DAYANANDA SAGAR ACADEMY OF TECHNOLOGY & MANAGEMENT

Mini Project Review – Advanced Java (BIS402)

Mini Project Title: ” Oral Cancer Detection using AI and Machine Learning“

Presented by:

Student Name(s) :ANISHA SHIVARAM USN:1DT22IS015

GEETHANJALI HS 1DT22IS054

DHEERAJ P B 1DT22IS048

Under the Guidance of:

Dr. Veena R S

Associate Professor, Department of ISE

DSATM

Bengaluru

Introduction

Oral cancer is one of the most prevalent forms of cancer worldwide, with a significant impact on public health. Early detection plays a crucial role in improving survival rates and the effectiveness of treatment. However, traditional methods of diagnosing oral cancer, such as visual inspections and biopsies, often suffer from limitations in accuracy and timeliness.

To address these challenges, our project leverages **Artificial Intelligence (AI) and Machine Learning (ML)** techniques combined with **image processing** and **image matching** to create a more efficient and reliable system for detecting oral cancer at an early stage. By training our AI models on a diverse set of medical images, we aim to automate the process of identifying potential signs of oral cancer, making it faster and more accessible for healthcare providers.

In this presentation, we will walk you through our methodology, the AI models we used, and the results of our system, which can potentially revolutionize the way oral cancer is detected and diagnosed

Implementation Details

1. Data Collection

•**Dataset**: Publicly available oral cancer images, including healthy and cancerous samples, are used for training.

•**Data Augmentation**: Techniques like rotation, flipping, and scaling enhance dataset diversity.

2.Image Preprocessing

•**Enhancement**: Image quality is improved using contrast adjustment and histogram equalization.

•**Segmentation**: Techniques like thresholding and edge detection isolate the oral tissues.

•**Normalization**: Images are resized and standardized for uniformity.

•**Feature Extraction**: Key features such as texture, shape, and intensity are extracted from the images.

3. Image Matching and Comparison

•**Image Matching**: Techniques like feature and template matching compare new images with known cases.

Similarity Metrics: Structural Similarity Index (SSIM) and Mean Squared Error (MSE) measure image similarities.

4. Evaluation

•**Metrics**: Model performance is assessed using accuracy, precision, recall, and F1-score.

•**Cross-validation**: k-fold cross-validation ensures robustness and generalization.

Software Requirements

Hardware Requirements:

1. **Processor**: At least a dual-core processor (Intel i3 or equivalent).
2. **Memory (RAM)**: Minimum of 4GB (8GB or more recommended for better performance).
3. **Storage**: At least 1GB of free storage space for application files and datasets.
4. **Graphics**: No special requirements; a basic integrated GPU suffices.
5. **Operating System**: Windows, Linux, or macOS.

Software Requirements:

1. **Programming Language**: Python 3.7 or later.
2. **Libraries and Frameworks**:
    1. NumPy (for numerical computations).
    2. Pandas (for data handling and analysis).
    3. Scikit-learn (for machine learning model handling).
    4. Pickle (for loading the serialized machine learning model).
3. **Database**: CSV files are used as the dataset storage medium.
4. **Web Browser**: Any modern browser (Google Chrome, Mozilla Firefox, etc.).
5. **IDE/Editor**: VSCode, PyCharm, or Jupyter Notebook for development.

Methodology

- **Data Loading and Preprocessing**
- **Hyperparameter Tuning:**

Keras Tuner: This library is used to automatically search for the optimal hyperparameters, such as the number of filters, kernel size, and learning rate, to improve model performance

Model Training and Evaluation:

Training: The model is trained on the training dataset, adjusting its weights and biases to minimize the loss function (binary cross-entropy in this case).

Validation: The model's performance is evaluated on a validation dataset to monitor overfitting and adjust hyperparameters.

Testing: The final trained model is evaluated on a separate test dataset to assess its generalization ability.

Image Preprocessing: New images are preprocessed in the same way as the training data.

Model Inference: The preprocessed image is fed into the trained model, and the model predicts the probability of the image belonging to the "cancer" class.

Threshold-Based Classification: A threshold value is used to classify the image as "cancer" or "no cancer" based on the predicted probability.

ALGORYTHM

Convolutional Neural Network (CNN): The model architecture uses a **CNN**, which is a deep learning algorithm specifically designed for processing structured grid data, such as images. The layers in the CNN help it learn spatial hierarchies of features (e.g., edges, textures, shapes, and objects)

Here’s how this work:
The code trains a **Convolutional Neural Network (CNN)** to recognize and classify images. Here's how it works:
1.Convolution Layers: The model looks for simple patterns in the image, like edges and shapes.
2.Pooling Layers: It reduces the image size while keeping the important features.
3.Dense Layers: It combines the features learned to understand more complex patterns.
4.Output Layer: The model makes a final prediction about what the image is (e.g., a park, beach, etc.).
It learns by being shown many images and adjusts itself to get better at predictions over time. The model then predicts the scene in new images based on what it has learned


Thank
you