Multi Class Classification on Retinal Eye Diseases using Swin Transformer

A
Project Report

Submitted in partial fulfillment of
the requirements for the award of the degree of

Post Graduate Diploma
in
Big Data Analytics (PG-DBDA)
from
C-DAC ACTS, Chennai

Submitted by

Kamarajugadda Jyothi Phani Vaibhav (240860825003)

Under the guidance of

Ms. Irene, C-DAC, Chennai

Declaration

This is to certify that the Project report work entitled **"Multi Class Classification on Retinal Eye Diseases using Swin Transformer"** is a bonafide work carried out by **Kamarajugadda Jyothi Phani Vaibhav** bearing **PRN. 240860825003** in partial fulfillment of the requirements for the degree of POST GRADUATE DIPLOMA in BIG DATA ANALYTICS (PG-DBDA) discipline to C-DAC, Chennai during the academic year 2024- 2025.

The results embodied in this report have not been submitted to any other University or Institution for the award of any degree or diploma.

Kamarajugadda Jyothi Phani Vaibhav 240860825003

Acknowledgment

I wish to take this opportunity to express my deep gratitude to all those who helped us in various ways during my Project report work. It is my pleasure to acknowledge the help of all those individuals who were responsible for foreseeing the successful completion of my Project report.

I express my sincere gratitude to my guide **Ms. Irene, C-DAC, Chennai** for her support and provided valuable insights and constructive criticism at every stage of the project period.

I sincerely thank to **Dr. Sumithra Radhakrishnan & Ms. Sunandha, C-DAC, Chennai** for their encouragement during the course.

I am thankful to faculty members and peers for their kind help and co-operation over the course period.

Furthermore, I would like to acknowledge the online communities and documentation sources—including open-source forums, official libraries, and technology guides—that offered a wealth of information and best practices. The ability to tap into a global network of developers and experts was vital in ensuring the robustness and scalability of this project.

Finally, Special thanks to my parents for their moral support and encouragement throughout this course.

Kamarajugadda Jyothi Phani Vaibhav 240860825003

Contents

Abstract

Retinal eye diseases are among the leading causes of vision impairment globally, with various conditions such as diabetic retinopathy, age-related macular degeneration, and glaucoma requiring accurate and early diagnosis. In this project, I propose a multi-class classification framework to identify and classify different retinal diseases using the Swin Transformer, a state-of-the-art vision transformer model. We leverage the ODIR-5K dataset, which contains images of retinal fundus, along with their corresponding diagnostic labels. By employing the Swin Transformer model, known for its hierarchical feature extraction capabilities and efficient handling of high-resolution images, I aim to improve the accuracy of disease detection in retinal scans.

We begin by preprocessing the dataset, including augmenting the images to enhance model generalization, and applying label encoding to categorize eight distinct classes of retinal diseases. My approach utilizes a deep learning pipeline incorporating transfer learning with a pre-trained Swin Transformer model, optimizing the network with class-weighted loss functions to account for class imbalances. Extensive training, validation, and evaluation metrics such as accuracy, precision, recall, F1 score, ROC AUC, and Cohen's Kappa are computed to assess the model's performance. I demonstrated that my model achieves competitive performance in terms of classification accuracy and generalization, showing its potential for real-world applications in ophthalmology, providing timely and accurate diagnosis of retinal diseases and assisting healthcare professionals in their decision-making processes.

The results indicate that the Swin Transformer model can significantly enhance the diagnostic capability for retinal diseases, offering a robust tool for automated medical image analysis and contributing to the advancement of AI-driven healthcare solutions.

List of Figures

Chapter 1

INTRODUCTION

1.1 About the Project

The ODIR 5K dataset is a comprehensive collection of retinal images used to classify different eye diseases. This project focuses on utilizing the Swin Transformer, a state-of-the-art vision transformer model, for the multi-class classification task. The model will identify various eye diseases, including diabetic retinopathy, glaucoma, cataracts, and more.

1.2 Project Objective

The objective of this project is to use the Swin Transformer model to classify retinal images into several categories based on the presence of specific eye diseases. The aim is to leverage the strengths of the transformer architecture to extract meaningful patterns from medical images and achieve high classification accuracy.

1.3 Problem Statement

In the field of healthcare, particularly in ophthalmology, early diagnosis of eye diseases is crucial for effective treatment and prevention of blindness. This project focuses on developing a deep learning model that can automatically classify different types of ocular diseases from fundus images (photographs of the back of the eye). The goal is to create a predictive model that can detect multiple eye conditions, such as diabetes, glaucoma, cataracts, hypertension, age-related macular degeneration (AMD), and more, based on features extracted from these images. Given a dataset of ocular images and diagnostic keywords like ODIR-5K, the model, like Swin Transformer, should accurately classify and predict whether a given eye image is associated with any of these diseases, helping doctors and healthcare professionals with faster and more accurate diagnoses.

1.4 Scope of the Project

This project will involve:

- Preprocessing and augmentation of the ODIR 5K dataset.
- Training a Swin Transformer model for multi-class classification.
- Evaluating model performance using standard metrics such as accuracy, precision, recall, F1-score, kappa score, Sensitivity, Specificity ROC score.
- Visualizing results and providing insights for future improvements.

1.5 Limitations of the Project

The main limitations of the project are:

- The dataset's limited size which leads to overfitting, especially with a deep model like the Swin Transformer.

- Imbalanced class distribution affects the model's ability to generalize across all disease categories.
- The computational demands of training a Swin Transformer model requires significant resources, such as high-end GPUs for a long time.

Chapter 2

Literature Survey

2.1 Ophthalmology and Ophthalmological disease

Ophthalmology is the branch of medicine that focuses on the diagnosis, treatment, and prevention of eye disorders and diseases. It encompasses a wide range of topics, including the structure and function of the eye, vision problems, eye diseases, and surgical treatments. Ophthalmologists are medical doctors who specialize in this field, and they are trained to perform eye exams, prescribe medications, and carry out surgical procedures to correct or treat various conditions.

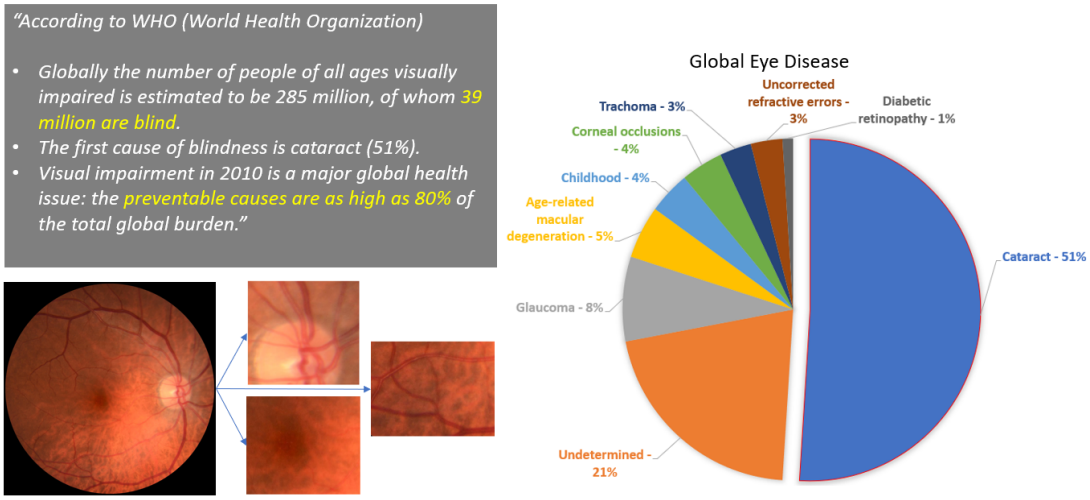

Figure 2.1: Ophthalmology and Ophthalmological diseases

There are four main aspects of Ophthalmology:

2.1.1 Anatomy and Physiology of the Eye

The eye is a complex organ, and understanding its anatomy is crucial for diagnosing and treating eye diseases. The key components include:

- Cornea: The clear, outer layer that helps focus light.
- Lens: The structure that changes shape to focus light onto the retina.
- Retina: The layer of cells at the back of the eye that detects light and sends signals to the brain.
- Optic Nerve: The nerve that transmits visual information from the retina to the brain.

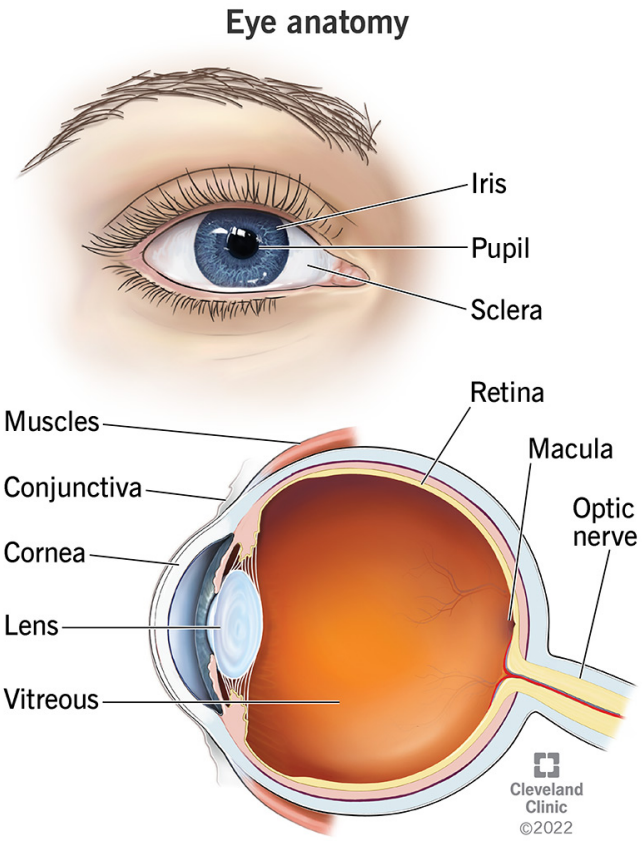

Figure 2.2: Anatomy of an eye (From Cleveland Clinic 2022)

2.1.2 Ophthalmological Diseases

There are more than 30 non rare Ophthalmological Diseases around the world. Few of them are Age-Related Macular Degeneration (AMD), Cataracts, Color Blindness, Diabetic Retinopathy, Farsightedness (Hyperopia), Glaucoma, Nearsightedness (Myopia).
Refractive Errors:

- Myopia (Nearsightedness): Difficulty seeing distant objects clearly.
- Hyperopia (Farsightedness): Difficulty seeing close objects clearly.
- Astigmatism: Blurred vision caused by an irregular shape of the cornea or lens.
- Presbyopia: Age-related difficulty in focusing on near objects.

Cataracts: A cataract is a clouding of the eye's lens that leads to a decrease in vision. It's most commonly seen in older adults and can be treated with surgery to replace the cloudy lens with a clear artificial one.
Glaucoma: A group of eye diseases that damage the optic nerve, often due to high intraocular pressure. If left untreated, glaucoma can lead to permanent vision loss. There are two main types:

- Open-Angle Glaucoma: The most common form, developing slowly with no obvious symptoms.
- Angle-Closure Glaucoma: A more sudden form that causes severe pain and requires immediate treatment.

Macular Degeneration: A condition that affects the retina, particularly the macula, the part of the retina responsible for central vision. Age-related macular degeneration (AMD) can cause vision loss in older adults, particularly affecting reading and recognizing faces.

Diabetic Retinopathy: A complication of diabetes that damages the blood vessels of the retina. Over time, it can cause vision impairment and even blindness. Early detection and control of diabetes are key to preventing this condition.

2.1.3 Diagnostic and Treatment Methods

Eye Exams: Regular eye exams are crucial for detecting eye problems early, especially in people with risk factors like diabetes, high blood pressure, or a family history of eye disease.

Diagnostic Tests:

- Visual acuity tests: To measure how well you can see at different distances.
- Tonometry: Measures intraocular pressure to detect glaucoma.
- Ophthalmoscopy: Allows the doctor to view the retina and optic nerve.
- Slit-lamp examination: Used to inspect the cornea, lens, and retina.
- Fluorescein Angiography: A test to check for blood vessel problems in the retina.

Treatments:

- Medications: Eye drops or oral medications are used to treat infections, inflammation, and conditions like glaucoma.
- Laser Therapy: Used for conditions like diabetic retinopathy, macular degeneration, or glaucoma.
- Surgery: Surgical treatments may be needed for conditions like cataracts, retinal detachment, or strabismus. LASIK surgery is a common procedure for refractive errors like myopia.

2.1.4 Prevention

Regular eye check-ups, wearing sunglasses to protect from UV rays, maintaining a healthy diet rich in vitamins, and managing underlying health conditions like diabetes and hypertension are key steps in preventing eye diseases.

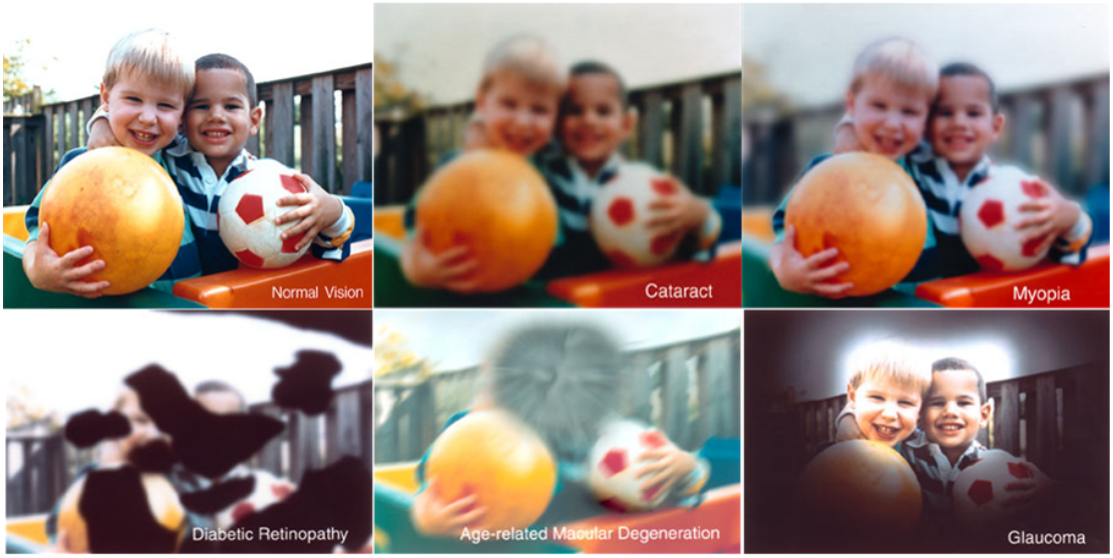

Figure 2.3: Vision from different Ophthalmological Diseases

2.2 Image Transformers

2.2.1 Introduction to Image Transformers

Traditionally, Convolutional Neural Networks (CNNs) have dominated the field of image processing tasks, including image classification, object detection, and segmentation. However, the success of transformers in natural language processing (NLP) motivated researchers to explore their potential in image-related tasks. Image Transformers are a novel architecture designed to overcome the limitations of CNNs, such as their inherent locality biases, by leveraging self-attention mechanisms to capture long-range dependencies in image data.

Transformers were first introduced in NLP tasks (Vaswani et al., 2017) through the Attention Is All You Need paper, and this shifted the paradigm for sequential data modeling. Given their success, researchers began to adapt this architecture for visual data, with early approaches such as Vision Transformer (ViT), DeiT, and others making significant strides toward competitive performance with CNN-based models.

2.2.2 Image Classification: Multi-Class and Multi-Label

Image classification is one of the fundamental tasks in computer vision. It involves assigning an image to a category or a set of categories, and it can be broadly categorized into multi-class classification and multi-label classification.

- **Multi-Class Classification:** In multi-class classification, each image belongs to exactly one class. The goal is to classify an image into one of several possible categories. Common datasets for multi-class classification include CIFAR-10, ImageNet, and MNIST. The output layer in a multi-class classification model typically consists of a single neuron for each class, where the predicted label corresponds to the class with the highest probability.
- **Multi-Label Classification:** Unlike multi-class classification, where only one class is assigned to an image, multi-label classification involves assigning multiple labels to an image. This is common in scenarios like scene understanding or object detection, where an image can contain multiple objects or scenes. In this setup, the model outputs a probability for each label independently, allowing multiple labels to be predicted for a single image. Datasets like COCO and Pascal VOC are commonly used for multi-label classification tasks.

Both tasks have been traditionally handled by CNNs, but image transformers have shown great potential in improving performance, especially when the data has complex relationships or requires long-range context.

2.2.3 Journey of Image Transformers

Vision Transformer (ViT)

The first notable attempt to use transformers for image classification was the Vision Transformer (ViT) (Dosovitskiy et al., 2020), which applied the transformer architecture directly to image patches. Unlike CNNs that operate over local neighborhoods, ViT divides an image into fixed-size patches, flattens them, and then treats these patches as a sequence of tokens (similar to words in NLP) to be processed by the transformer.

- **ViT Architecture:** In ViT, the input image is divided into non-overlapping patches (e.g., 16x16 pixels), and each patch is flattened into a vector, which is then fed into a transformer model. The self-attention mechanism of transformers allows the model to capture long-range dependencies and global context within the image, making ViT particularly powerful for tasks requiring complex reasoning across an image.
- **Challenges:** One limitation of ViT is that it requires a large amount of data to train effectively. To overcome this, the Data-efficient Image Transformer (DeiT) was introduced by Facebook AI in 2021, which uses knowledge distillation to train ViT efficiently even with smaller datasets.

Swin Transformer

The Swin Transformer (Liu et al., 2021), short for Shifted Window Transformer, marked a significant milestone in the evolution of image transformers. Swin introduced several key innovations over ViT to improve efficiency and performance.

- **Local and Global Attention:** Swin Transformer uses a hierarchical design, where the resolution of image patches progressively reduces at each stage of the network. To compute self-attention, Swin uses a window-based approach, applying attention only within fixed-size local windows (e.g., 7x7 patches). To address the limitation of local attention, Swin introduces a shifted window mechanism, where the windows are shifted between successive layers to allow for interactions between adjacent windows. This approach balances local and global attention, which improves computational efficiency while still capturing long-range dependencies.
- **Hierarchical Structure:** Swin Transformers utilize a hierarchical feature map that progressively reduces the resolution of image patches. This hierarchical nature makes Swin Transformers more scalable and efficient, allowing them to work well on high-resolution images, such as those in the COCO dataset.
- **Performance:** Swin Transformers achieved state-of-the-art results on several benchmark datasets such as ImageNet and COCO, outperforming CNN-based models like ResNet and achieving competitive results with large-scale transformers.

Current Developments and Future Directions

Following the success of ViT and Swin Transformer, image transformers continue to evolve. Researchers are working on more efficient transformer architectures that can handle both the computational costs and the data requirements of vision tasks.

- **Cross-Domain Transformers:** Some models aim to combine image transformers with other modalities, such as text, to create multimodal architectures that can handle tasks like visual question answering or image captioning.
- **Fine-Grained Transformers:** There is ongoing work on designing transformers that can focus on fine-grained details within an image while maintaining global context, potentially enhancing performance for tasks like fine-grained image classification and object detection.
- **Efficient Transformers:** Given that transformer-based models can be computationally expensive, various techniques like pruning, quantization, and knowledge distillation are being explored to make these models more efficient and deployable on resource-constrained devices.
- **Hybrid Models:** Recent work has focused on hybrid models that combine CNNs and transformers, leveraging the strengths of both architectures. For example, CNNs are good at extracting local features, while transformers excel at capturing global dependencies. Hybrid models combine these advantages for improved performance.

2.3 Research on Ophthalmological Disease Detection using Image Transformers

Introduction to Ophthalmology and Disease Detection

Ophthalmology, the branch of medicine focused on the eyes and visual system, has seen significant advancements with the help of artificial intelligence (AI) and computer vision. A key area of focus is the development of automated systems for the detection and diagnosis of ophthalmological diseases, such as diabetic retinopathy, age-related macular degeneration (AMD), glaucoma, cataracts, and retinal diseases. Image-based

diagnostic tools, particularly fundus images, optical coherence tomography (OCT), and retinal scans, play a critical role in detecting these diseases.
Historically, deep learning models such as Convolutional Neural Networks (CNNs) have been the go-to architectures for processing ophthalmic images due to their strong performance in image classification tasks. However, recent research has increasingly shifted towards the use of transformer-based models for visual data, inspired by the success of transformers in natural language processing. Transformers, particularly image transformers, have shown promising results in ophthalmology due to their ability to capture global dependencies and long-range contextual information, which is crucial for interpreting the intricate patterns in ophthalmic images.

2.4 Key Contributions in the Area

The application of image transformers in ophthalmology has seen a variety of innovations and contributions that have significantly advanced the field of automated disease detection, diagnosis, and prognosis.

- **Vision Transformers:**ViT introduced a new paradigm where images were split into patches, flattened, and treated as a sequence, similar to words in NLP. This approach proved effective in capturing global contextual relationships within images, which is essential in medical imaging tasks where understanding the full context is crucial for accurate diagnosis.
- **Swin Transformer:**The Swin Transformer introduces a novel approach by applying transformers to vision tasks while maintaining computational efficiency. Its hierarchical feature extraction process allows it to capture both local and global dependencies in an image, which is ideal for the complex patterns found in medical imaging. This approach has been successfully applied in medical image classification, including retinal disease detection.
- **Hybrid CNN-Transformer Models:** The hybrid CNN-Transformer architecture leverages CNNs for initial feature extraction followed by transformer-based models to capture the global relationships and contextual dependencies within the image. This allows the model to focus on the most relevant regions of an image while still maintaining a broader understanding of the entire image context.

- **Data-Efficient Transformers (DeiT):** DeiT introduced the concept of distilling knowledge from a large model to a smaller, more efficient one, making transformer-based models more practical for real-world applications with limited annotated data.
- **Multi-Modal Transformers:** Multi-modal transformers combine data from multiple imaging modalities to enhance the diagnostic capabilities of models. These models use attention mechanisms to cross-reference features from each modality, resulting in a more robust analysis and diagnosis.

2.5 Research Papers on Ophthalmological Disease Detection

Multi-class multi-label ophthalmological disease detection using transfer learning based convolutional neural network [21]
Brief information about the paper:

- This paper was published by Neha Gour , Pritee Khanna in 2020.
- An automated multi-class multi-label fundus image classification method is proposed.
- Two transfer learning-based models with four CNN architectures are proposed for fundus images.
- The results are evaluated on grand challenge database named ODIR database.
- The paper focuses on Accuracy, Sensitivity and Specificity parameters to evaluate the performance per class.

Multi-Class Eye Disease Classification Using Deep Learning [22]
Brief information about the paper:

- This paper was published by by Mariam Ayman Mohamed; Mustafa Adel Zakaria; Eman Hamdi; Rawan elSayed Tawfek; Taha Mohamed Taha; Yasmine M. Afify in 2023.

- An automated multi-class fundus image classification method is proposed.
- The system can accurately detect and classify normal, diabetic retinopathy, glaucoma, and cataracts with an overall accuracy rate of 92

BFENet: A two-stream interaction CNN method for multi-label ophthalmic diseases classification with bilateral fundus images [23]
Brief information about the paper:

- This paper was published by Xingyuan Ou, Li Gao, Xiongwen Quan, Han Zhang, Jinglong Yang, Wei Li in 2022.
- They propose a two-stream interactive CNN architecture for multi-label ophthalmic diseases classification with bilateral fundus images.
- They designed a feature enhancement module by using the attention mechanism to learn the interdependence between local and global information, which enhances the extracted feature under the two-stream interactive architecture.
- A multiscale module is designed to enriches the feature maps by superimposing feature information of different resolutions images extracted through dilated convolution.
- The paper focuses on Kappa, F1, AUC and Final score parameters to evaluate the performance.

Multi-Label Classification of Fundus Images With Graph Convolutional Network and Self-Supervised Learning [24]
Brief information about the paper:

- This paper was published by Jinke Lin; Qingling Cai; Manying Lin in 2021.
- They propose a two-stream interactive CNN architecture for multi-label ophthalmic diseases classification with bilateral fundus images.
- They designed a feature enhancement module by using the attention mechanism to learn the interdependence between local and global information, which enhances the extracted feature under the two-stream interactive architecture.

- A multiscale module is designed to enriches the feature maps by superimposing feature information of different resolutions images extracted through dilated convolution.
- The paper focuses on Kappa, F1, AUC and Final score parameters to evaluate the performance.

Dense Correlation Network for Automated Multi-Label Ocular Disease Detection with Paired Color Fundus Photographs [25]
Brief information about the paper:

- This paper was published by Jinke Lin; Qingling Cai; Manying Lin in 2021.
- They propose Propose two new multi-label classification networks - MCG-Net based on graph convolutional network and MCGS-Net based on graph convolutional network and self-supervised learning.
- The paper focuses on F1, AUC as parameters to evaluate the performance.

Chapter 3

Feature Engineering

3.1 Data Collection

Ocular Disease Intelligent Recognition (ODIR) is a structured ophthalmic database of 5,000 patients with age, color fundus photographs from left and right eyes and doctors' diagnostic keywords from doctors.
This dataset is meant to represent "real-life" set of patient information collected by Shanggong Medical Technology Co., Ltd. from different hospitals/medical centers in China. In these institutions, fundus images are captured by various cameras in the market, such as Canon, Zeiss and Kowa, resulting into varied image resolutions.
Annotations were labeled by trained human readers with quality control management. They classify patient into eight labels including:

- Normal (N),
- Diabetes Retinopathy(D),
- Glaucoma (G),
- Cataract (C),
- Age related Macular Degeneration (A),
- Hypertension (H),

- Pathological Myopia (M),
- Other diseases/abnormalities (O)

3.2 Data Preprocessing

Preprocessing steps included:

- Resizing images to 224x224 pixels to ensure compatibility with the Swin Transformer model.
- Normalizing class imbalances using scikit-learn.
- Augmenting the dataset with transformations such as rotation, flipping, and color jitter to prevent overfitting and increase robustness.

3.3 Feature Extraction

With Swin Transformer, feature extraction is performed through its transformer layers, which learn to capture both local and global features from the images. This is done via hierarchical patch-based attention mechanisms that allow the model to focus on different scales of the input image.

Chapter 4

Development and Coding

4.1 Technology Used

Technologies used in the project include:

- **Python**: The primary programming language [6].
- **KaggleHub**: For downloading and extracting the dataset [1].
- **PyTorch**: The deep learning framework used for implementing the Swin Transformer model [12][13].
- **NumPy, Pandas**: Libraries for data manipulation and handling [7], [8].
- **Matplotlib and Seaborn**: Libraries for data visualization [9],[10].
- **Scikit-Learn Documentation**: For Train-Test split, Label Encoding, Computing Metrics and Class Weights [11].
- **timm**: For getting Swin Transfomer from PyTorch Image Models [19].
- **NTLK(Natural Language ToolKit)**: For Word Clouds on Diagnostic Keywords [15], [16], [17].
- **Transformers**: For Swin Image Classification from Hugging Face [18].
- **Google Colab**: For Training and Testing on Free GPU computational power [5].

- **Python Imaging Library**: For loading and processing Images [20].
- **Miscellaneous**: Modules like os, random, warnings, string, re were used for various purposes.

4.2 Software Requirements

- IDE Anaconda/Google Colab
- Python 3.6 or higher
- Linux/Windows 8 or higher
- Latest Version of all libraries viz. KaggleHub, PyTorch, NumPy, Pandas, Matplotlib, Seaborn, sklearn, timm, NTLK, Transformers, PIL.

4.3 Hardware Requirements

- Processor Intel i5
- 2.9 GHz or Better CPU
- 8GB RAM
- Hard Disk 80GB
- 8GB GPU with minimum of 3000 CUDA Cores (For Running Locally)
- Minimum of 20 Mbps of Uninterrupted Internet (For Running Online)
- Input Devices viz. Keyboard and Mouse

4.4 Architecture Diagram

The architecture of the Swin Transformer is based on a hierarchical design, where the image is divided into non-overlapping patches, each processed by attention layers to

capture both local and global information. The model then uses these learned features to classify the image into one of the multiple disease categories.

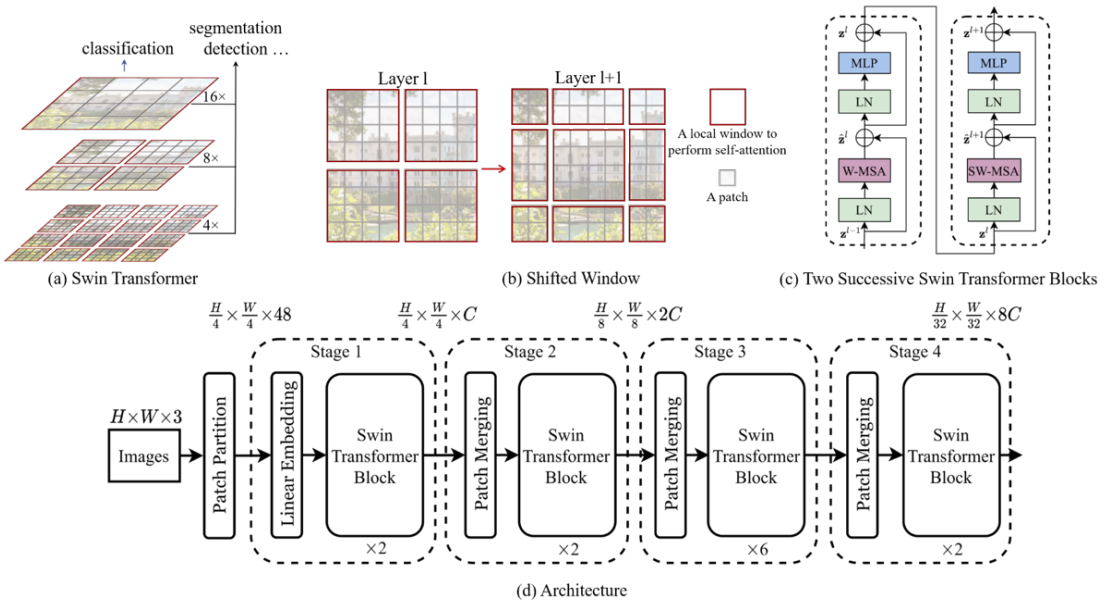

Figure 4.1: Architecture Diagram of the Swin Transformer Model

4.5 Explanation of the Model

The Swin Transformer model for this task uses a vision transformer-based architecture. The process is as follows:

- **Hierarchical Structure:** Unlike traditional Vision Transformers that process the image in one fixed-size grid (e.g., 16x16 patches), the Swin Transformer introduces hierarchical feature maps. It progressively down-samples the image as the network deepens, similar to convolutional neural networks (CNNs). This allows the model to capture information at different scales and be more computationally efficient.
- **Window-based Attention:** The attention mechanism in Swin Transformer is computed within non-overlapping local windows. These windows reduce the computational burden compared to global attention (which considers all tokens in the image at once). Each window performs self-attention independently, which is more efficient and allows the model to handle larger images.

The local windows help focus attention on smaller regions, which is similar to how convolutions capture local patterns in CNNs.

- **Shifted Windows:** A key feature of the Swin Transformer is the shifted window strategy. The windows are shifted between layers, meaning that for each layer, the set of pixels (patches) in each attention window slightly shifts. This allows the model to capture cross-window dependencies without dramatically increasing computational complexity.
  In Layer 1, the image is divided into fixed-size windows.
  In Layer 2, the windows are shifted by a few pixels, allowing information from adjacent windows to be communicated without fully merging the windows. This helps the model capture larger receptive fields in an efficient manner.

- **Linear Complexity:** A key advantage of Swin Transformer is that the attention mechanism within windows is computed with linear complexity. In traditional transformers, the attention complexity grows quadratically with the number of patches ($O(n^2)$), which is inefficient for images. By restricting attention to smaller, local windows, the Swin Transformer reduces the complexity, making it scalable to larger images.

In simple terms, The image is first divided into patches. Then a patch embedding layer is used to convert patches into feature vectors. Then these features are then passed through a series of transformer blocks that apply multi-head self-attention mechanisms. And the output features from the transformer blocks are processed through a classification head to predict the disease category.
The model uses the softmax activation function in the final layer to output a probability distribution over the classes.

4.6 Implementation

4.6.1 my-swin-model.py

```
# -*- coding: utf-8 -*-
"""13-02-25.ipynb

Automatically generated by Colab.

```

```
Original file is located at
    https://colab.research.google.com/drive/XXXXXXXXXX
    https://github.com/kjpvaibhav/PGD-Project

**Name:** Kamarajugadda Jyothi Phani Vaibhav

**PNR:** 240860825003

**Problem Statement:**

In the field of healthcare, particularly in ophthalmology, early diagnosis of eye diseases is crucial for effective treatment and prevention of blindness. This project focuses on developing a deep learning model that can automatically classify different types of ocular diseases from fundus images (photographs of the back of the eye). The goal is to create a predictive model that can detect multiple eye conditions, such as diabetes, glaucoma, cataracts, hypertension, age-related macular degeneration (AMD), and more, based on features extracted from these images. Given a dataset of ocular images and diagnostic keywords like ODIR-5K, the model, like Swin Transformer, should accurately classify and predict whether a given eye image is associated with any of these diseases, helping doctors and healthcare professionals with faster and more accurate diagnoses.



**Objective:**

To develop a Deep Learning model to classify the Ocular disease type, using the Microsoft Swin Transformer on the ODIR-5K dataset

# Importing Required Modules
"""

# For Dataset Download
import kagglehub

# Data Processing
import numpy as np
import pandas as pd

# Data Visualization
import matplotlib.pyplot as plt
```

```
import seaborn as sns

# NTLK
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from wordcloud import WordCloud, STOPWORDS
from nltk.stem import SnowballStemmer

# Machine Learning
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report,
     cohen_kappa_score, confusion_matrix, f1_score,
    precision_score, recall_score, roc_auc_score, roc_curve
from sklearn.preprocessing import LabelEncoder
from sklearn.utils.class_weight import compute_class_weight

# Deep Learning
import timm
import torch
import torch.nn as nn
import torch.optim as optim
from torch.optim.lr_scheduler import StepLR
from torch.utils.data import Dataset, DataLoader
from torchvision import transforms
from transformers import SwinForImageClassification

# Miscellaneous
import os
import itertools
import string
import re
from random import sample
from PIL import Image
import warnings
warnings.filterwarnings("ignore")

device = torch.device("cuda" if torch.cuda.is_available() else "
    cpu")

print(f"Using device: {device}")

"""# Loading Dataset"""

path = kagglehub.dataset_download("andrewmvd/ocular-disease-
    recognition-odir5k")
```

```
78
79 print("Path to dataset files:", path)
80
81 """# Exploratory Data Analysis"""
82
83 import os
84
85 def get_folder_info(parent_dir):
86     folder_info = []
87
88     for dirpath, dirnames, filenames in os.walk(parent_dir):
89         total_size = 0
90         num_folders = len(dirnames)
91         num_files = len(filenames)
92
93         for filename in filenames:
94             filepath = os.path.join(dirpath, filename)
95             total_size += os.path.getsize(filepath)
96
97         if num_files == 1:
98             single_file_size = os.path.getsize(os.path.join(
    dirpath, filenames[0]))
99             folder_info.append({
100                 'folder_name': dirpath,
101                 'num_folders': num_folders,
102                 'num_files': num_files,
103                 'total_size': single_file_size,
104                 'filenames': filenames[0]
105             })
106         else:
107             folder_info.append({
108                 'folder_name': dirpath,
109                 'num_folders': num_folders,
110                 'num_files': num_files,
111                 'total_size': total_size,
112                 'filenames': "Image Files"
113             })
114
115     return folder_info
116
117 folders = get_folder_info(path)
118
119 for folder in folders:
120     print(f"Folder: {folder['folder_name']}")
121     print(f"Number of subfolders: {folder['num_folders']}")
122     print(f"Number of files: {folder['num_files']}")
```

```
    print(f"Total size: {folder['total_size']} bytes")
    print('=' * 40)

    if folder['num_files'] == 1:
        single_file = folder['filenames']
        single_file_path = os.path.join(folder['folder_name'],
    single_file)
        print(f"Single file: {single_file}")
        print(f"Total size: {folder['total_size']} bytes")
        print(f"File path: {single_file_path}")
        print('-' * 40)

eyes = pd.read_csv(path + '/full_df.csv')
eyes

eyes.info()

eyes.describe(include="all")

eyes.columns

len(eyes)

len(eyes[eyes.duplicated()])

eyes.isnull().sum()

label_columns = ['N', 'D', 'G', 'C', 'A', 'H', 'M', 'O']
label_names = [
    "Normal",
    "Diabetic Retinopathy",
    "Glaucoma",
    "Cataract",
    "Age-related Macular Degeneration",
    "Hypertension",
    "Myopia",
    "Other diseases"
]

labels_dic = dict(zip(label_columns, label_names))
print(labels_dic)

print("Count of rows containing 'right':", eyes['filename'].str.
    contains('right').sum())
print("Count of rows containing 'left':", eyes['filename'].str.
    contains('left').sum())
```

```
166
167 """# Data Cleaning"""
168
169 eyes['Left-Fundus'].nunique()
170
171 photo_counts = eyes['Left-Fundus'].value_counts()
172 photo_more_than_once = photo_counts[photo_counts > 1].index.tolist
    ()
173 print(len(photo_more_than_once))
174
175 photo_more_than_once[0]
176
177 eyes[eyes['Left-Fundus'] == '0_left.jpg']
178
179 print(len(eyes[(eyes['labels'] == "['N']") & (eyes['N'] != 1)]))
180 print(len(eyes[(eyes['labels'] == "['D']") & (eyes['D'] != 1)]))
181 print(len(eyes[(eyes['labels'] == "['O']") & (eyes['O'] != 1)]))
182 print(len(eyes[(eyes['labels'] == "['C']") & (eyes['C'] != 1)]))
183 print(len(eyes[(eyes['labels'] == "['G']") & (eyes['G'] != 1)]))
184 print(len(eyes[(eyes['labels'] == "['A']") & (eyes['A'] != 1)]))
185 print(len(eyes[(eyes['labels'] == "['M']") & (eyes['M'] != 1)]))
186 print(len(eyes[(eyes['labels'] == "['H']") & (eyes['H'] != 1)]))
187
188 eyes.drop(columns=[ 'ID'] , inplace=True)
189
190 old_path_part = "../input/ocular-disease-recognition-odir5k/ODIR-5
    K/Training Images/"
191 new_path_part = path + "/ODIR-5K/ODIR-5K/Training Images/"
192 eyes['filepath'] = eyes['filepath'].replace(old_path_part,
    new_path_part, regex=True)
193
194 for filepath in eyes['filepath']:
195     assert os.path.exists(filepath), f"Missing file: {filepath}"
196
197 len(eyes[eyes['Patient Age'] == 1])
198
199 eyes['labels'] = eyes['labels'].str[2]
200 eyes.head()
201
202 eyes['labels'].nunique()
203
204 eyes['labels'].value_counts()
205
206 """# Data Visualization"""
207
208 plt.figure(figsize=(14, 5))
```

```
sns.countplot(x='labels', data=eyes , orient='h')
plt.title('Count of Target Labels in Eyes Dataset')
plt.xlabel('Labels')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()

eyes[eyes['Left-Diagnostic Keywords'] == 'low image quality']

eyes = eyes.loc[~(eyes['Left-Diagnostic Keywords'] == 'low image
    quality')]

print(len(eyes[(eyes['labels'] == "['N']") & (eyes['N'] != 1)]))
print(len(eyes[(eyes['labels'] == "['O']") & (eyes['O'] != 1)]))
eyes.head()

photos_unique = eyes.drop_duplicates(subset='Left-Fundus', keep='
    first')
eyes = photos_unique
eyes.reset_index(drop=True,inplace=True)
len(eyes)

eyes['Left-Diagnostic Keywords'].nunique()

eyes['Right-Diagnostic Keywords'].nunique()

eyes['Left-Diagnostic Keywords'].mode()

eyes['Left-Diagnostic Keywords'].value_counts()

eyes['Right-Diagnostic Keywords'].mode()

both_eyes_normal = eyes[
    (eyes['Right-Diagnostic Keywords'] == 'normal fundus') &
    (eyes['Left-Diagnostic Keywords'] == 'normal fundus')
]

both_eyes_normal.reset_index(inplace=True,drop=True)

len(both_eyes_normal)

both_eyes_not_normal = eyes[
    (eyes['Right-Diagnostic Keywords'] != 'normal fundus') &
    (eyes['Left-Diagnostic Keywords'] != 'normal fundus')
]

```

```
both_eyes_not_normal.reset_index(inplace=True,drop=True)

len(both_eyes_not_normal)

right_eye_normal = eyes[
    (eyes['Right-Diagnostic Keywords'] == 'normal fundus') &
    (eyes['Left-Diagnostic Keywords'] != 'normal fundus')
]

right_eye_normal.reset_index(inplace=True,drop=True)

len(right_eye_normal)

left_eye_normal = eyes[
    (eyes['Right-Diagnostic Keywords'] != 'normal fundus') &
    (eyes['Left-Diagnostic Keywords'] == 'normal fundus')
]

left_eye_normal.reset_index(inplace=True,drop=True)

len(left_eye_normal)

sns.pairplot(eyes)
plt.show()

data_num = eyes[['Patient Age', 'N', 'D', 'G', 'C', 'A', 'H', 'M',
    'O']]
data_num.head()

corr_exists = data_num.corr()
corr_exists

plt.figure(figsize=(15, 15))
sns.heatmap(corr_exists, annot=True)
plt.title('Correlation Heatmap of ODIR Dataset')
plt.show()

fig, axes = plt.subplots(2, 2, figsize=(14, 10))
fig.suptitle('Disease Distribution Across Different Groups',
   fontsize=16)

titles = [
    'Both Eyes Normal',
    'Both Eyes Not Normal',
    'Right Eye Normal',
    'Left Eye Normal'
```

```
]

dataframes = [both_eyes_normal, both_eyes_not_normal,
    right_eye_normal, left_eye_normal]

for df, ax, title in zip(dataframes, axes.ravel(), titles):
    disease_columns = ['N', 'D', 'G', 'C', 'A', 'H', 'M', 'O']
    disease_counts = df[disease_columns].sum()

    sns.barplot(x=disease_counts.index, y=disease_counts.values,
    color='skyblue', ax=ax)
    ax.set_title(title)
    ax.set_xlabel('Diseases')
    ax.set_ylabel('Frequency')

plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()

fig, axes = plt.subplots(2, 2, figsize=(14, 10))
fig.suptitle('Disease Distribution Across Different Groups',
    fontsize=16)

titles = [
    'Both Eyes Normal',
    'Both Eyes Not Normal',
    'Right Eye Normal',
    'Left Eye Normal'
]

dataframes = [both_eyes_normal, both_eyes_not_normal,
    right_eye_normal, left_eye_normal]

for df, ax, title in zip(dataframes, axes.ravel(), titles):
    disease_columns = ['N', 'D', 'G', 'C', 'A', 'H', 'M', 'O']
    disease_counts = df[disease_columns].sum()

    sns.barplot(x=disease_counts.index, y=disease_counts.values,
    color='skyblue', ax=ax)
    ax.set_title(title)
    ax.set_xlabel('Diseases')
    ax.set_ylabel('Frequency')

plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()

condition_normal_left = eyes['Left-Diagnostic Keywords'] == '
```

```
    normal fundus'
condition_normal_right = eyes['Right-Diagnostic Keywords'] == '
    normal fundus'

both_normal = (condition_normal_left) & (condition_normal_right)
both_abnormal = (~condition_normal_left) & (~
    condition_normal_right)
left_normal_right_abnormal = (condition_normal_left) & (~
    condition_normal_right)
right_normal_left_abnormal = (~condition_normal_left) & (
    condition_normal_right)

counts = {
    'Both Normal': both_normal.sum(),
    'Both Abnormal': both_abnormal.sum(),
    'Left Normal, Right Abnormal': left_normal_right_abnormal.sum
    (),
    'Right Normal, Left Abnormal': right_normal_left_abnormal.sum
    ()
}

labels = counts.keys()
sizes = counts.values()
colors = ['#ff9999','#66b3ff','#99ff99','#ffcc99']
explode = (0.1, 0.07, 0, 0)

plt.figure(figsize=(8, 6))
plt.pie(sizes, explode=explode, labels=labels, colors=colors,
    autopct='%1.1f%%', shadow=True, startangle=140)
plt.title('Distribution of Fundus Conditions')
plt.axis('equal')

plt.show()

fig, axes = plt.subplots(2, 2, figsize=(8, 8), sharex=True, sharey
    =True)

dataframes = [both_eyes_normal, both_eyes_not_normal,
    right_eye_normal, left_eye_normal]
titles = ['Both Eyes Normal', 'Both Eyes Not Normal', 'Right Eye
    Normal', 'Left Eye Normal']

for i, (df, title) in enumerate(zip(dataframes, titles)):
    row = i // 2
    col = i % 2
    sns.histplot(df['Patient Age'], kde=True, bins=10, ax=axes[row
```

```
    , col])
373     axes[row, col].set_title(title)
374     axes[row, col].set_xlabel('Age')
375     axes[row, col].set_ylabel('Frequency')
376
377
378 plt.tight_layout()
379 plt.show()
380
381 counts = {
382     'Both Eyes Normal': both_eyes_normal['Patient Sex'].
    value_counts(),
383     'Both Eyes Not Normal': both_eyes_not_normal['Patient Sex'].
    value_counts(),
384     'Right Eye Normal': right_eye_normal['Patient Sex'].
    value_counts(),
385     'Left Eye Normal': left_eye_normal['Patient Sex'].value_counts
    ()
386 }
387
388 plot_data = pd.DataFrame(counts).fillna(0).T.reset_index()
389 plot_data = plot_data.melt(id_vars='index', var_name='Gender',
    value_name='Count')
390 plot_data = plot_data.rename(columns={'index': 'Category'})
391
392 plt.figure(figsize=(8, 8))
393 sns.barplot(data=plot_data, x='Category', y='Count', hue='Gender',
     palette='viridis')
394
395 plt.title('Number of Males and Females by Category')
396 plt.xlabel('Category')
397 plt.ylabel('Number of Patients')
398 plt.legend(title='Gender')
399 plt.xticks(rotation=45)
400 plt.tight_layout()
401 plt.show()
402
403 fig, axes = plt.subplots(2, 2, figsize=(8, 8), sharex=True, sharey
    =True)
404
405 dataframes = [both_eyes_normal, both_eyes_not_normal,
    right_eye_normal, left_eye_normal]
406 titles = ['Both Eyes Normal', 'Both Eyes Not Normal', 'Right Eye
    Normal', 'Left Eye Normal']
407 colors = ['#66b3ff', '#ff9999']
408
```

```
for i, (df, title) in enumerate(zip(dataframes, titles)):
    row = i // 2
    col = i % 2
    sns.histplot(df, x='Patient Age', hue='Patient Sex', multiple='stack', palette=colors, bins=10, ax=axes[row, col], kde=True)
    axes[row, col].set_title(title)
    axes[row, col].set_xlabel('Age')
    axes[row, col].set_ylabel('Frequency')

plt.tight_layout()
plt.show()

fig, axes = plt.subplots(1, 3, figsize=(18, 6), sharex=True, sharey=True)

dataframes = [both_eyes_not_normal, right_eye_normal, left_eye_normal]

titles = ['Both Eyes Not Normal', 'Right Eye Normal', 'Left Eye Normal']
colors = ['#66b3ff', '#ff9999']

for i, (df, title) in enumerate(zip(dataframes, titles)):
    row = i // 3
    col = i % 3
    filtered_df = df.query('D == 1')
    if not filtered_df.empty:
        sns.histplot(filtered_df, x='Patient Age', hue='Patient Sex', multiple='stack', palette=colors, bins=10, ax=axes[col], kde=True)
    axes[col].set_title(title)
    axes[col].set_xlabel('Age')
    axes[col].set_ylabel('Frequency')

plt.tight_layout()
plt.show()

fig, ax = plt.subplots(figsize=(6, 6))

df = both_eyes_not_normal
title = 'Both Eyes Not Normal (H=1)'
colors = ['#66b3ff', '#ff9999']

filtered_df = df.query('H == 1')
if not filtered_df.empty:
    sns.histplot(filtered_df, x='Patient Age', hue='Patient Sex',
```

```
    multiple='stack', palette=colors, bins=10, ax=ax, kde=True)
449     ax.set_title(title)
450     ax.set_xlabel('Age')
451     ax.set_ylabel('Frequency')
452 else:
453     ax.text(0.5, 0.5, 'No data available for H = 1',
    horizontalalignment='center', verticalalignment='center',
    fontsize=14, color='red')
454
455 plt.tight_layout()
456 plt.show()
457
458 filtered_eyes = eyes[eyes['labels'] == 'G']
459 sample_images = sample(filtered_eyes['filepath'].tolist(), 16)
460
461 plt.figure(figsize=(12, 12))
462
463 for i in range(16):
464     image = Image.open(sample_images[i])
465
466     label = filtered_eyes.iloc[i]['labels']
467
468     plt.subplot(4, 4, i+1)
469     plt.imshow(image)
470     plt.title(label, color='k', fontsize=12)
471     plt.axis("off")
472
473 plt.show()
474
475 stemmer = SnowballStemmer("english")
476 stopword = set(stopwords.words('english'))
477
478 def clean(text):
479     text = str(text).lower()
480
481     text = re.sub(r'\[.*?\]', '', text)
482     text = re.sub(r'https?://\S+|www\.\S+', '', text)
483     text = re.sub(r'<.*?>+', '', text)
484     text = re.sub(r'[%s]' % re.escape(string.punctuation), '',
    text)
485     text = re.sub(r'\n', '', text)
486     text = re.sub(r'\w*\d\w*', '', text)
487
488     text = [word for word in text.split() if word not in stopword]
489     text = " ".join(text)
490
```

```
    text = [stemmer.stem(word) for word in text.split()]
    text = " ".join(text)

    return text

eyes['Right-Diagnostic Keywords'] = eyes['Right-Diagnostic
    Keywords'].apply(clean)

text = " ".join(i for i in both_eyes_not_normal['Right-Diagnostic
    Keywords'])
stopwords = set(STOPWORDS)
wordcloud = WordCloud(stopwords=stopwords, background_color="white
    ").generate(text)
plt.figure(figsize=(15, 10))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()

text = " ".join(i for i in both_eyes_not_normal['Left-Diagnostic
    Keywords'])
stopwords = set(STOPWORDS)
wordcloud = WordCloud(stopwords=stopwords, background_color="white
    ").generate(text)
plt.figure(figsize=(15, 10))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()

"""# Model Building"""

eyes = eyes[['filename', 'filepath', 'labels']]

label_encoder = LabelEncoder()

eyes['labels_encoded'] = label_encoder.fit_transform(eyes['labels'
    ])

for label, encoded_value in zip(label_encoder.classes_, range(len(
    label_encoder.classes_))):
    print(f"{label}: {encoded_value}")

eyes.head()

train_df, test_df = train_test_split(eyes, test_size=0.2,
    random_state=42, stratify=eyes['labels_encoded'])
train_df, val_df = train_test_split(train_df, test_size=0.1,
```

```
    random_state=42, stratify=train_df['labels_encoded'])

class ODIRDataset(Dataset):
    def __init__(self, dataframe, transform=None):
        self.dataframe = dataframe
        self.transform = transform

    def __len__(self):
        return len(self.dataframe)

    def __getitem__(self, idx):
        img_path = self.dataframe.iloc[idx]['filepath']
        image = Image.open(img_path).convert('RGB')
        label = self.dataframe.iloc[idx]['labels_encoded']

        if self.transform:
            image = self.transform(image)

        return image, label

train_transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(20),
    transforms.ColorJitter(brightness=0.2, contrast=0.2,
    saturation=0.2, hue=0.1),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229,
    0.224, 0.225])
])

val_transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229,
    0.224, 0.225])
])

train_dataset = ODIRDataset(train_df, transform=train_transform)
val_dataset = ODIRDataset(val_df, transform=val_transform)
test_dataset = ODIRDataset(test_df, transform=val_transform)

train_loader = DataLoader(train_dataset, batch_size=32, shuffle=
    True)
val_loader = DataLoader(val_dataset, batch_size=32, shuffle=False)
test_loader = DataLoader(test_dataset, batch_size=32, shuffle=
```

```
    False)

"""# Model Development"""

model = timm.create_model('swin_tiny_patch4_window7_224',
    pretrained=True, num_classes=8)

device = torch.device("cuda" if torch.cuda.is_available() else "
    cpu")
model.to(device)

class_weights = compute_class_weight('balanced', classes=np.unique
    (eyes['labels_encoded']), y=eyes['labels_encoded'])
class_weights = torch.tensor(class_weights, dtype=torch.float).to(
    device)

criterion = nn.CrossEntropyLoss(weight=class_weights)
optimizer = optim.AdamW(model.parameters(), lr=1e-4)
scheduler = StepLR(optimizer, step_size=5, gamma=0.1)

save_dir = '/content/downloadable'
os.makedirs(save_dir, exist_ok=True)

if os.path.exists(os.path.join(save_dir, 'training_history.csv')):
  training_history_df = pd.read_csv(os.path.join(save_dir, '
    training_history.csv'))
else:
  training_history_df = pd.DataFrame(columns=["Epoch", "Train Loss
    ", "Val Loss", "Val Acc"])

def train(model, dataloader, criterion, optimizer, device):
    model.train()
    running_loss = 0.0
    for images, labels in dataloader:
        images, labels = images.to(device), labels.to(device)

        optimizer.zero_grad()
        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        running_loss += loss.item()

    return running_loss / len(dataloader)
```

```
609 def validate(model, dataloader, criterion, device):
610     model.eval()
611     val_loss = 0.0
612     correct = 0
613     total = 0
614 
615     with torch.no_grad():
616         for images, labels in dataloader:
617             images, labels = images.to(device), labels.to(device)
618             outputs = model(images)
619             loss = criterion(outputs, labels)
620             val_loss += loss.item()
621 
622             _, predicted = torch.max(outputs.data, 1)
623             total += labels.size(0)
624             correct += (predicted == labels).sum().item()
625 
626     return (val_loss / len(dataloader)), (correct / total)
627 
628 def training_epoch(training_history_df, prev=0):
629   NUM_OF_EPOCHS = 10+prev
630   for epoch in range(prev, NUM_OF_EPOCHS):
631       train_loss = train(model, train_loader, criterion, optimizer
      , device)
632       val_loss, val_acc = validate(model, val_loader, criterion,
     device)
633       scheduler.step()
634       print(f"Epoch {epoch+1}/{NUM_OF_EPOCHS}, Train Loss: {
     train_loss:.4f}, Val Loss: {val_loss:.4f}, Val Accuracy: {
     val_acc: .4f}")
635 
636       new_row = pd.DataFrame([{"Epoch": epoch+1, "Train Loss":
     train_loss, "Val Loss": val_loss, "Val Acc": val_acc}])
637       if not new_row.empty:
638         training_history_df = pd.concat([training_history_df,
     new_row], ignore_index=True)
639         training_history_df.to_csv(os.path.join(save_dir, '
     training_history.csv'), index=False)
640       epoch_model_path = os.path.join(save_dir, f'
     swin_model_epoch_{epoch+1}.pth')
641       torch.save(model.state_dict(), epoch_model_path)
642   torch.save(model.state_dict(), os.path.join(save_dir, '
     swin_model_final.pth'))
643 
644 
645 if os.path.exists(os.path.join(save_dir, 'swin_model_final.pth')):
```

```
  print("Swin Model Found!\nLoading the model!")
  ans = input("Is it your final model (Y) or Do you want to train further (N):")
  if ans.lower() == 'n':
    model.load_state_dict(torch.load(os.path.join(save_dir, 'swin_model_final.pth')))
    prev = len(training_history_df)
    training_epoch(training_history_df, prev)
else:
    training_epoch(training_history_df)

"""# Model Evaluation"""

model.eval()

all_preds = []
all_labels = []
all_probs = []

with torch.no_grad():
    for images, labels in test_loader:
        images, labels = images.to(device), labels.to(device)
        outputs = model(images)
        probs = torch.softmax(outputs, dim=1)
        _, preds = torch.max(outputs, 1)

        all_preds.extend(preds.cpu().numpy())
        all_labels.extend(labels.cpu().numpy())
        all_probs.extend(probs.cpu().numpy())

all_preds = np.array(all_preds)
all_labels = np.array(all_labels)
all_probs = np.array(all_probs)

accuracy = accuracy_score(all_labels, all_preds)
print(f'Accuracy: {accuracy:.4f}')

print('\nClassification Report:')
print(classification_report(all_labels, all_preds, target_names=label_encoder.classes_))

kappa = cohen_kappa_score(all_labels, all_preds)
print(f'Cohen Kappa Score: {kappa:.4f}')

f1 = f1_score(all_labels, all_preds, average='weighted')
print(f'F1 Score (Weighted): {f1:.4f}')
```

```
689
690 precision = precision_score(all_labels, all_preds, average='
    weighted')
691 print(f'Precision (Weighted): {precision:.4f}')
692
693 recall = recall_score(all_labels, all_preds, average='weighted')
694 print(f'Recall (Weighted): {recall:.4f}')
695
696 roc_auc = roc_auc_score(all_labels, all_probs, multi_class='ovr',
    average='weighted')
697 print(f'ROC AUC Score (Weighted): {roc_auc:.4f}')
698
699 n_classes = len(label_encoder.classes_)
700 fpr = dict()
701 tpr = dict()
702 roc_auc = dict()
703
704 for i in range(n_classes):
705     fpr[i], tpr[i], _ = roc_curve(all_labels == i, all_probs[:, i
    ])
706     roc_auc[i] = roc_auc_score(all_labels == i, all_probs[:, i])
707
708 plt.figure(figsize=(10, 8))
709 for i in range(n_classes):
710     plt.plot(fpr[i], tpr[i], label=f'Class {label_encoder.classes_
    [i]} (AUC = {roc_auc[i]:.2f})')
711
712 plt.plot([0, 1], [0, 1], 'k--')
713 plt.xlabel('False Positive Rate')
714 plt.ylabel('True Positive Rate')
715 plt.title('ROC Curve for Each Class')
716 plt.legend(loc='lower right')
717 plt.show()
718
719 conf_matrix = confusion_matrix(all_labels, all_preds)
720 print('\nConfusion Matrix:')
721 print(conf_matrix)
722
723 plt.figure(figsize=(10, 8))
724 sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
725             xticklabels=label_encoder.classes_,
726             yticklabels=label_encoder.classes_)
727 plt.xlabel('Predicted Labels')
728 plt.ylabel('True Labels')
729 plt.title('Confusion Matrix')
730 plt.show()
```

```
731
732 def sensitivity_specificity_multiclass(conf_matrix):
733     num_classes = conf_matrix.shape[0]
734     sensitivity = []
735     specificity = []
736
737     for i in range(num_classes):
738         TP = conf_matrix[i, i]
739         FN = sum(conf_matrix[i, :]) - TP
740         FP = sum(conf_matrix[:, i]) - TP
741         TN = conf_matrix.sum() - (TP + FP + FN)
742
743         sens = TP / (TP + FN) if (TP + FN) != 0 else 0
744         spec = TN / (TN + FP) if (TN + FP) != 0 else 0
745
746         sensitivity.append(sens)
747         specificity.append(spec)
748
749     return sensitivity, specificity
750
751 sensitivity, specificity = sensitivity_specificity_multiclass(
    conf_matrix)
752
753 print('\nSensitivity and Specificity for Each Class:')
754 for i, (sens, spec) in enumerate(zip(sensitivity, specificity)):
755     print(f'Class {label_encoder.classes_[i]}:')
756     print(f'  Sensitivity: {sens:.4f}')
757     print(f'  Specificity: {spec:.4f}')
758
759 def accuracy_per_class(conf_matrix):
760     num_classes = conf_matrix.shape[0]
761     class_accuracy = []
762
763     for i in range(num_classes):
764         TP = conf_matrix[i, i]
765         FN = sum(conf_matrix[i, :]) - TP
766         FP = sum(conf_matrix[:, i]) - TP
767         TN = conf_matrix.sum() - (TP + FP + FN)
768
769         acc = (TP + TN) / conf_matrix.sum() if conf_matrix.sum()
    != 0 else 0
770         class_accuracy.append(acc)
771
772     return class_accuracy
773
774 class_accuracy = accuracy_per_class(conf_matrix)
```

```
775
776 print('\nAccuracy for Each Class:')
777 for i, acc in enumerate(class_accuracy):
778     print(f'Class {label_encoder.classes_[i]}: {acc:.4f}')
779
780 plt.figure(figsize=(10,6))
781
782 plt.plot(training_history_df['Epoch'], training_history_df['Train
    Loss'], label='Train Loss', color='blue')
783 plt.plot(training_history_df['Epoch'], training_history_df['Val
    Loss'], label='Validation Loss', color='red')
784
785 plt.title('Training and Validation Loss Over Epochs')
786 plt.xlabel('Epoch')
787 plt.ylabel('Loss')
788 plt.legend()
789 plt.show()
```

Listing 4.1: *myswin model.py*

4.6.2 File Structure of Dataset

```
Folder: /root/.cache/kagglehub/datasets/andrewmvd/ocular-disease-recognition-
odir5k/versions/2
Number of subfolders: 2
Number of files: 1
Total size: 1494762 bytes
========================================
Single file: full\_df.csv
Total size: 1494762 bytes
File path: /root/.cache/kagglehub/datasets/andrewmvd/ocular-disease-recognition-
odir5k/versions/2/full\_df.csv
----------------------------------------
Folder: /root/.cache/kagglehub/datasets/andrewmvd/ocular-disease-recognition-
odir5k/versions/2/ODIR-5K
Number of subfolders: 1
Number of files: 0
Total size: 0 bytes
========================================
Folder: /root/.cache/kagglehub/datasets/andrewmvd/ocular-disease-recognition-
odir5k/versions/2/ODIR-5K/ODIR-5K
```

```
Number of subfolders: 2
Number of files: 1
Total size: 268198 bytes
========================================
Single file: data.xlsx
Total size: 268198 bytes
File path: /root/.cache/kagglehub/datasets/andrewmvd/ocular-disease-recognition-
odir5k/versions/2/ODIR-5K/ODIR-5K/data.xlsx
----------------------------------------
Folder: /root/.cache/kagglehub/datasets/andrewmvd/ocular-disease-recognition-
odir5k/versions/2/ODIR-5K/ODIR-5K/Testing Images
Number of subfolders: 0
Number of files: 1000
Total size: 209848981 bytes
========================================
Folder: /root/.cache/kagglehub/datasets/andrewmvd/ocular-disease-recognition-
odir5k/versions/2/ODIR-5K/ODIR-5K/Training Images
Number of subfolders: 0
Number of files: 7000
Total size: 1425587322 bytes
========================================
Folder: /root/.cache/kagglehub/datasets/andrewmvd/ocular-disease-recognition-
odir5k/versions/2/preprocessed\_images
Number of subfolders: 0
Number of files: 6392
Total size: 397121132 bytes
========================================
```

4.6.3 Dataset

Complete Dataset:

	ID	Patient Age	Patient Sex	Left-Fundus	Right-Fundus
0	0	69	Female	0_left.jpg	0_right.jpg

```
1        1           57        Male     1\_left.jpg     1\_right.jpg
2        2           42        Male     2\_left.jpg     2\_right.jpg
3        4           53        Male     4\_left.jpg     4\_right.jpg
4        5           50      Female     5\_left.jpg     5\_right.jpg
....    ....          ....         ....            ....             ....
6387  4686           63        Male  4686\_left.jpg  4686\_right.jpg
6388  4688           42        Male  4688\_left.jpg  4688\_right.jpg
6389  4689           54        Male  4689\_left.jpg  4689\_right.jpg
6390  4690           57        Male  4690\_left.jpg  4690\_right.jpg
6391  4784           58        Male  4784\_left.jpg  4784\_right.jpg

                              Left-Diagnostic Keywords
0                                             cataract
1                                         normal fundus
2     laser spot,moderate non proliferative retinopathy
3                            macular epiretinal membrane
4                 moderate non proliferative retinopathy
....                                                ....
6387                severe nonproliferative retinopathy
6388              moderate non proliferative retinopathy
6389                   mild nonproliferative retinopathy
6390                   mild nonproliferative retinopathy
6391  hypertensive retinopathy,age-related macular d....

                               Right-Diagnostic Keywords  N  D  G  C  A  H  M
0                                         normal fundus  0  0  0  1  0  0  0
1                                         normal fundus  1  0  0  0  0  0  0
2                 moderate non proliferative retinopathy  0  1  0  0  0  0  0
3                      mild nonproliferative retinopathy  0  1  0  0  0  0  0
4                 moderate non proliferative retinopathy  0  1  0  0  0  0  0
....                                                ....  .. .. .. .. .. .. ..
6387                proliferative diabetic retinopathy  0  1  0  0  0  0  0
6388              moderate non proliferative retinopathy  0  1  0  0  0  0  0
6389                                      normal fundus  0  1  0  0  0  0  0
6390                   mild nonproliferative retinopathy  0  1  0  0  0  0  0
6391  hypertensive retinopathy,age-related macular d....  0  0  0  0  1  1  0

      0                                        filepath labels
```

```
0     0  ../input/ocular-disease-recognition-odir5k/ODI....  ['N']
1     0  ../input/ocular-disease-recognition-odir5k/ODI....  ['N']
2     1  ../input/ocular-disease-recognition-odir5k/ODI....  ['D']
3     1  ../input/ocular-disease-recognition-odir5k/ODI....  ['D']
4     0  ../input/ocular-disease-recognition-odir5k/ODI....  ['D']
....  ..                                                 ....    ....
6387  0  ../input/ocular-disease-recognition-odir5k/ODI....  ['D']
6388  0  ../input/ocular-disease-recognition-odir5k/ODI....  ['D']
6389  0  ../input/ocular-disease-recognition-odir5k/ODI....  ['D']
6390  0  ../input/ocular-disease-recognition-odir5k/ODI....  ['D']
6391  0  ../input/ocular-disease-recognition-odir5k/ODI....  ['H']

                        target       filename
0     [1, 0, 0, 0, 0, 0, 0, 0]    0\_right.jpg
1     [1, 0, 0, 0, 0, 0, 0, 0]    1\_right.jpg
2     [0, 1, 0, 0, 0, 0, 0, 0]    2\_right.jpg
3     [0, 1, 0, 0, 0, 0, 0, 0]    4\_right.jpg
4     [0, 1, 0, 0, 0, 0, 0, 0]    5\_right.jpg
....                          ....            ....
6387  [0, 1, 0, 0, 0, 0, 0, 0]  4686\_left.jpg
6388  [0, 1, 0, 0, 0, 0, 0, 0]  4688\_left.jpg
6389  [0, 1, 0, 0, 0, 0, 0, 0]  4689\_left.jpg
6390  [0, 1, 0, 0, 0, 0, 0, 0]  4690\_left.jpg
6391  [0, 0, 0, 0, 0, 1, 0, 0]  4784\_left.jpg

[6392 rows x 19 columns]
```

Information on Dataset:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6392 entries, 0 to 6391
Data columns (total 19 columns):
 \#   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   ID                         6392 non-null   int64
 1   Patient Age                6392 non-null   int64
 2   Patient Sex                6392 non-null   object
```

```
 3   Left-Fundus                6392 non-null   object
 4   Right-Fundus               6392 non-null   object
 5   Left-Diagnostic Keywords   6392 non-null   object
 6   Right-Diagnostic Keywords  6392 non-null   object
 7   N                          6392 non-null   int64
 8   D                          6392 non-null   int64
 9   G                          6392 non-null   int64
 10  C                          6392 non-null   int64
 11  A                          6392 non-null   int64
 12  H                          6392 non-null   int64
 13  M                          6392 non-null   int64
 14  O                          6392 non-null   int64
 15  filepath                   6392 non-null   object
 16  labels                     6392 non-null   object
 17  target                     6392 non-null   object
 18  filename                   6392 non-null   object
dtypes: int64(10), object(9)
memory usage: 948.9+ KB
```

Describing the Dataset:

	ID	Patient Age	Patient Sex	Left-Fundus	Right-Fundus	
count	6392.000000	6392.000000	6392	6392	6392	
unique	NaN	NaN	2	3358	3358	
top	NaN	NaN	Male	4690_left.jpg	4690_right.jpg	
freq	NaN	NaN	3424	2	2	
mean	2271.150814	57.857947	NaN	NaN	NaN	
std	1417.559018	11.727737	NaN	NaN	NaN	
min	0.000000	1.000000	NaN	NaN	NaN	
25\%	920.750000	51.000000	NaN	NaN	NaN	
50\%	2419.500000	59.000000	NaN	NaN	NaN	
75\%	3294.000000	66.000000	NaN	NaN	NaN	
max	4784.000000	91.000000	NaN	NaN	NaN	

	Left-Diagnostic Keywords	Right-Diagnostic Keywords	N
count	6392	6392	6392.000000

unique	196	205	NaN	
top	normal fundus	normal fundus	NaN	
freq	2796	2705	NaN	
mean	NaN	NaN	0.328692	
std	NaN	NaN	0.469775	
min	NaN	NaN	0.000000	
25\%	NaN	NaN	0.000000	
50\%	NaN	NaN	0.000000	
75\%	NaN	NaN	1.000000	
max	NaN	NaN	1.000000	

	D	G	C	A	H
count	6392.000000	6392.000000	6392.000000	6392.000000	6392.000000
unique	NaN	NaN	NaN	NaN	NaN
top	NaN	NaN	NaN	NaN	NaN
freq	NaN	NaN	NaN	NaN	NaN
mean	0.332134	0.062109	0.062891	0.049906	0.031758
std	0.471016	0.241372	0.242786	0.217768	0.175370
min	0.000000	0.000000	0.000000	0.000000	0.000000
25\%	0.000000	0.000000	0.000000	0.000000	0.000000
50\%	0.000000	0.000000	0.000000	0.000000	0.000000
75\%	1.000000	0.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000

	M	O
count	6392.000000	6392.000000
unique	NaN	NaN
top	NaN	NaN
freq	NaN	NaN
mean	0.047872	0.248436
std	0.213513	0.432139
min	0.000000	0.000000
25\%	0.000000	0.000000
50\%	0.000000	0.000000
75\%	0.000000	0.000000
max	1.000000	1.000000

	filepath	labels

count	6392	6392
unique	6392	8
top	../input/ocular-disease-recognition-odir5k/ODI....	['N']
freq	1	2873
mean	NaN	NaN
std	NaN	NaN
min	NaN	NaN
25\%	NaN	NaN
50\%	NaN	NaN
75\%	NaN	NaN
max	NaN	NaN

	target	filename
count	6392	6392
unique	8	6392
top	[1, 0, 0, 0, 0, 0, 0, 0]	4784_left.jpg
freq	2873	1
mean	NaN	NaN
std	NaN	NaN
min	NaN	NaN
25\%	NaN	NaN
50\%	NaN	NaN
75\%	NaN	NaN
max	NaN	NaN

Columns in the dataset:

```
Index(['ID', 'Patient Age', 'Patient Sex', 'Left-Fundus', 'Right-Fundus',
       'Left-Diagnostic Keywords', 'Right-Diagnostic Keywords', 'N', 'D', 'G',
       'C', 'A', 'H', 'M', 'O', 'filepath', 'labels', 'target', 'filename'],
      dtype='object')
```

Labels Counts:

```
labels
```

```
N    2873
D    1608
O     708
C     293
G     284
A     266
M     232
H     128
Name: count, dtype: int64
```

4.6.4 Swin model

```
SwinTransformer(
  (patch\_embed): PatchEmbed(
    (proj): Conv2d(3, 96, kernel\_size=(4, 4), stride=(4, 4))
    (norm): LayerNorm((96,), eps=1e-05, elementwise\_affine=True)
  )
  (layers): Sequential(
    (0): SwinTransformerStage(
      (downsample): Identity()
      (blocks): Sequential(
        (0): SwinTransformerBlock(
          (norm1): LayerNorm((96,), eps=1e-05, elementwise\_affine=True)
          (attn): WindowAttention(
            (qkv): Linear(in\_features=96, out\_features=288, bias=True)
            (attn\_drop): Dropout(p=0.0, inplace=False)
            (proj): Linear(in\_features=96, out\_features=96, bias=True)
            (proj\_drop): Dropout(p=0.0, inplace=False)
            (softmax): Softmax(dim=-1)
          )
          (drop\_path1): Identity()
          (norm2): LayerNorm((96,), eps=1e-05, elementwise\_affine=True)
          (mlp): Mlp(
            (fc1): Linear(in\_features=96, out\_features=384, bias=True)
            (act): GELU(approximate='none')
            (drop1): Dropout(p=0.0, inplace=False)
            (norm): Identity()
```

```
        (fc2): Linear(in\_features=384, out\_features=96, bias=True)
        (drop2): Dropout(p=0.0, inplace=False)
      )
      (drop\_path2): Identity()
    )
    (1): SwinTransformerBlock(
      (norm1): LayerNorm((96,), eps=1e-05, elementwise\_affine=True)
      (attn): WindowAttention(
        (qkv): Linear(in\_features=96, out\_features=288, bias=True)
        (attn\_drop): Dropout(p=0.0, inplace=False)
        (proj): Linear(in\_features=96, out\_features=96, bias=True)
        (proj\_drop): Dropout(p=0.0, inplace=False)
        (softmax): Softmax(dim=-1)
      )
      (drop\_path1): DropPath(drop\_prob=0.009)
      (norm2): LayerNorm((96,), eps=1e-05, elementwise\_affine=True)
      (mlp): Mlp(
        (fc1): Linear(in\_features=96, out\_features=384, bias=True)
        (act): GELU(approximate='none')
        (drop1): Dropout(p=0.0, inplace=False)
        (norm): Identity()
        (fc2): Linear(in\_features=384, out\_features=96, bias=True)
        (drop2): Dropout(p=0.0, inplace=False)
      )
      (drop\_path2): DropPath(drop\_prob=0.009)
    )
  )
)
(1): SwinTransformerStage(
  (downsample): PatchMerging(
    (norm): LayerNorm((384,), eps=1e-05, elementwise\_affine=True)
    (reduction): Linear(in\_features=384, out\_features=192, bias=False)
  )
  (blocks): Sequential(
    (0): SwinTransformerBlock(
      (norm1): LayerNorm((192,), eps=1e-05, elementwise\_affine=True)
      (attn): WindowAttention(
        (qkv): Linear(in\_features=192, out\_features=576, bias=True)
```

```
    (attn\_drop): Dropout(p=0.0, inplace=False)
    (proj): Linear(in\_features=192, out\_features=192, bias=True)
    (proj\_drop): Dropout(p=0.0, inplace=False)
    (softmax): Softmax(dim=-1)
  )
  (drop\_path1): DropPath(drop\_prob=0.018)
  (norm2): LayerNorm((192,), eps=1e-05, elementwise\_affine=True)
  (mlp): Mlp(
    (fc1): Linear(in\_features=192, out\_features=768, bias=True)
    (act): GELU(approximate='none')
    (drop1): Dropout(p=0.0, inplace=False)
    (norm): Identity()
    (fc2): Linear(in\_features=768, out\_features=192, bias=True)
    (drop2): Dropout(p=0.0, inplace=False)
  )
  (drop\_path2): DropPath(drop\_prob=0.018)
)
(1): SwinTransformerBlock(
  (norm1): LayerNorm((192,), eps=1e-05, elementwise\_affine=True)
  (attn): WindowAttention(
    (qkv): Linear(in\_features=192, out\_features=576, bias=True)
    (attn\_drop): Dropout(p=0.0, inplace=False)
    (proj): Linear(in\_features=192, out\_features=192, bias=True)
    (proj\_drop): Dropout(p=0.0, inplace=False)
    (softmax): Softmax(dim=-1)
  )
  (drop\_path1): DropPath(drop\_prob=0.027)
  (norm2): LayerNorm((192,), eps=1e-05, elementwise\_affine=True)
  (mlp): Mlp(
    (fc1): Linear(in\_features=192, out\_features=768, bias=True)
    (act): GELU(approximate='none')
    (drop1): Dropout(p=0.0, inplace=False)
    (norm): Identity()
    (fc2): Linear(in\_features=768, out\_features=192, bias=True)
    (drop2): Dropout(p=0.0, inplace=False)
  )
  (drop\_path2): DropPath(drop\_prob=0.027)
)
```

```
  )
)
(2): SwinTransformerStage(
  (downsample): PatchMerging(
    (norm): LayerNorm((768,), eps=1e-05, elementwise\_affine=True)
    (reduction): Linear(in\_features=768, out\_features=384, bias=False)
  )
  (blocks): Sequential(
    (0): SwinTransformerBlock(
      (norm1): LayerNorm((384,), eps=1e-05, elementwise\_affine=True)
      (attn): WindowAttention(
        (qkv): Linear(in\_features=384, out\_features=1152, bias=True)
        (attn\_drop): Dropout(p=0.0, inplace=False)
        (proj): Linear(in\_features=384, out\_features=384, bias=True)
        (proj\_drop): Dropout(p=0.0, inplace=False)
        (softmax): Softmax(dim=-1)
      )
      (drop\_path1): DropPath(drop\_prob=0.036)
      (norm2): LayerNorm((384,), eps=1e-05, elementwise\_affine=True)
      (mlp): Mlp(
        (fc1): Linear(in\_features=384, out\_features=1536, bias=True)
        (act): GELU(approximate='none')
        (drop1): Dropout(p=0.0, inplace=False)
        (norm): Identity()
        (fc2): Linear(in\_features=1536, out\_features=384, bias=True)
        (drop2): Dropout(p=0.0, inplace=False)
      )
      (drop\_path2): DropPath(drop\_prob=0.036)
    )
    (1): SwinTransformerBlock(
      (norm1): LayerNorm((384,), eps=1e-05, elementwise\_affine=True)
      (attn): WindowAttention(
        (qkv): Linear(in\_features=384, out\_features=1152, bias=True)
        (attn\_drop): Dropout(p=0.0, inplace=False)
        (proj): Linear(in\_features=384, out\_features=384, bias=True)
        (proj\_drop): Dropout(p=0.0, inplace=False)
        (softmax): Softmax(dim=-1)
      )
```

```
  (drop\_path1): DropPath(drop\_prob=0.045)
  (norm2): LayerNorm((384,), eps=1e-05, elementwise\_affine=True)
  (mlp): Mlp(
    (fc1): Linear(in\_features=384, out\_features=1536, bias=True)
    (act): GELU(approximate='none')
    (drop1): Dropout(p=0.0, inplace=False)
    (norm): Identity()
    (fc2): Linear(in\_features=1536, out\_features=384, bias=True)
    (drop2): Dropout(p=0.0, inplace=False)
  )
  (drop\_path2): DropPath(drop\_prob=0.045)
)
(2): SwinTransformerBlock(
  (norm1): LayerNorm((384,), eps=1e-05, elementwise\_affine=True)
  (attn): WindowAttention(
    (qkv): Linear(in\_features=384, out\_features=1152, bias=True)
    (attn\_drop): Dropout(p=0.0, inplace=False)
    (proj): Linear(in\_features=384, out\_features=384, bias=True)
    (proj\_drop): Dropout(p=0.0, inplace=False)
    (softmax): Softmax(dim=-1)
  )
  (drop\_path1): DropPath(drop\_prob=0.055)
  (norm2): LayerNorm((384,), eps=1e-05, elementwise\_affine=True)
  (mlp): Mlp(
    (fc1): Linear(in\_features=384, out\_features=1536, bias=True)
    (act): GELU(approximate='none')
    (drop1): Dropout(p=0.0, inplace=False)
    (norm): Identity()
    (fc2): Linear(in\_features=1536, out\_features=384, bias=True)
    (drop2): Dropout(p=0.0, inplace=False)
  )
  (drop\_path2): DropPath(drop\_prob=0.055)
)
(3): SwinTransformerBlock(
  (norm1): LayerNorm((384,), eps=1e-05, elementwise\_affine=True)
  (attn): WindowAttention(
    (qkv): Linear(in\_features=384, out\_features=1152, bias=True)
    (attn\_drop): Dropout(p=0.0, inplace=False)
```

```
      (proj): Linear(in\_features=384, out\_features=384, bias=True)
      (proj\_drop): Dropout(p=0.0, inplace=False)
      (softmax): Softmax(dim=-1)
    )
    (drop\_path1): DropPath(drop\_prob=0.064)
    (norm2): LayerNorm((384,), eps=1e-05, elementwise\_affine=True)
    (mlp): Mlp(
      (fc1): Linear(in\_features=384, out\_features=1536, bias=True)
      (act): GELU(approximate='none')
      (drop1): Dropout(p=0.0, inplace=False)
      (norm): Identity()
      (fc2): Linear(in\_features=1536, out\_features=384, bias=True)
      (drop2): Dropout(p=0.0, inplace=False)
    )
    (drop\_path2): DropPath(drop\_prob=0.064)
  )
  (4): SwinTransformerBlock(
    (norm1): LayerNorm((384,), eps=1e-05, elementwise\_affine=True)
    (attn): WindowAttention(
      (qkv): Linear(in\_features=384, out\_features=1152, bias=True)
      (attn\_drop): Dropout(p=0.0, inplace=False)
      (proj): Linear(in\_features=384, out\_features=384, bias=True)
      (proj\_drop): Dropout(p=0.0, inplace=False)
      (softmax): Softmax(dim=-1)
    )
    (drop\_path1): DropPath(drop\_prob=0.073)
    (norm2): LayerNorm((384,), eps=1e-05, elementwise\_affine=True)
    (mlp): Mlp(
      (fc1): Linear(in\_features=384, out\_features=1536, bias=True)
      (act): GELU(approximate='none')
      (drop1): Dropout(p=0.0, inplace=False)
      (norm): Identity()
      (fc2): Linear(in\_features=1536, out\_features=384, bias=True)
      (drop2): Dropout(p=0.0, inplace=False)
    )
    (drop\_path2): DropPath(drop\_prob=0.073)
  )
  (5): SwinTransformerBlock(
```

```
      (norm1): LayerNorm((384,), eps=1e-05, elementwise\_affine=True)
      (attn): WindowAttention(
        (qkv): Linear(in\_features=384, out\_features=1152, bias=True)
        (attn\_drop): Dropout(p=0.0, inplace=False)
        (proj): Linear(in\_features=384, out\_features=384, bias=True)
        (proj\_drop): Dropout(p=0.0, inplace=False)
        (softmax): Softmax(dim=-1)
      )
      (drop\_path1): DropPath(drop\_prob=0.082)
      (norm2): LayerNorm((384,), eps=1e-05, elementwise\_affine=True)
      (mlp): Mlp(
        (fc1): Linear(in\_features=384, out\_features=1536, bias=True)
        (act): GELU(approximate='none')
        (drop1): Dropout(p=0.0, inplace=False)
        (norm): Identity()
        (fc2): Linear(in\_features=1536, out\_features=384, bias=True)
        (drop2): Dropout(p=0.0, inplace=False)
      )
      (drop\_path2): DropPath(drop\_prob=0.082)
    )
  )
)
(3): SwinTransformerStage(
  (downsample): PatchMerging(
    (norm): LayerNorm((1536,), eps=1e-05, elementwise\_affine=True)
    (reduction): Linear(in\_features=1536, out\_features=768, bias=False)
  )
  (blocks): Sequential(
    (0): SwinTransformerBlock(
      (norm1): LayerNorm((768,), eps=1e-05, elementwise\_affine=True)
      (attn): WindowAttention(
        (qkv): Linear(in\_features=768, out\_features=2304, bias=True)
        (attn\_drop): Dropout(p=0.0, inplace=False)
        (proj): Linear(in\_features=768, out\_features=768, bias=True)
        (proj\_drop): Dropout(p=0.0, inplace=False)
        (softmax): Softmax(dim=-1)
      )
      (drop\_path1): DropPath(drop\_prob=0.091)
```

```
        (norm2): LayerNorm((768,), eps=1e-05, elementwise\_affine=True)
        (mlp): Mlp(
          (fc1): Linear(in\_features=768, out\_features=3072, bias=True)
          (act): GELU(approximate='none')
          (drop1): Dropout(p=0.0, inplace=False)
          (norm): Identity()
          (fc2): Linear(in\_features=3072, out\_features=768, bias=True)
          (drop2): Dropout(p=0.0, inplace=False)
        )
        (drop\_path2): DropPath(drop\_prob=0.091)
      )
      (1): SwinTransformerBlock(
        (norm1): LayerNorm((768,), eps=1e-05, elementwise\_affine=True)
        (attn): WindowAttention(
          (qkv): Linear(in\_features=768, out\_features=2304, bias=True)
          (attn\_drop): Dropout(p=0.0, inplace=False)
          (proj): Linear(in\_features=768, out\_features=768, bias=True)
          (proj\_drop): Dropout(p=0.0, inplace=False)
          (softmax): Softmax(dim=-1)
        )
        (drop\_path1): DropPath(drop\_prob=0.100)
        (norm2): LayerNorm((768,), eps=1e-05, elementwise\_affine=True)
        (mlp): Mlp(
          (fc1): Linear(in\_features=768, out\_features=3072, bias=True)
          (act): GELU(approximate='none')
          (drop1): Dropout(p=0.0, inplace=False)
          (norm): Identity()
          (fc2): Linear(in\_features=3072, out\_features=768, bias=True)
          (drop2): Dropout(p=0.0, inplace=False)
        )
        (drop\_path2): DropPath(drop\_prob=0.100)
      )
    )
  )
)
(norm): LayerNorm((768,), eps=1e-05, elementwise\_affine=True)
(head): ClassifierHead(
  (global\_pool): SelectAdaptivePool2d(pool\_type=avg, flatten=Identity())
```

```
    (drop): Dropout(p=0.0, inplace=False)
    (fc): Linear(in\_features=768, out\_features=8, bias=True)
    (flatten): Identity()
  )
)
```

4.6.5 Per Epoch values

```
Epoch 1/10, Train Loss: 1.7329, Val Loss: 1.4525, Val Accuracy:  0.4664
Epoch 2/10, Train Loss: 1.4441, Val Loss: 1.3828, Val Accuracy:  0.5149
Epoch 3/10, Train Loss: 1.2777, Val Loss: 1.4203, Val Accuracy:  0.3731
Epoch 4/10, Train Loss: 1.2613, Val Loss: 1.2363, Val Accuracy:  0.4328
Epoch 5/10, Train Loss: 1.1328, Val Loss: 1.2642, Val Accuracy:  0.4776
Epoch 6/10, Train Loss: 1.0112, Val Loss: 1.1461, Val Accuracy:  0.4851
Epoch 7/10, Train Loss: 0.9212, Val Loss: 1.1684, Val Accuracy:  0.5336
Epoch 8/10, Train Loss: 0.8877, Val Loss: 1.1479, Val Accuracy:  0.5037
Epoch 9/10, Train Loss: 0.8449, Val Loss: 1.1782, Val Accuracy:  0.4813
Epoch 10/10, Train Loss: 0.8306, Val Loss: 1.1760, Val Accuracy:  0.5597
```

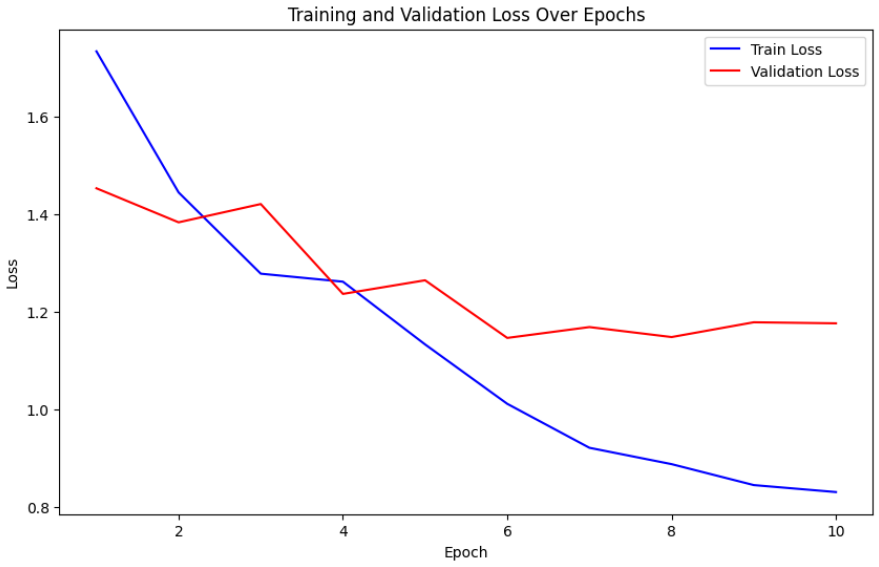

Figure 4.2: Training and Validation Loss over epochs

Chapter 5

Visualization and Testing

5.1 Data Visualization

Data visualization was performed to gain insights into the dataset. This includes:

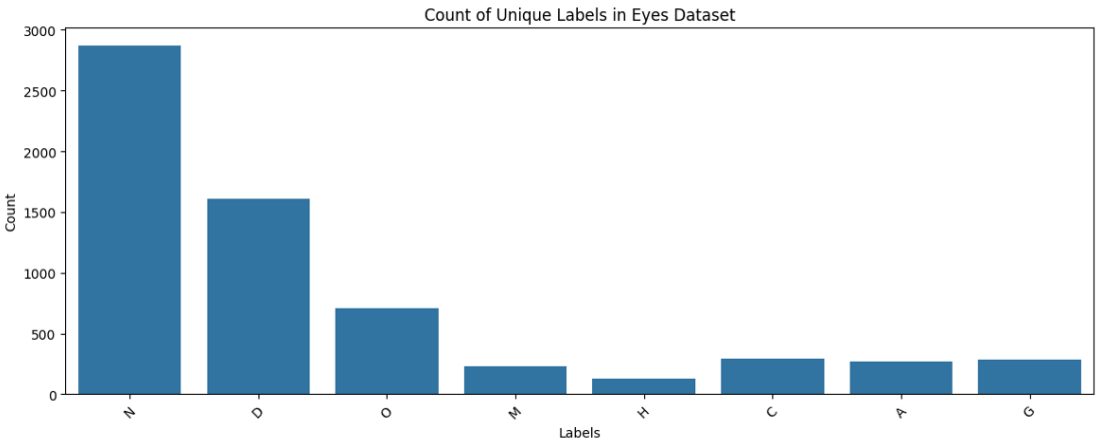

Figure 5.1: Distribution of diseases across the dataset

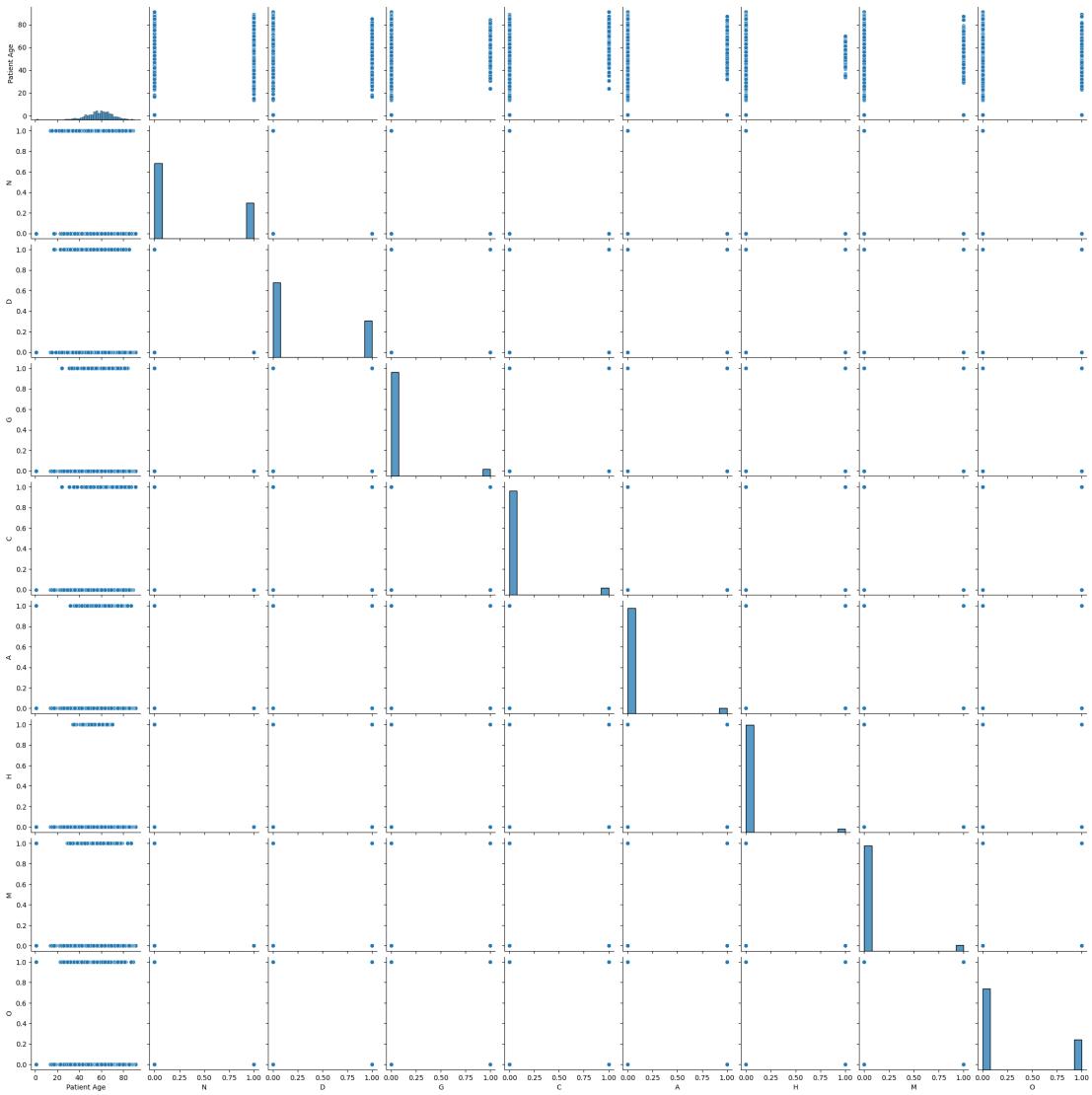

Figure 5.2: Pairplot of the ODIR Dataset

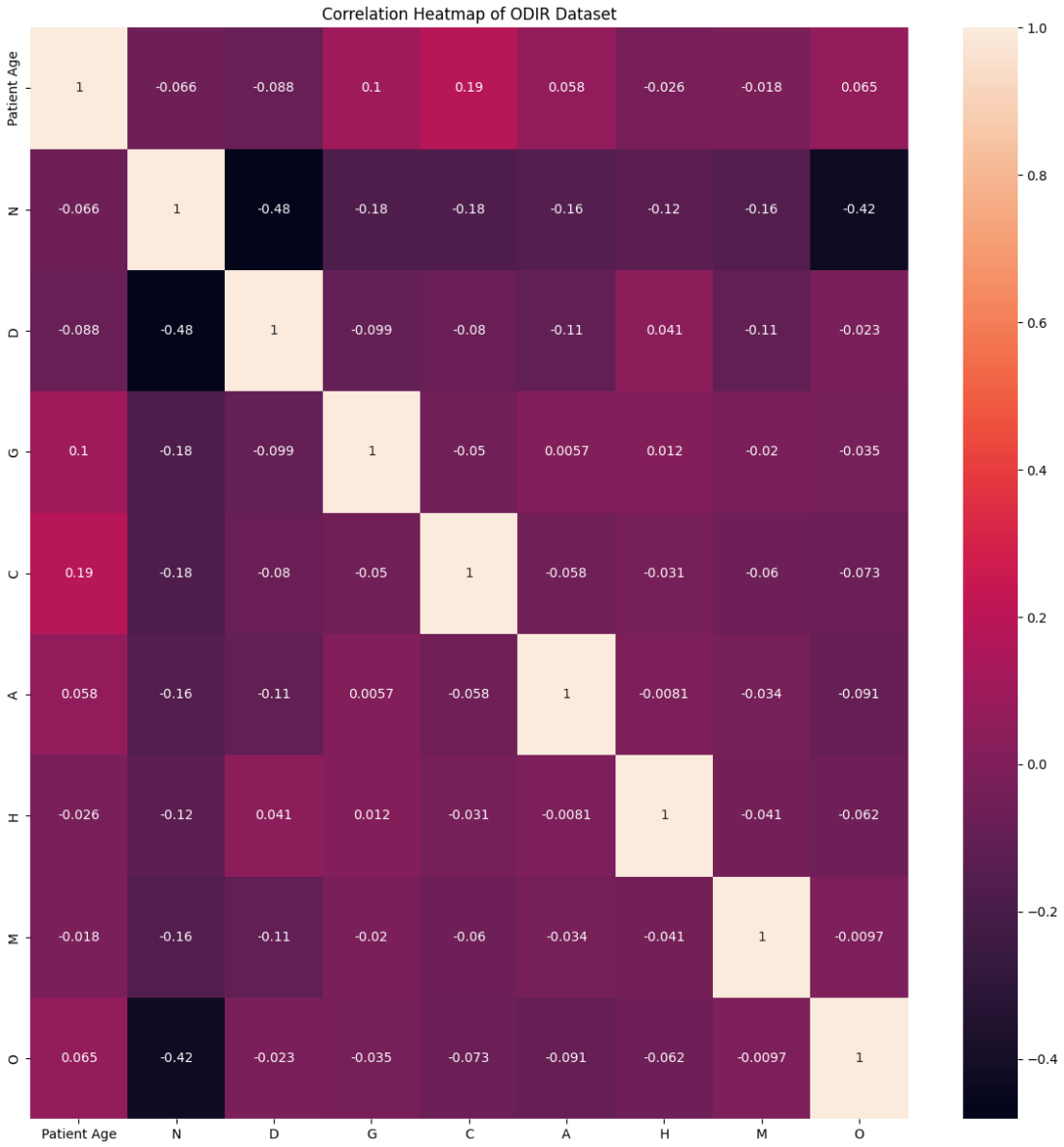

Figure 5.3: Correlation Heatmap of the ODIR Dataset

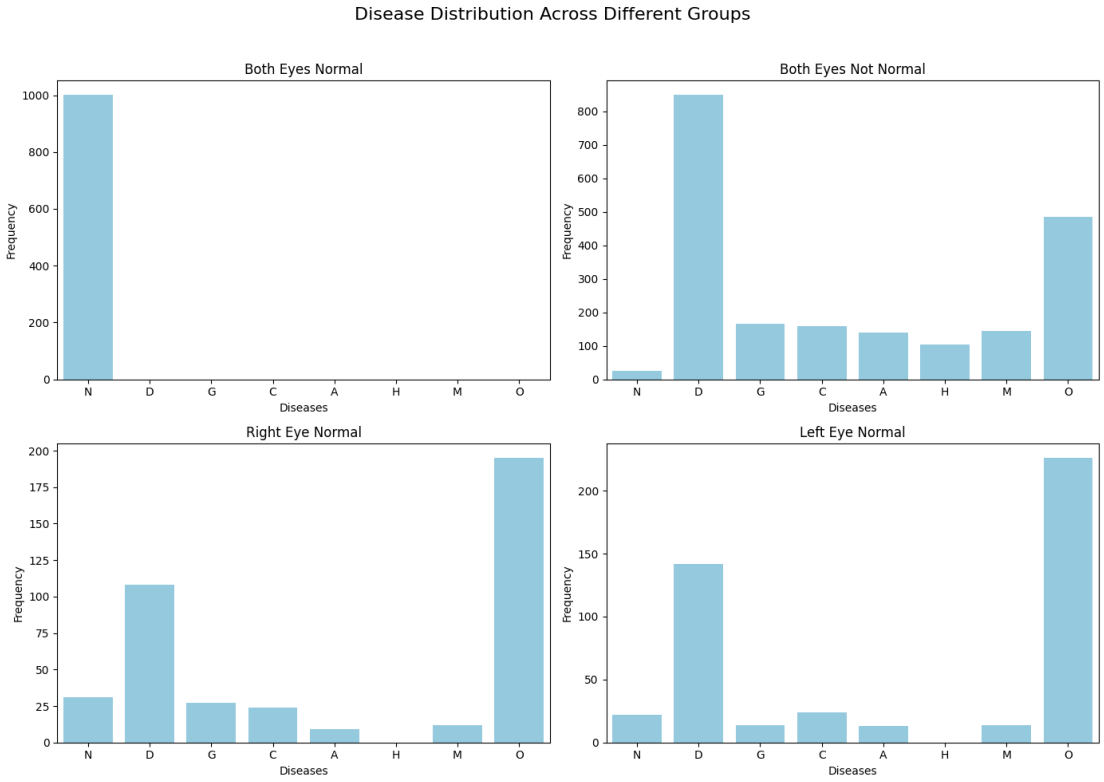

Figure 5.4: Disease Distribution Across Different Groups

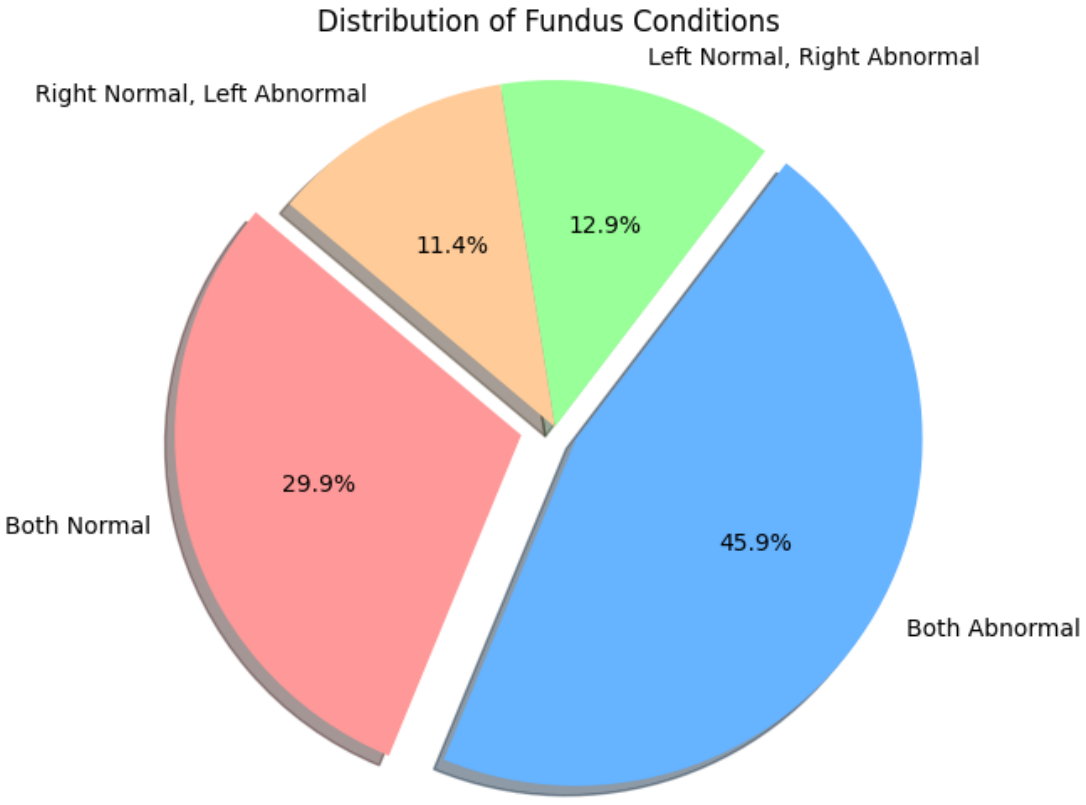

Figure 5.5: Distribution of Fundus Conditions

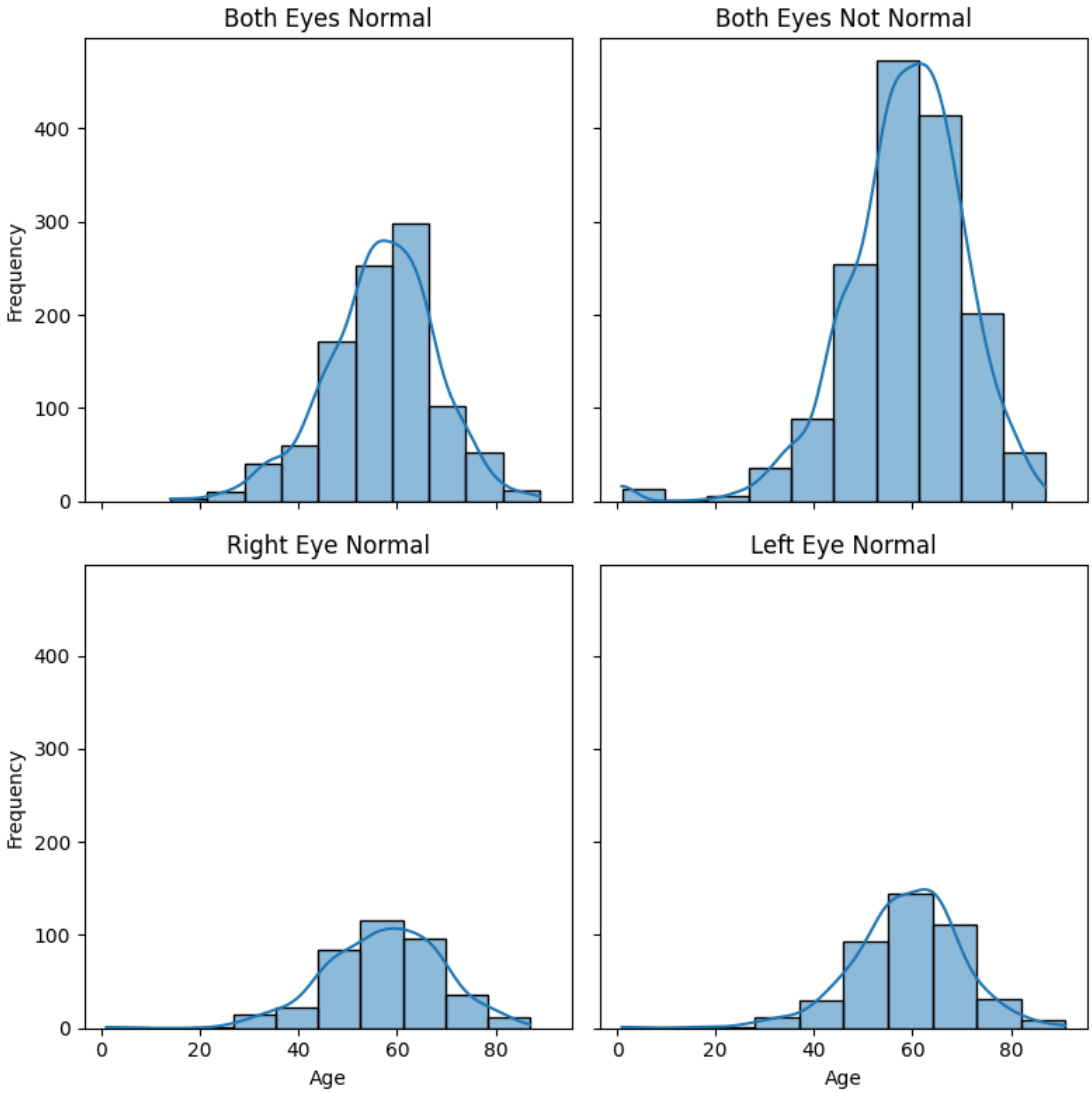

Figure 5.6: Distribution of Fundus Conditions against Patient's Age

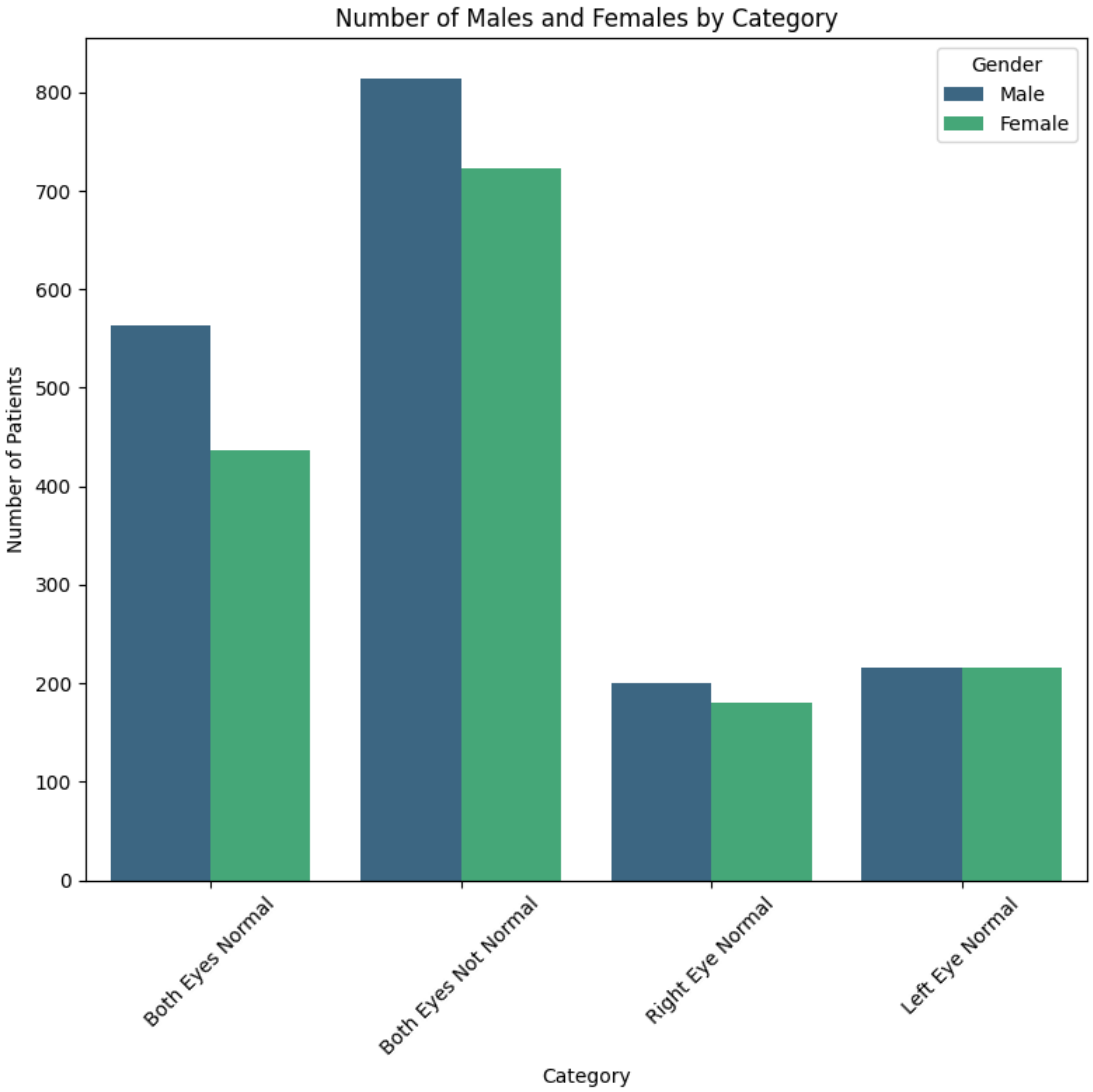

Figure 5.7: Distribution of Fundus Conditions against Patient's Gender

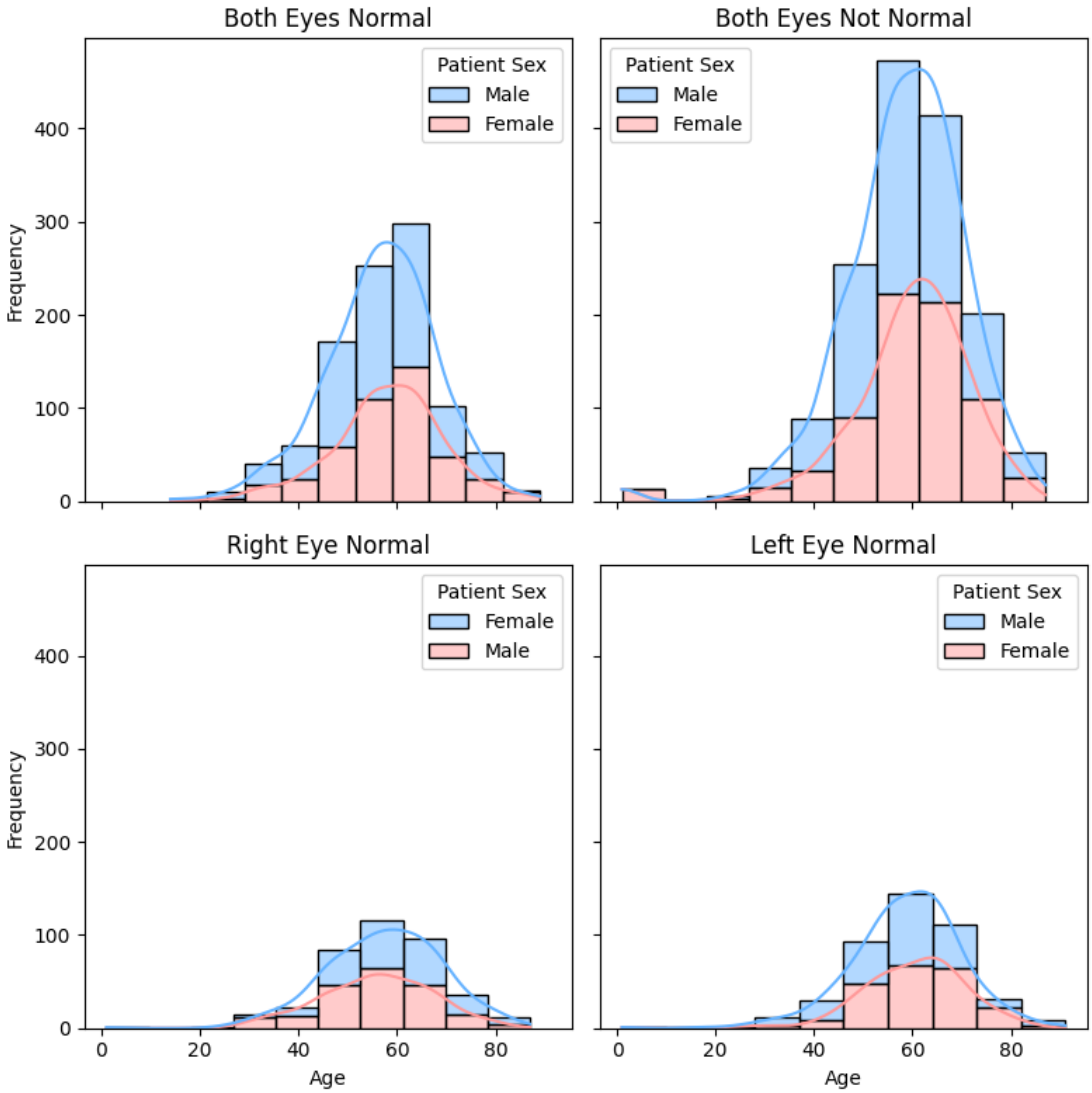

Figure 5.8: Distribution of Fundus Conditions against Patient's Age & Gender

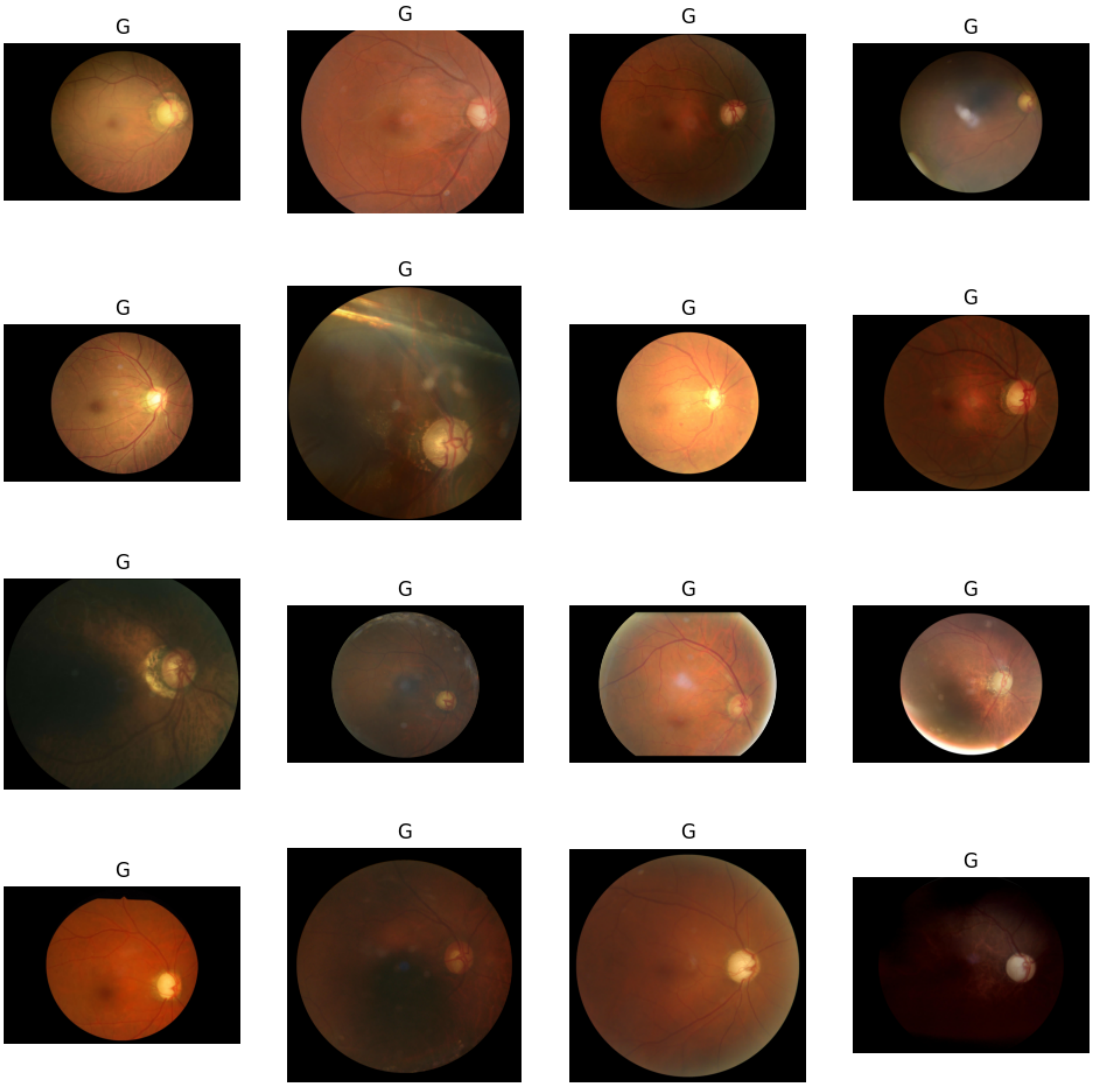

Figure 5.9: Sample Glaucoma Retinal Images in ODIR Dataset

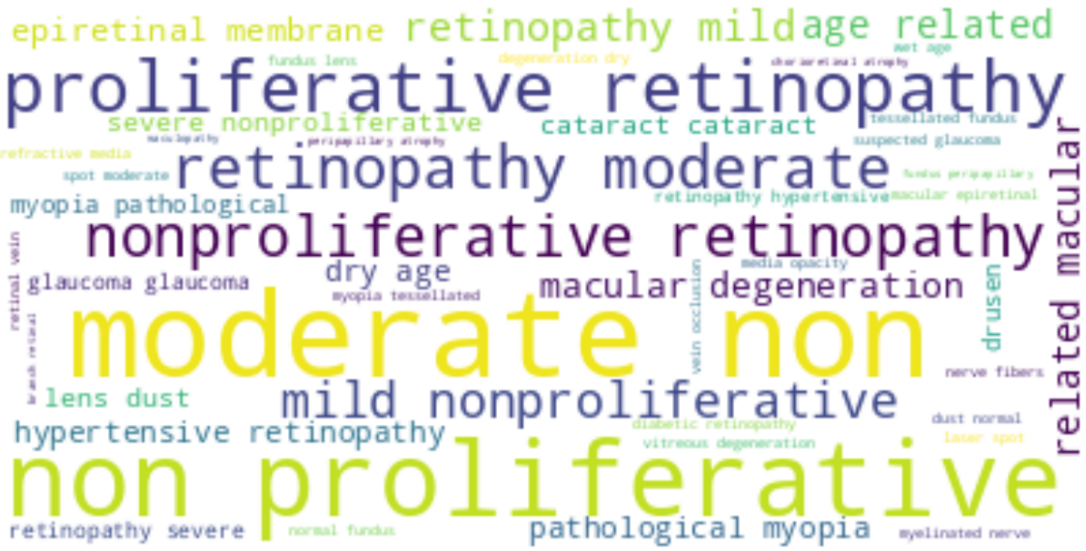

Figure 5.10: Word Cloud on Left Diagnostic Keywords

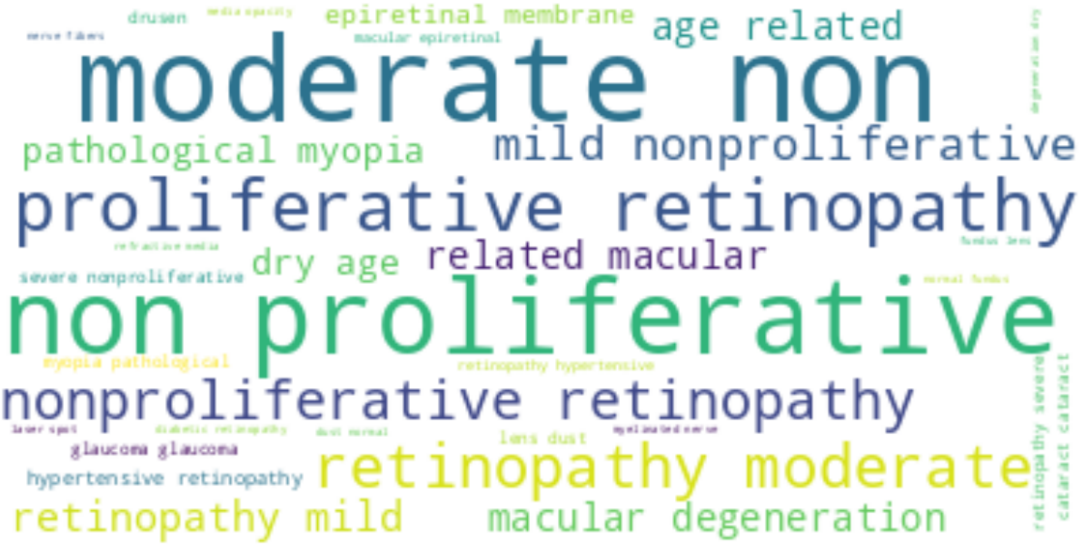

Figure 5.11: Word Cloud on Right Diagnostic Keywords

5.2 Evaluation Metrics for Classification

The model's performance was evaluated using several key metrics:

5.2.1 Definitions:

- **Accuracy Score:**

  The accuracy score measures the overall correctness of a model by calculating the ratio of correct predictions to the total number of predictions:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

- **Classification Report:**

  The classification report provides a summary of key classification metrics (precision, recall, F1-score) for each class in the model. It often includes macro and weighted averages.

  - **Precision:** The proportion of true positive predictions over the total predicted positives:

$$\text{Precision} = \frac{TP}{TP + FP}$$

  - **Recall:** The proportion of true positive predictions over the total actual positives:

$$\text{Recall} = \frac{TP}{TP + FN}$$

  - **F1-score:** The harmonic mean of precision and recall:

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

- **Cohen's Kappa Score:**

  The Cohen's Kappa score measures the agreement between two raters (or classifiers), adjusted for the agreement that would occur by chance. It is defined as:

$$\kappa = \frac{P_o - P_e}{1 - P_e}$$

  where:

  - P_o is the observed agreement.
  - P_e is the expected agreement by chance.

  The range is from -1 (complete disagreement) to 1 (complete agreement), with 0 indicating no agreement beyond chance.

- **Confusion Matrix:**

  The confusion matrix is a table that describes the performance of a classification model by comparing the predicted labels to the actual labels. It contains the following values:

- **True positives (TP):** Correctly predicted positive instances.
- **True negatives (TN):** Correctly predicted negative instances.
- **False positives (FP):** Incorrectly predicted positive instances.
- **False negatives (FN):** Incorrectly predicted negative instances.

- **F1 Score:**

  The F1 score is the harmonic mean of precision and recall:

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

- **Precision Score:**

  Precision is the proportion of true positive predictions over the total predicted positives:

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall Score:**

  Recall is the proportion of true positive predictions over the total actual positives:

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **ROC AUC Score:**

  The ROC AUC score measures the area under the receiver operating characteristic (ROC) curve. It evaluates the performance of a classifier across all classification thresholds:

$$0 \leq \text{AUC} \leq 1$$

  where 1 represents a perfect classifier and 0.5 represents a random classifier.

- **ROC Curve:**

  The ROC curve plots the true positive rate (recall) against the false positive rate (1 - specificity) at various thresholds. The axes are:

  - X-axis: False Positive Rate (FPR)
  - Y-axis: True Positive Rate (TPR)

- **Sensitivity:**

  Sensitivity (or recall) is the proportion of actual positives that were correctly identified:

$$\text{Sensitivity} = \frac{TP}{TP + FN}$$

- **Specificity:**

  Specificity is the proportion of actual negatives that were correctly identified:

$$\text{Specificity} = \frac{TN}{TN + FP}$$

5.2.2 Computed Evaluation Metrics:

- **Accuracy (Weighted):** 0.5992
- **F1 Score (Weighted):** 0.5195
- **Precision (Weighted):** 0.5309
- **Recall (Weighted):** 0.5261
- **Cohen Kappa Score:** 0.3559
- **ROC AUC Score (Weighted):** 0.7671
- **Classification Report:**

```
Classification Report:
              precision    recall  f1-score   support

           A       0.33      0.64      0.43        28
           C       0.72      0.87      0.79        30
           D       0.52      0.40      0.45       170
           G       0.31      0.67      0.42        27
           H       0.18      0.23      0.20        13
           M       0.73      0.89      0.80        27
           N       0.62      0.61      0.62       298
           O       0.28      0.17      0.21        78
```

accuracy			0.53	671
macro avg	0.46	0.56	0.49	671
weighted avg	0.53	0.53	0.52	671

- **Confusion Matrix:**

```
Confusion Matrix:
[[ 18   0   1   0   1   1   5   2]
 [  0  26   0   1   0   1   1   1]
 [  9   2  68   4   5   2  67  13]
 [  2   0   0  18   0   0   7   0]
 [  0   0   3   1   3   0   5   1]
 [  0   0   1   2   0  24   0   0]
 [ 13   4  42  29   8   3 183  16]
 [ 13   4  17   3   0   2  26  13]]
```

- **Accuracy, Sensitivity and Specificity for Each Class:**

Class	Accuracy	Sensitivity	Specificity
A:	0.9151	0.6429	0.9269
C:	0.9791	0.8667	0.9844
D:	0.7198	0.4471	0.8124
G:	0.9061	0.7778	0.9115
H:	0.9463	0.3077	0.9590
M:	0.9821	0.8889	0.9860
N:	0.6379	0.3926	0.8338
O:	0.8286	0.2692	0.9022

5.2.3 Existing Model V/S Proposed Model

Class	Accuracy	
	Two input VGG16+SGD optimizer	Proposed
Normal	0.66	0.6379
Cataract	0.96	0.8286

Diabetic retinopathy	0.93	0.9791
AMD	0.94	0.9151
Hypertension	0.95	0.9463
Myopia	0.94	0.9821
Glaucoma	0.67	0.9061
Other abnormalities	0.73	0.7198

Class	Sensitivity	
	Two input VGG16+SGD optimizer	Proposed
Normal	0.77	0.3926
Cataract	0	0.2692
Diabetic retinopathy	0.05	0.8667
AMD	0.06	0.6429
Hypertension	0	0.3077
Myopia	0.11	0.8889
Glaucoma	0.4	0.7778
Other abnormalities	0.74	0.4471

Class	Specificity	
	Two input VGG16+SGD optimizer	Proposed
Normal	0.21	0.8338
Cataract	1	0.9022
Diabetic retinopathy	0.94	0.9844
AMD	0.93	0.9269
Hypertension	0.99	0.959
Myopia	0.94	0.986
Glaucoma	0.6	0.9115
Other abnormalities	0.32	0.8124

Inferences:

- Proposed model is performing better than previous model on per class metrics.
- The overall accuracy, precision, F1-Scores and other metrics are average.

Chapter 6

Conclusion

In conclusion, this project successfully demonstrated the application of the Swin Transformer for multi-class classification on the ODIR-5K dataset. By leveraging the Swin Transformer's hierarchical feature extraction capabilities and its ability to capture both local and global patterns, the model was able to efficiently classify retinal images into multiple categories.
The Swin Transformer model has proven to be an effective tool for multi-class classification of retinal images. The model achieved high accuracy and demonstrated the ability to identify different types of eye diseases from medical images. Future work could focus on fine-tuning hyperparameters and addressing class imbalances to further improve the model's performance.

Key achievements of the project include:

- **Preprocessing and Data Augmentation:** Effective preprocessing and augmentation techniques ensured that the model received well-processed and diverse data, which contributed to better generalization and improved classification performance.
- **Swin Transformer Model:** The Swin Transformer, with its attention mechanism, allowed the model to focus on relevant regions within the retinal images, thereby improving the accuracy of classifications. Compared to traditional CNN-based models, the Swin Transformer exhibited superior performance, showcasing its

advantages in handling complex image data.

- **Model Performance:** The evaluation metrics, including accuracy, precision, recall, and F1-score, were above average, but the accuracy, specificity and sensitivity per class was significantly high indicating that the model performed well across different classes within the dataset. The model was able to learn the patterns in the retinal images, leading to accurate predictions.
- **Scalability and Adaptability:** The Swin Transformer demonstrated its potential for scalability and adaptability, showing that it could be applied to other medical image classification tasks with minimal modification.

The limitations of the project work, is the class imbalance and incompleteness of the openly available ODIR dataset. The missing correlation between patient's metadata and retinal images. The computational power to run swin tiny, small, base and large models for long epochs.
In future work, the model can be further refined by exploring different hyperparameter tuning strategies, incorporating additional data sources, or experimenting with hybrid architectures that combine the strengths of convolutional networks and transformers. This project highlights the potential of transformer-based models like Swin Transformer in advancing the field of medical image analysis, especially in tasks requiring multi-class classification.

REFERENCES

[1] *ODIR-5K dataset from Kaggle - https://www.kaggle.com/datasets/andrewmvd/ocular-disease-recognition-odir5k*

[2] *ODIR 2019 Grand Challenge - https://odir2019.grand-challenge.org/dataset/*

[3] *Swin Transformer: Hierarchical Vision Transformer using Shifted Windows - https://arxiv.org/abs/2103.14030*

[4] *An official implementation of Swin Transformer Implementation - https://github.com/microsoft/Swin-Transformer*

[5] *Google Colab - https://colab.research.google.com/*

[6] *Python Documentation - https://docs.python.org/3/index.html*

[7] *NumPy Documentation - https://numpy.org/doc/*

[8] *Pandas Documentation - https://pandas.pydata.org/docs/*

[9] *Matplotlib Documentation - https://matplotlib.org/stable/index.html*

[10] *Seaborn Documentation - https://seaborn.pydata.org/*

[11] *Scikit-Learn Documentation - https://scikit-learn.org/stable/user_guide.html*

[12] *PyTorch Documentation - https://pytorch.org/docs/stable/index.html*

[13] *TorchVision Documentation - https://pypi.org/project/torchvision/*

[14] *Pillow Documentation - https://pypi.org/project/pillow/*

[15] *NLTK Corpus Documentation - https://www.nltk.org/howto/corpus.html*

[16] *NLTK Stemming Documentation - https://www.nltk.org/howto/stem.html*

[17] *NLTK Wordcloud Documentation - https://pypi.org/project/wordcloud/*

[18] *Transfomers Documentation - https://pypi.org/project/transformers/*

[19] *Python Imaging Library Documentation - https://pypi.org/project/pillow/*

[20] *timm Documentation - https://pypi.org/project/timm/*

[21] *Multi-class multi-label ophthalmological disease detection using transfer learning based convolutional neural network by Neha Gour , Pritee Khanna - https://www.sciencedirect.com/science/article/abs/pii/S1746809420304432*

[22] *Multi-Class Eye Disease Classification Using Deep Learning by Mariam Ayman Mohamed; Mustafa Adel Zakaria; Eman Hamdi; Rawan elSayed Tawfek; Taha Mohamed Taha; Yasmine M. Afify - https://ieeexplore.ieee.org/abstract/document/10391159*

[23] *BFENet: A two-stream interaction CNN method for multi-label ophthalmic diseases classification with bilateral fundus images by Xingyuan Ou, Li Gao, Xiongwen Quan, Han Zhang, Jinglong Yang, Wei Li - https://www.sciencedirect.com/science/article/abs/pii/S0169260722001250*

[24] *Multi-Label Classification of Fundus Images With Graph Convolutional Network and Self-Supervised Learning by Jinke Lin; Qingling Cai; Manying Lin - https://ieeexplore.ieee.org/document/9349192*

[25] *Dense Correlation Network for Automated Multi-Label Ocular Disease Detection with Paired Color Fundus Photographs by Cheng Li; Jin Ye; Junjun He; Shanshan Wang; Yu Qiao; Lixu Gu - https://ieeexplore.ieee.org/abstract/document/9098340*

[26] *Multi-Label Classification of Fundus Images With EfficientNet by Jing Wang; Liu Yang; Zhanqiang Huo; Weifeng He; Junwei Luo - https://ieeexplore.ieee.org/document/9268081*

[27] *Multi-label classification of fundus images with graph convolutional network and LightGBM by Kai Sun, Mengjia He, Yao Xu, Qinying Wu, Zichun He, Wang Li, Hongying Liu, Xitian Pi - https://www.sciencedirect.com/science/article/abs/pii/S0010482522006527*

[28] *A Benchmark of Ocular Disease Intelligent Recognition: One Shot for Multi-disease Detection by Ning Li, Tao Li, Chunyu Hu, Kai Wang & Hong Kang - https://link.springer.com/chapter/10.1007/978-3-030-71058-3_11*

[29] *Source and Camera Independent Ophthalmic Disease Recognition from Fundus Image Using Neural Network by Md. Tariqul Islam; Sheikh Asif Imran; Asiful Arefeen; Mahmudul Hasan; Celia Shahnaz - https://ieeexplore.ieee.org/document/9065162*