*A project report on*

# EYE DISEASES DETECTION

*Submitted in partial fulfillment for the J component Review of*

## Master of Technology in Software Engineering (Integrated)

*by*

**RAJ KOYANI-21MIS1017**

**CHAITANYA GHOLAP-21MIS1079**



**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

April, 2024

## CONTENTS

**CHAPTER 1**

**INTRODUCTION**

* 1. Introduction……………………………………………………………………….3
  2. Project Overview………………………………………………………………….3

**CHAPTER 2**

**TECHNOLOGIES USED**

3.1 Technologies Used……………………………………………………………….6

**CHAPTER 3**

**PROPOSED SYSTEM**

4.1 Proposed System…………………………………………………………………8

**CHAPTER 4**

**IMPLEMENTATION**

5.1 Implementation…………………………………………………………………...14

**CHAPTER 5**

**RESULTS**

6.1 Results……………………………………………………………………………16

**CHAPTER 6**

**CONCLUSION AND FUTURE WORK**

7.1 Conclusion and Future Work…………………………………………………….20

**Chapter 1**

**Introduction**

* 1. **Introduction**

This project aims to develop a deep learning model for the accurate classification of eye diseases using state-of-the-art neural networks - Inception V3, VGG19, and Xception - via transfer learning. Leveraging their sophisticated architectures trained on vast datasets like ImageNet, these models offer efficient processing and feature extraction capabilities crucial for identifying subtle signs of eye diseases. Through transfer learning, the models are fine-tuned on a specific dataset of eye images, adapting their feature detection to the nuances of eye disease diagnostics. This approach enhances diagnostic accuracy, reduces training time, and ensures robustness against variations in new data, facilitating early and precise disease detection in clinical settings. The project's goal is to bridge advanced AI techniques with practical medical applications, fostering significant improvements in ophthalmology.

* 1. **Project Overview**

**Motivation for the Research**

The motivation for this research stems from the critical need for early and accurate diagnosis of eye diseases, which are major causes of visual impairment and blindness worldwide. Early detection is pivotal in preventing severe outcomes and enhancing the effectiveness of treatment options. Traditional diagnostic methods rely heavily on manual examination by specialists, which can be time-consuming, subjective, and inaccessible in underserved areas. Therefore, there is a pressing need for more scalable, objective, and accurate diagnostic methods.

Artificial intelligence (AI), particularly deep learning (DL), has emerged as a revolutionary technology in medical imaging, providing unprecedented accuracy and efficiency. The use of advanced neural networks for the classification of eye diseases could automate and improve diagnostic processes, making eye care more accessible and precise.

**Problem Statement**

The primary goal of this project is to develop an accurate and robust deep-learning model for the classification of various types of eye diseases, specifically focusing on Normal, Cataract, Diabetic Retinopathy, and Glaucoma. The utilization of state-of-the-art transfer learning techniques, including Inception V3, VGG19, and Xception V3, is a key strategy for achieving superior performance in image analysis and disease classification.

**Objectives**

* The primary objective of this project is to develop a deep learning model that can classify various eye diseases with high accuracy and efficiency. The specific goals are:
* To leverage the capabilities of top-tier neural networks, such as Inception V3, VGG19, and Xception, for the task of eye disease classification.
* To utilize transfer learning to adapt these pre-trained models to the specific characteristics of ocular diseases, thereby enhancing their diagnostic capabilities.
* To reduce the time and computational resources required for model training, while maintaining or improving the accuracy compared to traditional diagnostic methods.
* To evaluate the models' performance in real-world clinical settings, ensuring they can handle variations in new patient data.

**Methodology (Available and Adopted)**

**Available Methodologies**

Several methodologies exist for medical image analysis, ranging from traditional machine learning techniques to advanced deep learning models. Traditional methods often involve feature extraction followed by classification using algorithms such as support vector machines (SVM) or random forests. However, these methods typically require manual feature selection and are limited in their ability to capture complex patterns in data.

**Adopted Methodology**

This project adopts a transfer learning approach using deep neural networks, which have been pre-trained on the ImageNet dataset—a large, diverse dataset of general images. This provides a robust initial set of features that can be fine-tuned to specific tasks. The specific networks chosen for this project are:

**Inception V3:** Known for its mixed convolutional layers, allowing it to capture image details at various scales—an essential feature for identifying subtle indicators of eye diseases.

**VGG19:** With its deep convolutional layers, it is particularly adept at extracting textural and detailed features from images, making it suitable for the fine-grained analysis required in medical imaging.

**Xception:** Utilizes depthwise separable convolutions, which allow for efficient model training and adaptation, combining the strengths of Inception with improved computational efficiency.

The methodology involves fine-tuning these pre-trained networks on a dataset of labeled eye images. This dataset includes various common and rare eye conditions, ensuring the model learns to detect a wide range of diseases. The models' performance will be assessed based on their accuracy, precision, recall, and F1-score in classifying different eye diseases.

By leveraging advanced AI techniques and transfer learning, this project aims to enhance the diagnostic processes in ophthalmology, contributing to more effective, efficient, and accessible eye care.

**Chapter 2**

**Technologies used**

**2.1 Technologies Used**

1.Deep Learning:

Deep learning techniques form the backbone of the project's methodology. Specifically, convolutional neural networks (CNNs) are utilized for image classification tasks due to their ability to automatically learn features from raw data.

2.Transfer Learning:

Transfer learning is employed to leverage pre-trained neural network architectures, namely Inception V3, VGG19, and Xception. This approach allows the models to inherit knowledge gained from training on large-scale datasets like ImageNet and adapt it to the task of classifying eye diseases.

3.Python Libraries:

TensorFlow: TensorFlow is an open-source deep learning framework developed by Google. It provides tools for building and training neural networks.

Keras: Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow. It simplifies the process of building and training neural networks.

NumPy: NumPy is a fundamental package for scientific computing with Python, providing support for large, multi-dimensional arrays and matrices.

Matplotlib: Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension, NumPy. It is used for data visualization in the project.

4.Data Augmentation:

Data augmentation techniques are applied to increase the diversity of the training dataset, which helps prevent overfitting and improves the model's ability to generalize to unseen data. Augmentation techniques may include rotation, flipping, and scaling of images.

5.Image Processing:

Image processing libraries may be utilized for tasks such as reading, preprocessing, and augmenting the image data. These libraries help in handling the raw image data efficiently.

6.Model Evaluation:

Various metrics are used to evaluate the performance of the trained models, including accuracy, precision, recall, and F1-score. These metrics provide insights into the models' ability to correctly classify different eye diseases.

7.Deployment:

While not explicitly mentioned in the project report, deployment technologies may include frameworks like Flask or FastAPI for building web-based applications, which could serve as interfaces for the trained models, allowing them to be used in real-world clinical settings.

**Chapter 3**

**PROPOSED SYSTEM**

**3.1 Proposed System**

**Data Sources:**

Link to Eye Diseases Classification Dataset:

<https://www.kaggle.com/datasets/gunavenkatdoddi/eye-diseases-classification>

**Soft Computing Models:**

For the project, the soft computing models used are advanced neural network architectures that have proven highly effective in various image recognition tasks. Here’s a breakdown of each model mentioned:

**Inception V3:**

Architecture: Part of the Inception family, this model is designed with a series of symmetric and asymmetric building blocks, including convolutions, average pooling, max pooling, concats, dropouts, and fully connected layers. The modules in Inception V3 have optimized for computational efficiency with a focus on lowering dimensionality to reduce computational burden.

Usage: Inception V3 is particularly useful for feature extraction in images due to its convolutional approach that captures spatial hierarchies in data. For eye disease classification, it can effectively differentiate between subtle variations in eye images that distinguish different types of diseases.

**VGG19:**

Architecture: This model is characterized by its simplicity, using only 3x3 convolutional layers stacked on top of each other in increasing depth. It deepens the network significantly, which increases its capacity and captures finer details in the image.

Usage: VGG19 is renowned for its effectiveness in image recognition tasks and is particularly adept at capturing textures and patterns, making it valuable for identifying specific features in eye disease imagery like the patterns indicative of Diabetic Retinopathy or the cloudiness associated with Cataracts.

**Xception V3** (likely intended as Xception, which is sometimes thought of as a 'version 3' following Inception V1 and V2):

Architecture: Xception stands for "Extreme Inception". It modifies the original Inception architecture by replacing the Inception modules with depthwise separable convolutions. This not only makes the model lighter in terms of computational cost but also improves its performance on tasks where fine-grained feature extraction is crucial.

Usage: Given its efficiency and depthwise operational strategy, Xception is excellent for scenarios where high accuracy is required without a substantial increase in computational resources. It's particularly effective in handling complex image data like those needed for Glaucoma detection where intricate details must be discerned.

**Algorithm:**

A detailed step-by-step algorithm for developing a deep learning model for the classification of various types of eye diseases:

Step 1: Setup Environment

Install necessary Python libraries such as TensorFlow, Keras, NumPy, and Matplotlib.

Set up Kaggle API for data download.

Step 2: Download and Prepare Data

Use Kaggle command-line tools to download the eye disease dataset.

Unzip the downloaded dataset to a working directory.

Structure the dataset into separate folders for each disease type for easier access and management.

Step 3: Load and Preprocess Data

Initialize paths for each disease category (Glaucoma, Cataract, Normal, Diabetic Retinopathy).

Read images from these paths and create a DataFrame containing image paths and corresponding labels.

Step 4: Visualize Data

Display sample images from the dataset to understand the data's nature.

Use seaborn and matplotlib to visualize the distribution of different classes in the dataset.

Step 5: Data Augmentation

Apply image augmentation techniques such as rotation, flipping, and scaling to increase the diversity of the training data, which helps prevent overfitting and makes the model robust to variations in new data.

Step 6: Define the Model

Choose a suitable pre-trained model (Inception V3, VGG19, or Xception) as the base for transfer learning.

Add custom layers if necessary to tailor the model to the specific task.

Compile the model with an appropriate optimizer, loss function, and metrics for evaluation.

Step 7: Train the Model

Split the data into training and validation sets to monitor the model's performance and adjust parameters accordingly.

Train the model using the training data while validating on the validation set to track performance and make improvements.

Save the best model during training using callbacks like ModelCheckpoint based on validation accuracy.

Step 8: Evaluate the Model

Plot training and validation accuracy and loss graphs to visualize the learning process and identify issues like overfitting or underfitting.

Calculate class-wise accuracy to understand how well the model performs for each eye disease category.

Step 9: Save the Trained Model

Save the final model configuration and weights to a file using model.save() method for later use or deployment.

Step 10: Model Testing

Load new eye images to test the model's generalization capability.

Preprocess the test images to fit the model's input requirements.

Use the model to predict the disease type and display the predicted label.

Step 11: Post-Training Analysis

Analyze the errors and success cases to understand where the model performs well and where it doesn't.

Consider additional tuning or retraining if the performance on certain classes is unsatisfactory.

This algorithm provides a comprehensive guide to building and deploying a deep learning model for eye disease classification, from initial setup and data handling to training, evaluation, and deployment.

**Pseudo Code:**

1. Install necessary libraries (e.g., TensorFlow, Keras, NumPy)

2. Download and unzip the dataset containing eye disease images.

3. Load and preprocess the dataset:

- Read images from directories for different diseases (Glaucoma, Cataract, Normal, Diabetic Retinopathy).

- Convert images into a suitable format for model training.

- Label the images based on their corresponding disease.

4. Data Augmentation:

- Apply transformations like rotation, scaling, and flipping to increase dataset diversity.

5. Define the CNN model:

- Use pre-trained models such as Inception V3, VGG19, or custom layers.

- Add convolutional layers, activation functions, and pooling layers.

- Compile the model with an appropriate optimizer and loss function.

6. Train the model:

- Split the data into training and validation sets.

- Fit the model on the training data while validating on the validation set.

7. Evaluate the model:

- Plot training and validation loss and accuracy.

- Perform class-level accuracy assessment.

8. Save the trained model.

9. Test the model with new images:

- Preprocess the images to fit the model's input requirements.

- Predict and display the results.

**Flowchart:**

A diagram of a process

Description automatically generated

A diagram of a process flow

Description automatically generated

**Flowchart Elements:**

1.Start:

Begin the process.

2.Setup Environment:

Install necessary libraries (e.g., TensorFlow, Keras).

3.Data Acquisition:

Download data using Kaggle API.

Unzip the dataset.

4.Data Preparation:

Load image paths and labels into a DataFrame.

Preprocess images (resize, normalize).

5.Data Visualization:

Display sample images.

Plot distribution of classes.

6.Data Augmentation:

Apply transformations (rotate, flip, scale).

7.Model Definition:

Load pre-trained model (e.g., VGG19, Inception V3).

Add custom layers if necessary.

Compile the model.

8.Model Training:

Split data into training and validation sets.

Train the model using the training set.

Validate the model using the validation set.

Use callbacks for best model saving.

9.Model Evaluation:

Plot training and validation accuracy and loss.

Calculate class-wise accuracy.

10.Save Model:

Save the trained model for later use.

11.Model Testing:

Load new images.

Preprocess and predict using the model.

12. End:

Process completes.

**Equations:**

CNNs primarily involve convolution operations followed by non-linear activation functions, pooling, and fully connected layers. The core equation in a convolutional layer is:

𝑜𝑗𝑙=𝑓(∑𝑖∈𝑀𝑜𝑖𝑙−1∗𝑘𝑖𝑗𝑙+𝑏𝑗𝑙)ojl​=f(∑i∈M​oil−1​∗kijl​+bjl​)

Where:

𝑜𝑗𝑙ojl​ is the output of the j-th feature map in the l-th layer.

𝑓f is a non-linear activation function like ReLU.

𝑜𝑖𝑙−1oil−1​ are the outputs of the feature maps from the previous layer.

𝑘𝑖𝑗𝑙kijl​ are the convolutional kernels connecting feature map 𝑖i in layer 𝑙−1l−1 to feature map 𝑗j in layer 𝑙l.

𝑏𝑗𝑙bjl​ is the bias for the j-th feature map in the l-th layer.

denotes the convolution operation.

Pooling layers often use simple functions like max or average: 𝑜𝑗𝑙=max/avg(selected region from 𝑜𝑗𝑙−1)ojl​=max/avg(selected region from ojl−1​)

**Chapter 4**

**Implementation**

**4.1 Implementation**

**ML code:**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer code

Description automatically generated**

**FLASK:**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer code

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Chapter 5**

**Results**

**5.1 Results**

**Dataset Image Total :**

A graph of different colored rectangular shapes

Description automatically generated with medium confidence

**Model Performance:**

**Model Accuracy**

A graph of a graph showing the value of a train and val

Description automatically generated

**Model Loss:**

A graph with blue lines and numbers

Description automatically generated

**Performance Testing:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Parameter** | **Values** | **Screenshot** |
| 1. | Model Summary | A diagram of a computer  Description automatically generated | A screenshot of a computer program  Description automatically generated |
| 2. | Accuracy | Training Accuracy – 99.18% Validation Accuracy – 91.30% | A screenshot of a graph  Description automatically generated |

**Frontend Output:**

A screenshot of a computer

Description automatically generated

A close-up of a person's eye

Description automatically generated

**Chapter 6**

**Conclusion and Future Work**

**6.1 Conclusion and Future Work**

To conclude, similarly to any viable AI driven application, the Eye Disease Detection Model using CNN holds tremendous value for the public if this base prototype is carefully integrated into real world scenarios after careful scaleup. The key findings for us whilst doing this project were how different techniques such as ResNet, Classical implementation and Neural Networks fared against each other for this particular project. Being able to work on this was also a learning experience like nothing has been and the current small success of it definitely instills us with the confidence to work on this even further and hopefully have a market ready Eye Disease Classification Application at our disposal.

**FUTURE SCOPE:**

AI powered diagnostic services definitely commands attention due to their objective diagnosis and room for customization. With time the technology would only advance and win the faith of more and more potential users.

Talking particularly about our Eye Disease Detection Model, it could be levied to :

● Clinics

● Hospitals

● Government for camps and health drive purposes

● Medical Institutions

For quick and accurate diagnosis of diseases so that every practitioner could focus on treating the problem itself and connect with more people In need. In terms of development, we could hone the performance even further with more resources being allocated to research and development. More diseases could be included to support more conditions and the availability of the software can be widened.