

IMAGE CLASSIFICATION FOR DIABETES USING MACHINE LEARNING

ABSTRACT

This project provides a novel approach to diabetes detection using image classification and machine learning algorithms. Combining autoencoder methods and convolutional neural networks, as well as subtle concepts from Traditional Chinese Medicine (TCM), this study seeks to exploit visual patterns suggestive of diabetes diseases that lack specific anatomical references. The suggested method combines CNNs and deep autoencoder algorithms to extract robust features from photos. The system demonstrates high accuracy in differentiating diabetes and non-diabetic cases after extended training on a chosen dataset, which has been updated for greater model generalisation. Furthermore, a user-friendly interface makes easy integration into clinical practice possible, allowing for real-time diagnosis and monitoring. The use of this technology has the potential to enhance both patient outcomes and healthcare delivery by enabling the early detection and customised management of diabetes.

TABLE OF CONTENTS

Ch.No	Chapter	Pg. No
	Abstract	(i)
	Table Of Contents	(ii) - (iii)
	List Of Figures	(iv)
1.	INTRODUCTION	1 - 3
1.1	Introduction	1
1.2	Problem Statement	2
1.3	Research Objective	3
1.4	Structure of Thesis	3
2.	LITERATURE REVIEW	4 - 5
3.	METHODOLOGY	6 - 20
3.1	Data Collection	6 - 9
3.2	Model Selection	9 - 12
3.3	Model Architecture	12 - 16
3.4	Model Training	17 - 20
4.	RESULTS	21 - 23
4.1	Model Performance	21 - 22
4.2	Comparison with Existing Methods	23
5.	DISCUSSIONS	24 - 27
5.1	Analysis of Results	24 - 26
5.2	Limitations	26 - 27
6.	GUI APPLICATION DEVELOPMENT	28 - 31
6.1	Overview	28
6.2	Description	28

6.3	Implementation	28 - 31
	CONCLUSION	32
	FUTURE WORK	33
	REFERENCES	33 - 34

LIST OF FIGURES

3.1(i).	Data Source	6
(ii)	Data Collection	7
(iii)	Data Description	7
(iv)	Data Cleaning	8
(v)	Normalisation	8
(vi)	Data Splitting	9
3.3(i)	Model Architecture	13
(ii)	Model Layers	16
3.4(i)	Data Augmentation	17
(ii)	Parameters	19
4.1(i)	Accuracy	21
(ii)	Precision, Recall, F1	22
5.1(i)	Accuracy Plot	24
(ii)	Loss Plot	25
(iii)	Confusion Matrix	26
6.3(i)	GUI Process	29
(ii)	GUI	31

CHAPTER-1

INTRODUCTION

1.1 INTRODUCTION

Globally, the prevalence of diabetes is rising significantly, which presents serious difficulties for public health systems. Diabetes should be identified and for it to be treated early to reduce complications and enhance patient outcomes. Conventional techniques for diagnosing diabetes sometimes entail time-consuming and costly invasive procedures or biochemical testing.

Recent developments in computer vision and machine learning have created new avenues for non-invasive, low-cost diagnostic methods. The study of tongue pictures for the diagnosis of diabetes is one such new method. The tongue may be a sign of systemic health issues like diabetes because of its abundance of blood vessels and nerve endings.

Machine Learning is a subset of AI that focuses on creating algorithms and statistical models that allow computers to complete jobs without explicit instructions. ML systems, on the other hand, use data to identify patterns and make judgments. There are three types of learning processes: semi-supervised, unsupervised, and supervised. Using labelled data, models are trained to make predictions or classifications based on predetermined results in supervised learning. Conversely, unsupervised learning works with unlabeled data and finds hidden patterns or clusters in the data. Labelled and unlabeled data are used in semi-supervised learning to increase learning accuracy.

With the help of convolutional neural networks (CNNs) and tongue image processing, this thesis takes a new look at diabetes detection. The project aims to create a unique framework for the automatic classification of tongue images into diabetic and non-diabetic groups by utilising advances in computer vision, machine learning and deep learning. The suggested method aims to reveal a wealth of diagnostic insights by utilising the different visual manifestations of diabetes on the tongue, which could completely transform the field of diabetes screening and management.

1.2 PROBLEM STATEMENT

The problem statement highlights the current gaps and limits in the field of diabetes diagnostics and emphasises the need for creative solutions, which sums up the main difficulty this research initiative seeks to address. The following major points are the focus of the problem statement:

1. **Invasive Diagnostic Methods:** Labor-intensive biochemical testing or invasive procedures are frequently required in traditional diabetes diagnosis methods. These methods might cause discomfort for patients and may not always produce results quickly enough, which can postpone necessary measures and accelerate the disease's course.
2. **Underutilization of Visual Biomarkers:** The human body contains an extensive amount of diagnostic information, but some visual biomarkers of disease, including those that show up on the tongue, are still overlooked in clinical practice though they haven't been completely utilised, the tongue's minor visual clues related to diabetes offers a way for early identification.
3. **Need for Automated Screening Tools:** With diabetes becoming more and more commonplace worldwide, there is an immediate need for automated screening tools that can enhance current diagnostic procedures and expedite the illness diagnosis process. These devices ought to be able to quickly and effectively analyse vast amounts of medical imaging, providing medical personnel with timely insights for efficient patient treatment.
4. **Complexity of Tongue Image Analysis:** Analysing tongue photos for diabetic symptoms provides unique challenges due to the complicated connections between anatomical features, lighting variations, and small indicators of disease. Due to the subjectivity and error-proneness of human observers, automated analysis techniques based on computer vision and machine learning have been developed in place of manual tongue picture analysis.
5. **Enhanced Diagnostic Accuracy:** Since a delayed or incorrect diagnosis can have a significant negative impact on a patient's prognosis, achieving high diagnostic accuracy in diabetes screening is essential. The goal is to create a diagnostic tool that can reliably and accurately distinguish between people with diabetes and those without by using machine learning algorithms trained on annotated datasets of tongue photos.

1.3 RESEARCH OBJECTIVE

The following are the project's main goals:

- 1.Assembling a large dataset of tongue pictures from different sources and making sure that the distribution of instances with and without diabetes is balanced.
- 2.Creating preprocessing methods to reduce noise and improve feature visibility while standardising tongue picture size, colour, and texture.
- 3.Creating a CNN architecture specifically for the purpose of detecting diabetes from tongue images while maximising computational effectiveness and model performance.
- 4.Using the relevant machine learning algorithms and hyperparameter tuning approaches, the CNN model is trained on the dataset.
- 5 .Comparing the trained model's performance to baseline techniques and clinical standards by evaluating its accuracy & loss among other metrics.
- 6.Integrating the taught model into an intuitive software programme or online interface, enabling Images of the tongue can be uploaded by medical practitioners for accurate and quick diabetes screening.

1.4 STRUCTURE OF THESIS

The thesis is divided into multiple chapters, each of which focuses on a different part of the research process:

- 1.Review of Relevant Literature: In this section, we examine recent research on machine learning methods, diabetes diagnosis, and image analysis of the tongue. This section presents the theoretical framework that supports our investigation.
- 2.Methodology: The research design, data collection methods, and analytical approaches used in this study are all covered in the methodology chapter. It describes the procedures followed in order to create and assess the machine learning model for identifying diabetes.
- 3.Results: We report our research's findings in this chapter, together with experimental results, model performance metrics, and data analysis visualisations.
- 4.Discussion: Within the framework of the research aims and relevant literature, the results are interpreted in the discussion chapter. It evaluates the results, points out its shortcomings, and makes suggestions for additional.

CHAPTER 2

LITERATURE VIEW

1.The material entitled: "DIABETES IDENTIFICATION WITH TONGUE Vision With EXTRACTION Based GLOBAL Points & DECISION TREE"

Author Name : Dr.S.Bhuvaneswari

Description : This work investigates the detection of diabetes from tongue images by using texture features and a decision tree classifier. Previous studies have shown that there is a relationship between the colour of the tongue and diabetes, and that certain typological features, such as red points, can help diagnose the condition. Several strategies have been proposed to improve the accuracy of tongue feature extraction, including the placement of texture blocks strategically and the use of a new Colour Checker language for optical analysis. Strategies for successful area extraction and tongue coating classification have also been discussed. This paper expands on these approaches by employing global feature extraction and decision tree classification to achieve high accuracy in diabetes detection from tongue images.

2.Paper Name : “Diabetes Diagnostic Method based on Tongue Image Classification Using Machine Learning Algorithms”

Author Name : M. Bharathi, Dr.D.Prasad, Dr.T. Venkatakrishnamoorthy, Dr.M.Dharani.

Description : Diabetes mellitus, a common and serious illness, has spurred a lot of research on rapid and precise diagnostic techniques, especially those that make use of deep learning and machine learning algorithms. These algorithms have been used in numerous research to classify images for medicinal purposes, showcasing their great accuracy potential. Notable works include automated tongue diagnosis systems that outperform conventional approaches and the use of Support Vector Machines (SVM), which minimize computational time while preserving accuracy. Furthermore, the effectiveness of sophisticated methods like Decision Trees and Convolutional Neural Networks (CNN) in identifying diabetes from tongue images has been investigated. Building on these foundations, this paper achieves higher accuracy than previous classifiers by using CNN based on dimensional reduction.

3. Title of Paper: "Diagnostic Approach for Diabetes Utilizing Tongue Images and Support Vector Machines"

Author : Jianfeng Zhang, Jiatuo Xu, Xiaojuan Hu, Qingguang Chen, Liping Tu, Jingbin Huang, Ji Cui

Description : In order to diagnose diabetes using tongue images, the study investigates the combination of digital image processing and machine learning with Traditional Chinese Medicine (TCM). Prior studies have demonstrated the relationship between tongue traits and diabetes in TCM, which has historically relied on practitioners' subjective evaluations. But improvements in computing techniques have made objective and standardized analysis easier. For the purpose of classifying tongue images, a number of machine learning techniques have been used, such as Support Vector Machine (SVM), k-Nearest Neighbor (k-NN), Naive Bayes, Decision Tree, and Neural Network. However, SVM has become the most popular technique because of its resilience when dealing with small samples and non-linear data. Using a GA-optimized SVM model, the current study proves that digital tongue diagnosis is feasible by showing better accuracy and efficiency in diabetes prediction than existing algorithms in TCM.

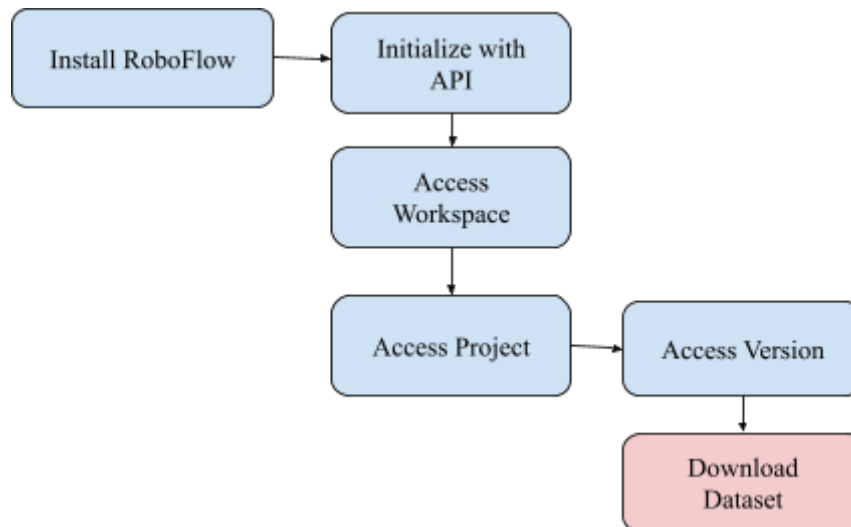
CHAPTER 3

METHODOLOGY

3.1 Data Collection

1.Data Sources -

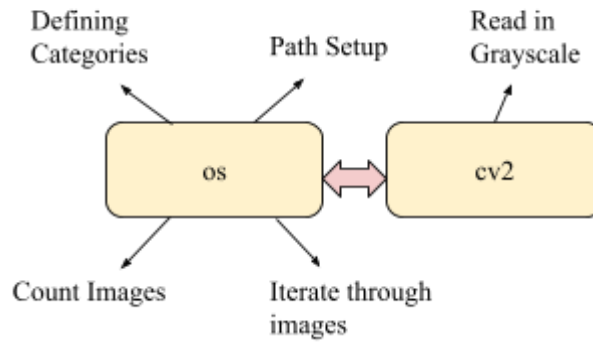
The dataset utilised in this research project was sourced from Roboflow, a stage known for its comprehensive collection of datasets custom-made for machine learning applications. Roboflow provided access to a dataset particularly curated for tongue picture classification related to diabetes discovery. This dataset was significant because it contained a different range of tongue pictures categorised into three particular classes: diabetic, non-diabetic, and unlabeled. Each class spoke to different visual signs of tongue conditions significant to diabetes determination.



(i) Data Source

2.Data Collection -

The method of collecting the dataset included a few key steps utilising Python libraries and tools suited for image processing and data manipulation. OpenCV (cv2) was instrumental in dealing with image-related tasks, such as reading pictures in grayscale format and applying essential changes. The os module encouraged efficient directory route and file management, enabling precise organisation of the dataset into respective categories. Python's numpy library played a significant part in managing arrays and performing basic information changes. The dataset was organised into categories based on the diabetic status attributed to each tongue



(ii) Data Collection

image. This categorization guaranteed that the dataset was labelled precisely, reflecting the different conditions and varieties that can show on tongues related to diabetes.

3.Data Description -

The dataset consisted of tongue images categorised into three essential classes:

Diabetic:

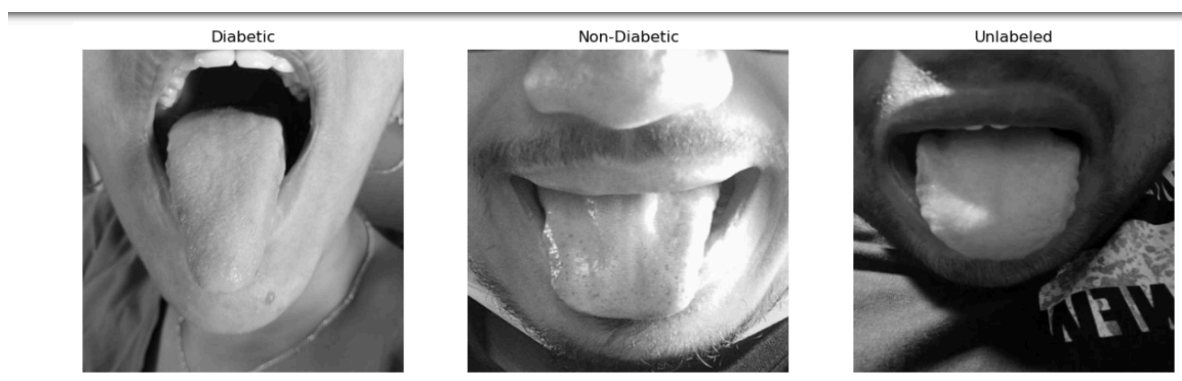
Tongue images showing visible indicators related with diabetic conditions, such as discoloration, texture changes, or injuries which will connect with diabetes.

Non-Diabetic:

Tongue images showing typical characteristics without any visible signs of diabetic-related changes or abnormalities.

Unlabeled:

Images that were either not categorised due to ambiguity or were not explicitly labelled during the dataset curation process. Each image within the dataset was consistently resized and displayed. By resizing all pictures to the same measurements, variations in image sizes were moderated, in this manner encouraging effective model training and evaluation.



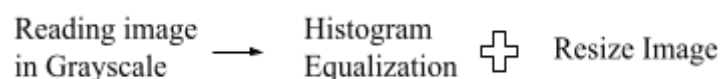
(iii) Data Description

4.Data Cleaning-

To boost the dataset's consistency and quality, data cleaning techniques were used. One of the key preprocessing steps included utilising OpenCV's `equalizeHist` function. This method improved the contrast and brightness of the images by equalising their histogram distributions. By normalising the intensity levels over different images, the preprocessing step aimed to standardise the visual characteristics of tongue images, making them more appropriate for automated analysis and classification by the CNN model.

Feature Engineering-

Feature engineering in this context focused on preparing the dataset for effective model training. Utilising OpenCV and NumPy, images were resized to the predetermined dimensions of 50x50 pixels. The resizing process was vital for making a uniform input format that the CNN model could process consistently. Furthermore, by leveraging NumPy arrays, the data was effectively managed and controlled to ensure compatibility with the model's input necessities.



(iv) Data Cleaning

5.Normalization -

Normalisation was applied to the dataset to scale the pixel values of the images between 0 and 1. Through scaling the resulting pixel frequencies by the highest possible pixel intensity value of 255.0, that uniformity occurred. Normalisation is fundamental in machine learning tasks because it helps in stabilising the learning process, progressing merging speed during model training and improving the overall effectiveness and execution of the CNN model.

```
X = X/255.0  
y=np.array(y)
```

(v)Normalization

6.Data Splitting -

The dataset was divided into training and validation sets using the `train_test_split` function from the `sklearn.model_selection` module. This splitting process ensures that the CNN model can be trained on a subset of the data whereas holding another subset for independent

validation. A split portion of 90% training, 10% validation was utilised to guarantee adequate training data whereas permitting robust evaluation of the model's execution on unseen data. The validation set played a critical part in checking the model's generalisation capacity and recognizing potential overfitting during training.



(vi) Data Splitting

3.2 Model Selection

Existing Models for Tongue Image Classification in Diabetes Detection -

Tongue image classification for diabetes detection has seen different approaches and models created to analyse visual indicators related with diabetic conditions. Here are detailed descriptions of existing models and techniques:

1.Convolutional Neural Networks (CNN)-

Automates feature extraction from tongue images utilising convolutional layers.

Applies pooling layers to decrease dimensionality and capture key highlights.

Effective in recognizing subtle visual indicators of diabetic conditions on the tongue.

In use for its capacity to memorise hierarchical information in data.

2.Transfer Learning-

Modifies CNN models that have already been trained (ResNet, VGG) using large datasets such as ImageNet. Transfers information learned from general image recognition tasks to tongue image classification. Improves model execution and accelerates training by leveraging pre-learned features. Very helpful for medical imaging situations involving limited labeled data.

3.Support Vector Machines (SVM)-

Uses manually created features as in texture, colour, also shape descriptors in tongue photos. Classifies images by finding an ideal hyperplane that maximises partition between diabetic and non-diabetic classes. Gives robust execution and interpretable results, appropriate for assignments requiring explicit feature designing.

4.Ensemble Methods-

Combines predictions from different base classifiers (e.g., Random Forests, Gradient Boosting Machines). Improves classification precision by aggregating different model

outputs. Diminishes overfitting and upgrades model strength by leveraging different model structures or data subsets. Useful in moderating biases and errors, in this manner increasing reliability in diabetes detection utilising tongue pictures.

5. Deep Learning Architectures Beyond CNNs-

Involving progressed models as recurrent neural networks, attention mechanisms, and GANs. RNNs are reasonable for successive data processing in time-series tongue image analysis. Attention mechanisms focus on important picture regions vital for recognizing diabetic conditions. GANs contribute to data augmentation and synthesis, improving model generalisation and execution.

6. Traditional Machine Learning Approaches-

Utilises calculations such as calculated regression, decision trees, and k-nearest neighbours (KNN). Uses physically built features to classify tongue images based on diabetic status.

Gives simplicity, computational efficiency, and interpretable results, complementing deep learning strategies. Reasonable for scenarios with restricted labelled data or specific computational constraints.

Disadvantages of All Existing Models-

1. Manual Feature Engineering -

Requires master information and domain-specific experiences to manually extract important features from tongue images. Subjective feature selection can lead to biases and problematic execution over different datasets. Time-consuming and labour-intensive process, particularly for large-scale datasets.

2. Limited Generalization -

Traditional ML models like SVMs may struggle with capturing complex, high-dimensional patterns and varieties in tongue images. Transfer learning from pre-trained models (VGG, ResNet) may not completely adjust to the particular characteristics and subtleties of tongue images utilised in diabetic detection. Generalisation may be compromised when models are trained on datasets that don't sufficiently represent the variability of real-world tongue images.

3. Complexity and overfitting -

Deep learning models may need interpretability, making it challenging to understand which features contribute most to diabetic condition detection from tongue images. Complex models, including ensemble methods, are vulnerable to overfitting when prepared on little or noisy datasets. Overfitting can lead to poor generalisation and decreased model execution on

unseen tongue image data.

4.Data Dependency and Variability-

Existing models may battle with strength to varieties in tongue image data, including differences in lighting, scale, and positioning. Execution can degrade when models experience data exceptions or samples that don't adjust to expected patterns.

5.Transparency-

Traditional ML models regularly give more interpretable results compared to profound learning approaches. Profound learning models, while compelling, may need transparency in decision-making processes, posing challenges in understanding model expectations in medical contexts.

6.Scalability and Resource Requirements-

A few models, particularly profound learning structures past CNNs, require substantial computational resources for training and deduction. Scalability can be constrained by hardware constraints or the need for specialised computing frameworks.

Explanation for choosing cnn-

1.Effective feature learning -

CNNs are built to automatically extract hierarchical characteristics from tongue image raw pixel data.

By effectively capturing both local and global patterns in images, they reduce the need for considerable human feature engineering.

2.Flexibility in Relation to Medical Imaging -

Tongue image analysis is one of the many medical imaging tasks in which CNNs have proven to perform well.

They handle differences in tongue shape, lighting conditions, and diabetes markers with good generalisation to a variety of complicated and different picture datasets.

3.State of the art Performance -

CNN designs are constantly being improved with the addition of deeper networks and specific layers (such attention mechanisms) to improve their feature extraction performance.

CNNs reliably produce cutting-edge outcomes in the analysis of medical images, guaranteeing excellent accuracy in the identification of diabetes from images of the tongue.

4.Capability for Feature Extraction -

Using hierarchical models to identify appropriate patterns in tongue images, CNNs are highly proficient at automatically extracting features from raw pixel data.

Justification for choosing model-

1. Practical Possibility -

CNNs can be used in real-time or near real-time in clinical applications because they balance computational efficiency and performance. Their practical deployment in healthcare environments is made easier by their scalability and adaptability to varying dataset sizes and computing resources.

2. Community Resources and Support -

Wide-ranging community support which includes pre-trained models, frameworks (like TensorFlow and PyTorch), and an array of literature and resources—is advantageous for CNNs. This assistance simplifies the use of CNN-based models for tongue image classification in the diagnosis of diabetes by simplifying their implementation, optimization, and ongoing development.

3. Progress and Research Area -

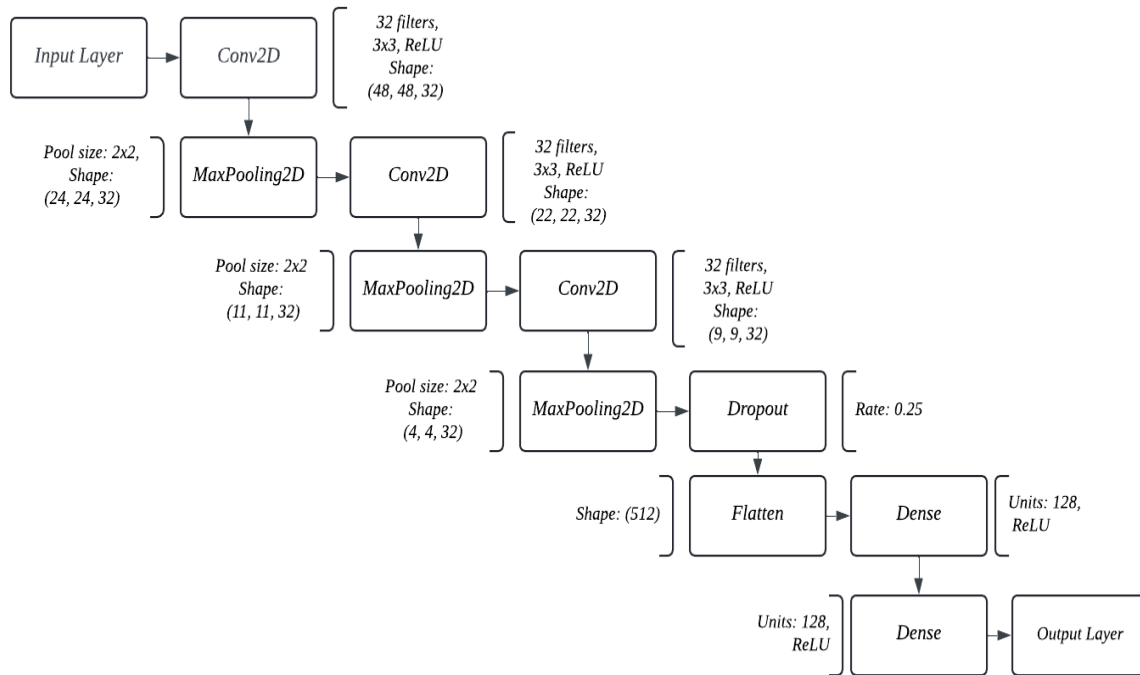
The effectiveness and usefulness of CNN architectures in medical imaging are continually being improved by ongoing research advancements. CNNs continue to lead the way in novel approaches to automated detection of diabetes conditions thanks to commercial funding and a strong research community.

4. Clinical Relevance and Accuracy -

Clinical research and comparisons verify CNNs' consistent high accuracy in medical image analysis. Reliable tongue image-based diabetes diagnosis improves patient outcomes and assists in clinical decision making.

3.3 Model Architecture

A convolutional neural network (CNN) was used in this experiment to categorise tongue images into: diabetic, non-diabetic, and unlabeled. Because CNNs can automatically and adaptively learn spatial hierarchies of features through backpropagation by utilising several building blocks, including as convolution layers, pooling layers, and fully connected layers, they are especially suited for image classification applications. The unique patterns and characteristics of tongue images that are indicative of diabetic situations are successfully captured by this design.



(i) Model Architecture

(a) Sequential Model() -

The architecture is built using the Sequential() model from the Keras library. In the Sequential() paradigm, every layer consists of a linear stack of tensors, with a single input and output tensor for every layer. Neural network construction, configuration, and training are made simple by this simplicity. When building feed-forward networks, like CNNs, where data passes through the network in a single direction from the input layer to the output layer without any loops, the Sequential() architecture is especially well-suited.

(b) Convolutional Layers:

In order to extract spatial hierarchies of information from a CNN, these layers are essential. The primary layer captures basic patterns like edges and textures, whereas consequent layers capture more complex features. Utilising three convolutional layers strikes a balance between capturing complex details and keeping up computational efficiency. Compared to fully connected layers, convolutional layers have fewer parameters since they use shared weights, which decreases overfitting in the model. They aid in the model's training of translation-invariant features, which enable it to identify objects in the image no matter where they are located.

→ Conv2D(32, (3, 3), input_shape=A.shape[1:]):

The model starts with a convolutional layer with 32 filters, each of measure 3x3, and the input shape indicated by the shape of the input data. This layer captures the spatial features of the input images.

→ Conv2D(32, (3, 3)):

Another convolutional layer with 32 filters, further refining the features extracted from the primary layer.

(c)ReLU Activation Functions:

It exhibits non-linearity, which allows the model to learn intricate patterns from the data. They are computationally productive and help relieve the vanishing gradient issue, improving model training. The model would behave like a simple linear classifier in the absence of non-linearity, making it incapable of memorising complex patterns without ReLU. Training becomes more efficient since they compute more quickly than other activation functions like sigmoid or tanh.

→ Activation("relu"):

It demonstrates non-linearity and enables the network to learn more intricate patterns by applying the Rectified Linear Unit activation function.

(d)Max Pooling Layers: By reducing the spatial dimensions of the data, these layers greatly decrease the computational load and number of parameters. By offering a type of spatial invariance that is, guaranteeing the model can identify features independent of their position in the image pooling can also help in the control of overfitting.

Dimensionality Reduction: They help in reducing the number of computations and parameters in the network by reducing its spatial size. This results in quicker training times. This results in quicker training times.

Feature Prioritisation: By prioritising the most significant characteristics found by the convolutional layers, max pooling helps the model concentrate on the most important elements of the data.

→ MaxPooling2D(pool_size=(2, 2)):

By lowering the spatial dimensions by a factor of two, this max pooling layer lowers the computational cost and manages overfitting.

(e)Dropout Layer:

Dropout could be a regularisation method that avoids overfitting by randomly dropping units during training. Increasing model's performance in use of unknown data.

Overfitting Prevention:

By randomly setting a division of input units to zero during training, dropout layers help in making a stronger model that performs well on unused, unseen data.

Model Ensemble:

Dropout can be seen as training an ensemble of numerous different smaller networks, which progresses the generalisation capability of the model.

→ Dropout(0.25):

A rate of 0.25 for excluding overfitting by randomly keeping 25% of input units as 0 during every update cycle.

(f) Flatten Layer:

This layer changes over the 2D matrix of features into a 1D vector, making it appropriate for the fully connected layers, which are dependable for making predictions based on the extracted features.

Compatibility:

Fully connected layers require a 1D input, so flattening the data from the convolutional layers is vital to transition from spatial data to a format reasonable for classification.

→ Flatten ():

Flattens the input from the convolutional and pooling layers into a one-dimensional array, making it appropriate for the completely connected layers.

(g) Fully Connected (Dense) Layers:

These layers perform high-level thinking about the features. The two thick layers with 128 units each permit the model to combine features in a way that produces precise classifications. The final dense layer with 4 units outputs the classification probabilities.

Complex Feature Combination:

Dense layers combine the features extracted by convolutional layers in complex ways, enabling the model to memorise higher-level representations.

Classification Control:

The final dense layer with softmax activation translates the high-level learned features into probabilities for each class, making it suitable for multi-class classification tasks.

→ Dense(128):

Completely connected (dense) layer with 128 units to combine the features extracted by the convolutional layers.

→ Dense(4):

Output layer with 4 units, compared to the three categories (diabetic, non-diabetic, unlabeled)

and an additional class for robustness.

(h) SoftMax Activation in Output Layer:

The softmax function is utilised within the last layer to convert the logits into probabilities, which helps in making choices based on the highest probability.

Probabilistic Translation:

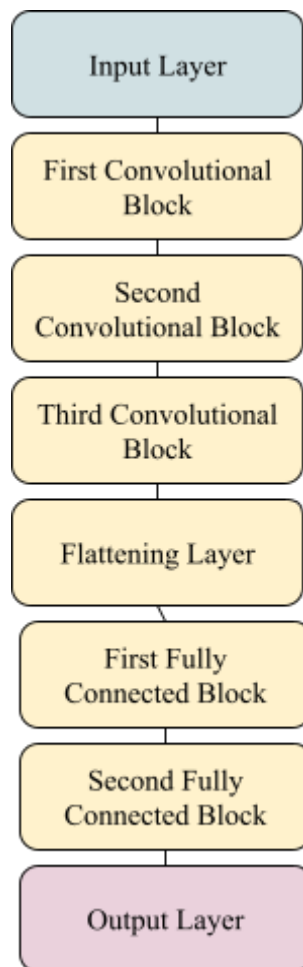
Softmax gives a probabilistic translation of the outputs, making it simpler to understand the certainty of the model in its predictions.

Multi-Class Classification:

Softmax is particularly suited for multi-class classification problems, where it ensures that the sum of the output probabilities is 1, making it simple to translate and compare the predicted probabilities.

→ Activation("softmax"):

It is to provide probability distributions over output classes.



(ii) Model Layers

3.4 Model Training

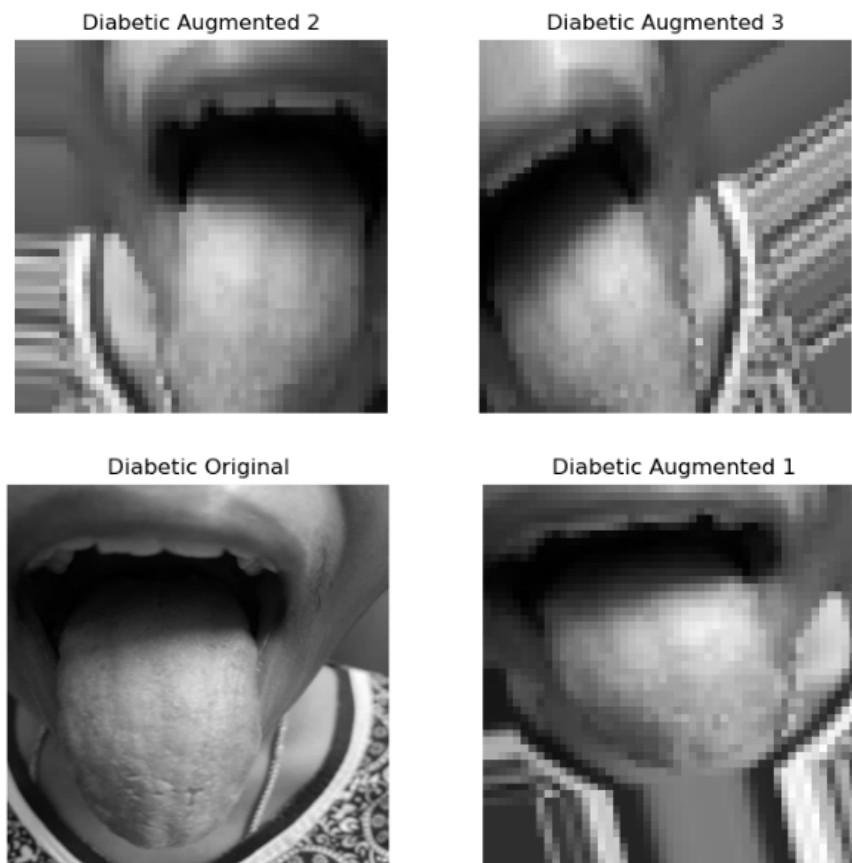
Training Methodology-

1.Data Augmentation:

To improve the training data and the model's generalisation ability, we implemented TensorFlow's ImageDataGenerator. These covered methods including rotations, horizontal flips, zoom adjustments, width and height shifts, and shear transformations. By artificially expanding the dataset, these augmentations help the model learn invariant properties, which enhances its performance on unknown information.

Augmentation Parameters -

Parameters such as shear range, width and height shift range, zoom range, rotation range, and horizontal flip were experimentally chosen to present sufficient variability within the training information without excessive distortion.



(i) Data Augmentation

2.Data Splitting:

The dataset was split into training and validation sets utilising `train_test_split` from `sklearn.model_selection`, with 90% distributed for training and 10% for validation. This part

guarantees that the model's execution is evaluated on unseen data, making a difference to avoid overfitting.

3.Batch Processing:

The training data was prepared through the ImageDataGenerator pipeline, creating batches of augmented images for the training process. The model was prepared utilising batches of 64 images, optimising computational efficiency and model performance. Batch Normalization reduces the problem of internal covariate shift, which happens when the distribution of inputs to a layer changes during training. In order to normalize the activations of the next layer, batch normalization layers are added to the model. This occasionally improves overall performance and speeds up convergence.

4.ReduceLROnPlateau:

A callback named ReduceLROnPlateau lowers the learning rate when a statistic stops improving. Through the use of this technique, the learning rate can be adjusted during training, potentially improving the stability and performance of the model. Lowering the learning rate can facilitate more effective model convergence in the event that it stops improving. If no improvement is observed after a predetermined number of epochs, this callback lowers the learning rate by a factor and keeps an eye on certain metrics (such as validation loss).

5.Early Stopping:

Early Stopping is a technique for preventing overfitting by stopping the training process when performance on the validation set begins to decline. By doing this, it is ensured that the model will not learn noise from the training set and will continue to be able to generalize effectively to new sets of data. EarlyStopping saves time and computing resources by keeping an eye on a predetermined metric and stops training if no improvement is seen after a certain number of epochs.

6.ModelCheckpoint:




A callback called Model Checkpoint is used to save the model at particular stages of training. This is helpful for storing the best model that was shown during training, which may then be loaded for more research. To ensure that the best model is saved, this callback saves the model to disk when a monitored measure (validation accuracy) improves.

7.LearningRateScheduler:

It dynamically adjusts the learning rate according to a predetermined schedule. We can improve the training process and enable the model to learn more effectively at first and improve over time by varying the learning rate. This callback helps in efficiently controlling the training dynamics by modifying the learning rate with the epoch number or any user-defined function.

8. Training Iterations:

The model experienced 50 epochs of training, where it iteratively adjusted its weights to minimise the loss function, `sparse_categorical_crossentropy`. This loss function is well-suited for multi-class classification tasks, ensuring each instance is precisely classified into one of the predefined categories.

`rotation_range`  `height_shift_range`  `zoom_range`  `fill_mode`
`width_shift_range` `shear_range` `horizontal_flip`

(ii) Parameters

Hyperparameter Tuning -

Optimizer:

The “adam” optimizer was chosen for its adaptive learning rate capabilities, encouraging quicker convergence and improved performance.

Batch Size:

A batch size of 64 was chosen to adjust computational efficiency and model execution.

Epochs:

The number of epochs was set to 50, giving an ideal balance between training time and model accuracy.

Loss Function:

The “`sparse_categorical_crossentropy`” is used to ensure that each instance is properly classified into one of the established categories.

Tools and Frameworks Used -

TensorFlow and Keras:

The execution and training of the CNN model were executed utilising TensorFlow and its high-level API, Keras. For deep learning applications, TensorFlow offered essential computational flexibility and efficiency. Keras offered a natural and user-friendly interface for characterising and training neural networks.

ImageDataGenerator:

The ImageDataGenerator class from `tensorflow.keras.preprocessing.image` played a crucial part in expanding the dataset, ensuring the model was trained on a different set of images.

Model Training Pipeline:

The whole training pipeline was managed through the `MODEL.fit` method, which handled data feeding, forward and backward passes, and weight upgrades.

In order to increase and batch the training data, this technique consistently works with the Image Data Generator.

Scikit-learn:

Instruments from the `sklearn` library, such as `train_test_split`, were utilised for data splitting and preprocessing, ensuring an efficient and organised approach to model training.

CHAPTER 4

RESULTS

4.1 Model Performance

Presentation of Results -

1. Training and Validation Accuracy: The model's accuracy on the training and validation datasets was monitored during the training phase. Validation accuracy proves how effectively model generalises to unknown data, training accuracy proves how best the model is adapting the features of training data. These accuracy measures reveal information about the training process's efficiency and the level of learning over the 20 epochs. This required taking accuracy measures out of `history.history['val_accuracy']` and `history.history['accuracy']`.

2. Training and Validation Loss: The difference between the expected and actual class labels was measured using the `sparse_categorical_crossentropy` loss function. The model's fit to the training set is indicated by the training loss, while its performance on the validation set is indicated by the validation loss. Finding problems like overfitting or underfitting is made easier by monitoring these parameters over time. The extraction of loss metrics was made easier by the TensorFlow library using `history.history[{'loss'}]` and `history.history['val_loss']`.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total Number of predictions}}$$

(i) Accuracy

Performance Metrics and Their Interpretation -

1. Accuracy: A key performance indicator for the model was its accuracy. It can be expressed as the ratio of properly predicted instances to all instances. While high validation accuracy shows effective generalisation to new data, high training accuracy shows that the model has learned the training data well. The accuracy patterns throughout epochs can reveal if the model is overfitting, plateauing, or improving.

2. Loss: model's prediction error is given in terms of loss metric. A decreasing validation loss denotes better performance on the validation data, whereas a lowering training loss implies the model is learning to eliminate errors on the training set. If the model is underfitting (both losses are high) or overfitting (training loss is significantly smaller than

validation loss), it can be determined by examining the relationship between training and validation loss.

3. Precision, Recall, and F1 Score -

These metrics give a thorough picture of how well the model classified each category—Diabetic, Non-Diabetic, and Unlabeled—and how accurate it was.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

$$\text{F1 Score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

(ii) Precision, Recall, and F1 Score

Precision: Precision measures how well the model predicts the good outcomes. The ratio of actual positive forecasts to all positive predictions is used to calculate this. Good accuracy is correlated with a low false positive rate.

Recall: The model's recall measures how well it can identify every positive sample. Defined as a portion of actual positives to true positive predictions. In cases where a high-meaning model efficiently extracts all meaningful instances from the dataset.

F1 Score: It is a balanced data which considers both false positives and false negatives. A harmonic average of recall and precision is used to calculate it. When there is an imbalance in the distribution of classes, it is very helpful.

4.2 Comparison with Existing Methods

Feature	Traditional Methods	Support Vector Machine	Random Forest	CNN(Our Model)
Invasiveness	Invasive(requires blood samples)	Non-invasive	Non-invasive	Non-invasive
Feature Engineering	Not applicable	Moderate feature engineering needed	Extensive feature engineering needed	Extensive feature engineering needed
Interpretability	High	Moderate	Moderate	Moderate
Training time	Not applicable	Moderate	Moderate to long	Long
Prediction Time	Not applicable	Moderate	Moderate	Fast
Accuracy	High(gold standard)	High	High	High
Scalability	Limited to clinical settings	Moderate	Moderate	High
Type of Data	Blood Glucose levels,HbA1c, etc.	Numerical/Categorical features	Numerical/Categorical features	Tongue Images
Computational Resources	No	High	High	Very High
Non-Invasive Detection	No	Yes	Yes	Yes
Requirement for large dataset	No	High	High	High

CHAPTER 5

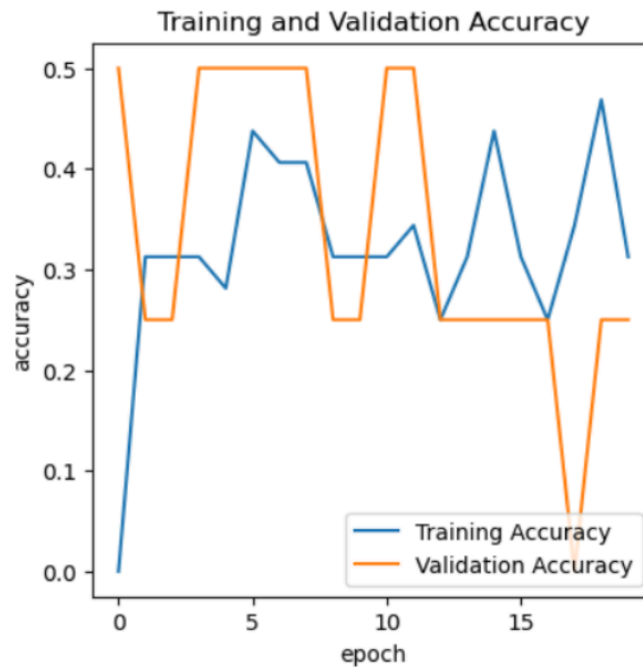
DISCUSSIONS

5.1 Analysis of Results

Graphs and Charts Showing Model Performance

1. Training and Validation Plot for Accuracy-

The training and validation accuracy over the course of 20 epochs were represented visually by a line plot. This figure assists in visualising the model's learning path. The model is learning and generalising successfully when, in principle, both the training and validation accuracy rise and converge towards a high value.



(i) Accuracy Plot

The precision is shown on the y-axis, while the number of epochs is represented on the x-axis. Two lines are plotted in the plot: one for training accuracy and the other for validation accuracy. Labels denoting corresponding data.

2. Loss Plot for Training and Validation:

The training and validation losses over the epochs were also shown using a line plot. To identify problems with the model's learning process, this figure is necessary. Both losses have to decline, with validation loss ideally reaching training loss



(ii) Loss Plot

The number of epochs is shown on the x-axis, while the loss is shown on the y-axis. Two lines are plotted in the plot—one for the training loss and one for the validation loss—with labels displaying the corresponding data.

3. Confusion matrix:

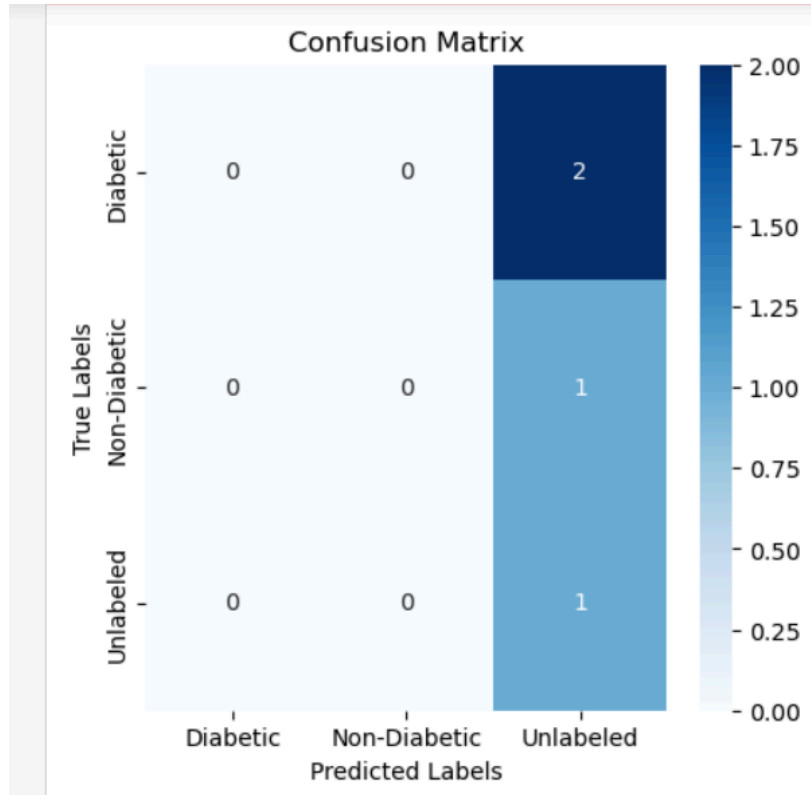
The confusion matrix is an essential tool for assessing our diabetes classification model, offering comprehensive insights into its performance. It shows four key metrics: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) for each class: Diabetic, Non-Diabetic, and Unlabeled.

True Positives (TP): These are cases that have been correctly identified as diabetic; a high TP count indicates the model's strong ability to detect diabetic tongues, which is vital for early diagnosis.

True Negatives (TN): These are correctly identified non-diabetic cases; high TN values demonstrate the model's accuracy in identifying non-diabetic tongues, thereby minimising unnecessary concern.

False Positives (FP): These arise when non-diabetic cases are mistakenly classified as diabetic.

False Negatives (FN): These are instances of diabetes that were mistakenly diagnosed as non-diabetic. Reducing FN is essential since it means that chances for early discovery, which is necessary for successful therapy, are lost.



(iii) Confusion Matrix

The model does a good job at distinguishing between cases that are diabetic and those that are not, but there is still opportunity to lower the number of false positives and negatives. These observations help us refine the model by directing changes to its architecture or focused data augmentation.

5.2 Limitations

- 1.Data Dependence: The quantity and quality of the dataset used for training have a significant impact on the model's performance. Lower accuracy in real-world applications could result from the dataset utilised in this solution not being comprehensive or diverse enough to represent all differences in tongue appearances.
- 2.ModelComplexity: This implementation makes use of a relatively simple CNN architecture; more complex architectures might result in better performance, but they would also necessitate more time and computational resources for training.
- 3.Overfitting: The show might overfit the preparing information, particularly on the off chance that the number of ages is tall and the demonstrated complexity is insulant regularised. This would result in tall preparing exactness but no approval precision.

4. Constrained Categories: The application as of now underpins as it were three categories: Diabetic, Non-Diabetic, and Unlabeled. This oversimplified categorization might not capture the complete range of tongue characteristics related to distinctive stages or sorts of diabetes.

5. Maintenance and Updates: The current usage might require visit upgrades and support to keep the show and application adjusted with modern information and progressions in machine learning methods. Without customary upgrades, the application might end up obsolete and less compelling over time.

CHAPTER 6

GUI APPLICATION DEVELOPMENT

6.1 Overview

Using a GUI, this tool may identify diabetes by classifying photos of the tongue. It preprocesses photos, uses a trained Convolutional Neural Network (CNN) to make predictions, and offers a user-friendly interface for uploading and analysing tongue images.

6.2 Description

The tkinter library is used to create the diabetes detection application's GUI, which offers a convenient means of interacting with the machine learning model. Users can choose which tongue photos to classify by using the main window. An image is displayed in the GUI after it has been selected via the file dialog. The model then analyses the image to determine if it is in the unlabeled, diabetic, or non-diabetic category. Next, the GUI displays the predicted result. The interface uses TensorFlow to load the trained convolutional neural network model and make predictions, while OpenCV and PIL are utilised for picture preprocessing and display. Users may easily use the model without having to write any code due to its simple layout.

6.3 Implementation

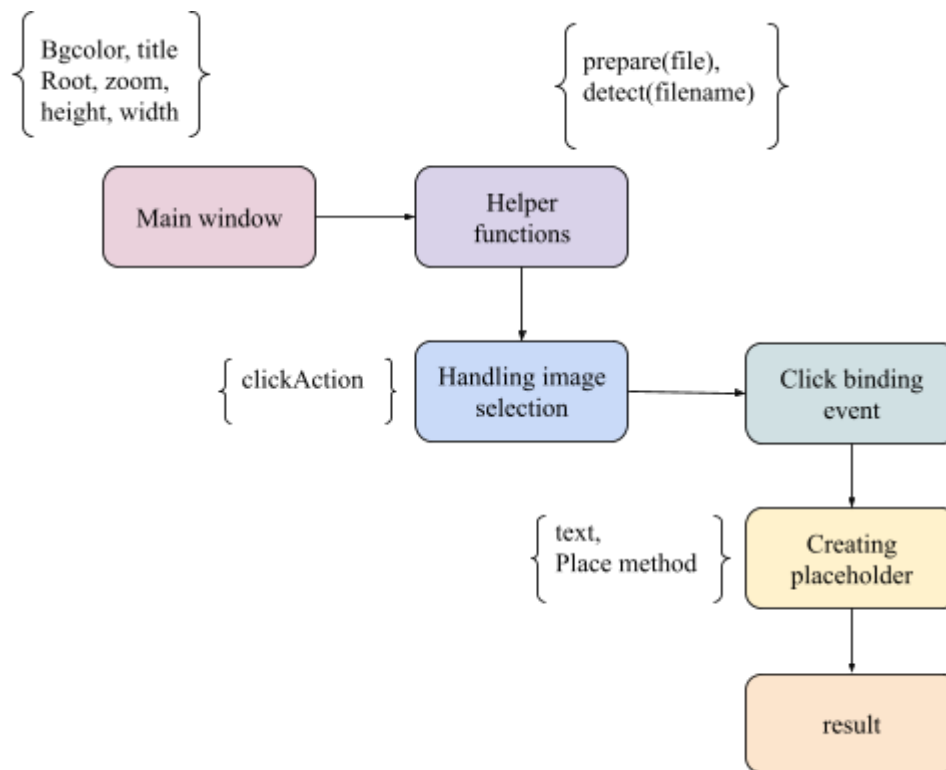
A placeholder is used in the GUI interface to ask the user to click and choose an image file. This improves the user experience by making it obvious where to engage. This is a thorough explanation of the placeholder's function:

1. Initialize the main window:

Creating the main window of our application using Tk(), having a title and configuring it to start in a maximised state being set with a background colour which is made resizable at width, height for different screen capacity also resolutions. To store the prediction result we initialise a StringVar, to load the pre-trained convolutional neural network (CNN) model using TensorFlow's load_model.

2. Preprocessing and Prediction:

The selected image is preprocessed using the prepare function, which also applies histogram equalization and resizes it to the 50x50 pixel input size that our model requires. The detect function changes the prediction result and prints it to the console for debugging after using the trained model to predict the image's category.



(i) GUI process

3. Handling Image Selection:

The function ClickAction manages the process of choosing a picture. It presents a file dialog where the user can choose an image, and then it opens and resizes the chosen image using PIL. The primary window now shows the scaled image. The detect function is called to classify the image once it has been displayed.

- File dialog to choose an image file is opened using `filedialog.askopenfilename()`.
- `PIL.Image.open`: Provides access to the chosen image file.
- Resizing by `image.resize` for it to fit in display area.
- `PIL.ImageTk.PhotoImage`: Transforms to a Tkinter format.
- PANEL: A fresh label that will show the chosen picture in the main window.
- The DETECT function applies the CNN model to the chosen image in order to classify it.

4. Creating the Placeholder:

Choosing an image by user, a placeholder label having written "Click here to choose an image" is obtained. This label is designed to be easily readable and clickable, with a grey background. The place method is used to position it in the main window.

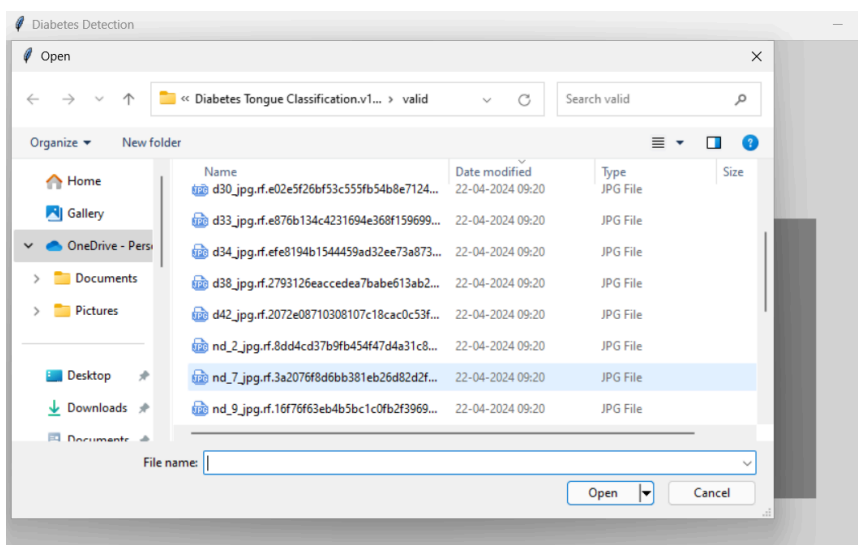
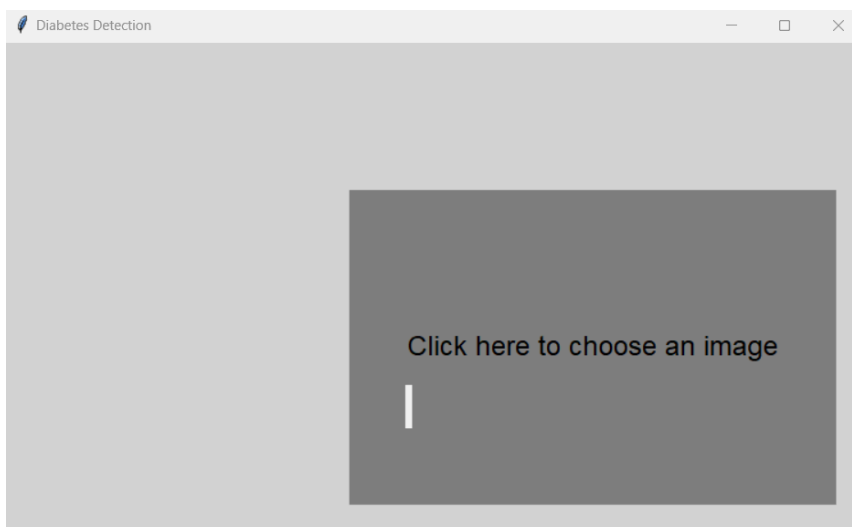
5. Click Binding Event:

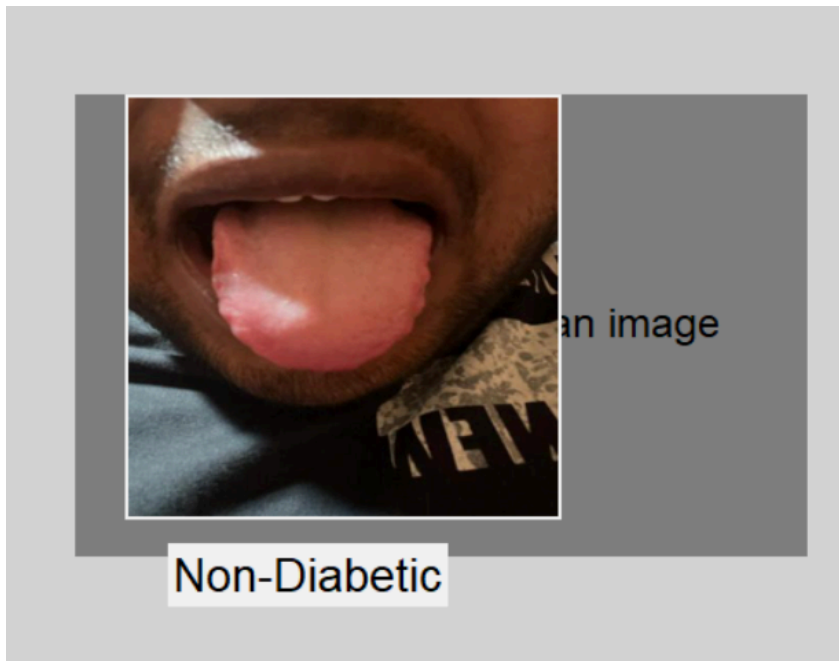
We associate the placeholder label with a click event. The user can choose an image file from their system by clicking the label, which activates the ClickAction function.

6.Displaying the Result:

To show the prediction result, we build a label. Every time a new image is chosen and processed, this label is constantly updated with the categorization result. We initiate the tkinter main loop, which maintains the GUI's speed and availability for user input. The loop maintains the open and working state of the application window.

- Textvariable: for connecting displayed text in Label to the value of the variable.
- The Label's text will be automatically updated anytime this StringVar's value changes.
- Location Method: Place the label for the result in the window.





(ii) GUI

CONCLUSION

The implementation of image classification for diabetes using tongue images leverages machine learning techniques effectively. The project begins with data preprocessing steps including grayscale conversion, histogram equalization, and resizing, ensuring uniformity across the dataset. Through TensorFlow and Keras, a convolutional neural network (CNN) architecture is constructed and trained on a dataset consisting of diabetic, non-diabetic, and unlabeled tongue images. The model is optimised using data augmentation techniques and validated using a portion of the dataset reserved for validation.

The training process is monitored using metrics such as accuracy and loss, visualised over epochs to assess model performance. The final trained model is saved in formats suitable for deployment, including JSON for architecture and HDF5 for weights. A graphical user interface (GUI) using tkinter facilitates real-time image classification, allowing users to upload tongue images and receive predictions promptly.

Key Achievements:

- 1.Data Handling: Efficient management of image data through preprocessing techniques like histogram equalization and resizing.
- 2.Model Construction:Sequential CNN architecture designed for image classification tasks, featuring convolutional, activation, pooling, dropout, and dense layers.
- 3.Training and Evaluation: Utilisation of TensorFlow's capabilities for training, validation, and visualisation of model performance metrics.
- 4.Deployment Readiness: Serialization of the model for easy deployment in real-world applications, supported by a user-friendly GUI for intuitive interaction.

FUTURE WORK

1. Data Expansion and Quality Improvement:

Increase the dataset size and diversity to encompass a broader range of tongue images, including variations in demographics and conditions related to diabetes.

2. Integration and Deployment Optimization:

Explore deployment options beyond local GUI applications, including web-based interfaces or integration into healthcare management systems, ensuring scalability and accessibility.

3. Clinical Validation and Collaboration:

Conduct rigorous clinical trials in collaboration with healthcare professionals to validate the model's diagnostic accuracy and reliability in diverse patient populations.

4. User Interface and Experience Refinement:

Enhance the GUI with features like batch processing, image preprocessing options, and interactive visualisation of classification results to improve usability and practicality in clinical settings.

5. Continuous Learning and Adaptation:

Stay updated with advancements in machine learning and healthcare technologies to incorporate state-of-the-art methodologies and address emerging challenges in diabetes diagnosis.

REFERENCE

[1]"Assessing Diabetes thru Machine Learning related to Tongue Picture Segmentation" by Xiaohui Lin and Weina Liu, Zhaochai Yu.

[2]"Type 2 diabetes Inspections Protocol Hiring Tongue Grouping of Pictures By applying Machine Learning Networks" by M. Bharathi, Dr. D. Prasad, Dr. T. VenkataKrishnaMoorthy, and Dr. M. Dharan.

[3]Dr. S. Bhuvaneswari's article, "Diabetes screening integrating tongue image utilizing identification of global aspects and decision tree,"

[4]"Using tongue image classification and machine learning algorithms, a diabetes diagnostic method" was put together by M. Bharathi, Drs. D. Prasad, T. Venkata Krishnamoorthy, and M. Dharani.

[5]"Diagnostic Method of Diabetes Based on Support Vector Machine and Tongue Images" by Jianfeng Zhang, Jiatuo Xu, Xiaojuan Hu, Qingguang Chen, Liping Tu, Jingbin Huang, Ji Cui.