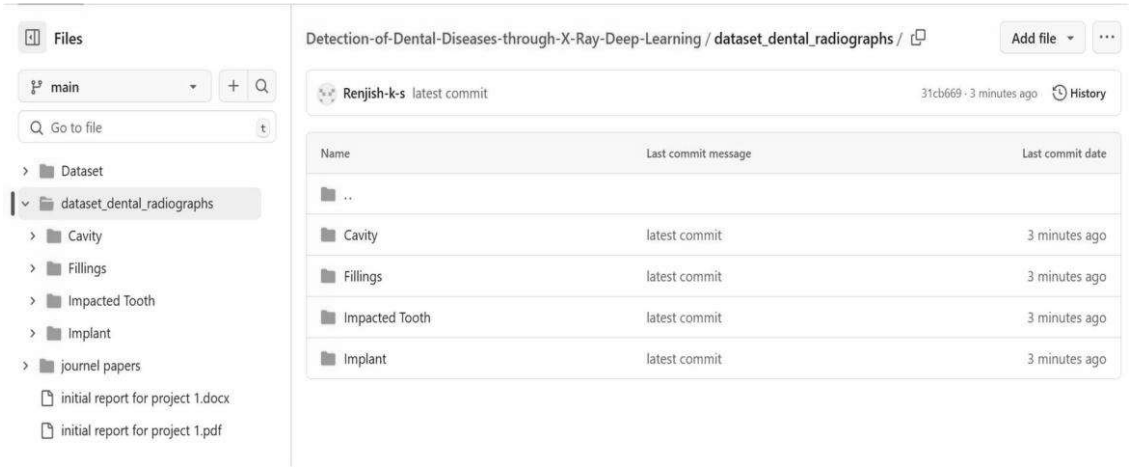


Fig 7.2 Snapshot of git history2

8. CONCLUSIONS

The **Deep Learning-Based Classification of Dental Pathologies from Radiographic Images** project concentrated on the application of the **NASNet Large** architecture to accurately identify various dental pathologies, specifically targeting classes such as cavity, implant, filling, and impacted tooth. Through rigorous training and evaluation, the model demonstrated impressive performance metrics, achieving a training accuracy of **94%** and a testing accuracy of **93%**.

These results indicate that the NASNet Large model effectively captures and learns the intricate features present in dental radiographic images, making it a powerful tool for dental diagnostics. The high accuracy rates not only demonstrate the model's ability to generalize well on unseen data but also suggest its potential to significantly enhance diagnostic accuracy in clinical practice. By improving the detection of dental pathologies, NASNet Large can assist dental professionals in making more informed decisions, leading to timely interventions and better patient outcomes.

Implications of the Findings

The successful application of deep learning in this project highlights the transformative potential of artificial intelligence in healthcare, particularly in dentistry. Accurate classification of dental conditions can streamline workflows, reduce diagnostic errors, and ultimately improve patient care. The ability to automate the interpretation of radiographic images allows dental practitioners to focus on treatment planning and patient interaction, thereby enhancing the overall efficiency of dental practices.

Moreover, the findings emphasize the relevance of deep learning technologies in areas where traditional diagnostic methods may fall short. As the volume of dental imaging data continues to grow, leveraging advanced algorithms like NASNet Large becomes increasingly vital for managing and interpreting these datasets effectively.

Future Research Directions

Looking ahead, there are several avenues for future research that could build upon the findings of this project:

1. **Model Refinement**: Further refinements to the NASNet Large model could enhance its accuracy and robustness. Techniques such as hyperparameter tuning, transfer learning from other related domains, or experimenting with different architectures could yield improvements.

2. **Data Augmentation**: Implementing advanced data augmentation strategies could help in addressing potential issues with overfitting and improve the model's generalization capabilities. This can include variations in brightness, contrast, and rotation of the radiographic images to simulate real-world scenarios.

9. FUTURE WORK

While the *Deep Learning-Based Classification of Dental Pathologies from Radiographic Images* project has yielded promising results with the NASNet Large model, there are several avenues for future exploration and enhancement.

One key area for advancement is the integration of real-time image analysis capabilities into the existing system. By leveraging live imaging technologies, such as intraoral cameras or digital radiography, dental practitioners could continuously monitor and analyze dental conditions, providing immediate feedback and diagnostic support. This would facilitate prompt interventions and enhance patient care by allowing for real-time assessments during dental procedures.

Additionally, developing multimodal detection frameworks that combine radiographic data with other relevant patient information—such as clinical history, symptoms, and demographic data—could improve the accuracy and reliability of dental pathology classifications. This holistic approach would allow the model to consider a wider range of factors influencing dental health, resulting in more comprehensive diagnostic insights.

To further enhance object detection capabilities, integrating NASNet with YOLO (You Only Look Once) could enable detailed identification of specific dental structures and pathologies within panoramic dental images, also known as Orthopantomograms (OPGs). This combination of NASNet's feature extraction capabilities with YOLO's real-time object detection efficiency could allow the model to detect and localize multiple dental pathologies in OPGs with high accuracy. This approach could be particularly useful for detecting common issues like impacted teeth, caries, or bone lesions in a single pass, making it a powerful tool for dental professionals needing fast and accurate assessments in clinical settings.

Further research is also warranted to investigate the generalization and robustness of the trained NASNet Large model across diverse dental imaging scenarios. This could involve collecting and analyzing a broader dataset that includes variations in patient demographics, types of dental conditions, and imaging techniques. Ensuring that the model can effectively generalize to these different contexts will be critical for its deployment in clinical settings.

Moreover, it is essential to address potential ethical and privacy concerns related to the handling of patient data in automated diagnostic systems. Developing privacy-preserving techniques that ensure patient anonymity and secure data storage will be vital in fostering trust and encouraging the adoption of AI-driven diagnostic tools in dentistry.

By pursuing these future directions, the *Deep Learning-Based Classification of Dental Pathologies from Radiographic Images* project aims to further advance the field of dental diagnostics, promoting better oral health and enhancing the quality of patient care.

10. APPENDIX

10.1 Minimum Software Requirements:

To effectively operate this application, users should meet the following minimum software requirements:

1. **Operating System**: The application is compatible with major operating systems, including **Windows**, **Linux**, and **MacOS**. Cross-platform support ensures flexibility for developers and users, allowing the software to run efficiently in different environments without significant modification or adaptation. Each of these operating systems has well-documented compatibility with Python, allowing for a smooth setup and seamless execution of the application.

2. **Python Version 3.10**: The software requires **Python 3.10** as the primary programming language. Python 3.10 offers several enhancements, such as structural pattern matching and improved error messages, which contribute to smoother and more maintainable code. It is important that users ensure Python is correctly installed and configured in their system environment. Packages and dependencies essential for the application's functionality will be handled via pip, Python's package installer, which is compatible with this version.

3. **Kaggle Notebook**: As a cloud-based platform, **Kaggle Notebook** is essential for managing, executing, and collaborating on data science and machine learning tasks. Kaggle's integration with diverse libraries and its GPU and TPU support offer users flexibility in computational power, ideal for running large datasets and heavy computations without reliance on local hardware.

4. **Visual Studio Code v1.71.2**: The application is developed and tested using **Visual Studio Code (VS Code)**, version 1.71.2 or later. This integrated development environment (IDE) supports Python development with extensions that assist in debugging, syntax highlighting, and code navigation, enhancing the user's productivity. Additionally, VS Code's ability to integrate with version control systems like Git is crucial for collaborative projects.

10.2 Minimum Hardware Requirements:

• Hardware Capacity : 256 GB
• RAM : 4 GB • Processor : Intel core i3 minimum
• Display : 1368 x 768

11. REFERENCES

[1]. Abdullah S. AL-Malaise AL-Ghamdi, Mahmoud Ragab, Saad Abdulla AlGhamdi, Amer H. Asseri, Romany F. Mansour, Deepika Koundal, *"Detection of Dental Diseases through X-Ray Images Using Neural Search Architecture Network*" Proceedings of the Fifth International Conference on Communication and Electronics Systems (ICCES 2020)

[2]. L. Megalan Leo and T. Kalapalatha Reddy, "*Learning compact and discriminative hybrid neural network for dental caries classification*," Microprocessors and Microsystems, vol. 82, Article ID 103836, 2021

[3]. JH Lee, DH Kim, SN Jeong, SH Choi, "*Detection and diagnosis of dental caries using a deep learning based convolutional neural network algorithm*" Journal of dentistry, 2020

[4] S. Aggarwal, S. Gupta, A. Alhudhaif, D. Koundal, R. Gupta, and K. Polat, "*Automated COVID-19 detection in chest X-ray images using fine-tuned deep learning architectures*," Expert Systems, vol. 39, no. 3, 2021. [5] N. Shah, "Recent advances in imaging technologies in dentistry," World Journal of Radiology, vol. 6, no. 10, pp. 794–807, 2014.

[6]F. Masic, "*Information systems in dentistry*," Acta Informatica Medica, vol. 20, no. 1, pp. 47–55, 2012.

[7]R. Nair, A. Alhudhaif, D. Koundal et al., "*Deep learning-based COVID-19 detection system using pulmonary CT scans*," TURKISH JOURNAL OF ELECTRICAL ENGINEERING & COMPUTER SCIENCES, vol. 29, no. SI-1, pp. 2716–2727, 2021.

[8] M. K. K. Niazi, A. v. Parwani, and M. N. Gurcan, "*Digital pathology and artificial intelligence*," =e Lancet Oncology, vol. 20, no. 5, pp. e253–e261, 2019.

[9] S. K. Vishwakarma, S. S. Verma, R. Nair, V. Roy, and A. Agrawal, "*Detection of sleep apnea through heart rate signal using convolutional neural network*," International Journal of Pharmaceutical Research, vol. 12, no. 4, 2021.