Software Engineering Department

Braude College of Engineering

EyeNet AI-Driven Early Eye Conditions Detection using Video Analysis

Project Code: 25-1-D-3

Capstone Project Phase B – 61998

July 2025

Gal Bitton

Ron Bendel

Supervisors:

Mrs. Elena Kramer

Dr. Dan Lemberg

Git repository link:

GalBitton/EyeNet-Eye-Disease-Detection

Table of Contents

Abstract

The sense of sight is the sense that we as humans use in every activity of our daily lives. The eyes are a sensitive organ and are exposed to infections and bacteria, and as a result, there are several external eye diseases, including cataracts, conjunctivitis, and styes. These diseases significantly affect daily functioning and vision if not detected early. Despite technological advances, accessible and effective diagnostic tools remain limited, and diagnosis today is still time-consuming, can be an expensive process, and sometimes even requires specialized medical equipment.

*In this project, we developed **'EyeNet'**, a **deep learning-based system** delivered as an **end-to-end user-friendly** application designed for identifying and classifying external eye conditions — specifically cataracts, conjunctivitis, and styes. The core model is based on **DenseNet121**, enhanced with attention mechanism, trained of a dataset of over 13K images per class with data augmentation to increase robustness to lighting changes and image noise. Our model achieved an accuracy of **98%**, with precision and recall scores of **97.95% and 97.92%**, respectively, demonstrating strong performance on the test set.*

*The system integrates a **Robyn Framework backend** and a **React frontend**, offering an intuitive interface for users to upload eye images or videos and receive reliable diagnostic feedback.*
'EyeNet' combines a Robyn backend with a React-based frontend, providing a user-friendly and accessible interface for users to upload videos of their eyes and receive accurate and reliable diagnostic results. This project demonstrates the feasibility of combining advanced deep learning models with accessible and affordable web-based technology to improve the detection of external eye diseases and reduce the diagnostic burden on healthcare systems.

Throughout the development process, we faced significant challenges in obtaining a high-quality, representative dataset, which underscored the critical role of data in medical AI systems. Despite these difficulties, our results show that with a well-curated dataset, it is possible to achieve remarkable *performance. We are optimistic that further improvements in data collection and model refinement will bring us even closer to real-world deployment and greater impact on public health.*

1. Introduction

Worldwide, at least 2.2 billion people suffer from near or far vision impairment. In at least one billion of these, vision impairment could be prevented or is still untreated [1]. Cataracts, for example, are one of the leading causes of visual impairment and blindness worldwide [2].

With the global prevalence of eye diseases such as cataracts, retinitis pigmentosa, and conjunctivitis, there is an urgent need for accessible diagnostic tools. If left untreated, these diseases can lead to significant discomfort and vision loss. Cataracts alone account for more than 51% of global blindness, making them a leading cause of preventable vision loss [3]. Today, traditional diagnostic methods include in-person consultations with a specialist and the use of specialized eye equipment that is often inaccessible to people in peripheral areas and is often very expensive [4].
In this project, we addressed these challenges by developing a cost-effective, scalable and user-friendly solution for early detection of eye conditions enabling users to identify and treat diseases in time, thereby facilitating and supporting the medical diagnostic process. Our system, **EyeNet**, is based on a deep personal story: First, we will share that one of the team members himself suffers from recurrent eye problems, and that our grandparents were diagnosed with cataracts at an early stage, and since we and our family live in a peripheral area, the diagnosis and treatment process was limited and expensive. The long queues for specialists prompted us to design an innovative solution. EyeNet leverages innovative artificial intelligence techniques to bridge the gap in accessibility to eye health, allowing people to proactively monitor their eye health using standard mobile cameras.

Deep learning has revolutionized the field of medical imaging by introducing powerful models capable of analyzing complex datasets [5]. Convolutional Neural Networks are particularly effective at extracting hierarchical features from visual data, making them a natural choice for us to diagnose eye conditions. Enhanced with attentional mechanisms, our CNN models should learn to focus on critical image regions, improving their ability to detect subtle symptoms related to cataracts, styes, and conjunctivitis [6].

EyeNet processes video content by identifying facial regions, isolating eyes, and analysing them frame by frame. With this approach, we essentially ensure that diagnostic accuracy is based on a sophisticated average of all frames in the video, allowing us to be accurate even under varying video conditions. With the help of CNNs and a user-friendly interface, we developed and evaluated EyeNet — a complete system that demonstrates the potential to prevent suffering from eye diseases and enable more effective and timely treatment.

2 Background and Related Work

In this chapter, we are going to discuss the global impact of external eye conditions, their symptoms, and traditional diagnostic methods. In addition, we will explore advancements in video analysis, object detection, and CNN technologies, laying the groundwork for our implemented AI-driven diagnostic solution.

2.1 Eye Conditions, Symptoms and their global impact

For us to effectively address the problem and develop a robust, stable, and effective solution, it is essential to first understand the various eye conditions, their global impact on public health, and the current diagnostic approaches used to identify them.

2.1.1 Eye Conditions and their global impact

As previously noted, external eye diseases, such as cataracts, styes, and conjunctivitis, are significant contributors to global visual impairment and discomfort. According to the World Health Organization (WHO), cataracts alone account for 51% of blindness worldwide, particularly in low- and middle-income countries where access to eye care remains a challenge [3]. Styes, although not a cause of blindness, are localized infections of the eyelid that cause pain and swelling, greatly impact daily activities, and create a strong sense of discomfort. Styes are also contagious. Similarly, conjunctivitis, commonly known as pink eye, causes irritation, redness, and discharge, which can spread rapidly through communities if left untreated [7].

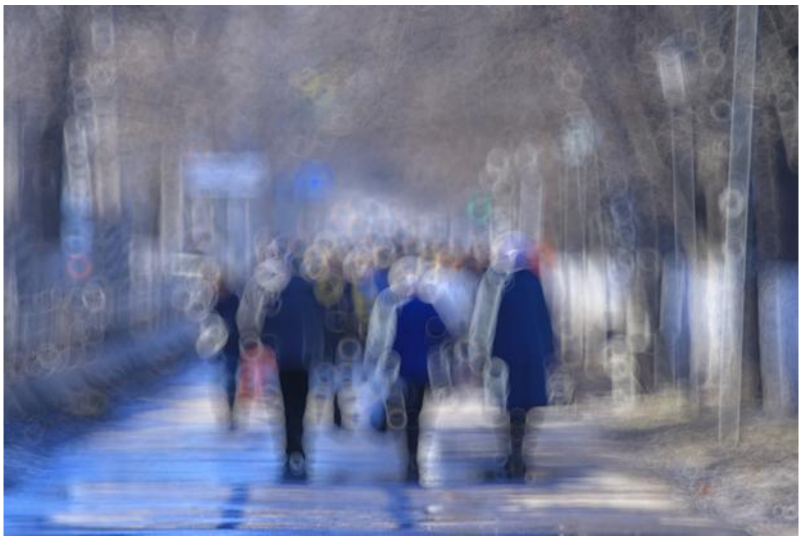


Figure 1. Vision with Untreated Cataracts
This figure illustrates the blurred and distorted vision caused by untreated cataracts, highlighting the impact of lens clouding on visual clarity and the importance of timely treatment to prevent vision loss.

2.1.2 Symptoms, Causes and Traditional Diagnostics

Each of these conditions presents unique symptoms and Causes:

Cataracts

A cataract is a clouding of the eye's natural lens, leading to blurred or impaired vision.
Primarily associated with aging, cataracts can also result from factors such as diabetes, prolonged exposure to ultraviolet radiation, smoking, and certain medications like corticosteroids.
Cataracts can cause cloudiness and blurriness in vision that may gradually spread, faded or yellow-tinged colors, progressive difficulty seeing at night, increased sensitivity to light, halos around lights, double vision or ghost-like images, and a gradual decline in visual clarity, which can lead to blindness if left untreated. [8]

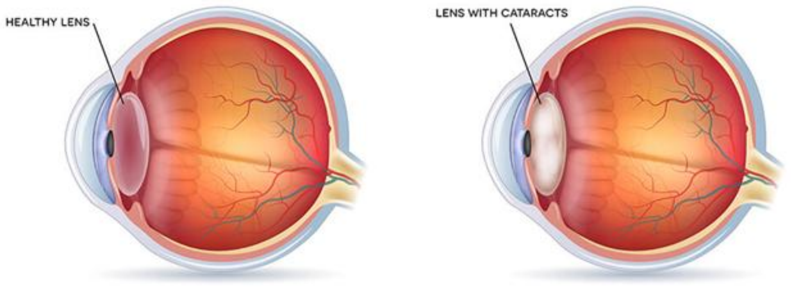


Figure 2 Example of eye with cataract

Styes

A stye is a painful, red bump near the edge of the eyelid, caused by an acute bacterial infection of an eyelash follicle or oil gland.

Styes are typically caused by Staphylococcus bacteria infecting the oil glands or hair follicles of the eyelid. Poor eyelid hygiene, touching the eyes with unclean hands, and using contaminated eye makeup can increase the risk. [9]

Symptoms include a red, swollen, and painful lump on the eyelid, tenderness, a feeling of something in the eye, and sometimes tearing or crusting around the eyelid.

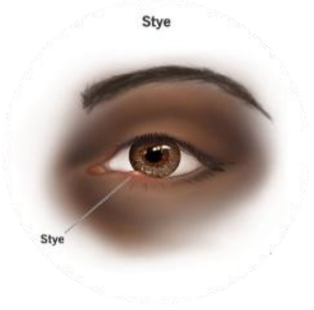


Figure 3 Example of eye with stye

Conjunctivitis

Conjunctivitis, commonly known as pink eye, is the inflammation or infection of the conjunctiva—the transparent membrane lining the eyelid and covering the white part of the eyeball—resulting in redness and swelling.

It can be caused by viruses, bacteria, allergens (such as pollen or pet dander), irritants like smoke or chlorine in swimming pools, or contact lens use.

Symptoms include redness in the white of the eye or inner eyelid, increased tearing, thick yellow discharge that crusts over the eyelashes (especially after sleep) in bacterial conjunctivitis, itchy eyes in allergic conjunctivitis, and a burning sensation. [10]

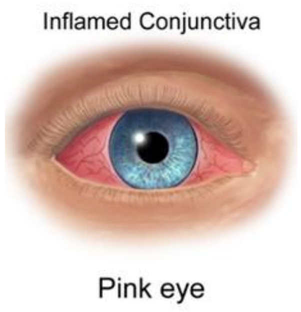


Figure 4 Example of eye with conjunctivitis

Current Diagnostics

Current diagnostic methods include **external examinations** using slit-lamp bio-microscopy and direct observation of symptoms. For cataracts, specialists rely on techniques such as visual acuity tests and ophthalmoscopy to assess the degree of lens opacity. Conjunctivitis is often diagnosed clinically based on the patient's symptoms and history, while sty**es** are visible to the naked eye. However, traditional diagnostic methods require a doctor's appointment and the presence of specialized equipment, which limits their reach in peripheral areas [11]. Internal eye conditions such as diabetic retinopathy and glaucoma are benefiting from advances in fundus photography and retinal imaging.

Unfortunately, external eye conditions remain underserved in terms of diagnostic technological tools, especially in non-clinical settings. In this project, we addressed this gap by developing an innovative diagnostic system that is portable, cost-effective, and easy to deploy.

2.1.3 The Importance of Early Detection

Early detection of external eye conditions is essential to prevent long-term complications. For example, untreated cataracts can progress to complete blindness as seen in Figure 1, while late diagnosis of conjunctivitis can lead to serious infections and corneal scarring. Early detection allows for timely medical intervention, reducing the burden on healthcare systems and improving the quality of life of patients [3].

Artificial intelligence-driven mobile solutions that use videos or photos taken with standard mobile devices can make eye health diagnosis efficient and easy to use, allowing people to monitor their eye health without needing immediate access to specialists and check their eye health anytime, anywhere.

2.2 Video Overview – Video & Image Processing

A central part of the solution we developed involves analysis and processing performed on video. In this section, we review the structure of video, video analysis techniques relevant for accurate diagnosis, and the challenges we addressed in implementing video-based diagnostics.

2.2.1 Video Architecture and Structure

Videos, unlike static images, consist of continuous sequences of frames that capture temporal and spatial information. A video is essentially a structured collection of still images (frames) displayed in rapid succession, typically at 24 to 60 frames per second (fps), creating the illusion of motion. Each video frame is a static snapshot, and collectively, they contain a wealth of visual data that can be analyzed for diagnostic purposes [12].

The architecture of video data includes several critical components:

Frames: A video is composed of individual images called frames. When these frames are displayed in rapid succession, they create the illusion of motion. Each frame captures a moment in time, and collectively, they form the visual content of the video.

Frame Rate: This refers to the number of frames displayed per second (fps). Common frame rates include 24 fps (standard for films), 30 fps (common for television), and 60 fps (used for high-definition video and sports). A higher frame rate results in smoother motion representation, enhancing the viewing experience.

Resolution: Resolution refers to the number of pixels in each frame of a video, determining its clarity and detail. For instance, a 1080p resolution means the frame has 1,920 pixels in width and 1,080 pixels in height, totalling approximately 2 million pixels. Higher resolutions, like 4K (3,840x2,160 pixels), offer even more detail but require more storage space and processing power.

Temporal Continuity: This aspect involves the motion patterns and changes that occur across sequential frames. Temporal continuity ensures that movements appear fluid and natural, which is crucial for maintaining the realism of the video.

Spatial Information: Spatial information refers to the details captured within individual frames, including colors, shapes, and regions of interest. This information is vital for tasks such as object recognition and scene understanding in image processing applications.

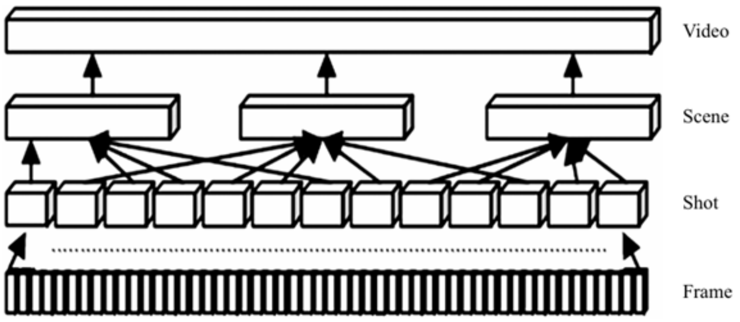


Figure 5 Schematic representation of a video structure

2.2.2 Analyzing Videos for Diagnostic Precision

Analyzing videos for diagnostics requires extracting individual frames to allow static image processing while retaining temporal relationships. This process, called **frame extraction**, ensures that videos are broken into manageable components for precise analysis. By combining spatial analysis (frame-by-frame) with temporal consistency, systems can detect patterns that remain unnoticed in single images [13].

By treating videos as a series of frames, systems like EyeNet can focus on static image processing techniques while leveraging temporal consistency for robust diagnosis. This hybrid approach ensures that challenges such as motion blur and lighting inconsistencies can be effectively mitigated. Moreover, this method allows the integration of advanced machine learning models that can analyse spatial and temporal data simultaneously. Such systems are particularly valuable in medical imaging, where early detection of abnormalities can significantly improve patient outcomes. Frame-based analysis also offers scalability, as it can process large amounts of video data efficiently. Ultimately, a combination of these techniques improved the accuracy and reliability of video-based diagnostic tools.

2.2.3 Challenges in Video-Based Diagnostics

While video-based diagnostics offer promising opportunities for automated analysis, they present several technical challenges that must be addressed to ensure reliability and accuracy [14]:

Variation in lighting: Variations in lighting between frames can blur visual features, especially in real-world environments.

Motion blur: User or camera movement can introduce blur, resulting in the loss of critical image details.

Irrelevant frames: Videos often contain unnecessary frames, such as frames with incomplete facial areas or closed eyes.

Camera angles: changes in head tilt and distance from the camera affect the consistency of extracted visual features.

To overcome these challenges, video data is usually split into frames for analysis. Pre-processing techniques such as frame stabilization, noise reduction and brightness normalization are applied to improve the quality of the frames and ensure easier and more efficient image processing for a learning machine.

Frame-based analysis ensures that critical information is stored and processed efficiently. In EyeNet, this step laid the foundation for accurately locating face and eye regions, ensuring that only high-quality and relevant frames were passed to the diagnostic model to produce an accurate and reliable diagnosis.

Challenge	Description	Solution/Technique
Lighting variations	Uneven illumination affects visibility of eye features	Brightness normalization

Motion blur	Blurred frames due to head or camera movement	Frame stabilization and sharpness enhancement
Dynamic camera angles	Changes in orientation or positioning of the face	Alignment through face and eye detection models
Occlusions and irrelevant frames	Frames where the eye region is obstructed	Frame filtering using bounding box validation
Noise in video frames	Pixel noise reduces clarity of subtle eye features	Noise reduction filters

Table 1 Challenges in video-based diagnostics and the technique to solve them

2.3 Object Detection in General

Object recognition is a basic task in computer vision that identifies and locates objects within images or videos. Unlike basic image classification, where the entire image is labelled in one class, object recognition recognizes multiple objects and knows how to differentiate between them and also identifies their location using bounding boxes. This technology serves as the basis for more advanced applications, such as facial recognition, autonomous driving and medical diagnosis.

In medical diagnosis, object detection enables the localization of anatomical structures, such as lesions, tumors or specific areas such as the eyes. By narrowing the focus to areas of interest, object detection improves the accuracy and efficiency of diagnostic tools. For example, in ophthalmology, face recognition and isolation of the eye region is critical for evaluating external eye conditions such as cataracts or conjunctivitis [15,16]. The ability to identify and isolate specific regions, such as the face and eyes, within video frames ensures that diagnostic systems can process only the most relevant information. This focus minimizes computational complexity and improves the accuracy of subsequent analyses.

With the help of transferring the desired information, you get more accurate and reliable results.

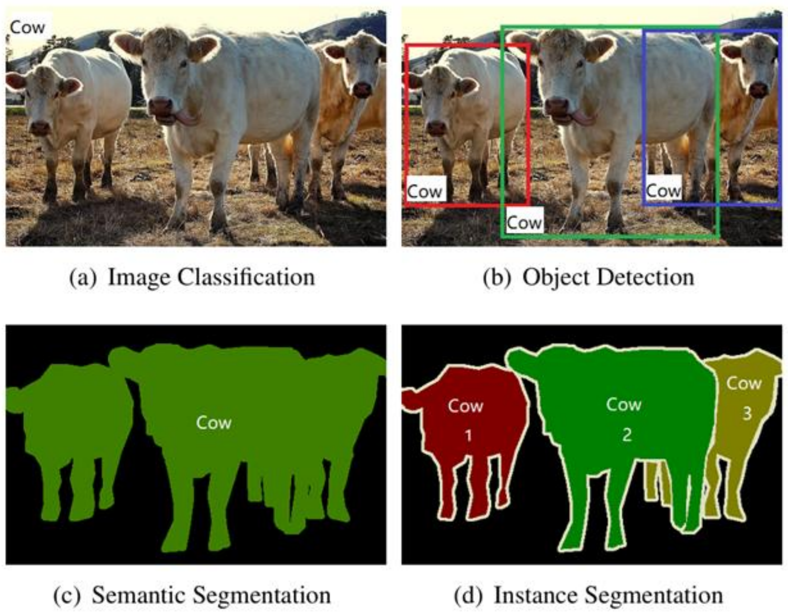


Figure 6 Comparison of different visual recognition tasks in computer vision. [16]

2.4 Existing Solutions for Face and Eyes Region Detection

Face and eye detection techniques have evolved significantly, offering powerful solutions for isolating landmarks and eye regions within video frames. These solutions are the fundamental component of medical systems that deal with external detection and eye diagnosis and enable accurate extraction of areas of interest for further analysis.

2.4.1 Classical Methods

Early methods relied on hand-crafted features and rule-based approaches:

Viola-Jones Algorithm: A pioneering real-time face detection technique using Haar-like features and cascading classifiers. Though efficient, it struggles with occlusions and variations in camera angles [17].

Haar Cascades: An OpenCV-based method optimized for lightweight face detection. While efficient, these methods lack robustness in dynamic environments and are limited by their inability to handle occlusions and varying camera angles [18].

2.4.2 Deep Learning-Based and Machine Learning-Based Solutions

Modern solutions leverage deep learning and convolutional neural networks to achieve higher accuracy and flexibility:

MTCNN (Multi-task Cascaded Convolutional Networks): Combines face detection with facial landmark localization, ensuring precise detection of the eye regions. It balances accuracy and multi-task capabilities but can be slower for large datasets or high-resolution inputs [19].

DLib Facial Landmark Detection: Utilizes a hybrid approach combining HOG (Histogram of Oriented Gradients) for feature extraction and SVM (Support Vector Machines) for classification. Detection is further refined using the *shape_predictor_68_face_landmarks.dat*, which identifies 68 facial landmarks, including the eyes. This approach effectively integrates classical methods, such as Haar cascades (*haarcascade_frontalface_default.xml* and *haarcascade_eye.xml*), with advanced landmark prediction for precise facial feature localization, even under challenging conditions [20].

YOLO (You Only Look Once): A real-time object detection model that efficiently detects faces and eyes within video frames [21].

2.4.3 Strength & Limitations

Algorithm	Strengths	Limitations
Viola-Jones	Fast, lightweight	Low accuracy in dynamic settings
MTCNN	High accuracy, multi-task learning	Slower processing on large inputs
YOLO	Real-time performance	Requires high GPU capabilities
OpenCV	Optimized for lightweight tasks	Limited robustness
DLib	Precise landmark detection	Computationally intensive

Table 2 Strengths and limitations for each algorithm/model

2.5 Existing Applications for Eye Condition Detection

In the domain of AI-driven solutions for diagnosing eye conditions, several applications have been developed to detect specific eye diseases through image analysis. We will cover some notable examples.

2.5.1 CRADLE (ComputeR-Assisted Detector of LEukocoria)

CRADLE, also known as the White Eye Detector, is a mobile application designed to detect leukocoria—a white pupil reflex often indicative of retinoblastoma or cataracts. CRADLE analyzes casual photographs using machine learning techniques to identify leukocoria, demonstrating high accuracy in early detection, often months before a clinical diagnosis. However, CRADLE is limited to detecting leukocoria and does not address other external eye diseases like styes or conjunctivitis. [22]

Figure 7 Logo of CRADLE

2.5.2 AEYE Health

AEYE Health is an AI-powered platform that uses deep learning algorithms to detect diabetic retinopathy and other retinal conditions from fundus images. Designed for use in primary care settings, it enables early diagnosis and treatment of diabetic eye diseases. While highly effective for internal eye conditions, AEYE Health does not extend to external eye diseases like cataracts, styes, or conjunctivitis. [23]

Figure 8 Logo of AEYE Health

2.5.3 MD EyeCare

MD EyeCare is a mobile-based application aimed at detecting a variety of eye conditions using deep learning algorithms. The app allows users to upload images of their eyes for analysis and provides preliminary diagnostic results for conditions like cataracts and dry eye syndrome. While it supports external eye condition detection, its focus is primarily on static image inputs, limiting its adaptability for video-based workflows. [24]

MD EyeCare

Quality Australian Ophthalmic Products

Figure 9 Logo of MD EyeCare

2.5.4 Comparison with Our Application & Model

Feature	CRADLE	AEYE Health	MD EyeCare	Our Solution (EyeNet)
Face Detection	None	Retinal imaging only	Static image detection	Automated face & eye detection
Eye Condition Detection	Leukocoria only	Diabetic retinopathy	Cataracts, Dry Eyes	Cataracts, Styes, Conjunctivitis
Input Type	Casual photos	Retinal images	Static images	Video & Image inputs
User-Friendly Interface	Yes	Yes	Yes	Yes
Cost	Free	Paid	Varies	Free
Scalability	Low	Moderate	Moderate	High

Table 3 Comparison between existing applications with our project

By analyzing the table above, it becomes evident that while existing solutions like CRADLE, AEYE Health, and MD EyeCare excel in specific areas, they lack versatility when diagnosing multiple external eye conditions using dynamic video inputs. EyeNet addresses these limitations by processing video data, automatically identifying face and eye regions using advanced object recognition models and leveraging improved CNNs for comprehensive analysis.

EyeNet's video-processing capabilities and automated pipeline provide a robust, accessible, and economical solution, ensuring accurate detection of multiple external eye conditions in real-world scenarios. Unlike other solutions, it is designed to handle dynamic inputs seamlessly, making it more adaptable and user-friendly.

2.6 Advances in Image Analysis and CNN

Our implemented solution leveraged existing image processing techniques and built on the proven strengths of CNNs in medical diagnostics to achieve high accuracy and reliability.

2.6.1 Advances in Image Analysis for Medical Diagnostics

Image analysis brought about a profound change in medical diagnosis, enabling the automatic extraction of significant insights from medical images. Historically, medical imaging has relied on manual interpretation of X-rays, MRIs, and microscopy images. Although these traditional methods have been effective, their diagnosis is time-consuming and depends on expert judgment. Modern computer vision techniques, on the other hand, have automatic image analysis, leading to improved accuracy, reproducibility and grading in diagnosis [5,25].

In ophthalmology, image analysis has been widely applied to detect internal eye diseases, such as diabetic retinopathy and glaucoma. Technologies such as fundus photography and optical coherence tomography (OCT) have enabled high-resolution imaging of the retina, helping to detect early disease progression. However, external eye diagnostics, which include conditions such as cataracts, conjunctivitis and styes remain less studied despite their prevalence and impact.

Major advances in image analysis for medical diagnosis include:

Segmentation: Decomposing an image into regions of interest to isolate anatomical structures (eg pupil, sclera, eyelid or iris).

Feature extraction: identifying disease markers such as changes in texture, color or opacity.

Classification: assigning images to categories (eg, diseased vs. healthy) based on extracted features.

These techniques provide the basis for systems such as EyeNet, which focus on detecting external eye states from standard video inputs. By combining accurate feature extraction with robust classification, modern image analysis ensures early detection of subtle abnormalities that the expert may not notice with his analysis and diagnosis.

2.6.2 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks are the cornerstone of modern medical image analysis, excelling in tasks such as feature extraction, object localization, and image classification. Unlike traditional machine learning methods, CNNs automatically learn **hierarchical features** from visual data, making them ideal for detecting both high-level patterns and subtle variations within medical images [26].

2.6.2.1 Overturning Layers

Convolutional layers apply filters (kernels) to the input image to extract spatial features such as edges, textures, and patterns. Early layers focus on simple features such as lines and corners, while deep layers recognize complex patterns such as shapes and areas.

The convolution operation preserves the spatial relationship between pixels by learning image features. Multiple filters are used in each layer to detect different features, which enriches the understanding of image finesse. The output of these layers, called feature maps, serve as input to subsequent layers, passing on the extracted spatial information.

2.6.2.2 Pooling Layers

Pooling layers perform operations that reduce the spatial dimensions of feature maps, preserving critical information while improving computational efficiency. Common pooling techniques include Maximum Pooling, which selects the maximum value within a certain area of the image whose size is determined in advance, and Average Pooling, which calculates the average. These layers contribute to reducing overfitting and improving model performance.

2.6.2.3 Fully Connected Layers

Fully connected layers connect the extracted features to the final output predictions. They map the learned patterns to specific classes, enabling tasks such as classification. For example, in medical imaging, a CNN might classify an image as "healthy eye" or "cataract" based on the features extracted by previous layers.

2.6.2.4 Activation Functions

Activation functions introduce non-linearities to the network, enabling CNNs to learn complex relationships within the data. Common activation functions include ReLU (Rectified Linear Unit), which helps address the vanishing gradient problem, and Sigmoid or Softmax for final output probabilities. These functions ensure that the network can model diverse patterns effectively.

ReLU, as one of the most widely used activation functions, introduces sparsity by setting all negative values to zero, improving computational efficiency and convergence speed.

Sigmoid activation squashes outputs between 0 and 1, making it suitable for binary classification tasks, while Softmax normalizes outputs into probabilities across multiple classes.

Without activation functions, the CNN would behave as a linear model, severely limiting its ability to model complex data patterns.

To summarize, in EyeNet the CNN framework underpinned our diagnostic model, enabling analysis of eye regions extracted from video frames. By learning features such as opacity (for cataracts), swelling (for styes), or redness (for conjunctivitis), the CNN processed each frame and made accurate predictions. We then aggregated the results from all frames to produce a final and reliable diagnosis.

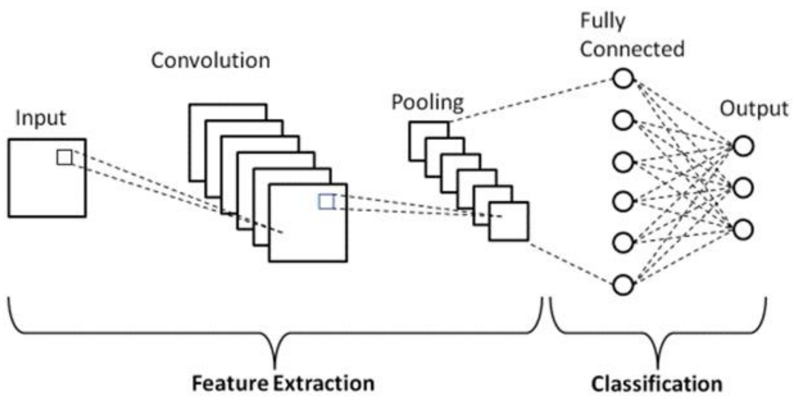


Figure 10 Basic architecture of CNN

3 Achievements

3.1 Outcomes

In our project, we developed a reliable, efficient, and high-quality system that provides accurate results for identifying eye diseases such as cataracts, styes, and conjunctivitis. We implemented advanced deep learning techniques, using DenseNet121 as the base model, enhanced with an attention mechanism to focus the learning process on the most relevant regions in each image.
The system reliably delivered diagnostic insights by processing videos captured on mobile devices, dividing them into frames, and analysing each frame independently using the EyeNet model to produce a final, aggregated diagnosis.
We combined our deep learning model with a user-friendly, publicly accessible web application, designed to be intuitive and easy to use. This application allows users to upload videos or images and receive diagnostic results quickly. Our frame-by-frame analysis approach ensured that each detected eye was classified independently, enabling detailed and comprehensive diagnoses.
This approach demonstrated robustness to challenging conditions, including varying lighting, different camera angles, and partial visibility in some video frames. In our testing, the model achieved an accuracy of approximately **97%**, with strong performance metrics in precision and recall as well, validating the effectiveness of our design.

3.2 Unique Features

Our implemented system incorporated several unique features that contributed to its effectiveness, usability, and robustness. These features addressed the key challenges of external eye diagnostics and ensured accessibility and accuracy in real-world use.

3.2.1 Attention-Enhanced DenseNet121 Architecture

We integrated spatial and channel attention layers into DenseNet121 to enhance the model's focus on disease-relevant regions while filtering out irrelevant background elements. This improved accuracy and robustness, especially under challenging conditions such as varying lighting or partial occlusion.

3.2.2 Video-Based Analysis

The system processed videos captured from mobile and computer cameras, analyzing frames individually to provide detailed insights for each detected eye. This frame-by-frame approach ensured comprehensive diagnostics for dynamic and real-world scenarios.

3.2.3 DLib for Detection

We used DLib for precise face and eye region detection, enabling robust localization even in videos with head movement or variable lighting. Its pre-trained models provided reliable detection and efficient preprocessing.

3.2.4 User-Friendly Interface

We developed a web-based interface that allowed users to upload images, record videos, view results, and interpret outputs effortlessly. The interface was designed for accessibility, presenting statistics in numerical form alongside verbal diagnostics and recommendations for ease of understanding.

3.2.5 Cross-Platform Compatibility

The application was designed to be accessible on both computer and mobile devices, supporting various browsers and video formats. This ensured widespread usability and convenience for diverse users.

3.3 Criteria for Success

High Accuracy in Detection: The system achieved a classification accuracy of approximately **97%**, surpassing the original target of 95%, and demonstrating strong diagnostic performance for external eye conditions.

Efficient Video Processing: Users recorded videos of **3–5 seconds**, and the system processed these videos in real time, providing results in just a few seconds, well within the target timeframe.

Robust Eye and Face Detection: The Dlib-based detection worked reliably even under challenging conditions, including head movements, partial occlusions, and varying lighting, ensuring accurate face and eye region localization.
User-Friendly Interface: The web-based interface was intuitive and easy to navigate, enabling users to upload videos and interpret results without technical expertise. Informal user feedback confirmed that the design was accessible and easy to use.
Cross-Platform Compatibility: The application functioned seamlessly on both computers and mobile devices, supporting popular browsers and common video formats as planned.
Transparent Reporting: The system provided clear and detailed outputs, including classification confidence scores. However, attention heatmaps were not included in the reporting in this phase.
Stable Performance: The application maintained stable operation throughout testing, with no crashes or critical errors observed under typical usage conditions.

4 Research/Engineering Process

The research and engineering phase of our project was divided into two primary semesters: the first dedicated to theoretical exploration, research, and designing the project's framework, and the second focused on the actual development and implementation of the system. This phased approach allowed us to build a solid foundation before transitioning into practical execution.

4.1 Research – Eye Condition Detection

We began our research by exploring the challenges and opportunities in developing a system to detect external eye conditions such as cataracts, styes, and conjunctivitis. This initial phase focused on gathering foundational knowledge and identifying potential solutions, driven by the following critical questions:

- What are the solutions currently on the market that solve the problem? What do they specialize in? How can they be improved?
- What datasets are available for external eye images, and what are the challenges in collecting, refining, and preparing such data for deep learning and machine learning tasks?
- Which pre-trained neural network models are most suitable for classifying external eye conditions, and what criteria should guide the selection process?
- How can existing deep learning models be enhanced to improve accuracy, robustness, and generalization for video data analysis?

- What practical considerations should be addressed when processing video data from mobile and computer cameras, including variations in resolution, lighting, and noise?

By addressing these questions, we built a strong foundationfor developing a scalable and reliable system capable of delivering accurate predictions while being user-friendly and adaptable to real-world scenarios. This research phase also helped us identify potential constraints and challenges, which shaped our subsequent approach.

4.2 Research – Dataset

One of the main key components to the success of a machine learning model lies in the quality and quantity of the data used for training. In our project, finding a large, high-quality dataset was a challenging task.

4.2.1 Data Collection Challenges

The main obstacle we faced when searching for the dataset was finding external eye images suitable for identifying conditions such as cataracts, styes and conjunctivitis. There are large amounts of datasets related to eye diseases and infections all over the internet but almost all of them often focus on internal eye images taken by specialized medical equipment. In our project, our dataset had to consist of external eye images – those taken using mobile phones or front-facing cameras.

To overcome this obstacle, we searched a wide range of websites, downloaded several different datasets that could serve our purpose, and merged them into a single dataset. The sources that provided us with the widest variety of external eye images were Kaggle and Roboflow Universe [27,28]. Despite an extensive search across the Internet, the volume of usable data was smaller than expected. The initial dataset contained an insufficient amount of external eye images, so we had to find a solution to the problem.

4.2.2 Data Augmentation

As we noted that we were able to collect a relatively small dataset, we used data augmentation techniques to artificially enlarge it and improve the generalizability of the model [29]. These techniques included:

Rotation: Generating new images by rotating the original ones to simulate different viewing angles.

Flipping: Horizontally flipping images to increase variability.

Brightness and Contrast Adjustments: Altering brightness and contrast to simulate different lighting conditions.

Scaling and Cropping: Creating variations in image size and focus to improve robustness.

After a challenging process of collecting a large and high-quality dataset, the final dataset consisted of approximately 13,000 images per class, covering four categories:

Healthy Eyes, Eyes with Cataracts, **Eyes with Styes**, **Eyes with Conjunctivitis**

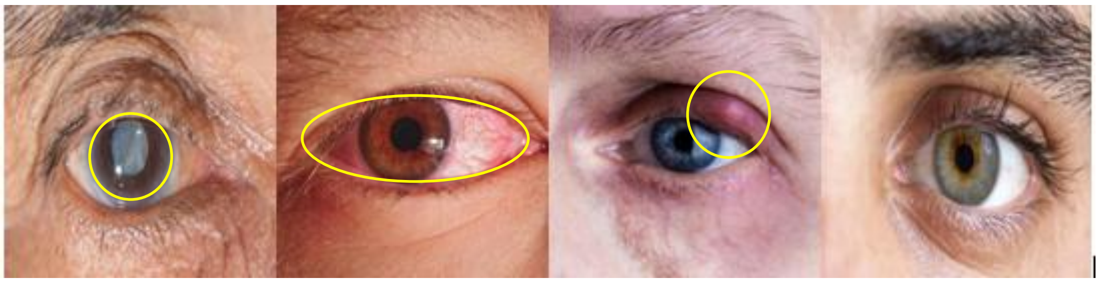


Figure 11 Visualization of sample dataset from left to right: Cataract eyes, Conjunctivitis eyes, Stye eyes and healthy eyes

Through this balanced distribution, the model learned equally from each category, reducing the risk of class bias during training.

4.2.3 Preprocessing and Data Refinement

Before training the models, we conducted several preprocessing steps:

Data Filtering: We manually checked the dataset to remove images that were irrelevant, unclear, or incorrectly labelled. For example, blurry images or images that did not prominently show the eye were excluded.

Cropping: Some images were manually cropped to focus on the eye area, ensuring consistency in input dimensions. This step was critical to reduce noise and improve the model's focus on relevant features.

An important point that we had to pay attention to in our project was the main problem that arises from images taken with mobile phones such as lighting variations, image noise, and obstructions. Using image processing techniques, we improved the quality of each image, thereby reducing factors that could confuse the model during the training process.

4.3 Research – Base Model Selection

Training deep learning models requires high-quality hardware and sufficient computing power.

In our project, we ran the experiments on our personal computers (AMD Ryzen 9 7950X, 64GB RAM, Nvidia GeForce RTX 4080 Super), these computers have limited computational power and although we improved the size of our dataset, it was still relatively small for training deep learning models from scratch. Therefore, we tested our objectives using pre-trained neural networks, fine-tuning their weights to our dataset.

There are several models available. Keras, for example, has nine pre-trained models for computer vision tasks, TL, forecasting, feature extraction, and fine-tuning. We chose to test our dataset on four pre-trained models: VGG16, ResNet50, Inception V3 and DenseNet-121.

4.3.1 Visual Geometry Group (VGG 16)

VGG16 is a widely used convolutional neural network architecture known for its depth and uniform design. It consists of 16 layers, including 13 convolutional layers utilizing 3×3 filters, and 3 fully connected layers. The model accepts input images of size 224×224×3 and employs max-pooling layers to progressively reduce spatial dimensions. This architecture enables VGG16 to effectively capture intricate features, making it suitable for various image classification tasks. [30]

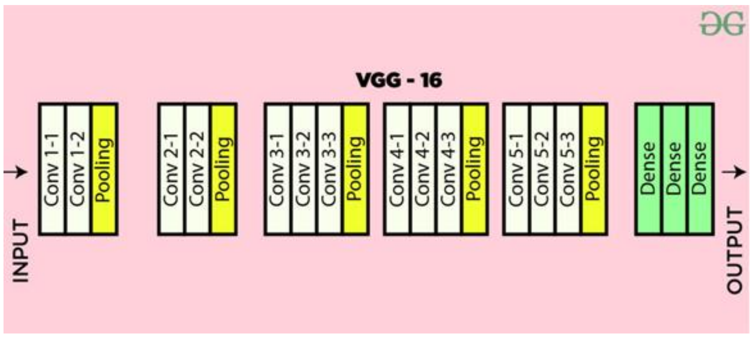


Figure 12 VGG16 Architecture

4.3.2 Residual Networks (ResNet50)

ResNet50 is a deep convolutional neural network with 50 layers that introduced the concept of residual learning to address challenges in training very deep networks. It employs residual blocks with shortcut connections, enabling efficient gradient flow and improved accuracy in deep architectures. ResNet50 is widely used for image recognition tasks due to its ability to learn complex features effectively. [31]

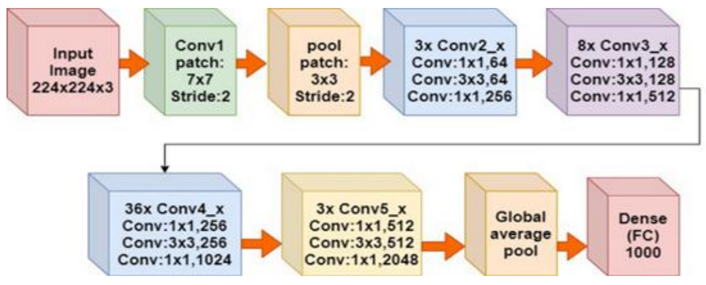

Input Image 224x224x3
Conv1 patch: 7x7 Stride:2
pool patch: 3x3 Stride:2
3x Conv2_x Conv:1x1,64 Conv:3x3,64 Conv:1x1,256
8x Conv3_x Conv:1x1,128 Conv:3x3,128 Conv:1x1,512
36x Conv4_x Conv:1x1,256 Conv:3x3,256 Conv:1x1,1024
3x Conv5_x Conv:1x1,512 Conv:3x3,512 Conv:1x1,2048
Global average pool
Dense (FC) 1000

Figure 13 ResNet50 Architecture

4.3.3 InceptionV3

Inception V3 is a deep convolutional neural network designed for efficient and accurate image classification. It enhances the original Inception model with techniques like factorized convolutions, label smoothing, and auxiliary classifiers, optimizing both computational performance and feature extraction for complex image recognition tasks. [32]

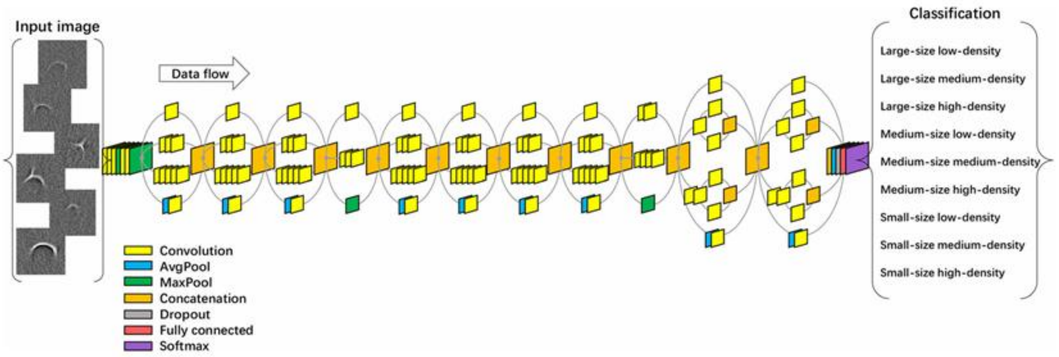


Figure 14 InceptionV3 Architecture

4.3.4 DenseNet121

DenseNet121, short for Dense Convolutional Network, is a 121-layer deep neural network that employs dense connectivity to improve feature propagation and reduce the number of trainable parameters. In this architecture, each layer is connected to every other layer, ensuring efficient reuse of features and mitigating the vanishing gradient problem. DenseNet121 consists of four dense blocks, each separated by transition layers that include convolution and pooling operations. With a total of approximately 8 million parameters. This model is particularly effective in extracting rich hierarchical features for tasks like image classification and object detection. [33]

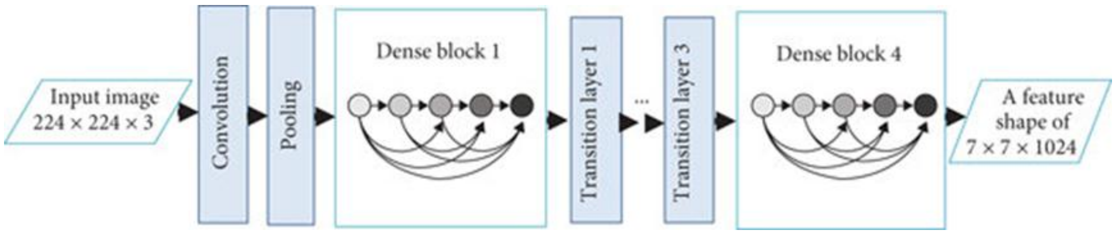


Figure 15 DenseNet121 Architecture

4.3.5 Model Performance Measures

Here are some of the measures for evaluating performance that was calculated. Using these criteria, we found the best classifier to detect eye diseases:

Accuracy

Measures the overall correctness of the model's predictions across all classes, **higher is better**.

$$Accuracy = \left(\frac{TP+TN}{TP+FN+FP+TN}\right) \times 100\%.$$

True Positive Rate (TPR)

Evaluates the model's ability to correctly identify positive cases, **higher is better**.

$$True\ Positive\ Rate\ (TPR) = \left(\frac{TP}{TP+FN}\right) \times 100\%$$

True Negative Rate (TNR)

Assesses how well the model identifies negative cases correctly, **higher is better**.

$$True\ Negative\ Rate\ (TNR) = \left(\frac{TN}{TN+FP}\right) \times 100\%$$

False Positive Rate (FPR)

Indicates the proportion of negative cases incorrectly classified as positive, **lower is better**

$$False\ Positive\ Rate\ (FPR) = \left(\frac{FP}{FP+TN}\right) \times 100\%$$

False Negative Rate (FNR)

Represents the proportion of positive cases missed by the model, **lower is better**.

$$False\ Negative\ Rate\ (FNR) = \left(\frac{FN}{FN+TP}\right) \times 100\%$$

Precision

Reflects the model's ability to correctly predict positive cases without false alarms, **higher is better**.

$$Precision = \left(\frac{TP}{TP+FP}\right) \times 100\%$$

F1 Score

Balances Precision and Recall to provide a single measure of model performance, **higher is better**.

$$F1\ Score = \left(2 \times \frac{Precision \times Recall}{Precision + Recall}\right) \times 100\%$$

4.3.6 Training Results

The dataset we used for training comprised 13,024 images **per class,** diverged into 9,114 training images, 1,955 validation images and 1,955 test images for each class (Healthy, Cataract, Conjunctivitis, Stye). The experiments were conducted on our personal computer powered by an AMD Ryzen 9 7950X CPU with 64 GB of memory, and Nvidia GeForce RTX 4080 Super GPU. All input images for the VGG-16, ResNet50, Inception V3, and DenseNet121 models were scaled to 224×224 pixels.

The pre-trained weights of VGG-16, ResNet50, Inception V3, and DenseNet121, initialized with the ImageNet dataset, were fine-tuned for this experiment. The confusion matrices for the tested models, as shown in Table 1, reported the **True Positives (TP)**, **False Negatives (FN)**, **False Positives (FP)**, and **True Negatives (TN)** for each of the four classes.

Class	Model	TP (True Positive)	FN (False Negative)	FP (False Positive)	TN (True Negative)
Healthy	VGG-16	1513	442	263	5602
	ResNet50	518	1437	165	5700
	Inception V3	1733	222	222	5643
	DenseNet 121	**1808**	**147**	**125**	**5740**
Cataract	VGG-16	1641	314	389	5476
	ResNet50	1416	539	1215	4650
	Inception V3	1788	167	167	5698
	DenseNet 121	**1810**	**145**	**92**	**5773**
Conjunctivitis	VGG-16	1713	242	799	5066
	ResNet50	901	1054	945	4920
	Inception V3	1717	238	**253**	**5612**
	DenseNet 121	**1844**	**111**	347	5518
Stye	VGG-16	1306	649	196	5669
	ResNet50	1364	591	1296	4569
	Inception V3	**1722**	**233**	218	5647
	DenseNet 121	1661	294	**133**	**5732**

Table 4 Confusion Matrix for each class, testing on 4 Keras pre-trained models

As can be analyzed from the table, DenseNet121 outperforms all three other models in almost every parameter for every class.

Class	Model	Accuracy(%)	TPR(%)	FNR(%)	FPR(%)	TNR(%)	Precision(%)	F1 Score(%)
Healthy	VGG-16	90.984	77.391	22.608	4.484	95.515	85.191	81.104
	ResNet50	79.514	26.496	73.503	2.813	97.186	75.841	39.272
	Inception V3	94.322	88.644	11.355	3.785	96.215	88.644	88.644
	DenseNet 121	**96.521**	**92.480**	**7.519**	**2.131**	**97.868**	**93.533**	**93.004**
Cataract	VGG-16	91.010	83.938	16.061	6.632	93.367	80.837	82.358
	ResNet50	77.570	72.429	27.570	20.716	79.283	53.819	61.753
	Inception V3	95.729	91.457	8.542	2.847	97.152	91.457	91.457
	DenseNet 121	**96.969**	**92.583**	**7.416**	**1.568**	**98.431**	**95.162**	**93.855**
Conjunctivitis	VGG-16	86.687	87.621	12.378	13.623	86.376	68.192	76.695
	ResNet50	74.437	46.086	53.913	16.112	83.887	48.808	47.408
	Inception V3	93.721	87.826	12.173	4.313	**95.686**	**87.157**	87.490
	DenseNet 121	**94.143**	**94.322**	**5.677**	**5.91**	94.083	84.162	**88.953**
Stye	VGG-16	89.194	66.803	33.196	3.341	9.658	86.950	75.556
	ResNet50	75.867	69.770	30.230	22.097	77.902	51.278	59.111
	Inception V3	94.232	**88.081**	**11.918**	3.716	96.283	88.762	88.421
	DenseNet 121	**94.539**	84.961	15.038	**2.267**	**97.732**	**92.586**	**88.610**

Table 5 Performance evaluation matrices for all 4 Keras pre-trained models.

The performance metrics for the applied models, evaluate their ability to classify the four eye conditions (Healthy, Cataract, Conjunctivitis, Stye) using measures such as Accuracy, True Positive Rate (TPR), False Negative Rate (FNR), False Positive Rate (FPR), True Negative Rate (TNR), Precision, and F1 Score.
These metrics provide a detailed assessment of the models' strengths and weaknesses in correctly identifying positive cases, minimizing errors, and maintaining high precision across all classes. DenseNet121 demonstrated superior performance across most metrics, with the highest accuracy and F1 scores, highlighting its robustness in this application.

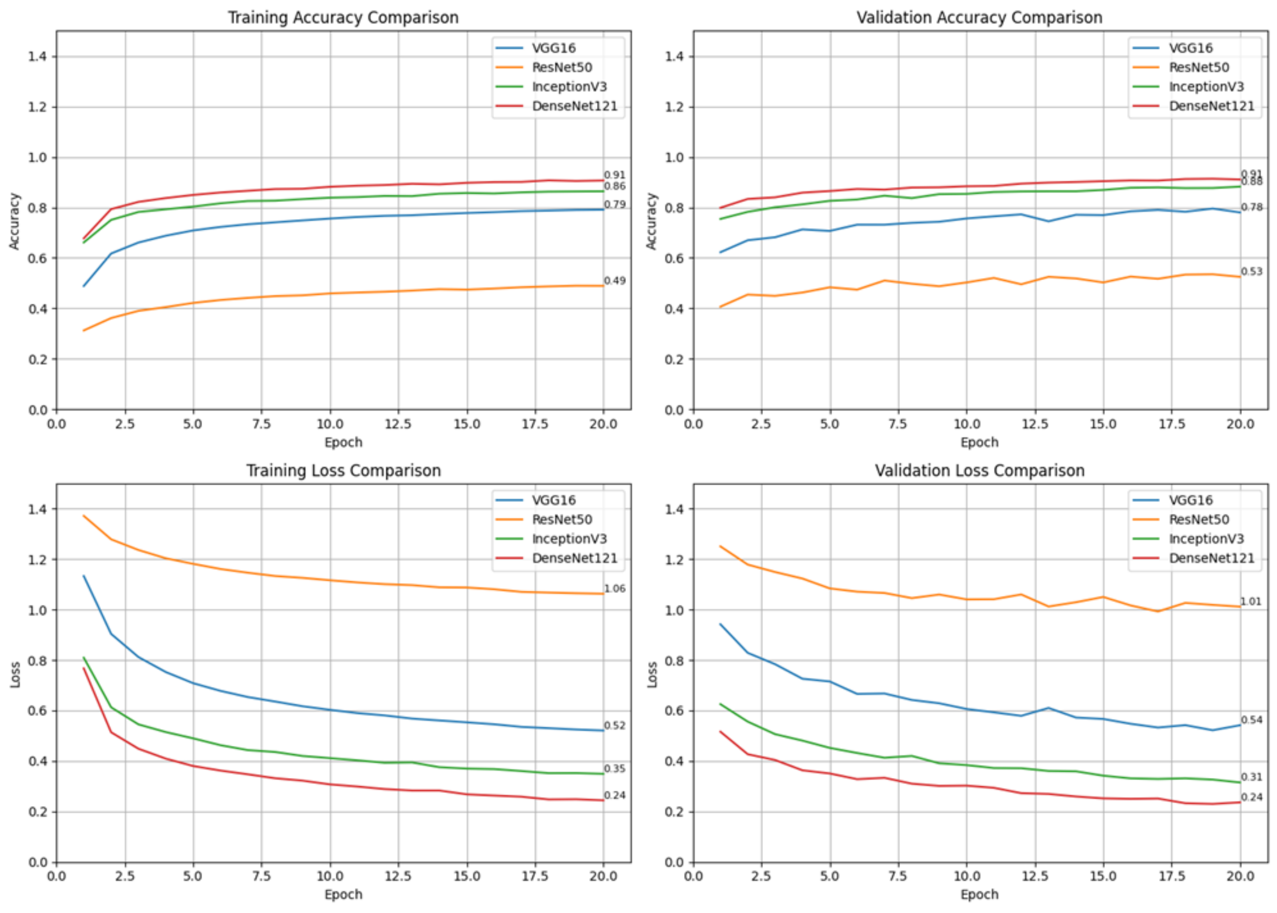


Figure 16 Visualization of the pre-trained models training and validation process

DenseNet121 demonstrated the best performance, achieving the highest training and validation accuracy (0.91 and 0.88, respectively) and the lowest loss values (0.24 for both). InceptionV3 followed with a strong validation accuracy of 0.86. VGG16 performed moderately, reaching a validation accuracy of 0.79, while ResNet50 showed the lowest performance, with a validation accuracy of 0.53 and higher loss throughout training. These results highlight the efficiency and reliability of DenseNet121 for our image classification task.

4.4 Research & Engineering – Enhance Selected Model

After testing several pre-trained models (VGG16, ResNet50, InceptionV3, and DenseNet121), we identified DenseNet121 as the most suitable architecture for our solution due to its superior performance in classifying external eye images. These evaluations provided valuable insight into the type of Convolutional Neural Network best suited for this task. Building on this foundation, we **enhanced DenseNet121 by integrating advanced attention mechanisms**, including Spatial Attention and Channel Attention. These mechanisms improved the model's ability to focus on the most relevant features within the input data, refining its classification accuracy and robustness.

4.4.1 Attention Mechanism Overview

We integrated attention mechanisms into the DenseNet121 architecture to improve its performance in classifying external eye images by enabling the model to focus dynamically on the most relevant features.

4.4.1.1 Attention Layers

The attention layers allowed the neural network to prioritize specific regions or features within an input image, improving feature localization and enabling the model to concentrate on critical areas indicative of eye conditions. [34]

4.4.1.2 Spatial Attention

This mechanism emphasizes **'where'** important features are located within the image. By generating an attention map that highlights crucial regions, the model focused on areas around the eyes that are most indicative of specific conditions. [35]

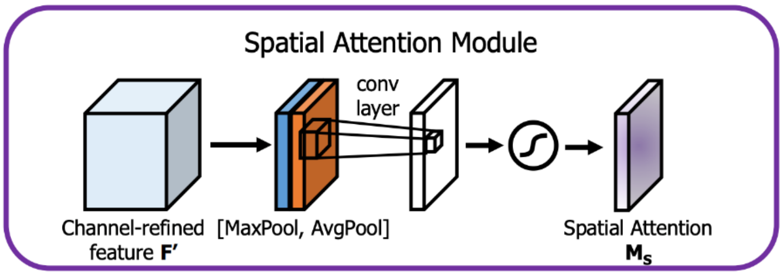


Figure 17 Architecture of Spatial attention layer

4.4.1.3 Channel Attention

This mechanism is focusing on **'what'** features are important, channel attention assigns different weights to various feature channels, each corresponding to specific visual attributes, the model prioritized channels capturing disease-relevant information, such as redness or swelling, thereby enhancing feature representation. [36]

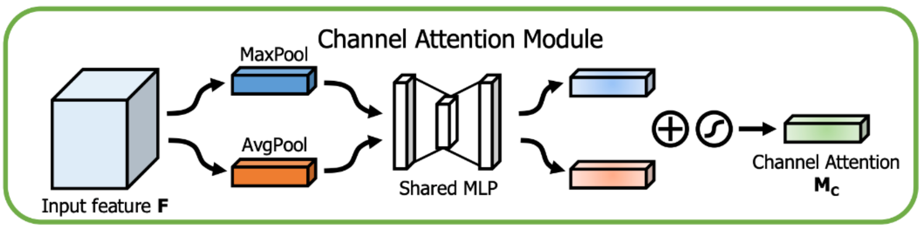


Figure 18 Architecture of Channel attention layer

4.4.2 Outcome Improvements

By integrating spatial attention, and channel attention mechanisms, we observed several improvements:

Improved Generalization: The model generalized better to varying conditions, such as differences in lighting, camera quality, or noise, by dynamically focusing on disease-relevant features.

Enhanced Learning Efficiency: The attention-enhanced DenseNet121 converged in fewer epochs and achieved higher accuracy compared to the baseline DenseNet121.

Visual demonstration: The attention mechanisms allowed visual interpretability (e.g., Grad-CAM visualizations), highlighting regions the model focused on for each prediction.

The integration of Spatial Attention and Channel Attention layers refined DenseNet121's feature extraction, improving its ability to focus on regions relevant to each disease: cataracts (pupil and iris), conjunctivitis (sclera), and styes (eyelid). These enhancements made the model more robust to noise, more efficient during training, and more interpretable, demonstrating its suitability for external eye disease classification.

4.5 Methodology and Development Process

For the development process, we followed the Agile methodology, which provided flexibility and adaptability as the project progressed. By working in iterative stages, we were able to refine our implementation continuously while addressing challenges as they arose. The development process included the following key phases:

Dataset Gathering and Preparation

- Collected datasets of external eye images from online sources, primarily Kaggle and Roboflow Universe.
- Performed data preprocessing, filtering out irrelevant, low-quality, or mislabeled images, and cropping to focus on eye regions.

- Applied data augmentation techniques, including rotation, flipping, brightness/contrast adjustments, and scaling, to expand and diversify the dataset.

Model Implementation and Testing

- Tested several pre-trained convolutional neural networks (VGG16, ResNet50, InceptionV3, DenseNet121) on static images.
- Based on evaluation metrics, DenseNet121 demonstrated the highest accuracy and F1 score and was selected as the base architecture.
- Enhanced DenseNet121 with Spatial and Channel Attention mechanisms to improve detection capabilities.
- Fine-tuned the improved model, named **EyeNet**, and evaluated it on static images, pre-recorded videos, and still lack of real-world images of patients.

Frontend Development

- Developed a user-friendly React-based interface, allowing users to upload images and record videos for analysis.
- Implemented timely and rapid feedback mechanisms to present results, including classifications and confidence scores.

Basic Backend-Frontend Communication

- Established the foundational RESTful API endpoints using Robyn Framework to handle communication between the backend and frontend.
- Enabled users to submit image and video files through the frontend and receive results seamlessly from the backend.

Video Processing Pipeline

- Implemented a pipeline to process uploaded videos by splitting them into individual frames.
- Integrated **YOLO-Face** to validate that each image contains a detectable face before sending it to the backend, and implemented a **Laplacian-based blur detection mechanism** to filter out blurry images that could degrade model performance.
- Integrated DLib for robust face and eye region detection across frames, ensuring consistent and accurate classification.
- Optimized the pipeline to handle different video formats, resolutions, and challenging conditions efficiently.

Model API Improvement and Refinement

- Enhanced the backend API for improved performance and scalability, delivering timely and accurate predictions.

- Added robustness features to handle edge cases, such as poor lighting, occlusions, and low video quality.

Refactoring and Optimization

- Refactored the codebase to improve modularity, readability, and maintainability.
- Optimized system performance to ensure efficiency and stability across various devices and environments.
- Dockerized the entire system to simplify deployment, scaling, and maintenance across environments.
- Deployed the backend server on **Google Cloud Platform (GCP)** for scalability, reliability, and accessibility.

5 Product

The EyeNet platform is structured into two main components: the **client-side frontend** and the **backend server**, communicating via RESTful APIs. This architecture ensures modularity, scalability, and clear separation of responsibilities.

On the **client side**, videos are decomposed into individual frames directly in the browser. Each frame is analysed locally with **YOLO-Face**, which detects whether a face is present. Only valid frames containing a face are sent to the backend for further processing, minimizing unnecessary computation and bandwidth usage.

On the **backend side**, the server receives the filtered frames and performs the core analysis. Using **Dlib**, it detects the face and eye regions within each frame, then crops the eye regions and runs inference with the trained **EyeNet model**. Finally, it aggregates predictions across frames and sends the diagnostic results back to the client for visualization.

The system is deployed on a **Google Cloud VM**, with **NGINX** serving as the reverse proxy and **ngrok** providing secure HTTPS tunnelling for public access. The frontend is implemented in React, while the backend leverages Robyn and TensorFlow for fast and scalable computation.

Figure 19Technologies stack used in the development and deployment of EyeNet

5.1 Backend Development

The backend serves as the computational core of the EyeNet system, responsible for processing client-uploaded videos and images, performing face and eye detection, running the trained EyeNet model, and returning diagnostic results.

To optimize performance and maintain modularity, the backend was designed to handle only valid frames (filtered on the client side) and to process them efficiently. Upon receiving frames, the backend detects face and eye regions using **Dlib**, crops the eye areas, and resizes each to 224×224 pixels. Pre-processed eye crops are then fed into the **EyeNet model**, implemented in TensorFlow and Keras, which classifies each frame into one of four categories: healthy, cataract, conjunctivitis, or stye. Predictions are aggregated across frames to generate the final result.

The backend is implemented using the **Robyn Framework**, chosen for its asynchronous performance and scalability, and is deployed on a **Google Cloud Platform VM** behind **NGINX** and **ngrok** for public HTTPS access. [37]

5.1.1 Backend Technology Stack

Framework: Robyn (asynchronous Python web framework).

Face and Eye Detection: After extracting frames, the backend detect and crop eye regions using **DLib**. DLib was selected for its balance of accuracy and efficiency, making it ideal for real-time applications. A figure of comparison of the existing methods which we reviewed above.

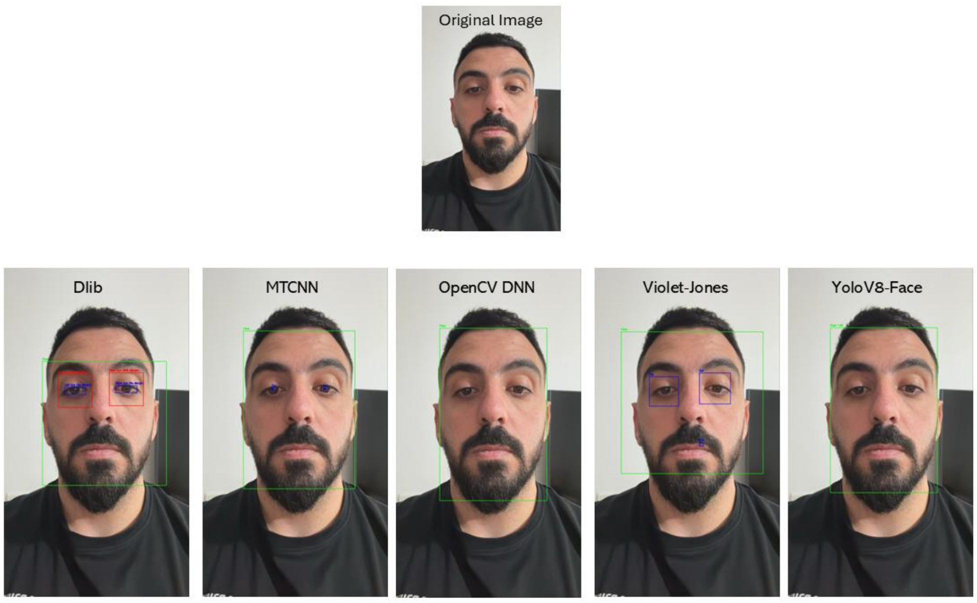
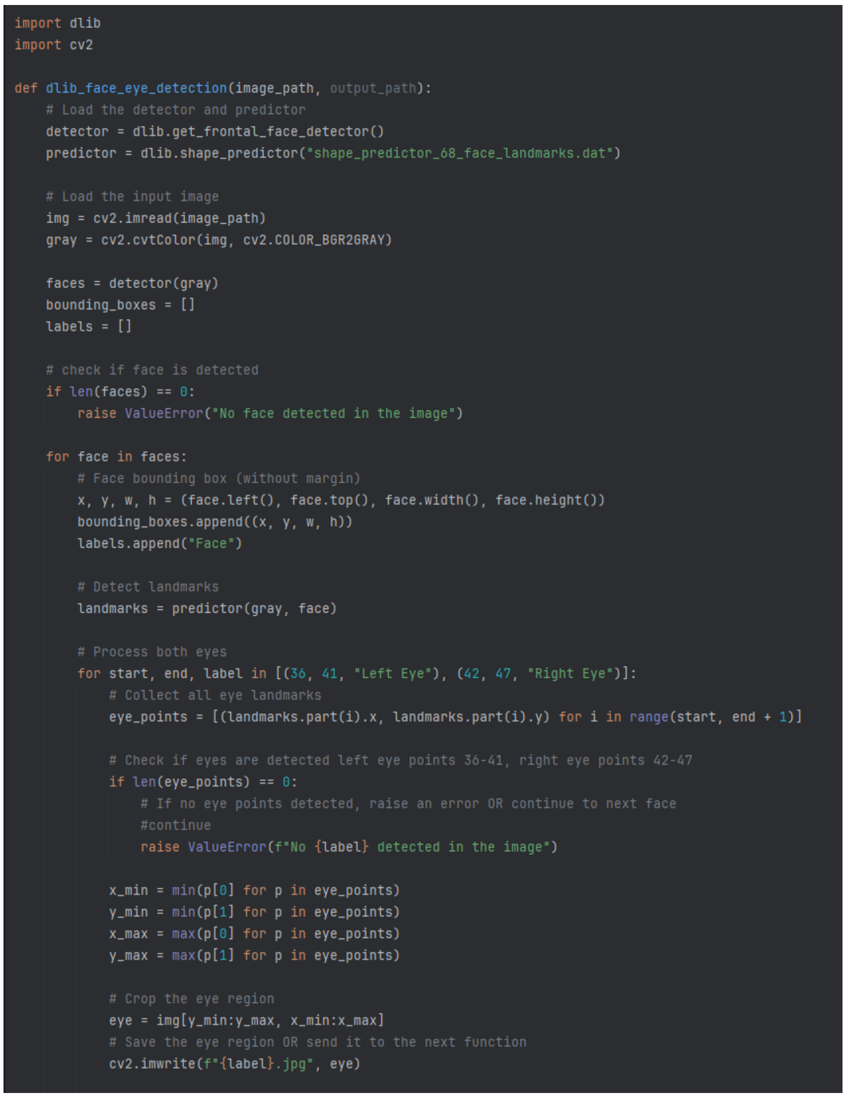


Figure 20 Comparing results of face and eye detection using 5 different models



```python
import dlib
import cv2

def dlib_face_eye_detection(image_path, output_path):
    # Load the detector and predictor
    detector = dlib.get_frontal_face_detector()
    predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")

    # Load the input image
    img = cv2.imread(image_path)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    faces = detector(gray)
    bounding_boxes = []
    labels = []

    # check if face is detected
    if len(faces) == 0:
        raise ValueError("No face detected in the image")

    for face in faces:
        # Face bounding box (without margin)
        x, y, w, h = (face.left(), face.top(), face.width(), face.height())
        bounding_boxes.append((x, y, w, h))
        labels.append("Face")

        # Detect landmarks
        landmarks = predictor(gray, face)

        # Process both eyes
        for start, end, label in [(36, 41, "Left Eye"), (42, 47, "Right Eye")]:
            # Collect all eye landmarks
            eye_points = [(landmarks.part(i).x, landmarks.part(i).y) for i in range(start, end + 1)]

            # Check if eyes are detected left eye points 36-41, right eye points 42-47
            if len(eye_points) == 0:
                # If no eye points detected, raise an error OR continue to next face
                #continue
                raise ValueError(f"No {label} detected in the image")

            x_min = min(p[0] for p in eye_points)
            y_min = min(p[1] for p in eye_points)
            x_max = max(p[0] for p in eye_points)
            y_max = max(p[1] for p in eye_points)

            # Crop the eye region
            eye = img[y_min:y_max, x_min:x_max]
            # Save the eye region OR send it to the next function
            cv2.imwrite(f"{label}.jpg", eye)
```

Figure 21 Code example of using DLib for face and eye detection

Model Inference: TensorFlow, Keras
Video Processing: OpenCV (cv2) for frame extraction
Deployment: GCP VM, NGINX, Ngrok
Environment: Python 3.10+, Docker, Docker-Compose

5.2 Frontend Development

The frontend provides a modern, responsive, and intuitive interface, enabling users to interact seamlessly with the EyeNet system. Built with **ReactJS**, it is designed for accessibility and usability across desktop and mobile platforms and ensures a smooth experience even when processing large video files.
Through the frontend, users can upload static images or recorded videos or capture a snapshot directly from their camera. The interface includes clear drag-and-drop zones and progress indicators to enhance usability. Client-side pre-processing is implemented in JavaScript: uploaded videos are decomposed into frames, **YOLO-Face** detects and filters frames that contain a face, and blurry frames are removed using a Laplacian-based blur detection method. Only valid frames are sent to the backend, reducing server load and network usage.
Results are retrieved asynchronously through RESTful APIs and displayed in a clear and actionable format. The results page shows a visual representation of the highest-confidence frame, average results across all frames, and allows users to export a PDF report of the analysis. [39]

5.2.1 Frontend Features

File Upload: Upload static images, record videos, or capture snapshots with camera integration.
Processing: Run YOLO-Face and blur detection in-browser to filter invalid frames.
Communicatio: Send valid frames to backend via RESTful API and handle responses asynchronously.
Results Display: Visualize predictions with confidence scores, highlight best frame, and offer PDF export..
Responsive Design: Fully responsive UI implemented with **Tailwind CSS** for accessibility on all devices.

5.3 Requirements

5.3.1 FR Requirements

No.	**Requirement**	**Status**
1	The system shall provide a user interface tool for users	Implemented
1.1	The system shall be cross-platform	Implemented
2	The system shall provide upload functionality	Implemented
2.1	The system shall allow users to upload videos	Not Implemented
3	The system shall integrate camera's functionality	Implemented
3.1	The system shall allow user to record video	Implemented
4	The system shall decompose the uploaded video into frames	Implemented
4.1	The system shall perform initial preprocessing on each frame	Implemented
4.2	The system shall isolate the detected eye object in each frame	Implemented
4.3	The system shall crop the frame to left and right eye	Implemented
5	The system shall utilize object detection techniques	Implemented
5.1	The system shall utilize eye tracking mechanism	Not Implemented
6	The system shall perform eye status classification	Implemented
6.1	The system shall save the outcome results from each frame	Implemented
6.2	The system shall calculate the average classification from all frames	Implemented
7	The system shall display the cropped eyes from chosen frame	Implemented
7.1	The system shall display the outcome results for each eye	Implemented
7.2	The system shall provide an option to export the results as PDF	Implemented
7.3	The system shall display a summary of analyzing	Implemented
8	The system shall provide error messages for issues	Implemented
9	The system shall provide an information about detected eye diseases	Implemented

Table 6 Functional Requirements

5.3.2 NFR Requirements

No.	Requirement	Type	Status
1	The system should ensure video uploads and processing are completed within a reasonable time frame.	Performance	Implemented – user can upload static image, and record live video. In both scenarios, processing takes less than 2ms per frame.
1.1	The system shall prioritize videos under 30 seconds for faster processing.	Performance	Implemented – instead of 30 seconds video, we let the user record video of 3-5 seconds length.
1.2	The system shall notify users if processing time exceeds the estimated duration.	Performance	Implemented – analysing animation lasts maximum 30 seconds.
1.3	The system will avoid performing unnecessary calculations in frames where no eyes are detected.	Performance	Implemented
2	The system must scale to handle increased user load and large video files without performance degradation.	Scalability	Irrelevant - requirement changed to smaller video files.
3	The system should be compatible with modern mobile and computer platforms.	Compatibility	Implemented
3.1	The supported platforms include iOS, Android, and Windows.	Compatibility	Implemented
3.2	The system shall support browsers like Chrome, Safari, and Edge.	Compatibility	Implemented
3.3	The system shall adapt to various screen sizes, especially for mobile phones.	Compatibility	Implemented – system is responsive
4	The system should ensure high precision and recall metrics for detecting and classifying eye conditions	Accuracy	Implemented – achieved 97% accuracy overall
4.1	The system shall periodically retrain models to improve accuracy over time.	Accuracy	Future Implementation
5	The system should provide a user-friendly interface that is intuitive and easy to navigate.	Usability	Implemented
5.1	The system shall include a dark mode option for improved usability in low-light environments.	Usability	Implemented
5.1.1	The system shall allow users to toggle between modes in real-time.	Usability	Implemented
5.2	The system shall include an intuitive navigation bar.	Usability	Implemented
5.3	The system shall include visual feedback (e.g., progress bars or	Usability	Implemented

	indicators) during video upload and processing.		
6	The system should be reliable and available, with minimal downtime.	Reliability	Implemented – Backend was deployed in GCP
6.1	The system should implement robust error handling and recovery mechanisms.	Reliability	Implemented
6.2	The system shall provide real-time status updates on service availability.	Reliability	Implemented
7	The system should support a wide range of video formats and resolutions.	Scalability	Implemented – supports 4 types of image formats, and 4 types of video formats
8	The system should support video uploads of up to 500MB without affecting processing efficiency.	Scalability	Irrelevant
9	The system should use advanced object detection algorithms for accurate eye isolation and tracking.	Technology	Implemented
10	The system will process inputs (e.g., videos, frames) in a format optimized for the classification model.	Optimization	Implemented
11	The system will prioritize preprocessing steps to reduce redundant computations.	Optimization	Implemented
12	The system should log all events and errors for debugging and monitoring purposes.	Auditability	Implemented
13	The system will allow integration of additional models for new eye condition detection.	Maintainability	Implemented – Code is modular and easy to maintenance

Table 7 Non-Functional Requirements

5.4 Model Structure

Our model is based on **DenseNet121**, augmented with spatial and channel attention mechanisms to enhance its ability to focus on the most informative regions of the input for accurate eye condition classification. The DenseNet backbone provides deep feature extraction, while the attention layers guide the model to emphasize relevant patterns within the eye region.
Input images are resized to 224×224×3 and passed through the DenseNet backbone (initially frozen and later fine-tuned), followed by the attention modules and a fully connected dense layer with L2 regularization and dropout for improved generalization. The model outputs one of four possible

classes: healthy, cataract, conjunctivitis, or stye. This architecture achieved significant improvements in both accuracy and interpretability compared to baseline CNNs.

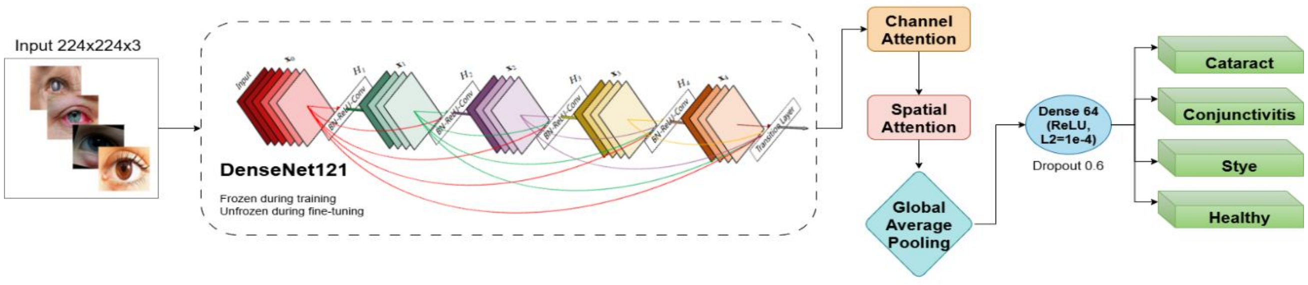


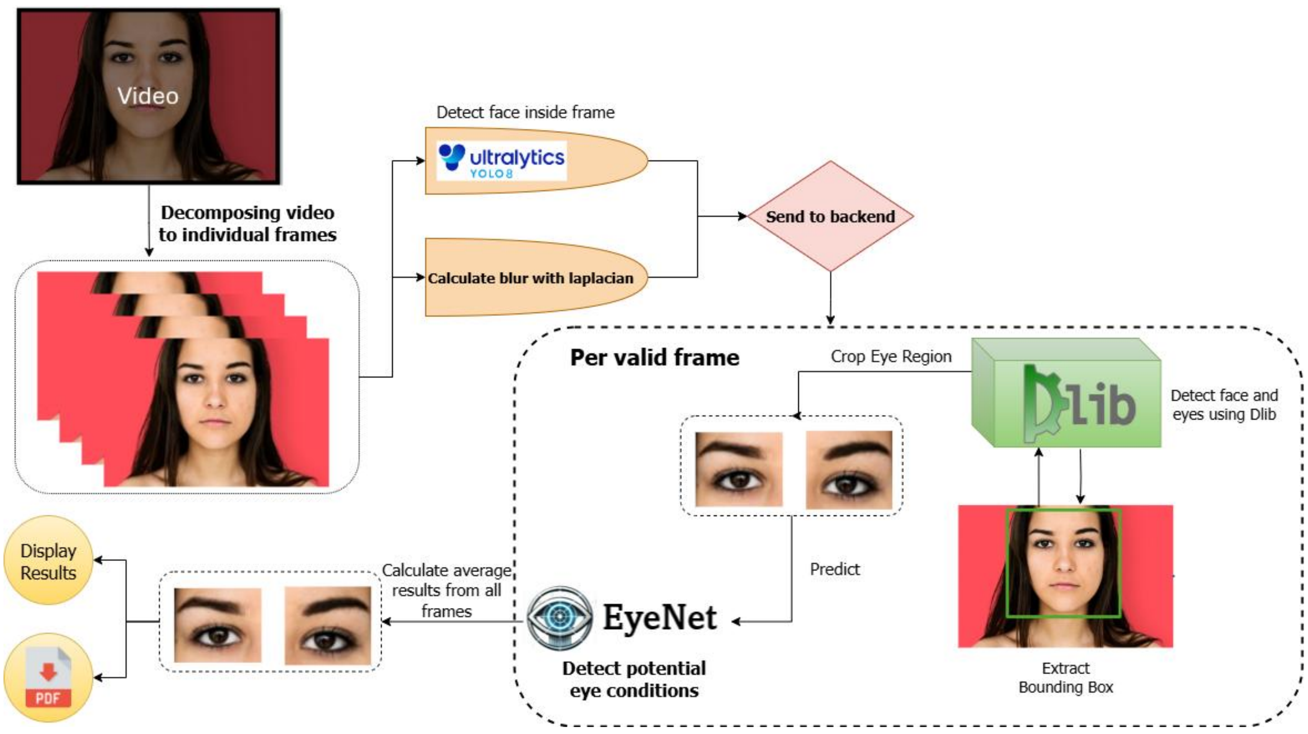
Figure 22 Visualization of EyeNet's model architecture. The model uses DenseNet121 as a base model and adds layers of attention mechanism to it to improve the model's performance.

5.5 Processing Pipeline

The analysis process begins with an input video, either uploaded or recorded via the web interface. The video is decomposed into individual frames to enable frame-level processing. On the client side, each frame undergoes **YOLO-Face detection** to ensure that only frames containing a valid face are kept, and blurry frames are discarded using Laplacian-based blur detection.

These filtered, valid frames are sent to the backend, where **DLib** detects the face and locates the eye regions within each frame. The detected eye regions are cropped, pre-processed, and passed to the **EyeNet model**, which classifies them into one of the four classes. Predictions are aggregated across all frames to compute an average result, providing robustness and reducing noise from individual frames.

The final output displays the analyzed results to the user and offers an option to export the findings as a PDF report for documentation or sharing purposes.



Video
Decomposing video to individual frames
Detect face inside frame
ultralytics YOLO8
Calculate blur with laplacian
Send to backend
Per valid frame
Crop Eye Region
Detect face and eyes using Dlib
Dlib
Predict
Extract Bounding Box
EyeNet
Detect potential eye conditions
Calculate average results from all frames
Display Results
PDF

Figure 23 Processing pipeline of EyeNet

5.6 System Use Case

The use case diagram details the system in general, the actors who take part in the processes in the system, and the actions the user can perform.

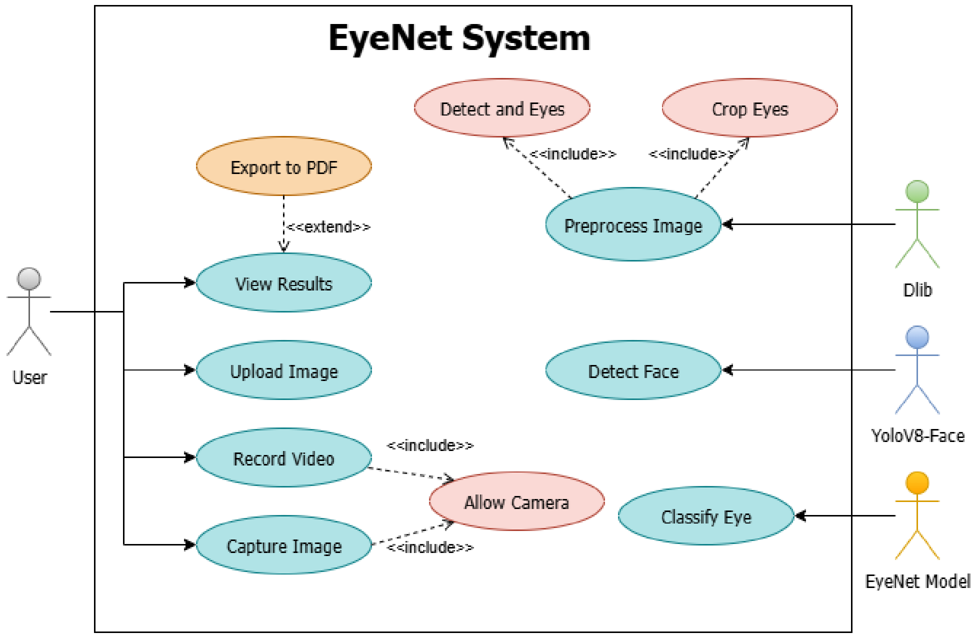


Figure 24 System's use case

5.7 System Activity Diagram

A system activity diagram describes the internal process that occurs in our system, which actor handles each action, and what the step-by-step sequence of actions is.

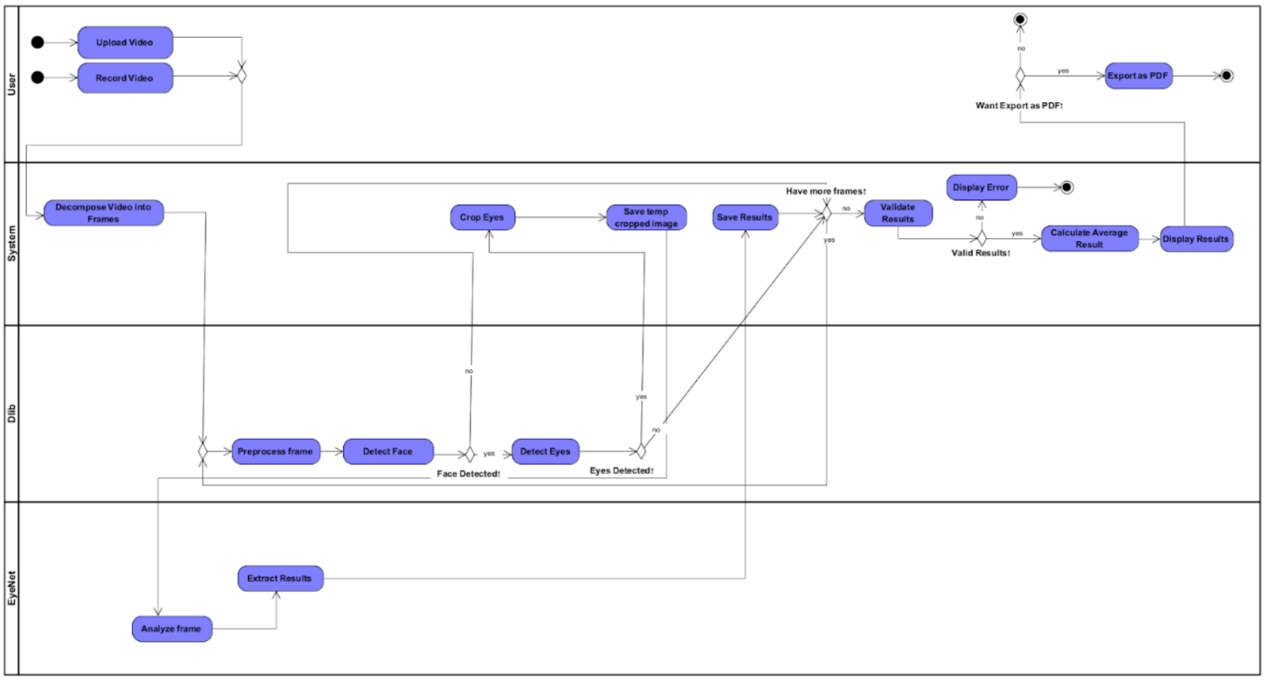


Figure 25 System's activity diagram

5.8 System Deployment Architecture

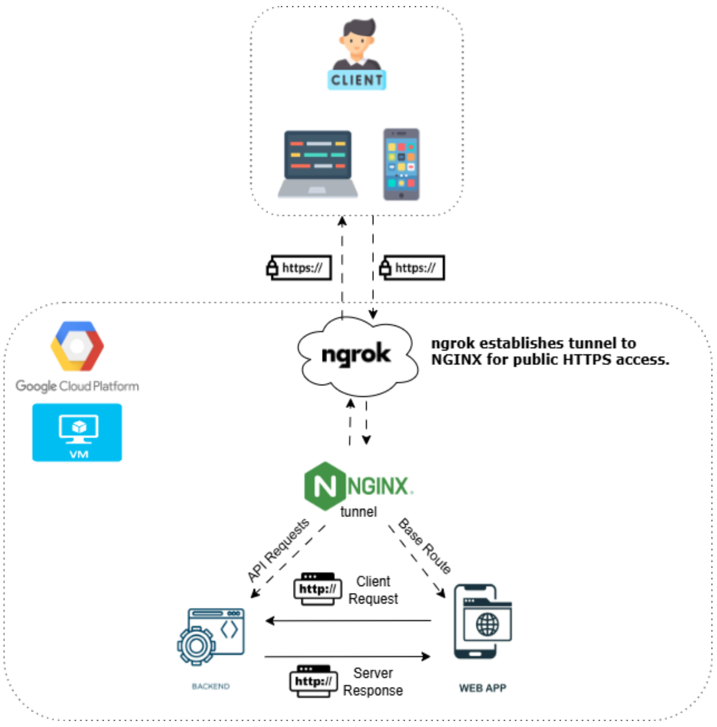


Figure 26 System deployment architecture of EyeNet

The EyeNet system is deployed on a **Google Cloud Platform (GCP)** virtual machine, ensuring scalability and reliability. The VM hosts both the backend and the frontend, served through an **NGINX reverse proxy** for efficient routing. To expose the service securely to the public over HTTPS without additional DNS configuration, we integrated **ngrok**, which establishes a secure public tunnel to the NGINX server.

Clients—whether on desktop or mobile devices—access the system through the ngrok-provided HTTPS endpoint. The frontend web application communicates with the backend via RESTful API requests routed through NGINX. This architecture allows modular separation of concerns, secure client-server communication, and ease of deployment and testing without the need for static IP or domain configuration.

6 Verification and Evaluation

6.1 Evaluation

The primary goal of this project was to develop a system capable of accurately detecting external eye conditions — cataracts, styes, and conjunctivitis — using videos or images captured by mobile or computer cameras. To evaluate the system, we focused on several key criteria. First, the classification accuracy was measured using precision, recall, F1 score, and overall accuracy. The system achieved a classification accuracy of **98%**, exceeding the initial target of 90%.

Additionally, The system processed videos of up to 5 seconds within **5-8 seconds**, meeting the target of under 15 seconds. Informal user testing confirmed the interface was intuitive and easy to navigate, with users able to upload videos, view results, and export findings without difficulty.

To ensure robustness, the system performed reliably under varied lighting, moderate occlusions, and different resolutions, although accuracy degraded slightly in extreme low-light scenarios. The system displayed appropriate and actionable error messages for corrupted files, unsupported formats, or videos without detectable eyes.

The evaluation process included extensive testing using different datasets and simulated challenging scenarios to assess the system's accuracy and resilience. All findings confirmed the project met its intended objectives and delivered a reliable and practical solution.

6.2 Verification

We verified the system by dividing it into independent modules — EyeNet Model, Video Processing, Camera Integration, Web Application, and Data Analysis — followed by full system integration testing. Tests included unit tests, manual QA, and end-to-end system testing.

The table below summarizes the results of the verification process:

No.	Module	Tested Function	Expected Result	Result
1	EyeNet Model	Classification Accuracy	Achieve classification accuracy >90% across all eye conditions.	Achieved 98% accuracy
2	EyeNet Model	False Positive and Negative Rates	Maintain a False Positive Rate (FPR) and False Negative Rate (FNR) <10%.	Achieved
3	EyeNet Model	Model Response Time	Provide classification results within 500ms per frame.	Achieved – each frame classification takes approximately 500ms.
4	Video Processing	Frame Decomposition	Decompose videos into individual frames accurately and efficiently.	Achieved
5	Video Processing	Eye Region Cropping	Detect and crop left and right eye regions accurately for all valid frames.	Achieved – yolo-face remove all invalid frames
6	Video Processing	Frame Preprocessing	Apply preprocessing (resizing, filtering) without altering critical features.	Achieved
7	Camera Integration	Video Recording	Record high-quality videos from the device camera and save them locally.	Achieved – saving the recorded video in local-storage
8	Camera Integration	Camera Permissions	Handle denied camera permissions gracefully with appropriate error messages.	Not Implemented – ask the user for permissions
9	Web-Application	Video Upload and Processing	Allow users to upload or record videos and process them without errors.	Achieved – user can upload image, snapshot static image or record video

10	Web-Application	User Interface Usability	Ensure the UI is intuitive and easy to navigate, validated by user feedback.	Achieved
11	Web-Application	Results Display	Display classification results for each eye with clear visualizations.	Achieved
12	Web-Application	Export Results as PDF	Generate accurate, well-formatted PDF reports of analysis results.	Achieved
13	Web-Application	Page Loading and Navigation	Load the landing page in under 2 seconds and provide seamless navigation.	Achieved
14	Data Analysis	Aggregated Classification Results	Calculate and display the average classification result for all processed frames.	Achieved
15	Data Analysis	Data Accuracy	Ensure all stored and analyzed data matches the expected results.	Achieved
16	System Integration	Processing Time for Short Videos (<30 sec)	Process short videos within 15 seconds.	Achieved – takes approximately 5 seconds
17	System Integration	Handling Large Video Files (500MB)	Process large videos efficiently without performance degradation.	Not Implemented – requirement changed to handle 3-5 secs of video
18	System Integration	Error Handling for Unsupported Formats	Display appropriate error messages for unsupported file formats.	Achieved
19	System Integration	Error Handling for Corrupted Files	Notify users when a corrupted file is uploaded.	Achieved
20	System Integration	Robustness in Poor Lighting Conditions	Maintain classification accuracy under varied lighting conditions.	Achieved

Table 8 Our modules, the test functions and the expected results

7 Results

7.1 Training Process

The training and validation loss (left) decreased steadily in Phase 1 and improved further after fine-tuning, reaching a final validation loss of **0.07**. Similarly, accuracy (right) improved throughout the training process, achieving **98.07%** validation accuracy after fine-tuning. These trends indicate effective learning with minimal overfitting and good generalization.

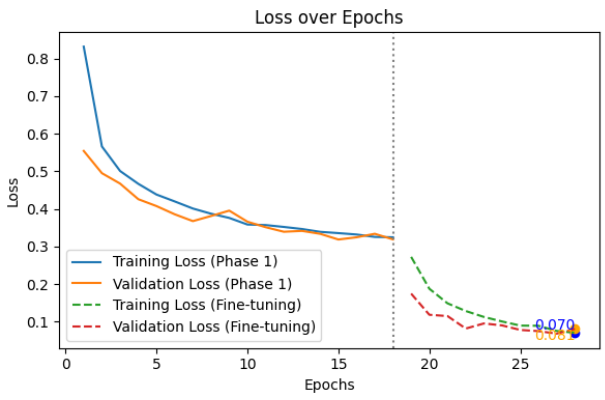


Figure 27 Validation Loss over Epochs

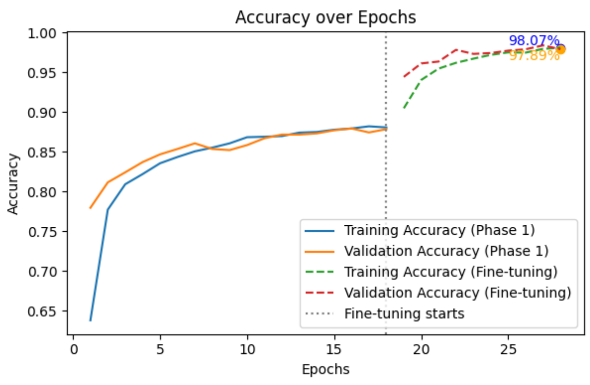


Figure 28 Training over Epochs

7.2 Performance By Class

The bar chart above illustrates the EyeNet classification performance on each class. All four eye conditions achieved high scores across all metrics: accuracy, precision, recall (TPR), F1 score, and TNR.

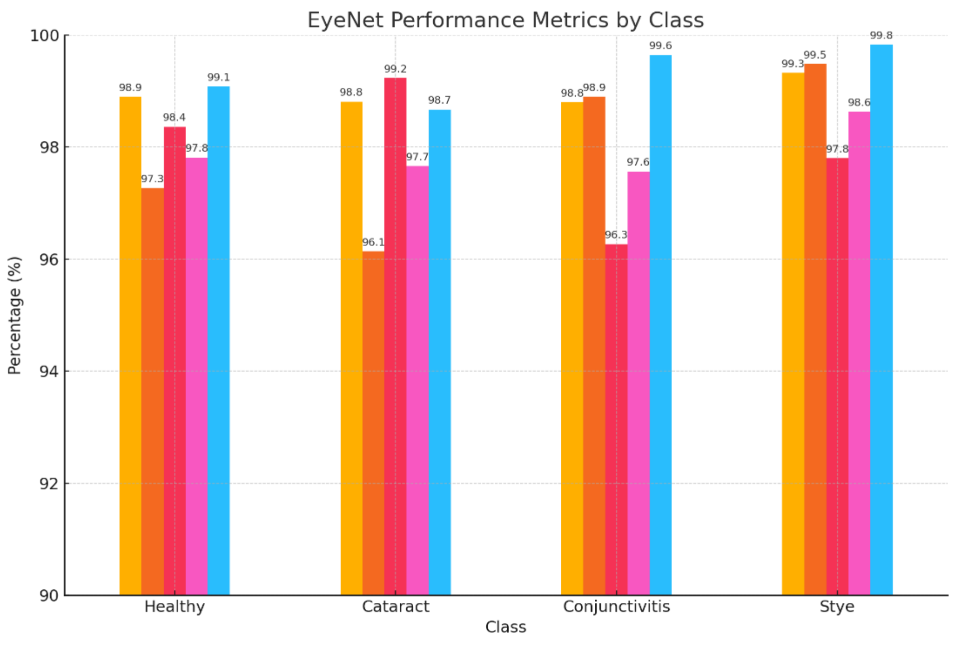


Figure 29 EyeNet Performance Metrics by Class

Among the four, cataract showed slightly lower metrics, consistent with its more subtle visual presentation, while stye achieved the highest performance. These results demonstrate the system's robustness and reliability across diverse conditions.

The tables below summarize the evaluation metrics of EyeNet compared to existing state-of-the-art models (VGG16, ResNet50, InceptionV3, DenseNet121). While previous sections presented these metrics for the baseline models, this section highlights the clear improvements achieved by EyeNet. Notably, EyeNet consistently outperformed all baselines across all classes (Healthy, Cataract, Conjunctivitis, Stye) in terms of Accuracy, Precision, Recall (TPR), F1 Score, and TNR, while achieving the lowest FNR and FPR. These results demonstrate the effectiveness of our architectural enhancements and validate the superiority of EyeNet over prior approaches in external eye condition classification.

Class	Model	Accuracy(%)	TPR(%)	FNR(%)	FPR(%)	TNR(%)	Precision(%)	F1 Score(%)
Healthy	VGG-16	90.984	77.391	22.608	4.484	95.515	85.191	81.104
	ResNet50	79.514	26.496	73.503	2.813	97.186	75.841	39.272
	Inception V3	94.322	88.644	11.355	3.784	96.215	88.644	88.644
	DenseNet121	96.521	92.48	7.519	2.131	97.868	93.533	93.004
	EyeNet	**98.9**	**98.363**	**1.637**	**0.921**	**99.079**	**97.269**	**97.813**
Cataract	VGG-16	91.01	83.938	16.061	6.632	93.367	80.837	82.358
	ResNet50	77.57	72.429	27.57	20.716	79.283	53.819	61.753
	Inception V3	95.729	91.457	8.542	2.847	97.152	91.457	91.457
	DenseNet121	96.969	92.583	7.416	1.568	98.431	95.162	93.855
	EyeNet	**98.811**	**99.233**	**0.767**	**1.33**	**98.67**	**96.135**	**97.659**
Conjunctivitis	VGG-16	86.687	87.621	12.378	13.623	86.376	68.192	76.695
	ResNet50	74.437	46.086	53.913	16.112	83.887	48.808	47.408
	Inception V3	93.721	87.826	12.173	4.313	95.686	87.157	87.49
	DenseNet121	94.143	94.322	5.677	5.91	94.083	84.162	88.953
	EyeNet	**98.798**	**96.266**	**3.734**	**0.358**	**99.642**	**98.896**	**97.564**
Stye	VGG-16	89.194	66.803	33.196	3.341	9.658	86.95	75.556
	ResNet50	75.867	69.77	30.23	22.097	77.902	51.278	59.111
	Inception V3	94.232	88.081	11.918	3.716	96.283	88.762	88.421
	DenseNet121	94.539	84.961	15.038	2.267	97.732	92.586	88.61
	EyeNet	**99.322**	**97.801**	**2.199**	**0.171**	**99.829**	**99.48**	**98.633**

Table 9 Performance Comparison Table

Class	Model	TP(True Positive)	FN(False Negative)	FP(False Positive)	TN(True Negative)
Healthy	VGG-16	1513	442	263	5602
	ResNet50	518	1437	165	5700
	Inception V3	1733	222	222	5643
	DenseNet121	1808	147	125	5740
	EyeNet	**1923**	**32**	**54**	4811
Cataract	VGG-16	1641	314	389	5476
	ResNet50	1416	539	1215	4650
	Inception V3	1788	167	167	5698
	DenseNet121	1810	145	92	5773
	EyeNet	**1940**	**15**	**78**	4787
Conjunctivitis	VGG-16	1713	242	799	5066
	ResNet50	901	1054	945	4920
	Inception V3	1717	238	253	5612
	DenseNet121	1844	111	349	5518
	EyeNet	**1882**	**73**	**21**	4844
Stye	VGG-16	1306	649	196	5669
	ResNet50	1364	591	1296	4569
	Inception V3	1722	233	218	5647
	DenseNet121	1661	294	133	5732
	EyeNet	**1912**	**43**	**11**	4854

Table 10 Confusion Matrix Table

7.3 Model Explainability with Grad-CAM

To interpret the model's decisions, Grad-CAM heatmaps were generated to highlight the regions of the eye that contributed most to predictions. Each row below corresponds to one class and shows: the original image, Grad-CAM heatmap, and overlay.

7.3.1 Healthy

For healthy eyes, the model focused on clear iris and sclera regions, as expected.

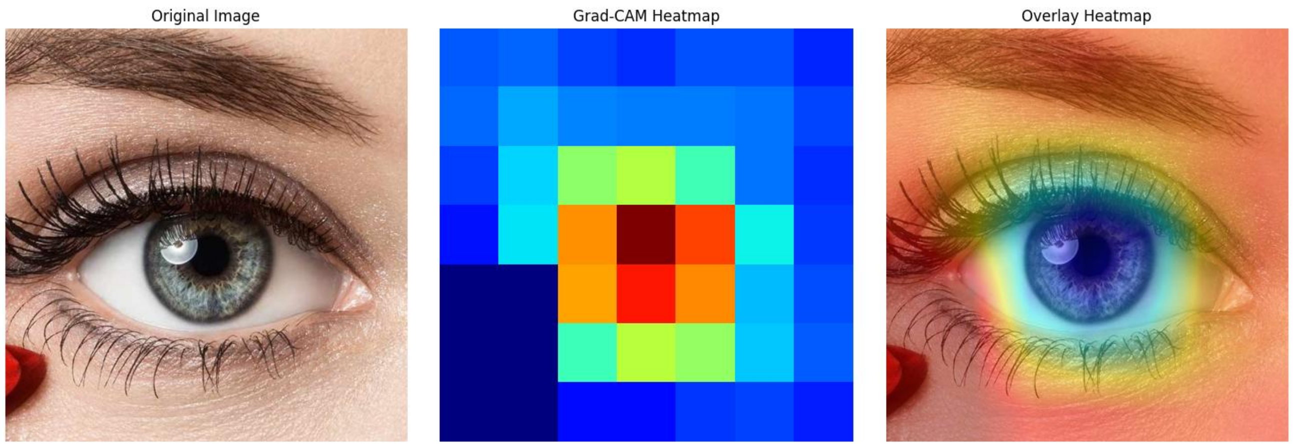


Figure 30 Grad-CAM Heatmap — Healthy Eye

7.3.2 Cataract

In cataract cases, the attention centered on the cloudy or opaque lens area, aligning with clinical expectations

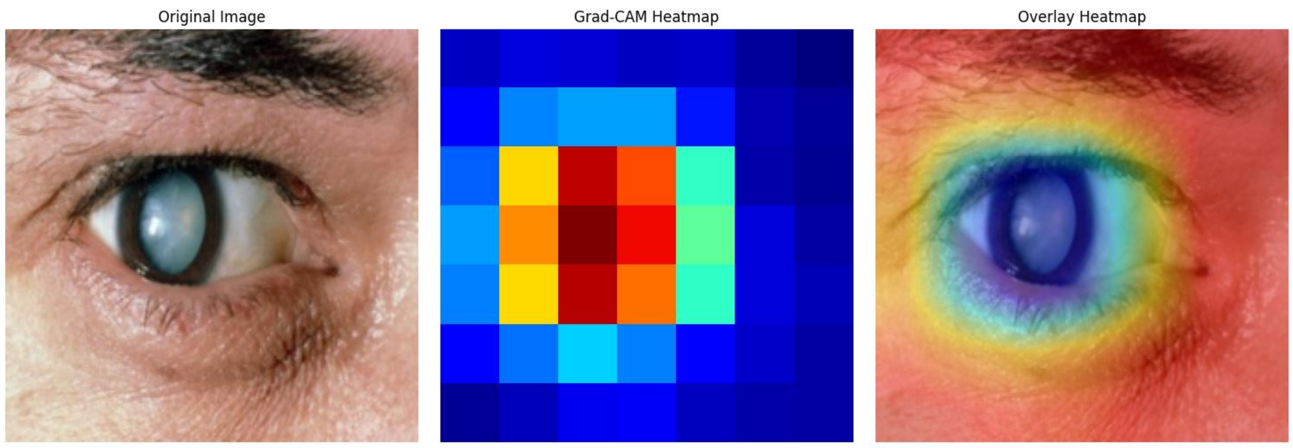


Figure 31 Grad-CAM Heatmap — Cataract Eye

7.3.3 Conjunctivitis

For conjunctivitis, the model highlighted the redness and irritation in the conjunctival area.

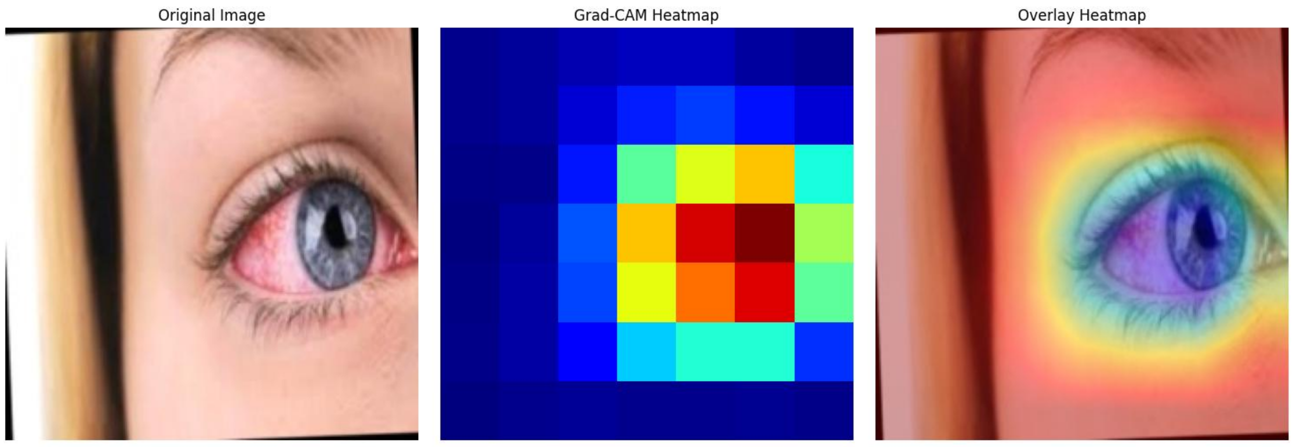


Figure 32 Grad-CAM Heatmap — Conjunctivitis Eye

7.3.4 Stye

For stye detection, the model concentrated on inflamed regions of the eyelid

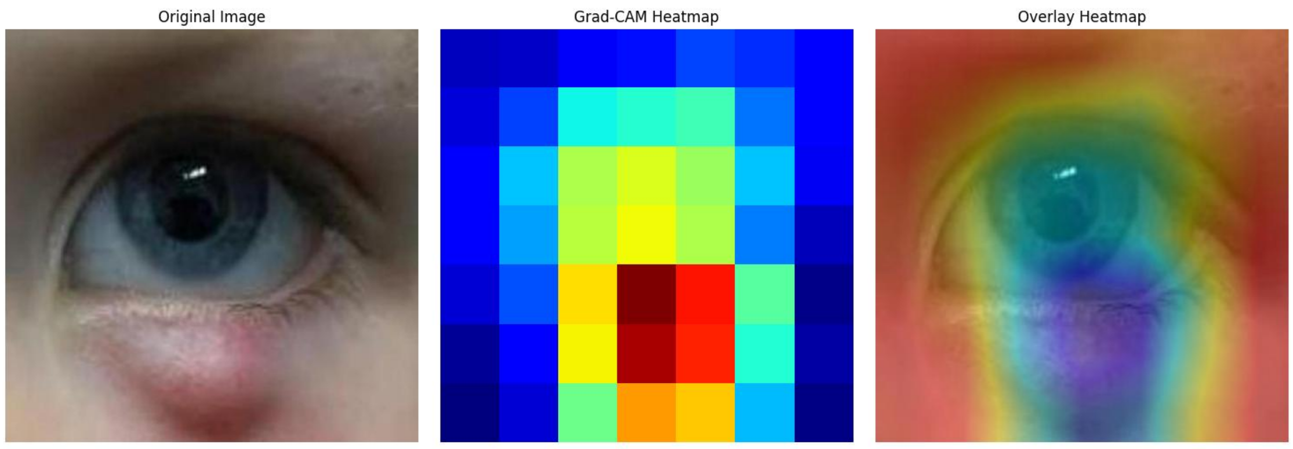


Figure 33 Grad-CAM Heatmap — Stye Eye

7.4 Discussion

The results demonstrate that EyeNet achieves excellent classification performance across all targeted eye conditions, exceeding the initial goal of 90% accuracy. The training curves validate the effectiveness of the training and fine-tuning phases. The model's ability to focus on medically-relevant regions, as evidenced by Grad-CAM visualizations, increases its interpretability and reliability. Minor performance differences between classes, such as slightly lower metrics on cataract, suggest avenues for further dataset enhancement and fine-tuning.

8 User Guide

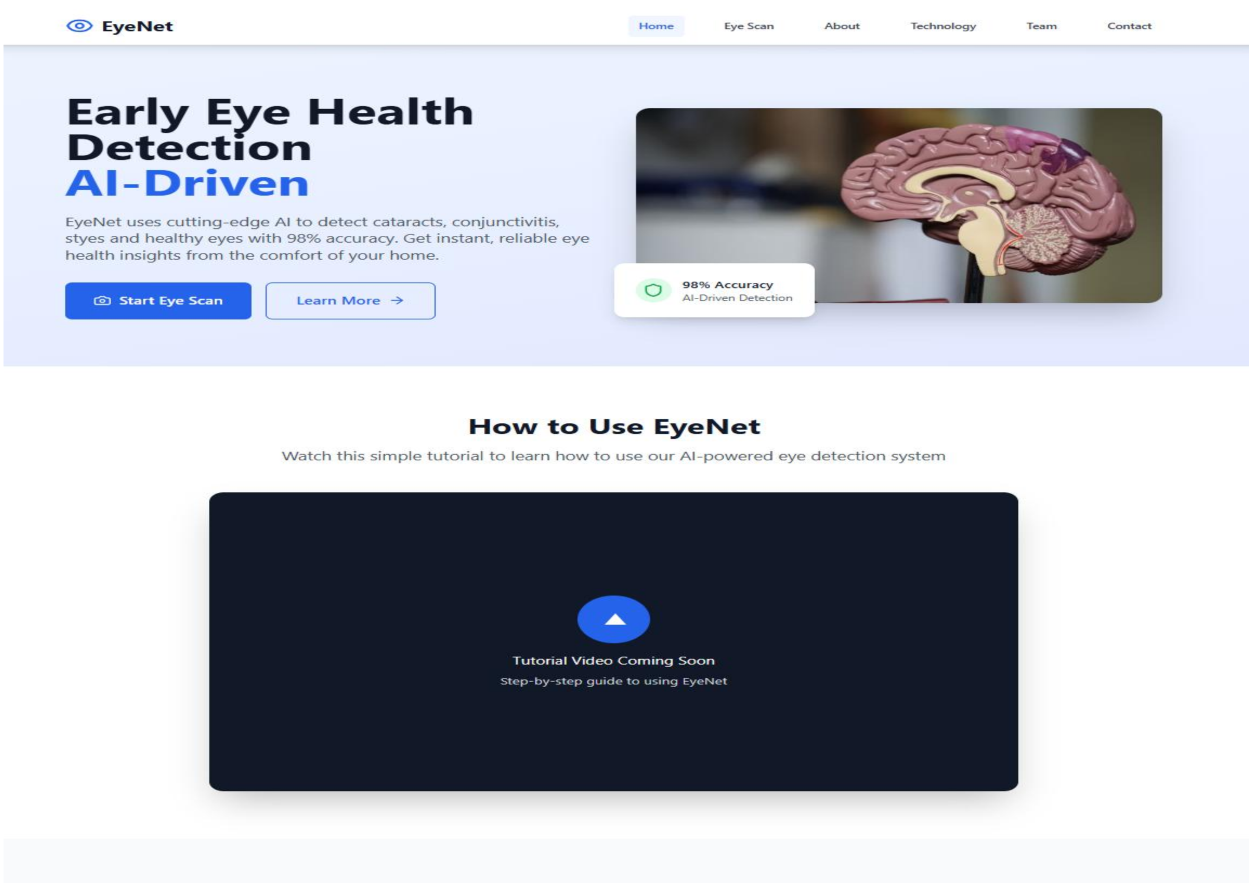
This document provides detailed instructions on how to use the EyeNet system. EyeNet is a web-based AI application designed to detect external eye conditions such as **cataracts, styes, and conjunctivitis** using images or short video recordings from a mobile or computer camera.

1.Accessing the System

Open your browser and navigate to the EyeNet web application URL (as provided in github). The system supports all major browsers, including Chrome, Safari, and Microsoft Edge, and is compatible with both desktop and mobile devices.

1.Home Page Overview

The EyeNet home page introduces users to the platform and its purpose: fast, AI-powered detection of external eye conditions like cataracts, styes, and conjunctivitis. At the center of the page, users see a prominent **“Start Eye Scan”** button, which leads directly to the diagnostic tool. A **“Learn More”** button is also available for those who want to understand the technology before using it. The homepage also features a section for a tutorial video and highlights EyeNet's key advantages such as high accuracy, instant results, and user-friendliness. Users can also explore other parts of the platform through the top navigation bar, including links to About, Technology, Team, and Contact. Overall, the home page is designed to quickly engage users and guide them toward scanning their eyes in just a few simple steps.



Early Eye Health Detection AI-Driven

EyeNet uses cutting-edge AI to detect cataracts, conjunctivitis, styes and healthy eyes with 98% accuracy. Get instant, reliable eye health insights from the comfort of your home.

How to Use EyeNet

Watch this simple tutorial to learn how to use our AI-powered eye detection system

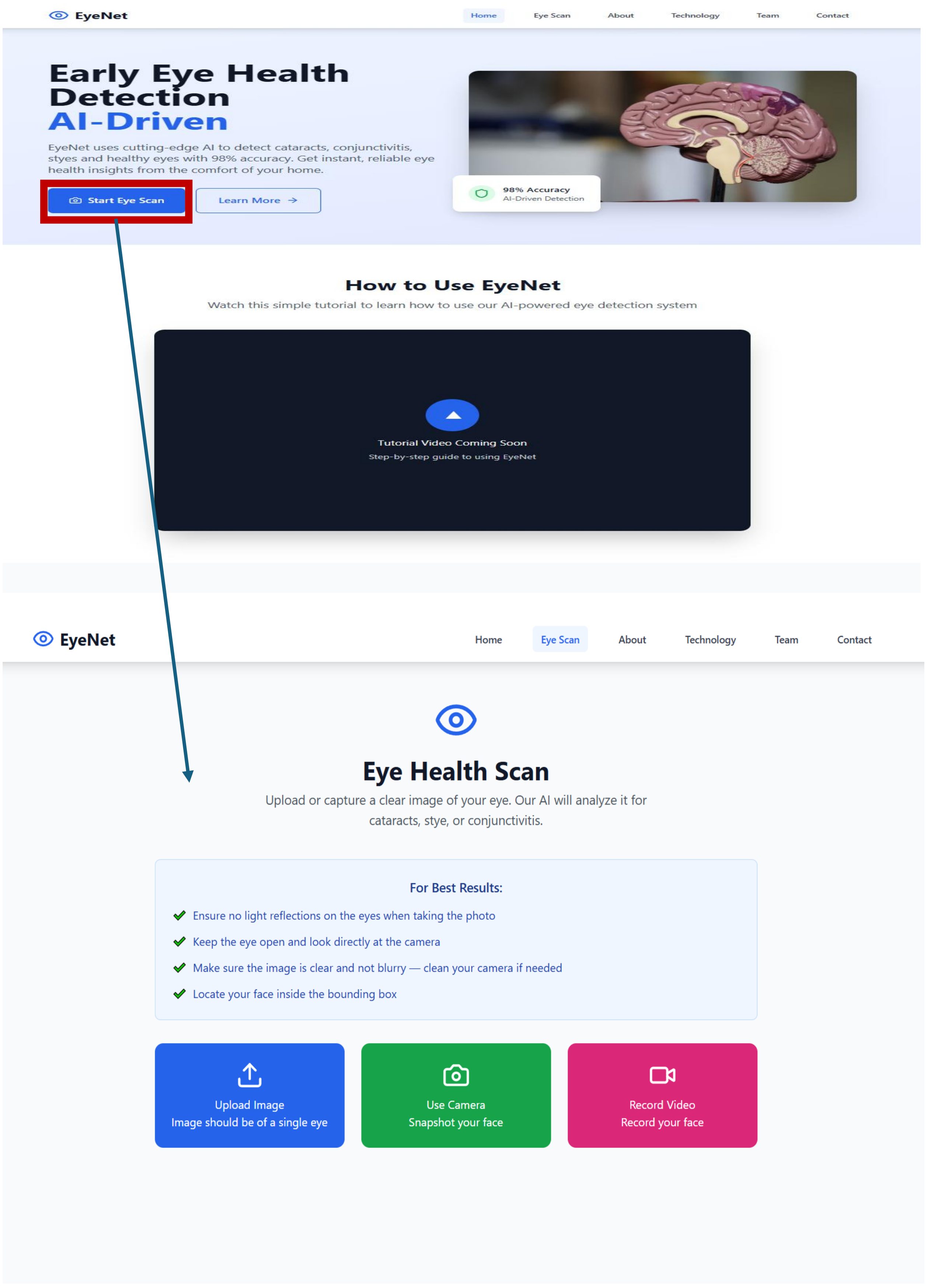
Tutorial Video Coming Soon

Step-by-step guide to using EyeNet

2. Click “Start Eye Scan”

Click the **blue “Start Eye Scan”** button located as marks in the image.

This will navigate you to the scan page where you can submit an image, video or capture image using webcam.


EyeNet

Home Eye Scan About Technology Team Contact

Early Eye Health Detection AI-Driven

EyeNet uses cutting-edge AI to detect cataracts, conjunctivitis, styes and healthy eyes with 98% accuracy. Get instant, reliable eye health insights from the comfort of your home.

Start Eye Scan Learn More →

98% Accuracy
AI-Driven Detection

How to Use EyeNet

Watch this simple tutorial to learn how to use our AI-powered eye detection system

Tutorial Video Coming Soon
Step-by-step guide to using EyeNet

EyeNet

Home Eye Scan About Technology Team Contact

Eye Health Scan

Upload or capture a clear image of your eye. Our AI will analyze it for cataracts, stye, or conjunctivitis.

For Best Results:

- ✓ Ensure no light reflections on the eyes when taking the photo
- ✓ Keep the eye open and look directly at the camera
- ✓ Make sure the image is clear and not blurry — clean your camera if needed
- ✓ Locate your face inside the bounding box

Upload Image
Image should be of a single eye

Use Camera
Snapshot your face

Record Video
Record your face

3. **Upload or Capture Your Eye Image or Video**

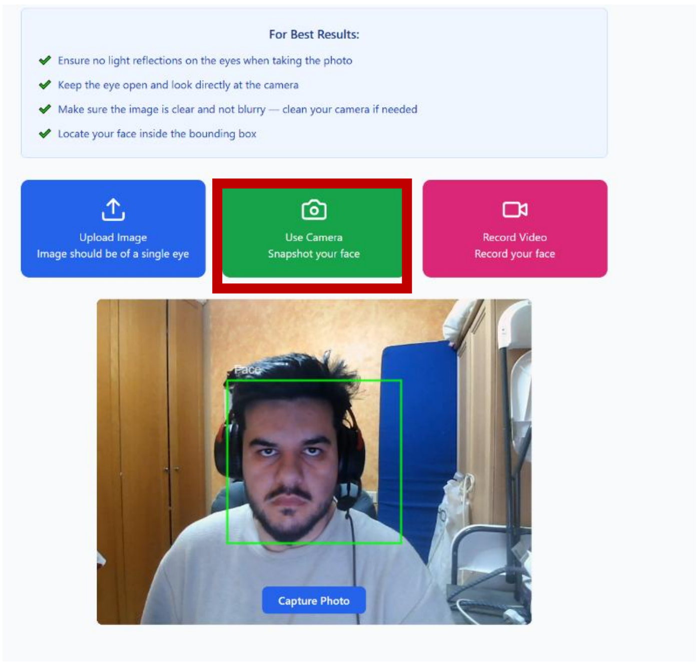
On the **Eye Scan** page, you can choose how to submit your eye data using one of the following three options. No matter which method you choose, make sure your entire face is visible and centered in the frame, as the system needs to detect both eyes accurately.

Upload Image

This option allows you to upload an existing photo from your device. The image must include one cropped eye clearly visible. Make sure the photo is sharp, and free of reflections or shadows.

Use Camera

This option lets you take a real-time snapshot using your device's camera. Ensure that your whole face is within the frame, looking directly at the camera with open eyes. This is ideal if you don't have a saved photo and want to capture one instantly.

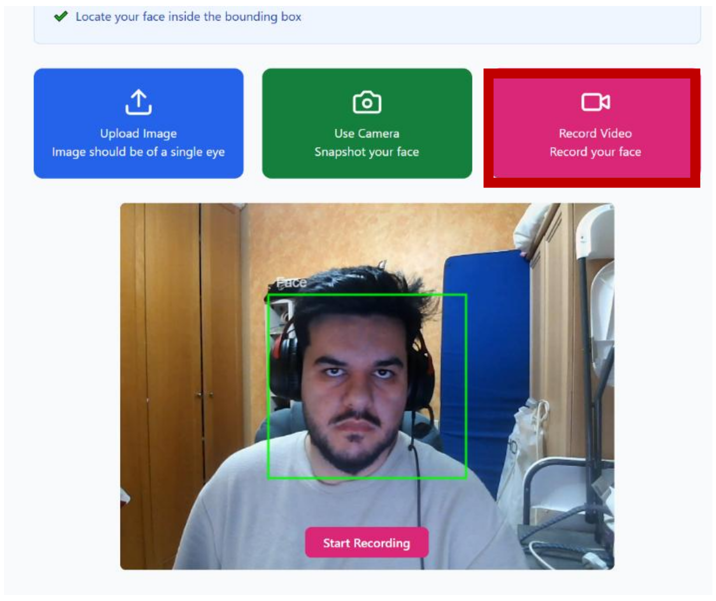


For Best Results:

- ✓ Ensure no light reflections on the eyes when taking the photo
- ✓ Keep the eye open and look directly at the camera
- ✓ Make sure the image is clear and not blurry — clean your camera if needed
- ✓ Locate your face inside the bounding box

Record Video

This option enables you to record a short video, usually 3 to 5 seconds long (configurable). During recording, keep your face steady, centered, and looking straight at the camera. The system will analyze multiple frames from the video to improve diagnostic accuracy.



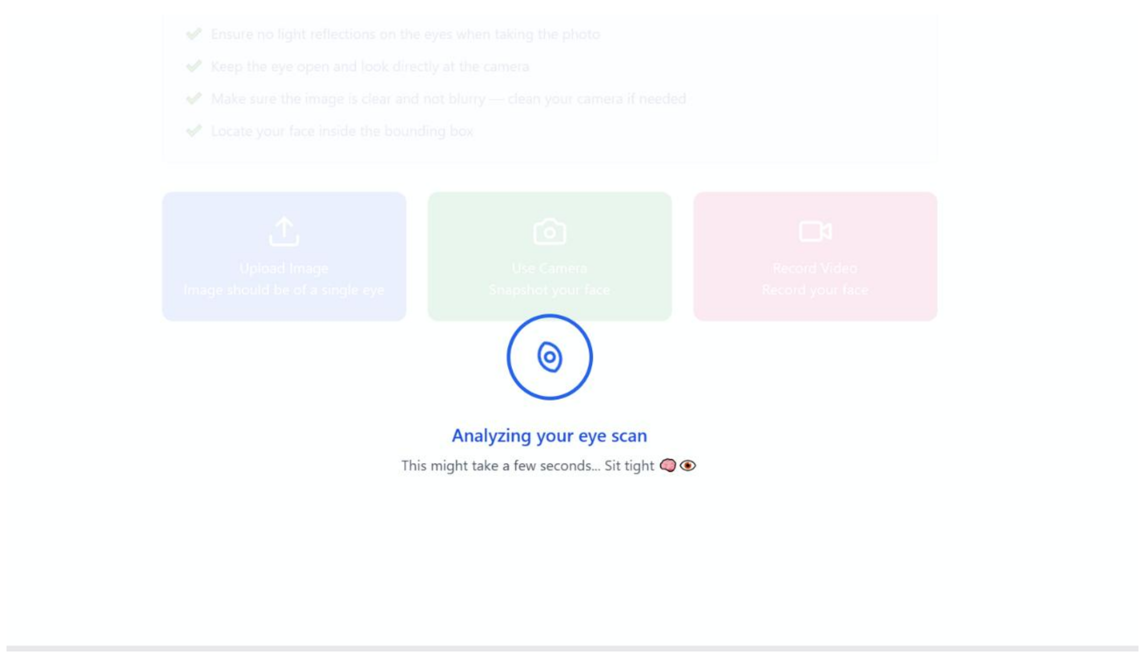
Before starting, you may be prompted to allow camera access to approve it to proceed. For the best results, avoid glare, ensure the image is not blurry, and keep your eyes open and clearly visible within the bounding box.

4. Start Analysis

After uploading an image, capturing a photo, or recording a short video, you will be prompted to begin the analysis. Click the **“Capture Photo”** button if you are using the camera snapshot option, or click **“Start Recording”** to begin a video scan. Once the photo is taken or the video is recorded, the system immediately transitions to the analysis phase without any additional action required from you.

A loading screen appears with the message **“Analyzing your eye scan”**, indicating that the system is processing your input. During this time, the AI detects your face, isolates the eye regions, and classifies each eye as healthy, or as having cataract, stye, or conjunctivitis. For video inputs, this process includes analyzing multiple frames to increase accuracy.

The analysis typically takes just a few seconds. Afterward, you’ll be automatically directed to a results page showing your diagnosis, confidence scores, and personalized suggestions.



✓ Ensure no light reflections on the eyes when taking the photo
✓ Keep the eye open and look directly at the camera
✓ Make sure the image is clear and not blurry — clean your camera if needed
✓ Locate your face inside the bounding box

Analyzing your eye scan

This might take a few seconds... Sit tight 🧠👁

5. Viewing Analysis Results

Once the scan is complete, the system automatically redirects to the Analysis Results page, where the user is presented with a structured summary of the diagnostic outcome.

The results are displayed separately for each eye but follow the same standardized format. For each eye, the system provides:

Diagnostic Summary

A concise label indicating the predicted condition based on the highest-confidence prediction generated by the AI model.

Confidence Scores

A set of horizontal bar charts showing the AI's confidence level (in percentage) for each possible diagnosis. This allows the user to assess how strongly the system favors the predicted outcome compared to other conditions.

Recommendations Section

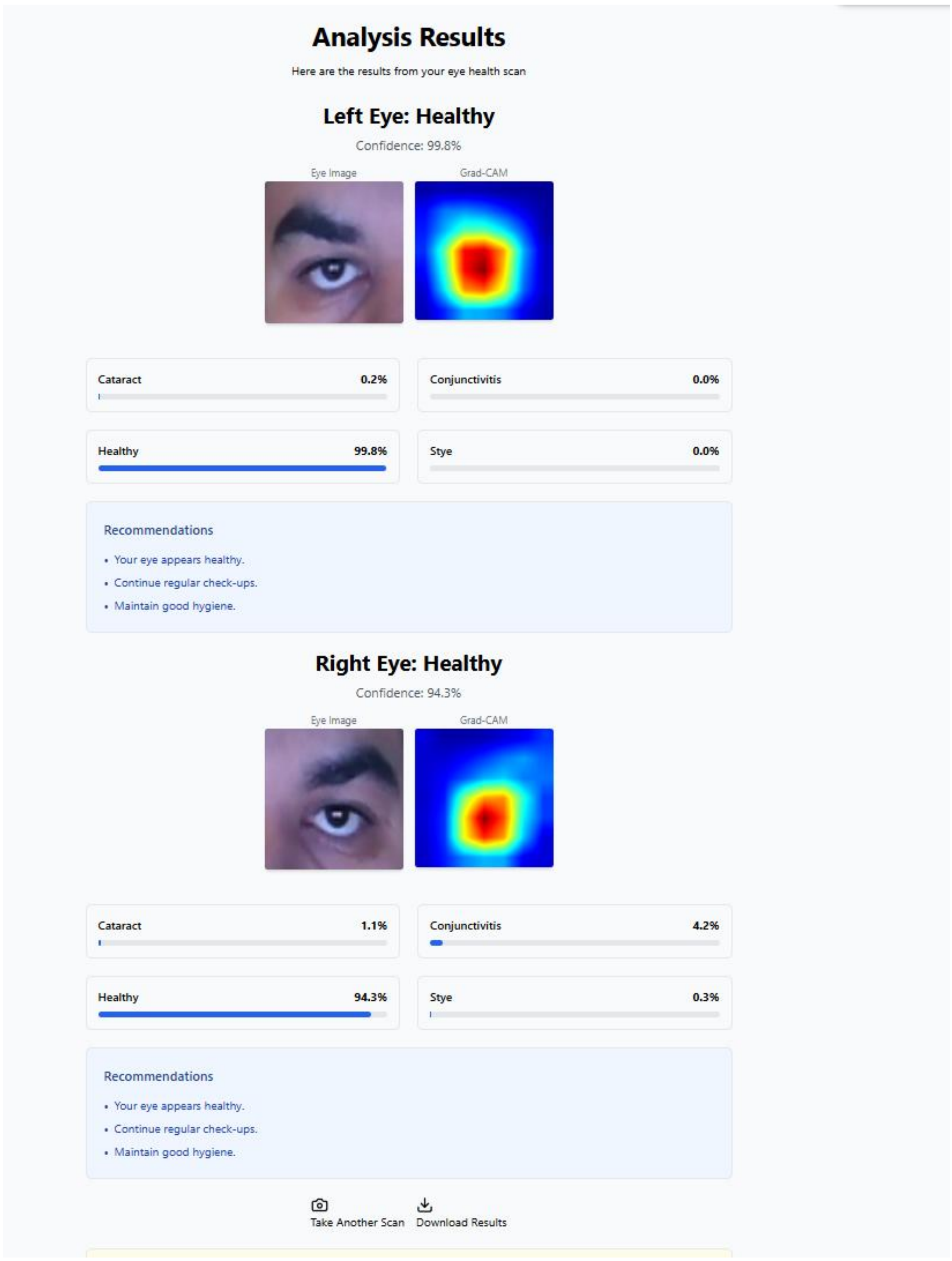
Below the diagnosis, the system provides a brief set of personalized recommendations. These may include general wellness advice follow-up suggestions or alerts if further attention may be needed.

At the bottom of the results page, users have access to two key options:

Take Another Scan : Returns to the Eye Scan page to initiate a new evaluation.

Download Results : Exports a PDF report that includes the scanned image, diagnosis, confidence scores, and recommendations. This report is suitable for personal records or consultation with a healthcare provider.

The Analysis Results interface is designed to be clear, transparent, and actionable, allowing users to quickly understand their eye health status and take appropriate next steps.



Analysis Results

Here are the results from your eye health scan

Left Eye: Healthy

Confidence: 99.8%

Cataract	0.2%	Conjunctivitis	0.0%
Healthy	99.8%	Stye	0.0%

Recommendations

- Your eye appears healthy.
- Continue regular check-ups.
- Maintain good hygiene.

Right Eye: Healthy

Confidence: 94.3%

Cataract	1.1%	Conjunctivitis	4.2%
Healthy	94.3%	Stye	0.3%

Recommendations

- Your eye appears healthy.
- Continue regular check-ups.
- Maintain good hygiene.

6. Downloading the PDF Report

At the bottom of the results page, users have the option to export their scan results by clicking the "Download Results" or "Export as PDF" button.

Upon clicking this button, the system generates a professionally formatted report containing the following information:

Date and Time of Scan: Clearly displayed at the top of the report for record-keeping and clinical reference.

Predicted Diagnosis: A summarized statement of the predicted condition based on the highest-confidence result.

Confidence Table: A tabular breakdown listing the confidence percentages for each possible condition:

- Cataract
- Conjunctivitis
- Stye
- Healthy

Personalized Recommendations: A short list of follow-up actions or health tips based on the diagnosis, such as maintaining good hygiene or attending regular check-ups.

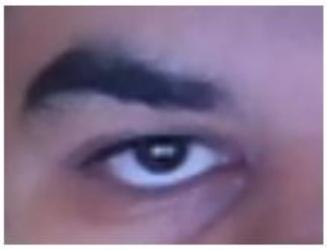
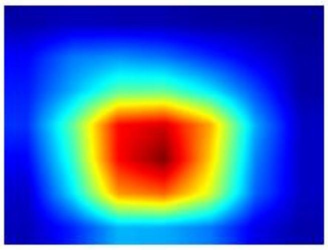
This downloadable PDF serves as an official copy of your scan results and is suitable for sharing with medical professionals, attaching to electronic health records, or saving for personal use.

EyeNet AI Scan Report

Generated on: 7/19/2025, 9:39:03 AM

Left Eye Prediction: Healthy

Confidence: 99.8%

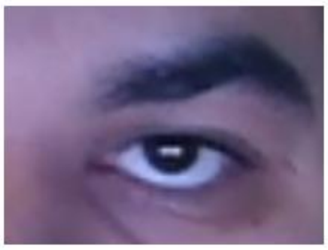
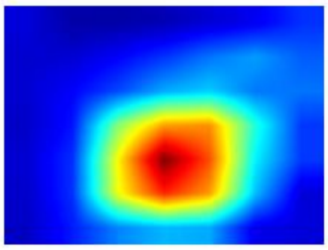



Condition	Confidence
Cataract	0.19%
Conjunctivitis	0.04%
Healthy	99.76%
Stye	0.01%

- **Your eye appears healthy.**
- **Continue regular check-ups.**
- **Maintain good hygiene.**

Right Eye Prediction: Healthy

Confidence: 94.3%

Condition	Confidence
Cataract	1.15%
Conjunctivitis	4.24%
Healthy	94.33%
Stye	0.29%

9 Maintenance Guide

This guide is intended to ensure the continued use and maintenance of the **EyeNet Eye Disease Detection System** after the completion of the project. It provides clear instructions for applying updates, improvements, and ongoing maintenance in order to support the system's lifecycle and future usability.

9.1 System Overview

The EyeNet system is a web-based application for detecting common eye diseases from external eye images. It consists of:
Backend: A Python application (Robyn + Uvicorn) that serves the machine learning model and API.
Frontend: A React-based web application for user interaction.
Model: A pre-trained Keras/TensorFlow .h5 file located in the backend.
Both components are deployed on a cloud-based server and communicate through standard HTTP interfaces.

9.2 Operating Environment

<u>**Hardware Requirements:**</u>
Processor: At least 2 vCPUs
Memory: Minimum 4 GB RAM (8 GB recommended for model inference)
Optional GPU: For improved inference performance, an NVIDIA GPU with CUDA 11.2+ is supported but not required.
<u>**Software Requirements:**</u>
Operating System: Ubuntu 20.04 LTS or newer
Python: Version 3.9 or 3.10 (used for the backend)
Node.js: Version 18 or higher (used for the frontend)
Pip: Python package manager
Docker & Docker-Compose : optional, but recommended for ease of deployment and development

These are the only required technologies specific to the system; no external database or third-party server software is required beyond standard web server infrastructure.

9.3 Installation Instructions

Below are the steps for installing and running the EyeNet system, focusing on the components developed during the project. Instructions are provided for both a manual setup and a Docker-based setup.

9.3.1 Option A: Using Docker (Recommended)

To simplify installation and ensure environment consistency, we provide a docker-compose.yml file that builds and runs both the backend and frontend in isolated containers.

Steps:

1. Install Docker and Docker Compose on the server/local machine
2. Clone the repository

```
git clone https://github.com/GalBitton/EyeNet-Eye-Disease-Detection.git
cd EyeNet-Eye-Disease-Detection/Project
```

3. Build and start the system

```
docker-compose up --build
```

4. Once running:

a. Frontend is accessible at: http://<server_ip>:5173

b. Backend is accessible at: http://<server_ip>:8000

For development purposes <server_ip> is usually localhost/127.0.0.1

9.3.2 Option B: Manual Setup

This approach requires installing dependencies for each component separately.

1. Clone the repository and navigate to the project directory
2. Backend setup:

```
cd EyeNet-Eye-Disease-Detection/Project/Backend
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt
uvicorn app.main:app --host 0.0.0.0 --port 8000
```

3. Frontend setup:

```
cd EyeNet-Eye-Disease-Detection/Project/Frontend
npm install
# you can use npm run dev – for developing purpose
npm run build
```

The frontend is then available at http://localhost:5173, and the backend at http://localhost:8000 (or http://<deployed_url>).

4. For deployment, you can deploy the Frontend/build directory to the web server (e.g. NGINX)

9.4 Updating the Model

To improve the accuracy of the system, the machine learning model can be updated as follows:

Add new labelled images to the dataset in the /train, /val, /test folders under the correct category (**cataract, conjunctivitis, stye, normal).**

Run the provided training script to create a new model file.

Replace the model.h5 file located in backend/models with the updated file.

- The .env file is used to store sensitive environment variables and is excluded from version control.
- Documentation for additional configuration options and scripts is available within the repository.

10 References

[1] Bourne, R. R. A., et al. "Magnitude, temporal trends, and projections of the global prevalence of blindness and distance and near vision impairment: a systematic review and meta-analysis." *The Lancet Global Health*, vol. 5, no. 9, 2017, pp. e888-e897.

[2] Flaxman, S. R., et al. "Global causes of blindness and distance vision impairment 1990-2020: a systematic review and meta-analysis." *The Lancet Global Health*, vol. 5, no. 12, 2017, pp. e1221-e1234.

[3] World Health Organization. "Blindness and vision impairment." [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment. [Accessed: Dec. 27, 2024].

[4] Ackland, P., et al. "The Lancet Global Health Commission on Global Eye Health: vision beyond 2020." *The Lancet Global Health*, vol. 9, no. 4, 2021, pp. e489-e551.

[5] Litjens, G., et al. "A survey on deep learning in medical image analysis." *Medical Image Analysis*, vol. 42, 2017, pp. 60-88.

[6] Wang, X., et al. "Residual attention network for image classification." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 3156-3164.

[7] Centers for Disease Control and Prevention (CDC). "Conjunctivitis (Pink Eye)." [Online]. Available: https://www.cdc.gov/conjunctivitis/index.html. [Accessed: Dec. 27, 2024].

[8] Health.com. "Cataracts: Overview." [Online]. Available: https://www.health.com/cataracts-overview-7376241. [Accessed: Dec. 27, 2024].

[9] Mayo Clinic. "Sty: Symptoms and Causes." [Online]. Available: https://www.mayoclinic.org/diseases-conditions/sty/symptoms-causes/syc-20378017#causes. [Accessed: Dec. 27, 2024].

[10] Mayo Clinic. "Pink Eye (Conjunctivitis): Symptoms and Causes." [Online]. Available: https://www.mayoclinic.org/diseases-conditions/pink-eye/symptoms-causes/syc-20376355. [Accessed: Dec. 27, 2024].

[11] Healthline. "Slit Lamp Exam: Purpose, Procedure and Results." [Online]. Available: https://www.healthline.com/health/slit-lamp-exam. [Accessed: Dec. 27, 2024].

[12] Bovik, A. *The Essential Guide to Video Processing*. 2nd ed., Academic Press, 2013.

[13] Feichtenhofer, C., et al. "Temporal modeling in convolutional neural networks for video understanding." *International Journal of Computer Vision*, vol. 128, no. 1, 2020, pp. 67-87.

[14] Birchfield, T. A. "Noise Reduction Techniques for Video Processing." *Proceedings of IEEE International Conference on Multimedia and Expo (ICME)*, 2005, pp. 654–657.

[15] Zhao, Z., et al. "Object Detection with Deep Learning: A Review." *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, 2019, pp. 3212–3232.

[16] Viola, P., and Jones, M. J. "Robust real-time object detection." *International Journal of Computer Vision*, 2004.

[17] Viola, P., and Jones, M. J. "An evaluation study of face detection by Viola-Jones algorithm." *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 5, 2020, pp. 38–46.

[18] Bradski, G. "The OpenCV Library." *Dr. Dobb's Journal of Software Tools*, 2000.

[19] Zhang, Z., et al. "MTCNN++: A CNN-based face detection algorithm inspired by MTCNN." *The Visual Computer*, vol. 39, no. 3, 2023, pp. 1–10.

[20] Singh, A., et al. "Evaluating Dlib, Haar Cascade, and MTCNN face detectors." *Proceedings of the International Conference on Machine Vision and Image Processing (MVIP)*, 2022, pp. 112–118.

[21] Redmon, J., et al. "A comparative study of MTCNN, Viola-Jones, SSD, and YOLO face detection algorithms." *IEEE Access*, vol. 10, 2022, pp. 50675–50689.

[22] Richards, B. D., et al. "Detection of leukocoria using a smartphone-based application: CRADLE white reflex app." *Journal of the American Association for Pediatric Ophthalmology and Strabismus (AAPOS)*, vol. 23, no. 6, 2019, pp. 307.e1-307.e5. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/31616792/. [Accessed: Dec. 27, 2024].

[23] AEYE Health. "AEYE Health - Transforming eye screening with AI." [Online]. Available: https://www.aeyehealth.com/. [Accessed: Dec. 27, 2024].

[24] MD EyeCare. "MD EyeCare." [Online]. Available: https://www.mdeyecare.org/. [Accessed: Dec. 27, 2024].

[25] Varadarajan, R., et al. "Deep learning for detecting diabetic retinopathy from eye fundus images." *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019, pp. 64-72.

[26] Lecun, Y., Bengio, Y., and Hinton, G. "Deep learning." *Nature*, vol. 521, no. 7553, 2015, pp. 436-444.

[27] Kaggle. "Kaggle: Your Machine Learning and Data Science Community." [Online]. Available: https://www.kaggle.com/. [Accessed: Dec. 27, 2024].

[28] Roboflow. "Roboflow: Accelerating Computer Vision Development." [Online]. Available: https://www.roboflow.com/. [Accessed: Dec. 27, 2024].

[29] Perez, L., and Wang, J. "The Effectiveness of Data Augmentation in Image Classification." *arXiv preprint*, arXiv:1712.04621, 2017.

[30] GeeksforGeeks. "VGG-16 CNN Model." [Online]. Available: https://www.geeksforgeeks.org/vgg-16-cnn-model/. [Accessed: Dec. 27, 2024].

[31] He, K., et al. "Deep Residual Learning for Image Recognition." *arXiv preprint*, arXiv:1512.03385, 2015. [Online]. Available: https://arxiv.org/abs/1512.03385. [Accessed: Dec. 27, 2024].

[32] Szegedy, C., et al. "Rethinking the Inception Architecture for Computer Vision." *arXiv preprint*, arXiv:1512.00567, 2015. [Online]. Available: https://arxiv.org/abs/1512.00567. [Accessed: Dec. 27, 2024].

[33] Huang, G., et al. "Densely Connected Convolutional Networks." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4700–4708. [Online]. Available: https://arxiv.org/abs/1608.06993. [Accessed: Dec. 27, 2024].

[34] Rahim, M. S. M., et al. "Is Attention all You Need in Medical Image Analysis? A Review." *IEEE Access*, vol. 10, 2022, pp. 70650–70665. [Online]. Available: https://ieeexplore.ieee.org/document/10376277. [Accessed: Dec. 27, 2024].

[35] Zhang, Y., et al. "Residual Dense Network Based on Channel-Spatial Attention for the Scene Classification of a High-Resolution Remote Sensing Image." *Remote Sensing*, vol. 12, no. 11, 2020, p. 1887. [Online]. Available: https://www.mdpi.com/2072-4292/12/11/1887. [Accessed: Dec. 27, 2024].

[36] Li, H., et al. "Channel-Attention-Based DenseNet Network for Remote Sensing Image Scene Classification." *IEEE Geoscience and Remote Sensing Letters*, vol. 17, no. 8, 2020, pp. 1351–1355. [Online]. Available: https://ieeexplore.ieee.org/document/9141394. [Accessed: Dec. 27, 2024].

[37] Python Software Foundation. "Python Programming Language – Official Website." [Online]. Available: https://www.python.org/. [Accessed: Dec. 27, 2024].

[38] Ramírez, S. "FastAPI – Modern, Fast (High-Performance) Web Framework for Python 3.7+." *FastAPI Documentation*, 2024. [Online]. Available: https://fastapi.tiangolo.com/. [Accessed: Dec. 27, 2024].

[39] Meta Platforms, Inc. "React – A JavaScript Library for Building User Interfaces." *React.dev*, 2024. [Online]. Available: https://react.dev/. [Accessed: Dec. 27, 2024].