**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Чорноморський національний університет імені Петра Могили**

**Факультет комп'ютерних наук**

**Кафедра інтелектуальних інформаційних систем**

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інтелектуальних інформаційних систем

_____Юрій КОНДРАТЕНКО

«____»_____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

# ІНТЕЛЕКТУАЛЬНА СИСТЕМА РОЗПІЗНАВАННЯ ХВОРОБ ОКА З ВИКОРИСТАННЯМ НЕЙРОННИХ МЕРЕЖ

Спеціальність 122 Комп'ютерні науки

Освітня програма «Комп'ютерні науки»

***Здобувач*** _____ Дмитро ГАВРИЛЮК

«___»_____ 2025 р.

***Керівник*** канд. техн. наук, доцент _____ Євген СІДЕНКО

«___»_____ 2025 р.

**Миколаїв – 2025**

Чорноморський національний університет імені Петра Могили

(повне найменування закладу вищої освіти)

| | |
|---|---|
| Факультет | Комп’ютерних наук |
| Кафедра | Інтелектуальних інформаційних систем |
| Рівень вищої освіти | Перший (бакалаврський) |
| Освітній ступень | Бакалавр |
| Спеціальність | 122 Комп’ютерні науки |
| Освітня програма | Комп’ютерні науки |

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем

_____ Юрій КОНДРАТЕНКО

«____» _____ 2025 р.

**ЗАВДАННЯ**
**на кваліфікаційну роботу здобувача**

**Гаврилюка Дмитра Валерійовича**

(прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «Інтелектуальна система розпізнавання хвороб ока з використанням нейронних мереж»

Керівник роботи: Сіденко Євген Вікторович, доцент кафедри інтелектуальних інформаційних систем, канд. техн. наук, доцент.

Затверджена наказом ЧНУ ім. Петра Могили від «18» листопада 2024 р. № 321.

2. Строк представлення кваліфікаційної роботи «___» _____ 2025 р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні: система, здатна класифікувати хвороби ока за допомогою знімків з офтальмоскопії очного дна, зберігання даних пацієнтів та їх діагнозів, пошук діагнозів за даними пацієнта; початковими даними є медичні знімків з офтальмоскопії очного дна.

4. Перелік питань, що підлягають розробці: аналіз сучасного стану задачі діагностування хвороб з використанням нейронних мереж; огляд існуючих методів

машинного навчання; вибір та налаштування відповідної архітектури нейронної мережі для класифікації хвороб ока.

5. Перелік графічних матеріалів: презентація.

**Керівник роботи** _____ Євген СІДЕНКО
*(Особистий підпис)* *(Власне ім'я ПРІЗВИЩЕ)*

**Здобувач** _____ Дмитро ГАВРИЛЮК
*(Особистий підпис)* *(Власне ім'я ПРІЗВИЩЕ)*

Дата видачі завдання «18» грудня 2024 р.

**КАЛЕНДАРНИЙ ПЛАН**
**кваліфікаційної роботи**

Тема: Інтелектуальна система розпізнавання хвороб ока з використанням нейронних мереж

| № | Найменування роботи | Початок | Закінчення | Примітки |
|---|---|---|---|---|
| 1 | Отримання завдання на виконання КР | 18.12.2024 | 20.12.2024 | |
| 2 | Аналіз предметної області та постановка задачі | 21.12.2024 | 30.01.2025 | |
| 3 | Огляд літературних джерел за темою кваліфікаційної роботи, зокрема огляд публікацій та аналогічних систем, щодо діагностування хвороб ока | 31.01.2025 | 01.03.2025 | |
| 4 | Аналіз існуючих архітектур штучних нейронних мереж, що застосовуються для класифікації медичних зображень | 02.03.2025 | 01.04.2025 | |
| 5 | Реалізація обраних технологій з аналізом отриманих результатів | 02.04.2025 | 15.05.2025 | |
| 6 | Перший попередній захист КР на засіданні комісії кафедри | 16.05.2025 | 16.05.2025 | |
| 7 | Корегування роботи за результатами попереднього захисту | 29.05.2025 | 29.05.2025 | |
| 8 | Другий попередній захист КР на засіданні комісії кафедри | 30.05.2025 | 30.05.2025 | |
| 9 | Доробка та остаточне оформлення КР | 31.05.2025 | 10.06.2025 | |
| 10 | Подання КР, її електронної копії та інших документів (відгуку, рецензії) до захисту | 11.06.2025 | 17.06.2025 | |

**Керівник роботи** _____ Євген СІДЕНКО
*(Особистий підпис)* *(Власне ім'я ПРІЗВИЩЕ)*

**Здобувач** _____ Дмитро ГАВРИЛЮК
*(Особистий підпис)* *(Власне ім'я ПРІЗВИЩЕ)*

Дата складання календарного плану
«29» січня 2025 р.

# АНОТАЦІЯ

до кваліфікаційної роботи
здобувача групи 402 ЧНУ ім. Петра Могили

**Гаврилюка Дмитра Валерійовича**

на тему: **“ ІНТЕЛЕКТУАЛЬНА СИСТЕМА РОЗПІЗНАВАННЯ ХВОРОБ ОКА З ВИКОРИСТАННЯМ НЕЙРОННИХ МЕРЕЖ”**

**Актуальність** даного дослідження полягає у необхідності підвищення ефективності для своєчасної діагностики захворювань очей. Що вирішальне значення для збереження зору. Так як багато офтальмологічних хвороб, зокрема глаукома, діабетична ретинопатія, вікова макулодистрофія, на початкових етапах можуть не мати явних симптомів. Без своєчасного виявлення та лікування ці хвороби поступово прогресують, що може призвести до незворотної втрати зору.

**Об’єктом** роботи є процес діагностики хвороб ока за допомогою медичних зображень.

**Предметом** роботи є нейронні мережі для розпізнавання медичних зображень та автоматизації процесу діагностики захворювань ока.

**Метою** роботи є діагностування хвороб ока за допомогою нейронних мереж.

В результаті виконання роботи було досліджено сучасні методи діагностики хвороб ока. Створено інтелектуальну систему для розпізнавання найбільш розповсюджених хвороб ока з використанням нейронних мереж, протестовано її ефективність а також розроблено інтерфейс користувача.

Дана робота складається з чотирьох розділів. У першому розділі досліджено хвороби ока, методи їх діагностики та сучасні системи. Другий розділ присвячений опису математичних моделей та методів, що використані у роботі. У третьому розділі наведено результати проєктування системи та підготувку даних. В четвертому — розробка системи, її тестування та інформаційні діаграми. Загальний обсяг роботи — 76 сторінок. Кваліфікаційна робота містить 1 додаток, 45 рисунків, 2 таблиці і 27 джерел посилання.

**Ключові слова**: нейроні мережі, хвороби ока, класифікація, ResNet50.

# ABSTRACT

to the qualification work by the student of the group 402 of Petro Mohyla Black Sea National University

**Havryliuk Dmytro**

## “INTELLIGENT SYSTEM FOR RECOGNISING EYE DISEASES USING NEURAL NETWORKS”

The relevance of this study lies in the need to increase the efficiency of timely diagnosis of eye diseases. Which is crucial for preserving vision. Since many ophthalmological diseases, in particular glaucoma, diabetic retinopathy, age-related macular degeneration, may not have obvious symptoms in the initial stages.

Without timely detection and treatment, these diseases gradually progress, which can lead to irreversible vision loss.

A object of the work is the process of diagnosing eye diseases using medical images.

A subject of the work is neural networks for recognizing medical images and automating the process of diagnosing eye diseases.

A purpose of the work is to diagnose eye diseases using neural networks.

As a result of the work, modern methods for diagnosing eye diseases were investigated. An intelligent system for recognizing the most common eye diseases using neural networks This work consists of four chapters. The first chapter examines eye diseases, their diagnostic methods, and modern systems. The second chapter is devoted to the description of mathematical models and methods used in the work.

The third section presents the results of system design and data preparation. The fourth section presents system development, testing, and information diagrams.

The overall scope of the work is 76 pages. Thesis contains 1 application, 45 figures, 2 tables and 27 references in it.

**Key words:** neural networks, eye diseases, classification, ResNet50

# ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ ..... 4

ВСТУП..... 5

1 АНАЛІЗ СФЕРИ ДІАГНОСТИКИ ХВОРОБ ОКА. ПОСТАНОВКА ЗАДАЧІ ..... 6

1.1 Огляд захворювань очей..... 6

1.2 Методи діагностики хвороб ока ..... 14

1.3 Аналіз наявних аналогів ..... 15

1.4 Постановка задачі..... 17

Висновки до розділу 1 ..... 18

2 МЕТОДИ ТА ІНСТРУМЕНТИ ДЛЯ ДІАГНОСТУВАННЯ ХВОРОБ ОКА ЗА ДОПОМОГОЮ НЕЙРОННИХ МЕРЕЖ ..... 19

2.1 Методи машинного навчання ..... 19

2.2 Вибір мови програмування ..... 23

2.3 Архітектура нейронної мережі ..... 24

Висновки до розділу 2 ..... 28

3 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДІАГНОСТУВАННЯ ХВОРОБ ОКА ..... 30

3.1 Опис вхідних даних та структури системи ..... 30

3.2 Аналіз даних ..... 35

3.3 Попередня обробка..... 37

Висновки до розділу 3 ..... 41

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ..... 43

4.1 Створення та навчання моделі ResNet50 ..... 43

4.2 Підбір параметрів та оцінювання моделі..... 47

4.3 Інтерфейс та БД ..... 52

4.3 Тестування ..... 55

Висновки до розділу 4 ..... 59

ВИСНОВКИ..... 60

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ..... 61

ДОДАТОК А Лістинг коду..... 65

# СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ВМД — Вікова макулярна дегенерація

КТ — Комп'ютерна томографія

ОКТ — Оптична когерентна томографія

УЗД — Ультразвукова діагностика

ШІ — Штучний інтелект

CNN — Convolutional Neural Networks

FDA — Food and Drug Administration

GUI — Graphical User Interface

# ВСТУП

Сучасна медицина стрімко розвивається завдяки впровадженню передових технологій, серед яких особливу роль відіграють штучний інтелект і методи машинного навчання. Однією з актуальних проблем залишається своєчасне й точне виявлення складних захворювань, зокрема офтальмологічних, що суттєво впливають на якість життя. Хвороби, як-от катаракта, глаукома та ретинопатія, на ранніх стадіях часто не мають явних симптомів, що ускладнює їхню діагностику та лікування. Застосування нейронних мереж дає змогу покращити точність діагностики, своєчасно виявляти патології та запобігати серйозним ускладненням.

Хоча традиційні методи діагностики залишаються ефективними, вони потребують значного часу й залежать від досвіду лікаря. У регіонах із браком кваліфікованих спеціалістів і високим ризиком людських помилок це створює додаткові труднощі. В таких умовах інтелектуальні системи стають важливим інструментом: вони дозволяють автоматизувати діагностику і підвищити її точність завдяки потужностям нейронних мереж.

**Об'єкт роботи** — процес діагностики хвороб ока за допомогою медичних зображень.

**Предмет роботи** — нейронні мережі для розпізнавання медичних зображень та автоматизації процесу діагностики захворювань ока.

**Метою роботи** — діагностування хвороб ока за допомогою нейронних мереж.

# 1 АНАЛІЗ СФЕРИ ДІАГНОСТИКИ ХВОРОБ ОКА. ПОСТАНОВКА ЗАДАЧІ

## 1.1 Огляд захворювань очей

Захворювання очей охоплюють широкий спектр патологій, що вражають різні структури зорового аналізатора — від рогівки до сітківки та зорового нерва. Вони можуть бути як вродженими, так і набутими, а також виникати внаслідок інфекцій, травм, вікових змін або системних захворювань. Очні хвороби є серйозною медико-соціальною проблемою, оскільки можуть призвести до значного зниження якості життя, втрати працездатності та повної сліпоти. Найбільш поширеними є такі патології, як катаракта, глаукома, макулодистрофія, діабетична ретинопатія та рефракційні порушення [1].

### 1.1.1 Діабетична ретинопатія

Діабетична ретинопатія [2] — це ускладнення діабету, яке вражає очі. Вона спричинена пошкодженням кровоносних судин світлочутливої тканини в задній частині ока (сітківки).

Спочатку діабетична ретинопатія може не викликати жодних симптомів або мати лише легкі проблеми із зором. Але вона може призвести до сліпоти. Цей стан може розвинутися у будь-кого, хто має діабет 1 або 2 типу. Чим довше у вас діабет і чим менше контролюється рівень цукру в крові, тим більша ймовірність розвитку цього ускладнення для очей.

На ранніх стадіях діабетичної ретинопатії симптоми можуть відсутня. У міру прогресування захворювання можуть з’явитися:

- плями або темні тяжі, що плавають у вашому полі зору;
- затуманений зір;
- коливання зору;
- темні або порожні ділянки у полі зору;
- втрата зору.

Причинами захворювання є цукрове блокування крихітних кровоносних

судин, які живлять сітківку ока, припиняючи її кровопостачання. В результаті око намагається виростити нові кровоносні судини. Але ці нові кровоносні судини не розвиваються належним чином і можуть легко протікати. Існує два типи діабетичної ретинопатії.

Рання діабетична ретинопатія — це більш поширена форма, яка називається непроліферативною діабетичною ретинопатією. При непроліферативній діабетичній ретинопатії стінки кровоносних судин сітківки слабшають. На стінках дрібних судин з'являються крихітні випинання, з яких іноді витікає рідина і кров у сітківку. Більші судини сітківки також можуть почати розширюватися і ставати нерівномірними в діаметрі.

Прогресуюча діабетична ретинопатія. Діабетична ретинопатія може прогресувати до більш важкого типу, відомого як проліферативна діабетична ретинопатія. При цьому типі пошкоджені кровоносні судини закриваються, спричиняючи ріст нових, аномальних кровоносних судин у сітківці. Ці нові кровоносні судини є крихкими і можуть просочуватися в прозору, желеподібну речовину, яка заповнює центр ока, а саме склоподібне тіло (див. рис. 1.1).

Врешті-решт, рубцева тканина від росту нових кровоносних судин може призвести до того, що сітківка відокремиться від задньої частини ока. Якщо нові кровоносні судини перешкоджають нормальному відтоку рідини з ока, в очному яблукі може підвищуватися тиск. Цей тиск може пошкодити зоровий нерв, що призведе до глаукоми.

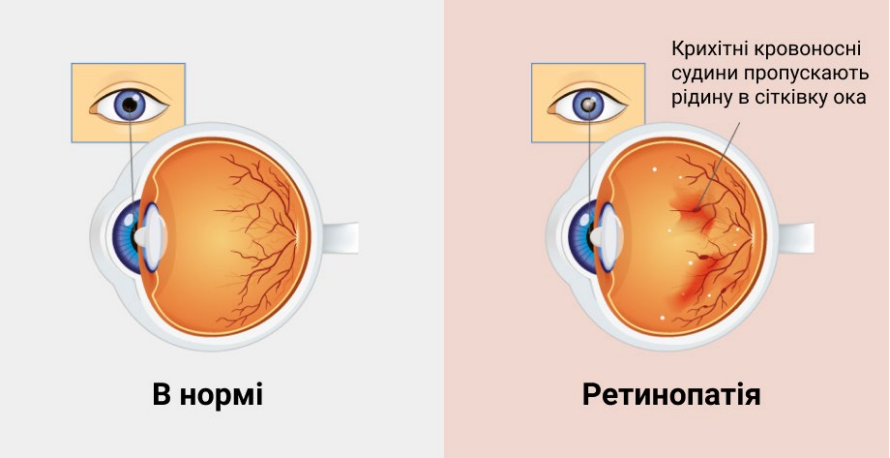


Рисунок 1.1 – Діабетична ретинопатія [2]

### 1.1.2 Глаукома

Глаукома [3] — це захворювання очей, яке пошкоджує зоровий нерв. Це пошкодження може призвести до втрати зору або сліпоти. Зоровий нерв передає візуальну інформацію від ока до мозку і є життєво важливим для хорошого зору. Пошкодження зорового нерва часто пов'язане з високим внутрішньоочним тиском. Але глаукома може виникнути навіть при нормальному внутрішньоочниму тиску.

Глаукома зазвичай не викликає жодних симптомів на початковому етапі. Вона, як правило, розвивається повільно протягом багатьох років і спочатку вражає краї зору, тобто периферійний зір як на рис. 1.2.

Рисунок 1.2 – Втрата зору при глаукомі [4]

З цієї причини багато людей не усвідомлюють, що у них глаукома, і часто її виявляють лише під час звичайної перевірки зору. Зазвичай уражаються обидва ока, хоча зір одного може бути гірший.

Існує кілька різних типів глаукоми:

– найпоширеніший називається первинною відкритокутовою глаукомою, яка розвивається повільно протягом багатьох років. Це пов'язано з тим, що дренажні канали в оці з часом поступово закупорюються (див. рис. 1.3);

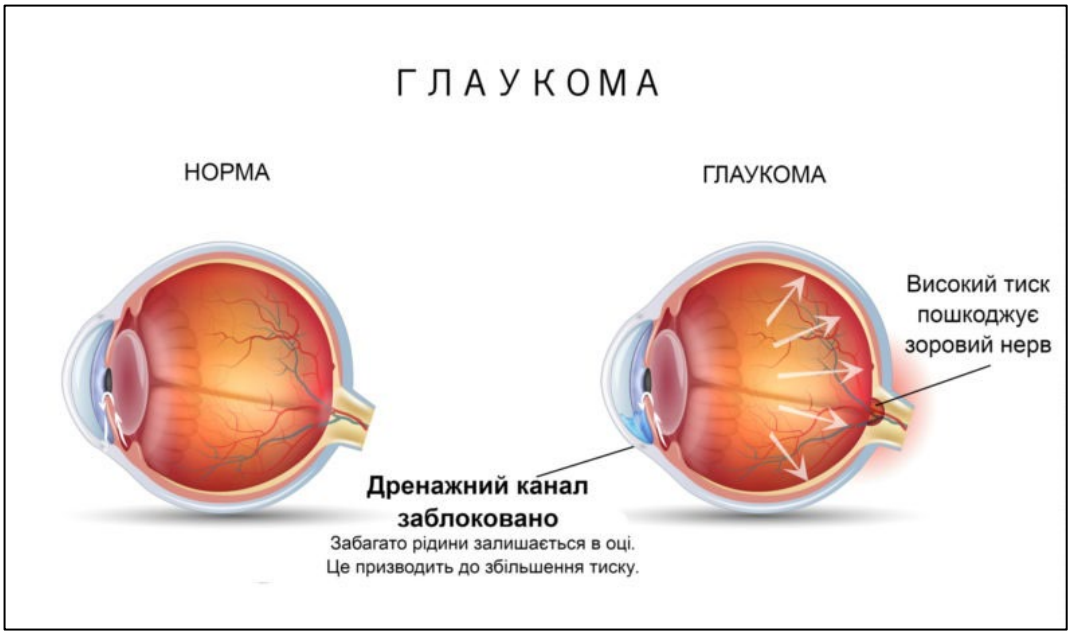


Рисунок 1.3 – Відкритокутова глаукома [5]

– гостра закритокутова глаукома — рідкісніший тип, спричинений раптовою закупоркою дренажних каналів ока, що швидко підвищити внутрішньоочний тиск. В результаті райдужка перекриває дренажний кут, ніби листок паперу в стічному отворі умивальника. Закритокутова глаукома може призвести до втрати зору, якщо не розпочати лікування негайно;

– вторинна глаукома — викликана основним захворюванням очей, таким як запалення ока (увеїт);

– дитяча глаукома (вроджена глаукома) - рідкісний тип, який трапляється у дуже маленьких дітей, спричинений аномалією розвитку ока.

### 1.1.3. Катаракта

Катаракта [6] — це помутніння (непрозорість) кришталика ока, яке спричиняє прогресуючу, безболісну втрату зору. Більшість катаракт розвивається у людей старше 55 років, але іноді вони трапляються у немовлят та маленьких дітей або внаслідок травми, впливу ультрафіолетового випромінювання, променевої терапії, хронічних захворювань (наприклад, діабету) чи тривалого прийому певних ліків, зокрема кортикостероїдів. Зазвичай катаракта розвивається на обох очах, хоча не завжди симетрично — один кришталик може мутніти швидше за інший.

Кришталик розташований всередині ока за райдужною оболонкою, кольоровою частиною ока. Зазвичай він фокусує світло на сітківці, яка перетворює світлові імпульси на електричні сигнали та передає їх через зоровий нерв до мозку, формуючи зображення. Однак, якщо кришталик помутнів через катаракту, світло розсіюється або блокується, тому кришталик більше не може правильно його фокусувати. Це спричиняє проблеми із зором, такі як затуманення зображення, зниження контрастності, підвищена чутливість до світла, двоїння в оці або необхідність частішої заміни окулярів, як показано на рис. 1.4.

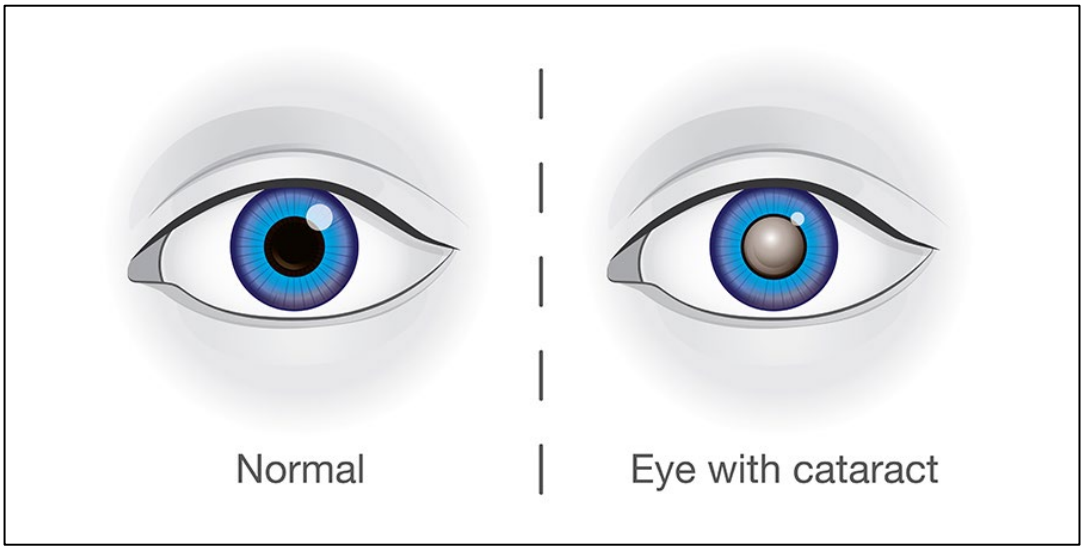


Рисунок 1.4 – Помутніння кришталика [7]

Кришталик складається переважно з білків та води. Помутніння кришталика відбувається через структурні зміни в білках та волокнах кришталика, які з віком або під впливом пошкоджувальних чинників втрачають свою прозорість і починають злипатися, утворюючи мутні ділянки.

Кришталик має шарувату будову, подібну до цибулі. Зовнішній шар — це капсула, яка захищає внутрішні структури та підтримує форму. Шар всередині капсули — це кора, що складається з новіших волокон, а внутрішній центральний шар — ядро, утворене старішими, більш щільними волокнами. Катаракта може розвинутися в будь-якій з цих ділянок: ядерна катаракта (в ядрі), кортикальна (в корі) або субкапсулярна (під капсулою), і кожна з них має свої особливості перебігу та симптоми.

### 1.1.4. Вікова макулярна дегенерація

Вікова макулярна дегенерація (ВМД) [8] — це хронічне захворювання очей, яке вражає макулу — центральну частину сітківки, відповідальну за чітке пряме бачення. Це одна з провідних причин втрати зору серед людей віком від 55 років і старше.

ВМД виникає, коли з віком макула пошкоджується, що призводить до розмиття центрального зору. Це ускладнює читання, розпізнавання облич, водіння та виконання дрібної роботи. Захворювання не призводить до повної сліпоти, але суттєво впливає на якість життя.

Існує два основних типи ВМД:

– суха ВМД (атрофічна): найпоширеніший тип, який прогресує повільно. Виникає через витончення макули з віком. Має три стадії: ранню, проміжну та пізню;

– волога ВМД (неоваскулярна): менш поширена, але більш агресивна форма. Характеризується ростом аномальних кровоносних судин під макулою, що може призвести до швидкої втрати зору. Суха ВМД може перейти у вологу форму на пізніх стадіях. Раннє виявлення та своєчасне лікування можуть уповільнити

прогресування захворювання та зберегти зір.

### 1.1.5. Артеріальна гіпертензія

**Артеріальна гіпертензія** [9] — це стан, за якого артеріальний тиск у судинах постійно підвищений, що створює додаткове навантаження на серце та кровоносну систему. Це одне з найпоширеніших серцево-судинних захворювань у світі та ключовий фактор ризику для розвитку інсульту, серцевого нападу, серцевої недостатності, ураження нирок та зору. Незважаючи на свою небезпеку, гіпертензія часто тривалий час протікає безсимптомно, тому її називають «тихим убивцею».

З віком ризик розвитку гіпертензії зростає, однак вона дедалі частіше зустрічається і в молодших людей, особливо внаслідок малорухомого способу життя, неправильного харчування, надмірного споживання солі, алкоголю та хронічного стресу. Також важливу роль відіграє спадковість: якщо один з батьків мав гіпертензію, ймовірність її розвитку у нащадків значно підвищується.

Причини підвищення тиску можуть бути первинними, тобто виникати самостійно, або вторинними — бути наслідком іншого захворювання, ендокринних порушень або дії певних лікарських засобів. Вторинна гіпертензія частіше виявляється у молодих людей і зазвичай має раптовий, різкий початок.

Постійно підвищений тиск пошкоджує стінки артерій, зменшуючи їхню еластичність і сприяючи утворенню атеросклеротичних бляшок. У результаті серцю доводиться працювати інтенсивніше, щоб перекачувати кров, що з часом призводить до гіпертрофії серцевого м’яза і підвищеного ризику серцевої недостатності. Порушення кровопостачання мозку може стати причиною інсульту, а ураження судин нирок — хронічної ниркової недостатності. Також високий тиск впливає на сітківку ока, викликаючи гіпертонічну ретинопатію, яка може погіршити зір.

Попри хронічний характер гіпертензії, вона добре піддається контролю. Люди, які дотримуються рекомендацій лікаря, регулярно приймають ліки та стежать за тиском, можуть жити повноцінним, активним життям без ускладнень.

### 1.1.6. Патологічна міопія

Патологічна міопія [10] (також відома як міопічна дегенерація) — це тяжка форма короткозорості, яка супроводжується структурними змінами в задньому сегменті ока внаслідок надмірного подовження очного яблука. На відміну від звичайної короткозорості, яка зазвичай стабілізується в підлітковому віці, патологічна міопія часто продовжує прогресувати в дорослому житті. Цей стан може призвести до серйозних порушень зору та є однією з провідних причин сліпоти в усьому світі.

Згідно з прогнозами, до 2050 року майже половина населення світу страждатиме на міопію, а близько 10% матимуть високу міопію, що підвищує ризик розвитку патологічної міопії. Висока міопія зазвичай визначається як рефракційна помилка понад -6 діоптрій або аксіальна довжина ока більше 26–26,5 мм .

Захворювання не завжди супроводжується різким падінням зору на ранніх етапах, що ускладнює його вчасне виявлення. Одним із найнебезпечніших проявів патологічної міопії є розвиток неоваскуляризації — коли організм починає формувати аномальні кровоносні судини у спробі компенсувати нестачу кисню в тканинах сітківки. Ці судини нестійкі, легко розриваються, що призводить до крововиливів і утворення рубців, які остаточно псують зорову інформацію, яку сітківка передає мозку.

Крім того, у пацієнтів з патологічною міопією нерідко спостерігається розвиток макулярних уражень — зокрема фовеошизису (розщеплення шару сітківки в макулі), куполоподібної макули, а також різноманітних типів макулярного відшарування. Ці стани вимагають точного моніторингу за допомогою оптичної когерентної томографії, оскільки своєчасна діагностика значно підвищує ефективність лікування. Варто зазначити, що патологічна міопія може також ускладнюватись відшаруванням сітківки — грізним станом, що загрожує повною втратою зору, якщо не втрутитися хірургічно.

## 1.2 Методи діагностики хвороб ока

Для діагностики захворювань ока використовуються сучасні методи дослідження. Методи діагностики хвороб ока поділяються на клінічні, інструментальні та лабораторні. Вибір конкретного методу залежить від симптомів, підозрюваної патології та загального стану пацієнта.

До клінічних методів належать візуальний огляд, збір анамнезу та основні офтальмологічні тести. Найбільш поширеним є перевірка гостроти зору за допомогою таблиць Сивцева або Ландольта. Також застосовуються методи визначення кольорового зору (таблиці Ішихари як на рис. 2.1.), тестування полів зору та перевірка рефракції.

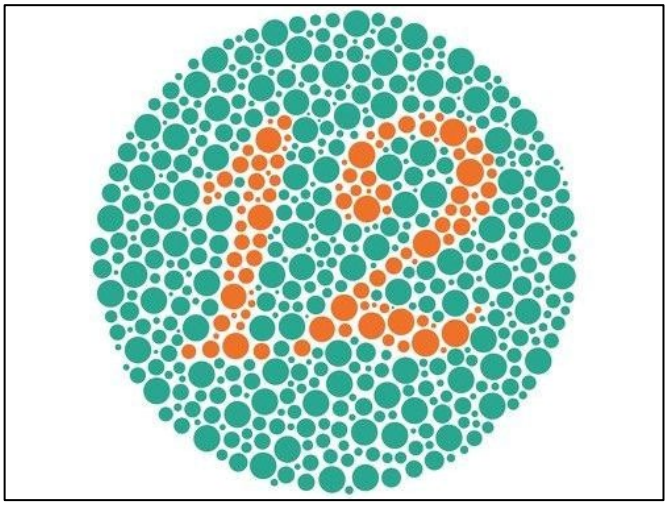

Рисунок 1.5 – Приклад тесту Ішихари

Інструментальна діагностика охоплює широкий спектр сучасних технологій, які описані нижче.

**Офтальмоскопія** — це метод огляду внутрішніх структур ока за допомогою офтальмоскопа. Лікар оцінює стан сітківки, диска зорового нерва та судинного русла. Метод дозволяє виявити ознаки глаукоми, діабетичної ретинопатії, дегенеративних змін сітківки та інших патологій. Він є основним у рутинному офтальмологічному огляді.

**Тонометрія** — це процедура вимірювання внутрішньоочного тиску, що є критично важливим для виявлення глаукоми. Підвищений тиск може свідчити про ризик ураження зорового нерва. Існують контактні та безконтактні методи тонометрії, останні є безболісними та швидкими.

**Біомікроскопія** — це мікроскопічне дослідження переднього відрізка ока з використанням вузького пучка світла. Метод дає змогу детально оглянути рогівку, кон'юнктиву, передню камеру ока, кришталик та райдужку. Завдяки високій точності лікар може виявити запальні, дегенеративні чи травматичні зміни. Біомікроскопія є основним методом при діагностиці катаракти, кератитів та увеїтів.

**ОКТ** (оптична когерентна томографія) — неінвазивне дослідження, що створює пошарове зображення сітківки, зорового нерва та макули. Завдяки високій роздільній здатності ОКТ дозволяє виявляти найменші структурні зміни, які не видно під час офтальмоскопії. Метод широко застосовується для моніторингу прогресування глаукоми, вікової макулодистрофії та набряку сітківки.

**УЗД ока** (сонографія) — це метод дослідження ока за допомогою ультразвукових хвиль, який дозволяє візуалізувати внутрішні структури, особливо за умов оптичної непрозорості. УЗД ефективне при діагностиці відшарування сітківки, пухлин, крововиливів та сторонніх тіл усередині ока.

Останніми роками активно розвиваються цифрові та штучно-інтелектуальні технології. Зокрема, автоматизований аналіз зображень сітківки, машинне навчання для виявлення патологій на ранніх етапах та телемедицина. Ці методи підвищують доступність діагностики та точність виявлення змін.

### 1.3 Аналіз наявних аналогів

Сучасні інтелектуальні системи відіграють дедалі важливішу роль у діагностиці офтальмологічних захворювань, особливо завдяки високій точності, швидкості обробки великих обсягів даних та здатності виявляти патології на ранніх стадіях. На основі штучного інтелекту, машинного навчання та комп'ютерного зору створюються моделі, здатні аналізувати офтальмологічні знімки з точністю,

що порівнянна з людиною-фахівцем.

Однією з найуспішніших сфер застосування є автоматичне виявлення діабетичної ретинопатії. Наприклад, системи на основі глибоких нейронних мереж здатні класифікувати ступінь ураження сітківки за знімками очного дна. Також активно впроваджуються алгоритми для діагностики глаукоми, вікової макулодистрофії та відшарування сітківки.

Водночас важливими залишаються питання стандартизації даних, прозорості алгоритмів, клінічної валідації та етичних аспектів використання ШІ в медицині. Інтелектуальні системи не замінюють лікаря, але стають ефективним інструментом підтримки клінічних рішень, сприяючи швидшій та точнішій діагностиці захворювань ока.

**IDx-DR** [11] — це перша інтелектуальна система, схвалена FDA (США) для автономної діагностики діабетичної ретинопатії. Вона не потребує участі лікаря: пацієнту роблять знімок очного дна, а алгоритм самостійно аналізує зображення і видає висновок про наявність або відсутність патології. IDx-DR активно використовується в загальній медичній практиці, зокрема в сімейній медицині, для масового скринінгу пацієнтів з діабетом.

**DeepMind (Eye)** [12] розроблена у співпраці з британською системою охорони здоров’я (NHS), ця система аналізує оптичні когерентні томограми (ОКТ) сітківки. Вона здатна виявити понад 50 захворювань, включаючи макулодистрофію та набряк сітківки.

**Eyenuk EyeArt** [13] сертифікована в США та ЄС, система EyeArt спеціалізується на автоматичному виявленні діабетичної ретинопатії. Вона працює в режимі повної автономності: після завантаження знімків очного дна програма самостійно генерує детальний звіт про стан сітківки. EyeArt дозволяє проводити масовий скринінг пацієнтів без участі офтальмолога, що особливо корисно в умовах великого навантаження на медичні заклади.

**ARDA (Automated Retinal Disease Assessment).** [14] Система створена для використання в країнах, що розвиваються, де офтальмологічна допомога часто є

недоступною. ARDA виконує автоматичний аналіз зображень очного дна для виявлення ознак ретинопатії, глаукоми та макулодистрофії.

Таблиця 1.1 – Інтелектуальні системи у офтальмології

| Назва системи | Призначення | Технології | Особливості |
|---|---|---|---|
| **IDx-DR** | Виявлення діабетичної ретинопатії | Глибоке навчання, CNN | Перша система з дозволом FDA для автономної діагностики без участі лікаря |
| **Google DeepMind (Eye)** | Виявлення понад 50 патологій сітківки, макулодистрофії | Глибоке навчання, аналіз ОКТ-зображень | Показує діагноз і рекомендації, порівнянні з експертною думкою офтальмолога |
| **Eyenuk EyeArt** | Скринінг діабетичної ретинопатії | Машинне навчання та аналіз зображень | Отримала сертифікацію CE, FDA, автоматично генерує звіти для лікаря |
| **ARDA (Automated Retinal Disease Assessment)** | Діагностика патологій очного дна | Глибокі нейромережі | Розроблена для масштабного скринінгу у слаборозвинених країнах |

## 1.4 Постановка задачі

Загальна проблематика питання полягає в:

– розробці ефективних методів виявлення пошкоджень органів зору на основі медичних зображень з допомогою нейронної мережі;

– необхідності підвищення точності та розширення спектру методів діагностики офтальмологічних захворювань, таких як катаракта, глаукома, міопія тощо.

Актуальність даного дослідження полягає у необхідності підвищення ефективності для своєчасної діагностики захворювань очей. Що вирішальне значення для збереження зору. Так як багато офтальмологічних хвороб, зокрема глаукома, діабетична ретинопатія, вікова макулодистрофія, на початкових етапах можуть не мати явних симптомів. Без своєчасного виявлення та лікування ці хвороби поступово прогресують, що може призвести до незворотної втрати зору. Ось чому раннє виявлення є надзвичайно важливим:

– лікування на ранніх стадіях є значно ефективнішим, ніж на пізніх;
– збереження працездатності;
– зменшення витрат на лікування, профілактика та рання терапія дешевші, ніж тривале лікування ускладнень або інвалідність.

Завдяки сучасним технологіям, зокрема штучному інтелекту та нейронним мережам, стає можливим проводити масовий скринінг та виявляти патології навіть у віддалених або малозабезпечених регіонах, де бракує лікарів-офтальмологів.

**Висновки до розділу 1**

Розглянуто основні захворювання сітківки ока, які є одними з найпоширеніших причин втрати зору в усьому світі. Зокрема, проаналізовано особливості клінічного перебігу патологій. Було визначено, що більшість з них мають хронічний перебіг і часто не супроводжуються вираженими симптомами на початкових стадіях, що значно ускладнює своєчасну діагностику та лікування.

У зв’язку з цим особливе значення має застосування методів комп’ютерного зору, зокрема глибокого навчання, для виявлення та класифікації патологій на основі зображень очного дна. Ці технології забезпечують можливість автоматизованого аналізу медичних зображень з високим рівнем точності, що сприяє підвищенню ефективності раннього виявлення захворювань, зниженню витрат часу для лікарів та зменшенню ризику людських помилок. Автоматизовані системи також мають потенціал для впровадження у віддалених або малозабезпечених регіонах, де доступ до кваліфікованої офтальмологічної допомоги є обмеженим.

Таким чином, поєднання медичних знань із сучасними підходами в галузі комп’ютерного зору та штучного інтелекту відкриває нові перспективи в сфері офтальмології. Подальші дослідження в цьому напрямку є важливими для створення надійних систем ранньої діагностики, що здатні зменшити глобальний тягар очних захворювань та покращити якість життя пацієнтів.

# 2 МЕТОДИ ТА ІНСТРУМЕНТИ ДЛЯ ДІАГНОСТУВАННЯ ХВОРОБ ОКА ЗА ДОПОМОГОЮ НЕЙРОННИХ МЕРЕЖ

## 2.1 Методи машинного навчання

Машинне навчання відіграє ключову роль у розвитку сучасної медицини, особливо в галузі діагностики. Завдяки здатності аналізувати великі обсяги медичних даних та виявляти приховані закономірності, воно дозволяє покращити точність, швидкість і об’єктивність медичних рішень. Однією з перспективних задач, у якій застосовуються ці методи, є класифікація хвороб ока за фотографіями сітківки.

Для реалізації цієї задачі було обрано метод навчання з учителем, оскільки він найбільше відповідає її природі: ми маємо вхідні дані та відповідні діагнози (мітки), які дозволяють моделі навчатися на прикладах та надалі передбачати діагноз для нових, ще не бачених зображень.

Цей підхід забезпечує:

– контрольований процес навчання з відомим результатом (що критично важливо в медицині);

– можливість точного прогнозу, а не лише кластеризації чи групування;

– адаптацію під конкретні класи захворювань, що дозволяє створити спеціалізовану систему діагностики.

Навчання з учителем особливо ефективне, коли доступна якісна розмічена база даних, що і є основою в даному випадку.

**Нейронна мережа** [19] — це обчислювальна модель, що імітує структуру та принципи роботи біологічної нервової системи, зокрема мозку людини. Вона складається з великої кількості взаємозв’язаних елементів — нейронів, організованих у шари як на рис. 2.1

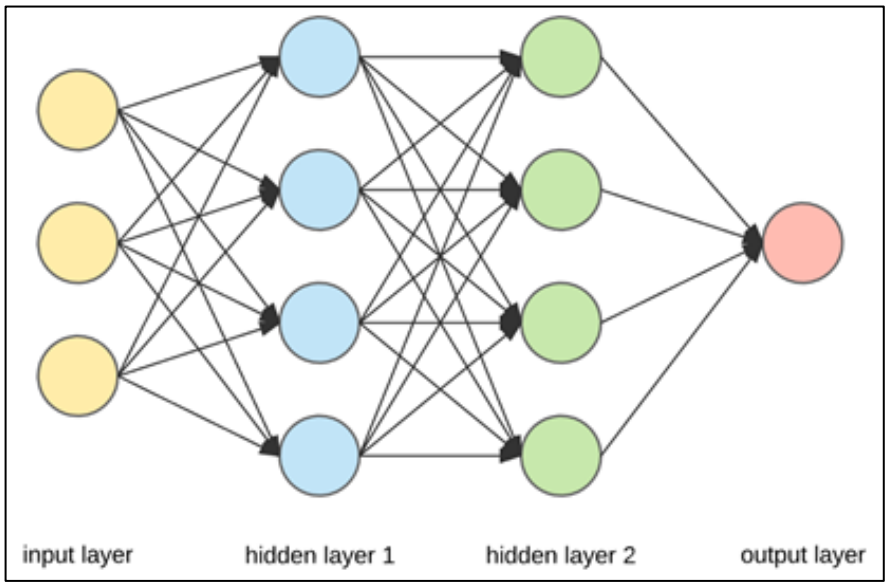


Рисунок 2.1 – Шари в нейронній мережі [19]

Нейронні мережі відкривають нові можливості для медичної діагностики, забезпечуючи вищу швидкість і точність порівняно з традиційними методами, такими як оцінка симптомів або базові медичні тести. Сучасні діагностичні системи, побудовані на основі нейронних мереж, спрямовані на автоматичне розпізнавання характерних ознак захворювань із вхідних даних.

Одним із найпоширеніших напрямів застосування нейронних мереж у медицині є аналіз медичних зображень, зокрема рентгенівських знімків. У таких випадках зазвичай застосовують глибокі нейронні мережі, зокрема конволюційні, які здатні самостійно виявляти візуальні ознаки, характерні для певних патологій.

**Глибоке навчання** [20] відіграє ключову роль у сучасних методах медичної діагностики завдяки здатності розпізнавати складні закономірності в великомасштабних і високовимірних медичних даних, особливо в зображеннях. Його головною перевагою є використання багаторівневих нейронних мереж, які самостійно виявляють значущі ознаки на зображеннях, що дозволяє уникнути трудомісткої ручної обробки.

Основні переваги застосування глибокого навчання в медицині:

– глибокі мережі здатні розпізнавати характерні риси на різних рівнях

абстракції — від простих геометричних елементів (ліній, контурів) до складних патологічних змін тканин;

– моделі демонструють високу чутливість і специфічність навіть при мінімальних змінах у зображеннях, що є критично важливим у контексті уникнення помилок у діагнозах;

– глибокі мережі здатні ефективно працювати з великими обсягами даних. Це робить їх надзвичайно корисними в умовах лікарень і діагностичних центрів, де щоденно обробляються сотні або тисячі медичних зображень;

– системи, що базуються на глибокому навчанні, можуть точно виділяти уражені області, що дозволяє лікарю точніше оцінити стан пацієнта та обрати оптимальну тактику лікування;

– можуть навчатися на великому архіві історичних клінічних даних для прогнозування перебігу хвороб і оцінки ризиків;

– глибокі мережі здатні покращувати власну продуктивність з часом, пристосовуючись до нових типів даних. Це підвищує точність діагностичних висновків і дозволяє зберігати актуальність систем в умовах постійного оновлення медичної інформації.

**Згорткові нейронні мережі** (Convolutional Neural Networks, або CNN) [16] — це різновид штучних нейронних мереж, спеціально адаптований для аналізу зображень. Завдяки своїй архітектурі CNN здатні автоматично виявляти характерні ознаки на зображеннях, що робить їх надзвичайно ефективними в задачах комп’ютерного зору, таких як класифікація, розпізнавання об’єктів і сегментація. У медичній діагностиці CNN використовуються для автоматичного виявлення візуальних проявів, пов’язаних із конкретними захворюваннями, що дозволяє значно підвищити точність і швидкість встановлення діагнозу. CNN здатні самостійно навчатися витягувати релевантні ознаки із зображень без потреби в ручному формуванні ознакових векторів. Завдяки архітектурі, яка включає згорткові шари, шари підвибірки та повнозв’язані шари, ці мережі ефективно виявляють як низькорівневі, так і високорівневі особливості на

медичних знімках. Це дозволяє застосовувати CNN для класифікації, сегментації та локалізації уражених ділянок на таких зображеннях, як рентгенограми, КТ або офтальмологічні знімки очного дна. У результаті CNN активно застосовуються в системах підтримки прийняття рішень, знижуючи навантаження на лікарів і підвищуючи об’єктивність діагностики.

Структура та принцип роботи згорткової нейронної мережі [19].

1. **Вхідний шар:** CNN отримує необроблене зображення у вигляді матриці піксельних значень. Кожне зображення представляється у вигляді матриці значень пікселів. Розмір вхідного шару визначається розмірами вхідного зображення.

2. **Згортковий шар:** цей шар є ключовим елементом CNN. Він використовує набір невеликих фільтрів, які ковзають по зображенню, виконуючи згорткову операцію. У кожній позиції фільтр обчислює скалярний добуток між своїми вагами та відповідними пікселями зображення. Так виявляються локальні ознаки, наприклад краї або текстури. Результатом є карти ознак, які демонструють, де і які шаблони були виявлені.

3. **Активаційна функція:** після згорткової операції до кожного значення у картах ознак застосовується функція активації. Найчастіше використовується функція ReLU (Rectified Linear Unit), яка замінює всі від’ємні значення на нуль, а додатні залишає без змін. Це додає моделі здатність до нелінійного навчання, що є необхідним для складних завдань розпізнавання.

4. **Шар об'єднання (Pooling):** цей шар зменшує розміри карт ознак, при цьому зберігаючи найважливішу інформацію. Найпопулярнішим методом є max pooling — він ділить карту ознак на невеликі області та залишає лише максимальні значення з кожної області. Така процедура зменшує обчислювальне навантаження та підвищує стійкість мережі до невеликих змін положення об’єктів на зображенні.

5. **Повністю зв’язаний шар:** на цьому рівні мережа переходить від вивчених ознак до прийняття рішень. Всі нейрони цього шару з’єднані з усіма нейронами попереднього шару. Формується відповідність між виявленими

ознаками і кінцевими класами. Наприкінці використовується функція SoftMax, яка перетворює результати на ймовірності приналежності зображення до кожного з можливих класів.

6. **Вихідний шар:** фінальний рівень CNN, що формує підсумок класифікації. Він містить стільки нейронів, скільки є класів. За допомогою SoftMax мережа визначає ймовірність належності зображення до кожного класу. Клас з найвищою ймовірністю обирається як передбачений результат.

## 2.2 Вибір мови програмування

Для реалізації моделі машинного навчання, яка виконує класифікацію захворювань ока за медичними зображеннями, було обрано мову програмування Python. Це один із найпоширеніших і найзручніших інструментів у сфері штучного інтелекту, який активно використовується як у наукових дослідженнях, так і в реальних медичних системах.

Python надає широкий набір бібліотек і фреймворків для роботи з нейронними мережами, що дозволяє легко будувати, навчати та оцінювати моделі. Зокрема, для створення згорткової нейронної мережі в цій роботі використовуються такі інструменти.

TensorFlow [21] — це потужна бібліотека з відкритим кодом від Google, яка забезпечує гнучкість і високу продуктивність при побудові моделей глибокого навчання. Вона підтримує роботу як на CPU, так і на GPU, що дозволяє значно прискорити процес навчання великих моделей.

Останні версії TensorFlow з повною підтримкою GPU доступні лише при використанні операційної системи Linux або середовища Windows Subsystem for Linux (WSL) спеціальної технології від Microsoft, яка дозволяє запускати повноцінне Linux-середовище безпосередньо у Windows без потреби у віртуальній машині чи подвійному завантаженні. WSL дає змогу використовувати всі можливості Linux-дистрибутивів у звичному середовищі Windows, що ідеально підходить для роботи з бібліотеками, які краще або виключно працюють під Linux.

Таким чином, використання WSL дозволить запускати та навчати сучасні глибокі моделі з апаратним прискоренням (GPU) навіть на комп'ютерах з Windows, забезпечуючи кращу продуктивність і сумісність із сучасними інструментами глибинного навчання.

Keras [22] — високорівнева надбудова над TensorFlow, яка спрощує створення нейронних мереж завдяки зручному та зрозумілому API. Вона дозволяє швидко будувати архітектуру моделі, експериментувати з шарами та гіперпараметрами без зайвих технічних складностей. Містить підтримку рекурентних та згорткових нейронних мереж.

Обидві бібліотеки мають велику спільноту, активну документацію та велику кількість готових прикладів і туторіалів, що є перевагою під час розробки медичних моделей. Крім того, вони інтегруються з іншими популярними бібліотеками Python — такими як NumPy, Pandas, Matplotlib — що дозволяє зручно обробляти дані, аналізувати результати та візуалізувати процес навчання.

Для реалізації графічного інтерфейсу користувача (GUI) було обрано бібліотеку Gradio — сучасний інструмент для швидкого створення веб-інтерфейсів у середовищі Python [23].

Gradio дозволяє легко інтегрувати машинне навчання та інші обчислювальні моделі в інтерактивні веб-застосунки, що працюють у браузері. Завдяки простому синтаксису, підтримці зображень, тексту, аудіо та відео, Gradio є зручним рішенням для розробки прототипів та демонстрацій без необхідності створювати окремий десктопний застосунок або писати HTML/CSS-код.

На відміну від класичних десктопних рішень, таких як PyQt, Gradio працює прямо у веббраузері, що дозволяє запускати інтерфейс навіть у середовищах без графічної оболонки.

### 2.3 Архітектура нейронної мережі

Для задачі класифікації зображень сітківки за ознаками захворювання було розглянуто кілька сучасних згорткових нейронних мереж, які показали

ефективність у медичних застосуваннях. Нижче наведено короткий огляд кожної з них.

**VGG16** [24] — класична глибока CNN-архітектура з послідовним розміщенням згорткових і пулінгових шарів (див. рис. 2.2). Проста у реалізації, але має велику кількість параметрів, що призводить до тривалого часу навчання та ризику перенавчання на невеликих наборах даних.

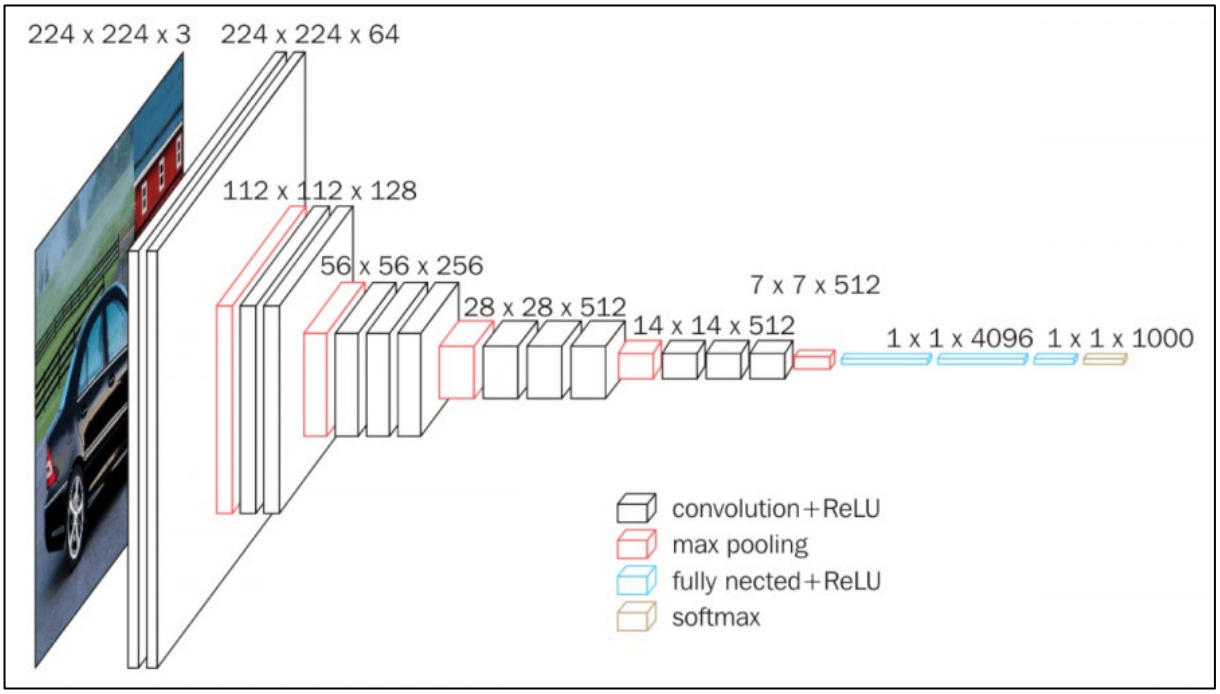


Рисунок 2.2 – Архітектура VGG16 [25]

**InceptionV3** [26] — глибока модель з ефективним розподілом обчислень завдяки використанню модулів "Inception". Забезпечує хорошу точність при меншій кількості параметрів, але має складну структуру, що ускладнює її модифікацію та інтерпретацію.

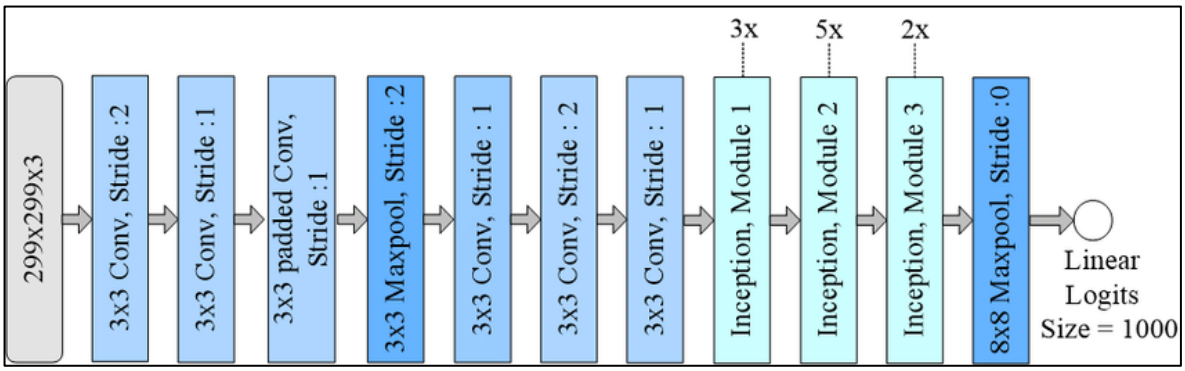


Рисунок 2.3 – Архітектура InceptionV3 [26]

**DenseNet121** [27] — архітектура, в якій кожен шар отримує на вхід виходи з усіх попередніх шарів. Це сприяє покращеному передаванню градієнта і повторному використанню ознак, але потребує більше оперативної пам'яті при тренуванні що не є пріоритетом в умовах розпізнаванні сітківки.

**ResNet50** [28] — архітектура з залишковими з'єднаннями (див. рис. 2.3), які усувають проблему затухання градієнта, дозволяючи ефективно навчати глибокі моделі. Має баланс між глибиною, точністю та стабільністю роботи при трансферному навчанні.

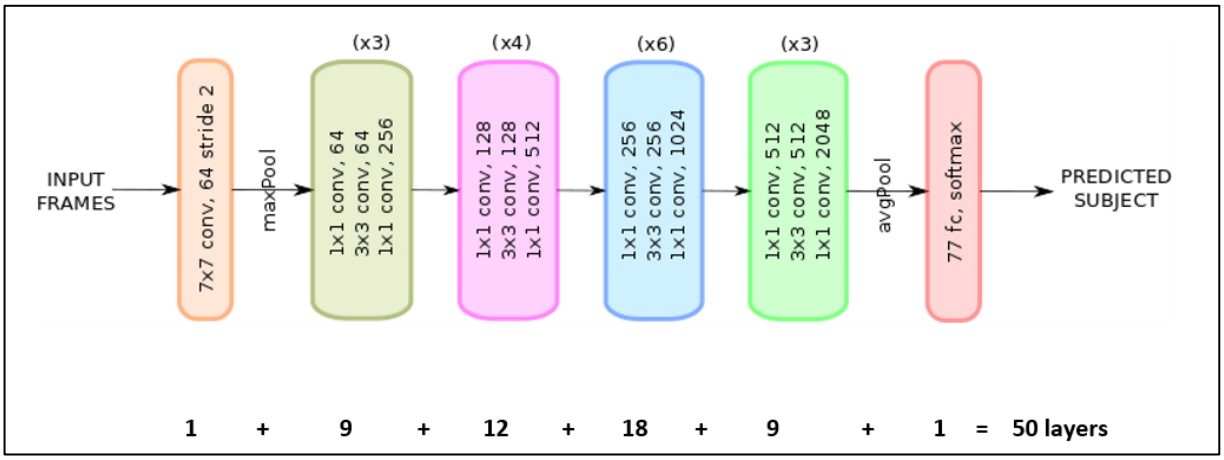


Рисунок 2.4 – Архітектура ResNet50 [29]

Серед проаналізованих варіантів **ResNet50** було обрано як оптимальну архітектуру завдяки її високій точності**,** стійкості до перенавчання**,** гнучкості у

використанні трансферного навчання **та** відносно невеликій кількості параметрів у порівнянні з класичними моделями на кшталт VGG16. Її успішне застосування в багатьох задачах медичної діагностики робить її надійним вибором і для аналізу зображень сітківки.

Архітектура ResNet50 складається з 50 шарів, включно з згортковими, нормалізаційними, пулінговими шарами та повнозв'язним класифікатором на виході. Структура мережі реалізована за принципом залишкових блоків (*residual blocks*), що дозволяють передавати інформацію без змін між шарами, покращуючи ефективність навчання глибокої мережі.

Основні компоненти архітектури:

а) вхідний шар:

1) згортковий шар з фільтром 7×7, stride = 2;

2) подальший шар максимального пулінгу 3×3, stride = 2;

б) чотири основні групи залишкових блоків де кожна група містить по кілька блоків, де кожен блок включає послідовність згорток 1×1 → 3×3 → 1×1;

в) на додачу застосовується пропускний шлях (*identity shortcut*) для збереження сигналу;

г) глобальний середній пулінг (Global Average Pooling);

д) класифікаційний шар: Повнозв'язний лінійний шар зі зміненою кількістю вихідних нейронів відповідно до кількості класів захворювань (n);

е) активаційна функція SoftMax для багатокласової класифікації.

**Переваги використання ResNet50**:

– дозволяє виявляти складні ознаки на зображеннях сітківки;

– залишкові зв'язки усувають проблему деградації точності при збільшенні глибини мережі;

– можливість трансферного навчання надає змогу ефективно адаптувати модель до медичних зображень при обмеженому обсязі навчальних даних;

– висока точність класифікації**,** підтверджена результатами численних досліджень у галузі медичного комп'ютерного зору;

– архітектура ResNet50 має 50 шарів, але уникає проблеми затухання градієнта та деградації точності завдяки residual-з'єднанням.

**Недоліки використання ResNet50**:

– модель є досить глибокою, що потребує значних ресурсів особливо при тренуванні на великих зображеннях;

– через велику кількість шарів навчання може тривати довше порівняно з менш складними архітектурами;

– при недостатньому обсязі даних або неправильному налаштуванні регуляризації модель може перенавчитися, особливо якщо використовуються всі шари без заморожування;

– великий розмір моделі.

## Висновки до розділу 2

Розглянуто основні теоретичні та практичні аспекти застосування машинного навчання в медичній діагностиці, зокрема в задачі класифікації хвороб ока за зображеннями сітківки. Машинне навчання, а особливо глибоке навчання, довело свою ефективність у виявленні патологічних змін на медичних зображеннях, завдяки здатності автоматично витягувати релевантні ознаки з вхідних даних.

Було обґрунтовано вибір підходу навчання з учителем, що дозволяє моделі навчатися за допомогою попередньо розмічених даних та забезпечує високу точність і надійність результатів — надзвичайно важливі характеристики в медичних задачах.

Серед різних архітектур згорткових нейронних мереж, що активно застосовуються в медицині, оптимальним вибором для поставленої задачі виявилася ResNet50. Ця модель поєднує глибоку структуру, залишкові з’єднання, високу точність класифікації та можливість трансферного навчання, що робить її ефективною навіть при обмеженій кількості даних.

Також було обґрунтовано вибір мови програмування Python як

найзручнішого інструменту для реалізації проєкту, завдяки її широкому набору бібліотек для машинного навчання (TensorFlow, Keras) та створення графічного інтерфейсу користувача (Gradio).

У результаті проведеного аналізу було створено теоретичну та технічну основу для реалізації системи автоматичної діагностики офтальмологічних захворювань на основі зображень, що відкриває можливості для підвищення якості медичного обслуговування та зменшення навантаження на фахівців.

# 3 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДІАГНОСТУВАННЯ ХВОРОБ ОКА

## 3.1 Опис вхідних даних та структури системи

Інформаційна система діагностування хвороб ока орієнтована на обробку цифрових медичних зображень, зроблених на офтальмоскопі. Ці знімки можуть бути зроблені на різних пристроях та з різною роздільною здатністю.

Характеристики вхідного зображення:

– зображення з офтальмоскопа з розширеннями png або jpeg;

– кольорове зображення RGB, 24 біти з роздільною здатністю 224×224 або більше. У процесі передобробки зображення масштабується до стандартного розміру, сумісного з моделлю нейромережі.

Крім цього необхідними даними є інформація про пацієнта: ім’я, вік, стать, скарги, тощо.

Також необхідними даними є база даних знімків для попереднього навчання нейронної мережі з відповідними помітками про поставлені діагнози.

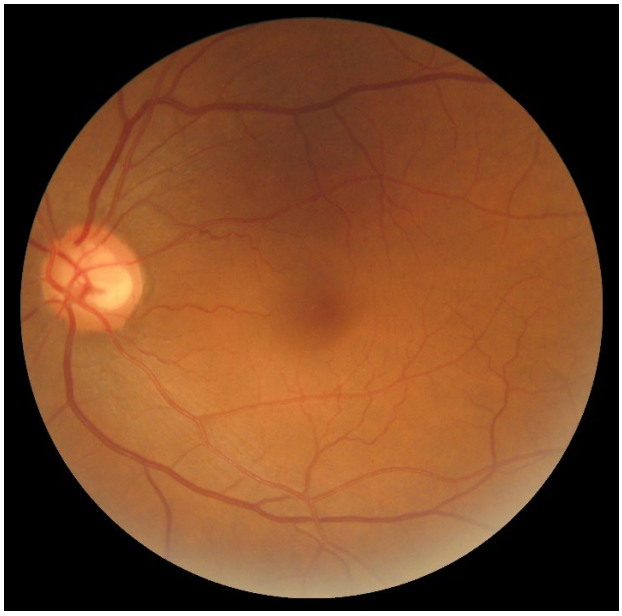

Рисунок 3.1 – Приклад вхідного зображення

### 3.1.1 Структури системи

Система має модульну структуру, яка забезпечує гнучкість, масштабованість та розмежування задач. Загалом, її можна подати у вигляді трьохрівневої архітектури.

1. Рівень подання (графічний інтерфейс користувача (GUI), реалізований за допомогою бібліотеки Gradio який дозволяє):
   - завантажувати дані, тобто зображення та дані пацієнта;
   - переглядати результат класифікації;
   - експортувати вже наявні дані.
2. Рівень логіки (обробляє основну логіку роботи системи):
   - масштабування та нормалізація вхідних зображень;
   - прогнозування діагнозу за допомогою навченої нейронної мережі;
   - отримання діагнозу;
   - забезпечення комунікації між інтерфейсом та моделлю;
   - запис даних у відповідну базу.
3. Рівень даних:
   - файли зображень з ID;
   - файл моделі нейронної мережі;
   - база даних пацієнтів.

### 3.1.2 Діаграма послідовностей

Діаграма послідовностей [30] що використовується для моделювання поведінки системи шляхом відображення послідовності взаємодій між об'єктами або компонентами, ілюструє процес взаємодії ключових елементів системи діагностики захворювань ока. Вона показує порядок виконання дій та передачу даних між учасниками системи, зокрема акторами та функціональними модулями.

На діаграмі відображено взаємодію між лікарем, графічним інтерфейсом, модулем обробки зображень, модулем аналізу симптомів і модулем збереження даних у базі даних (див. рис. 3.1).

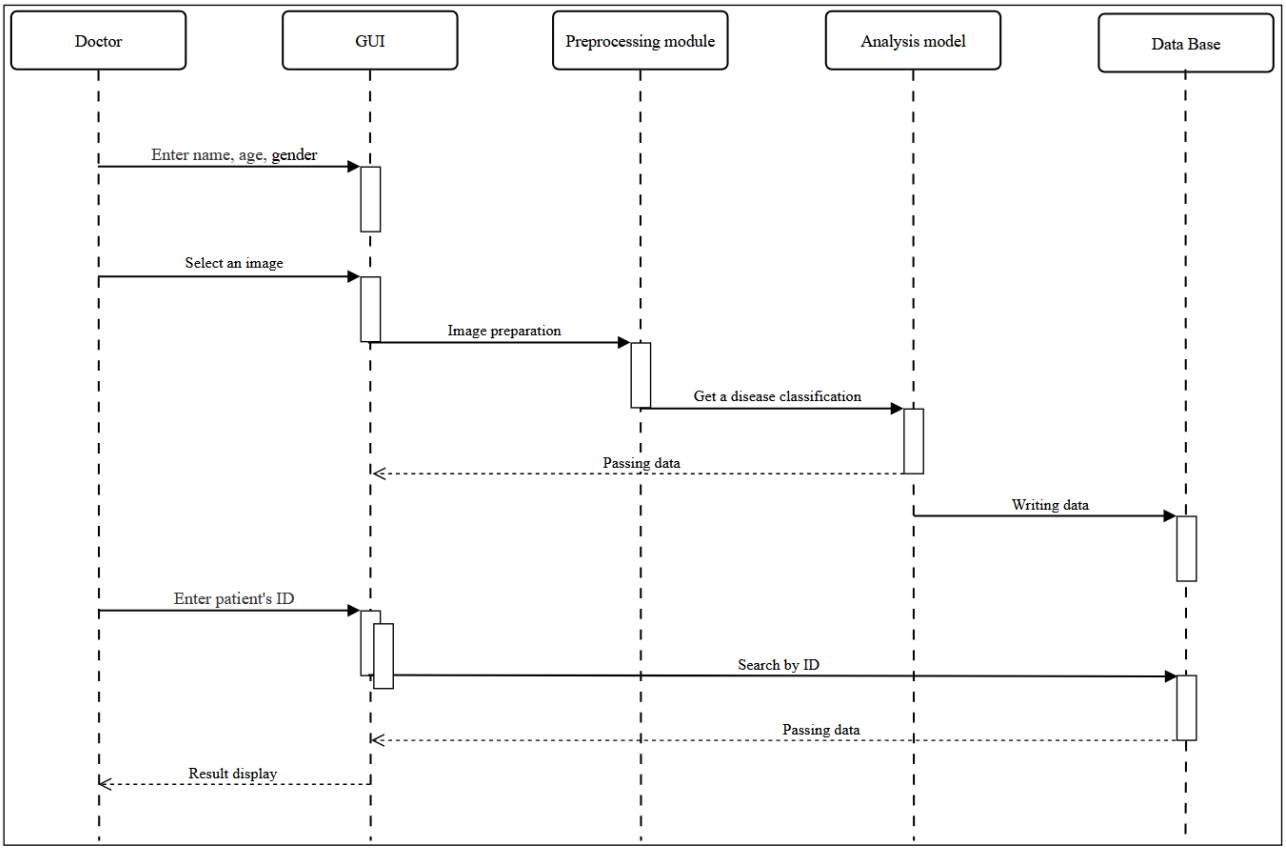


Рисунок 3.2 – Діаграма послідовностей

Компоненти діаграми:

– Doctor — користувач системи (лікар);
– GUI — графічний інтерфейс користувача;
– Preprocessing module — модуль попередньої обробки зображення;
– Analysis model — навчена нейронна мережа, яка виконує діагностику;
– Data Base — база даних, у якій зберігаються результати.

Основні сценарії наведено нижче:

а) класифікація захворювання ока:

1) Doctor вводить дані пацієнта: ім’я, вік, стать;

2) вибирається зображення ока;

3) GUI надсилає зображення до модуля Preprocessing, де воно підготовлюється;

4) попередньо оброблене зображення передається до Analysis model, яка проводить діагностику;

5) отриманий результат передається назад через GUI лікарю;

6) результати також записуються у базу даних для збереження;

б) пошук історії пацієнта за прізвищем:

1) лікар вводить прізвище пацієнта;

2) GUI надсилає виконує функцію пошуку;

3) знайдені дані виводяться лікарю.

### 3.1.3 Макет інтерфейсу

Процес створення макету застосунка є важливим етапом у розробці, оскільки дозволяє заздалегідь продумати інтерфейс користувача (UI) та забезпечити зручність взаємодії з системою. Макет слугує візуальним представленням основних елементів інтерфейсу та логіки переходів між екранами до початку програмної реалізації.

Основною метою створення макету є:

- визначення структури майбутнього застосунка;
- візуалізація розташування елементів інтерфейсу;
- узгодження дизайну з технічними та функціональними вимогами.

Макет складається з таких основних екранів (головна сторінка, сторінка пошуку існуючих пацієнтів), які більш детально розглянуто нижче.

**Головна сторінка** — надає користувачеві доступ до основного функціоналу. Містить рядки для вводу даних пацієнта, а саме ім’я, прізвище, стать, дата народження; окремі меню для завантаження знімків очей, рядок для опису скарг пацієнта — звичайна практика для медичних застосунків; місце для виводу діагнозу. Макет представлено на рис. 3.2.

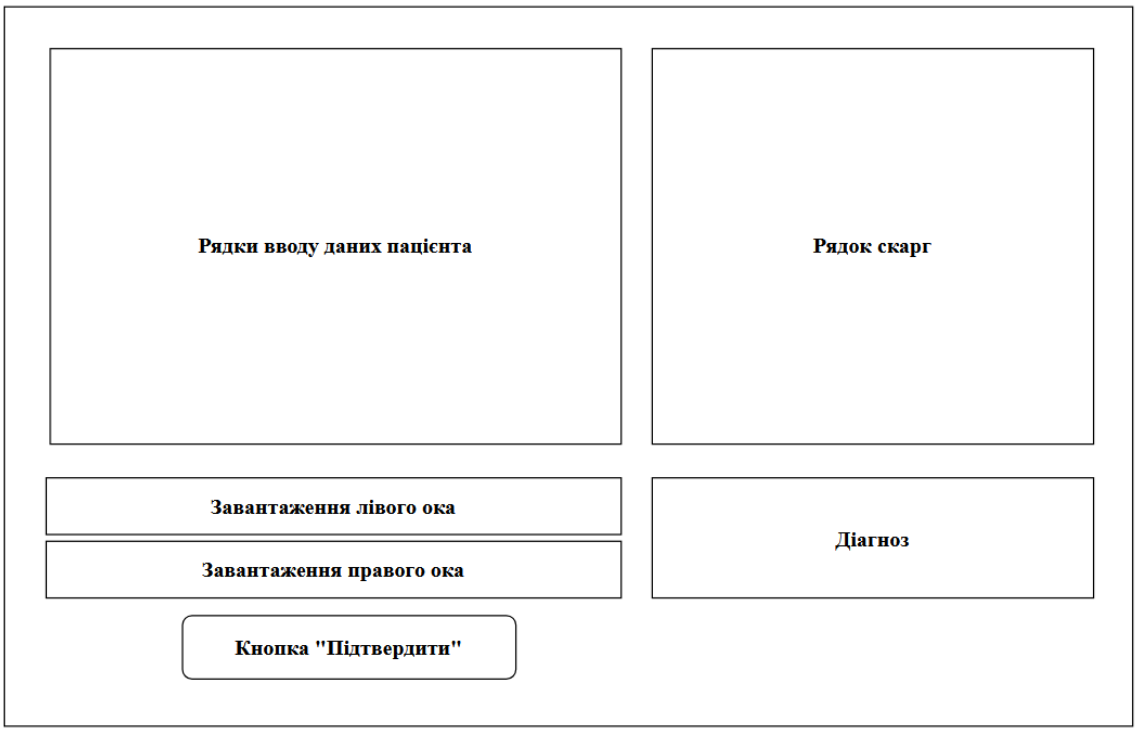


Рисунок 3.3 – Макет головної сторінки

**Сторінка пошуку існуючих пацієнтів** — забезпечує пошук по базі даних пацієнтів. Містить рядок для прізвища, кнопку «Пошук» та таблицю з знайденими даними.

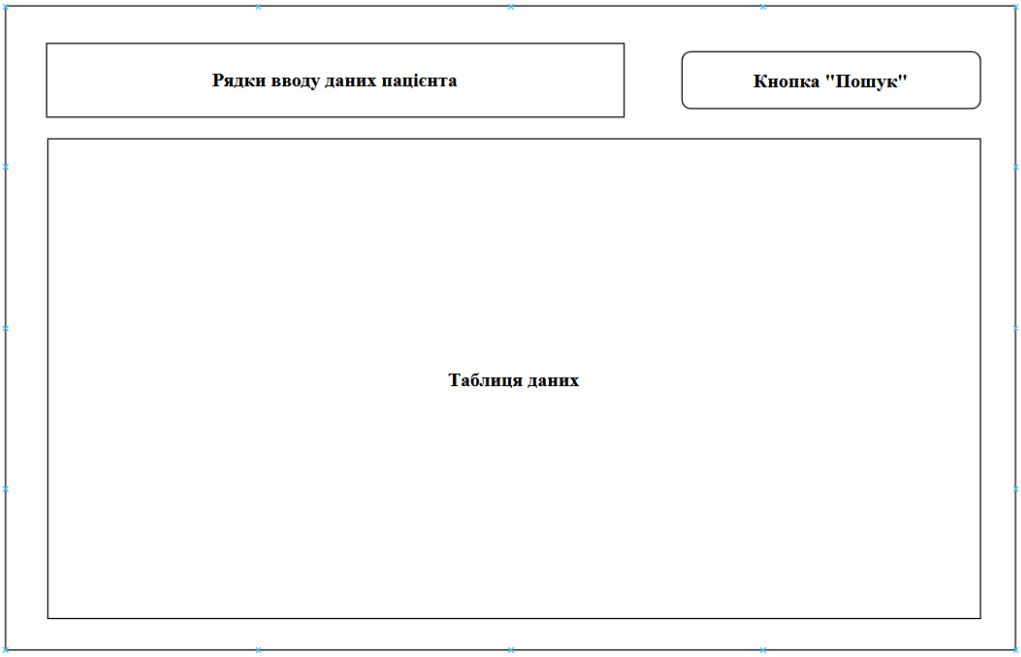


Рисунок 3.4 – Макет сторінки пошуку

## 3.2 Аналіз даних

У рамках дослідження з розробки автоматизованої системи розпізнавання захворювань ока було обрано датасет **ODIR**, який є повним, анотаційно багатим і клінічно наближеним джерелом офтальмологічних даних, відкрито доступних для дослідників.

ODIR (Ocular Disease Intelligent Recognition) [31] — це структурована база даних, що була створена компанією **Shanggong Medical Technology Co., Ltd.** у співпраці з різними лікарнями та медичними центрами Китаю. Вона містить клінічні записи 5 000 пацієнтів, що включають:

– демографічну інформацію (вік, стать);
– кольорові фундус-зображення обох очей (правого і лівого);
– ключові діагностичні терміни, зазначені лікарями.

Дані зібрані з реального клінічного середовища, що робить цей набір особливо наближеним до ситуацій, які можуть виникати у практичній офтальмології. Важливою особливістю є те, що зображення отримані з використанням різноманітного обладнання, що забезпечує варіативність у форматах та роздільній здатності знімків. Цей фактор створює додаткову складність для автоматизованих моделей, але водночас робить треновані системи більш стійкими до змін у вхідних даних.

Анотації до кожного зображення створювались кваліфікованими офтальмологами з подальшим контролем якості, що значно підвищує достовірність та точність розмітки.

ODIR охоплює широкий спектр офтальмологічних захворювань, що дозволяє створювати багатокласові класифікаційні моделі. Кожен пацієнт може бути класифікований за наявністю одного або декількох із наступних захворювань:

– normal (N) — відсутність патологій;
– diabetes (D) — діабетична ретинопатія;
– glaucoma (G) — глаукома;

– cataract (C) — катаракта;
– age-related Macular Degeneration (A) — вікова макулодистрофія;
– hypertension (H) — гіпертонічна ретинопатія;
– pathological Myopia (M) — патологічна міопія;
– other abnormalities (O) — інші відхилення як на рис. 3.5.

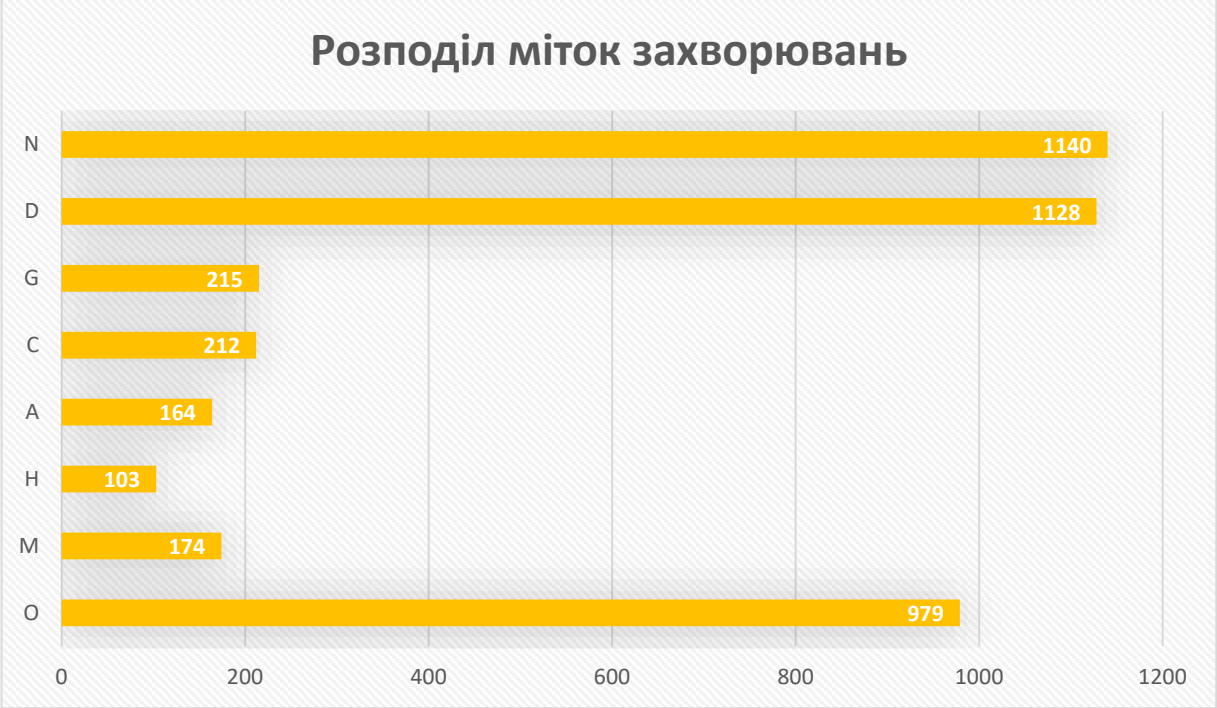


Рисунок 3.5 – Діаграма розподілу міток

Завдяки такій класифікації датасет дозволяє досліджувати не лише мультикласові, але й **багатоміткові** моделі, адже один пацієнт може мати декілька діагнозів одночасно.

Набір складається з двох папок тренувальної розміром 1.3 ГБ, як в свою чергу складається із знімків формату .jpg лівих та правих очей в загальній кількості 7000 файлів та тестувальної розміром 200 МБ яка складається із знімків формату .jpg в загальній кількості 1000 файлів.

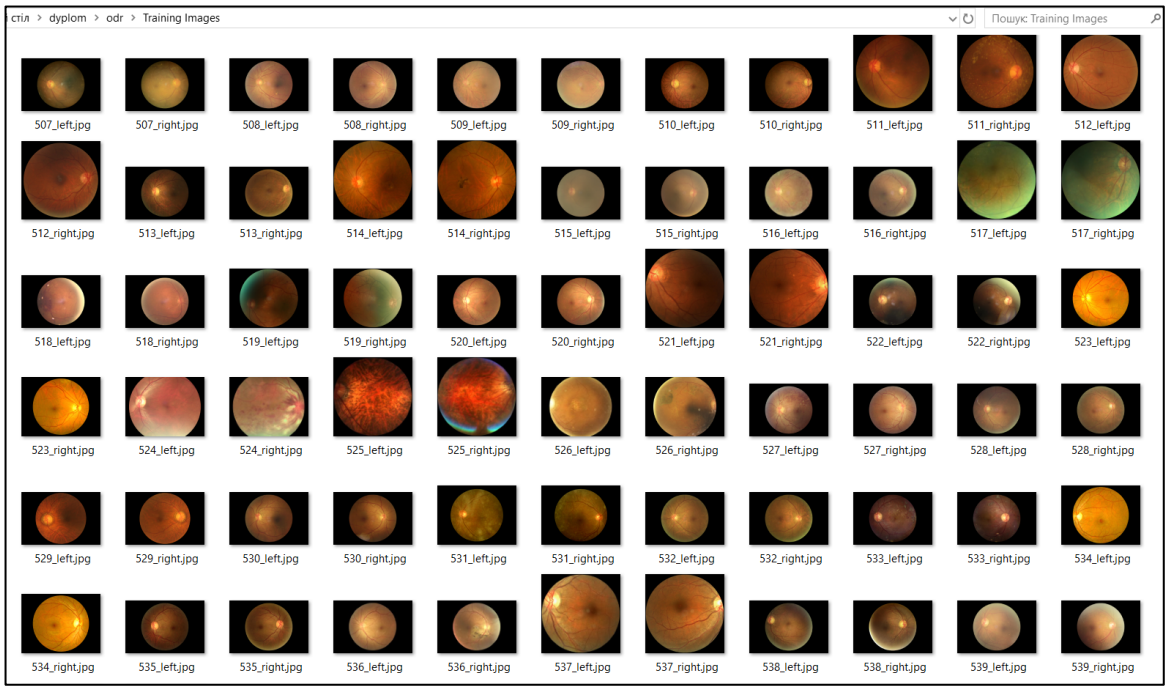


Рисунок 3.6 – Вибірка зображень

### 3.3 Попередня обробка

Перед початком навчання моделі глибокого навчання необхідно провести етап попередньої обробки даних. Також цю обробку варто проводити для нових даних вже під час безпосередньо роботи натренерованої мережі. Це важливий етап, який суттєво впливає на якість та ефективність навчання нейронної мережі. Враховуючи, що ODIR є реальним клінічним датасетом, зібраним із різних джерел, попередня обробка є особливо критичною. Для цього створимо окремий який буде мати відповідний функціонал який наведено нижче.

**Обрізка чорних полів або круглих рамок**: часто фундус-зображення містять чорні фони або круглі обрізи. Вони можуть бути видалені, щоб залишити лише корисну область — сітківку.

```python
def crop_black_borders(img, threshold=10):
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    _, thresh = cv2.threshold(gray, threshold, 255, cv2.THRESH_BINARY)
    coords = cv2.findNonZero(thresh)
    if coords is None:
        return None
    x, y, w, h = cv2.boundingRect(coords)
    return img[y:y+h, x:x+w]
```

Рисунок 3.7 – Обробка чорних полів

Функція crop_black_borders працює таким чином:

– зображення переводиться в чорно-біле (градації сірого), щоб спростити подальшу обробку;

– порогове бінарне перетворення, пікселі яскравість яких вище порогу threshold, перетворюються в білий (255), а інші — в чорний (0);

– пошук координат всіх білих пікселів;

– побудова прямокутника навколо корисної області;

– обрізання зображення до цієї зони.

**Зміна розміру зображень**. Зображення у датасеті мають різну роздільну здатність, оскільки отримані з різних офтальмоскопів та різних камер. Для уніфікації всіх зображень вони приводяться до стандартного фіксованого розміру 224×224 пікселі.

```python
resized = cv2.resize(cropped, (224, 224))
```

Рисунок 3.8 – Збереження зображення розміром 224x224

**Фільтрація або видалення неякісних зображень**. Частина зображень може бути сильно розмитою або мати серйозні артефакти, такі зразки доцільно вилучити, оскільки вони можуть заважати навчанню.

```python
for side in ['Left', 'Right']:
    filename = row[f'{side}-Fundus']
    img_path = os.path.join(img_dir, filename)

    if not os.path.exists(img_path):
        success = False
        break

    img = cv2.imread(img_path)
    if img is None:
        success = False
        break

    cropped = crop_black_borders(img)
    if cropped is None:
        success = False
        break
```

Рисунок 3.9 – Фільтрування зображень

Це дозволяє автоматично прибрати сміття з датасету та зберегти тільки якісні зображення для подальшого навчання моделі.

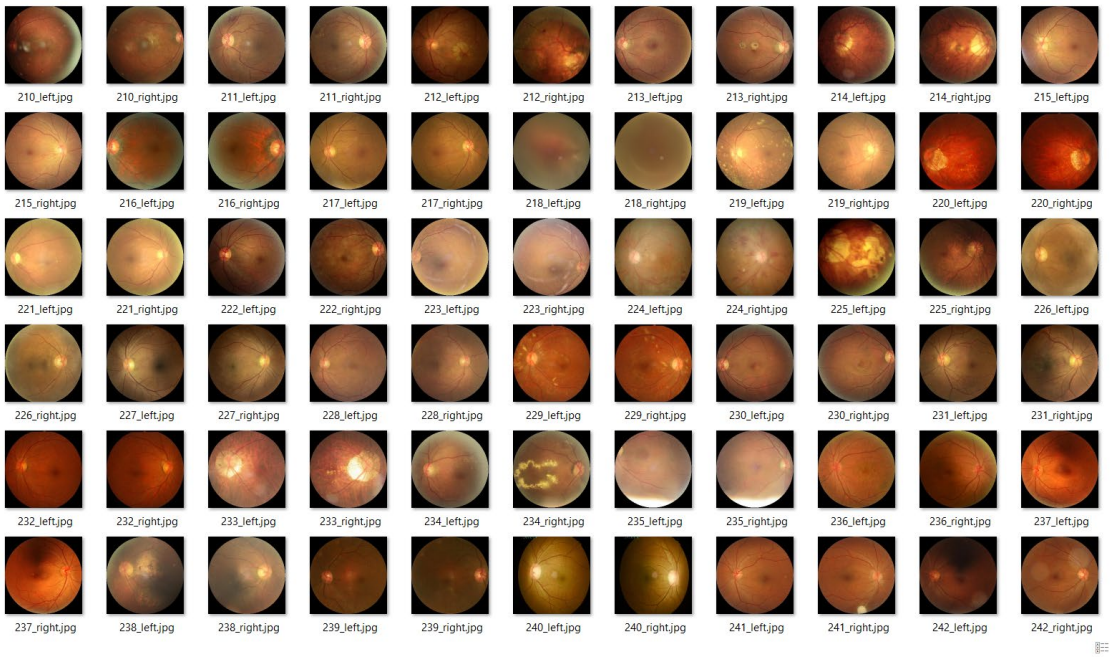


Рисунок 3.10 – Вибірка зображень після фільтрації

**Сортування зображень**. Під час підготовки набору даних для навчання нейронної мережі важливо організувати зображення у чітку структуру. Наприклад, сортувати зображення по окремих папках відповідно до їх класу.

Основні переваги такої організації:

– фреймворки глибокого навчання, такі як TensorFlow та Keras, підтримують автоматичне зчитування зображень, якщо вони відсортовані по папках, це дозволяє значно спростити та автоматизувати процес створення датасету;

– у випадку, коли хвороби зазначені лише в CSV-файлі, для обробки доведеться писати додатковий код, який буде відповідно зчитувати шляхи до зображень і зіставляти їх з мітками. Сортування по папках дозволяє уникнути цього кроку.

**Аугментація зображень** — це процес штучного розширення навчального набору даних шляхом модифікації наявних зображень із збереженням їхнього семантичного змісту. На відміну від повної генерації нових даних, аугментація передбачає варіювання існуючих прикладів шляхом застосування таких перетворень, як обертання, віддзеркалення, зміна яскравості, масштабування, контрасту тощо.

У контексті глибокого навчання, зокрема для задач комп’ютерного зору, ефективність моделей значною мірою залежить від обсягу та різноманіття вхідних даних. Оскільки збирання великої кількості оригінальних зображень часто є трудомістким і ресурсоємним процесом, аугментація даних виступає як один з ключових інструментів підвищення якості навчання моделей.

У межах даного дослідження аугментація зображень реалізується безпосередньо в процесі навчання — модифікації зображень здійснюються динамічно. Такий підхід сприяє покращенню здатності моделі до узагальнення, підвищує її стійкість до варіацій у вхідних даних та зменшує ризик перенавчання.

```
data_augmentation = tf.keras.Sequential([
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.1),
    layers.RandomContrast(0.1),
    layers.RandomBrightness(0.1),
])
```

Рисунок 3.11 – Аугментація зображень

*Tf.keras.Sequential* — клас для побудови послідовних моделей.

*Layers.RandomRotation(0.1)* шар застосовує випадкове обертання до зображень. Параметр 0.1 вказує на максимальний коефіцієнт обертання, що означає, що зображення будуть обертатися на випадковий кут в діапазоні від -36 до +36 градусів.

*Layers.RandomZoom(0.1)* випадкове збільшення або зменшення масштабу. Масштаб буде випадково вибраний у діапазоні ±10%.

*Layers.RandomContrast(0.1)* випадкові зміни контрасту зображення.

*Layers.RandomBrightness(0.1)* шар застосовує випадкові зміни яскравості.

**Висновки до розділу 3**

У розділі було розглянуто архітектуру, компоненти та основні етапи функціонування інформаційної системи діагностування хвороб ока, орієнтованої на обробку цифрових медичних зображень. Було описано модульну трирівневу структуру системи, яка забезпечує розподіл обов’язків між рівнями подання, логіки та даних, а також забезпечує гнучкість і масштабованість при подальшій розробці або розширенні функціоналу.

Представлено діаграму послідовностей, яка моделює типові сценарії використання системи, включаючи процес класифікації захворювання ока та пошук історії пацієнта. Це дозволяє чітко візуалізувати взаємодію між ключовими компонентами: лікарем, інтерфейсом, модулями обробки зображень, аналізу та

базою даних.

Особливу увагу приділено аналізу даних, де було обґрунтовано вибір датасету ODIR як надійного джерела для навчання та тестування моделі. Завдяки своїй структурованості, наявності анотацій і різноманітності знімків, цей набір даних забезпечує хорошу основу для побудови стійких та точних багатокласових або багатоміткових моделей.

Окремо розглянуто процес попередньої обробки зображень, що є важливим етапом як перед навчанням, так і під час реального використання моделі. Було описано методи видалення чорних полів, зміни розміру зображень, фільтрації неякісних даних, сортування зображень та їх аугментації. Ці процедури спрямовані на покращення якості вхідних даних і, як наслідок, підвищення точності та узагальнювальної здатності моделі.

# 4 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

## 4.1 Створення та навчання моделі ResNet50

Для навчання моделі було використано набір зображень очного дна, попередньо розділений по класах та впорядкований у відповідні підкаталоги за діагнозами. Папка *sorted_images* містить підкаталоги з назвами відповідних класів, які автоматично зчитуються під час завантаження датасету.

Дані розділяються на **тренувальний** та **валідаційний набори** з допомогою функції *image_dataset_from_directory()*. Валідаційний набір, на якому модель тестуватиметься кожну епоху становить 20% від усіх даних.

```
data_dir = 'sorted_images/'
img_size = (224, 224)
batch_size = 32
epochs = 30

train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir,
    validation_split=0.2,
    subset="training",
    seed=123,
    image_size=img_size,
    batch_size=batch_size
)
validation_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir,
    validation_split=0.2,
    subset="validation",
    seed=123,
    image_size=img_size,
    batch_size=batch_size
)
```

Рисунок 4.1 – Зчитування даних

Зважаючи на те що база даних містить не врівноважену кількість зображень для кожного з класів, застосовано балансування класів за допомогою модуля *sklearn.utils.class_weight*. Також для покращення здатності моделі застосовується аугментація.

```
class_weights = class_weight.compute_class_weight(
    class_weight='balanced',
    classes=np.unique(all_labels),
    y=all_labels
)
class_weights = dict(enumerate(class_weights))
```

Рисунок 4.2 – Балансувач класів

Як основну частину класифікатора було обрано попередньо натреновану модель ResNet50, яка вже добре вивчена та оптимізована для задач розпізнавання зображень.

Модель було використано без верхньої частини *include_top=False*, що дозволяє інтегрувати власні шари під специфіку конкретного набору даних. Крім того, збережено попередньо натреновані ваги *weights=imagenet*, що дозволяє скоротити час навчання та підвищити точність на обмеженій кількості медичних зображень. Мережу було частково заморожено — перші 110 шарів не оновлюються під час навчання.

Такий підхід називається **тонке перенавчання** (англ. *fine-tuning*). Він дозволяє адаптувати ознаки, які вже "вивчила" ResNet50, до особливостей нового датасету, не знищуючи при цьому знання, отримані на великій базі ImageNet.

```
base_model = ResNet50(include_top=False, input_shape=(224, 224, 3), pooling='avg', weights='imagenet')
base_model.trainable = True

fine_tune_at = 110
for layer in base_model.layers[:fine_tune_at]:
    layer.trainable = False
```

Рисунок 4.3 – Створення моделі

Після цього додаються власні шари класифікації а саме:

– вхідний **Dense** шар де кожен нейрон цього шару з’єднаний з усіма виходами попереднього шару. Цей шар виконує лінійну трансформацію ознак, витягнутих зображенням, і передає їх далі з використанням нелінійної функції

активації *ReLU*. *ReLU* дозволяє моделі краще адаптуватися до складних нелінійних залежностей даних, зберігаючи обчислювальну ефективність;

– **dropout** шар — цей шар випадковим чином "вимикає" 20% нейронів під час кожної ітерації навчання. Його головна мета — запобігти перенавчанню (overfitting), тобто ситуації, коли модель просто запам’ятовує дані а не їх ознаки Dropout змушує модель не покладатися лише на певні шляхи проходження інформації, що підвищує стійкість класифікації нових зображень;

– вихідний **Dense**-шар кількість нейронів у якому дорівнює кількості класів а функція активації — **softmax**. Цей шар перетворює числові виходи на ймовірності.

```
model = Sequential([
    data_augmentation,
    base_model,
    layers.Dense(256, activation='relu'),
    layers.Dropout(0.2),
    layers.Dense(num_classes, activation='softmax')
])
```

Рисунок 4.4 – Шари класифікації

Перед початком навчання, модель компілюється з використанням оптимізатора **Adam.**

**Adam** (Adaptive Moment Estimation). Це один з найпопулярніших та найефективніших алгоритмів градієнтного спуску. Він автоматично адаптує швидкість навчання для кожного параметра моделі, що дозволяє досягати хороших результатів без складного підбору гіперпараметрів.

Для запобігання перенавчанню навчанню використовується колбек **EarlyStopping**. Він перериває тренування, якщо валідаційна похибка не покращується протягом 5 епох.

Використовується спрощена функція втрат **sparse_categorical_crossentropy**,

яка приймає цілочисельні мітки класів, що знижує обчислювальне навантаження та спрощує обробку.

```
model.compile(
    optimizer=Adam(learning_rate=0.001),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

early_stop = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
```

Рисунок 4.5 – Компіляція моделі

Після завершення тренування результати було візуалізовано у вигляді графіків:

– **точність (accuracy)** — як змінюється здатність моделі правильно класифікувати зображення на тренувальній і валідаційній вибірках;

– **функція втрат (loss)** — як сильно змінюється помилка моделі під час навчання.

Також обчислено класифікаційний звіт та матрицю плутанини, які дають змогу оцінити точність класифікації для кожного класу окремо.

```
y_true, y_pred = [], []
for images, labels in validation_ds:
    preds = model.predict(images)
    y_pred.extend(np.argmax(preds, axis=1))
    y_true.extend(labels.numpy())

print("\nClassification Report:\n")
print(classification_report(y_true, y_pred, target_names=class_names))

print("\nConfusion Matrix:\n")
print(confusion_matrix(y_true, y_pred))
```

Рисунок 4.6 – Обчислення оцінки моделі

Для можливості використовування моделі, створений скрипт **predictor**. В якому відбувається завантаження моделі, та зображень ока, попередня обробка, так

і як в навчальному датасеті, віддзеркалювання зображення лівого ока, та передбачення класу захворювання.

```python
def classify_image(image: Image.Image, is_left_eye: bool = False):
    img = np.array(image.convert("RGB"))
    img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)

    if is_left_eye:
        img = cv2.flip(img, 1)
    cropped = crop_black_borders(img)

    resized = cv2.resize(cropped, (224, 224))
    img_array = resized / 255.0
    img_array = np.expand_dims(img_array, axis=0)

    predictions = model.predict(img_array)[0]
    predicted_index = np.argmax(predictions)
    label = class_names[predicted_index]
    confidence = round(float(predictions[predicted_index]), 2)

    return label, confidence
```

Рисунок 4.7 – Функція класифікації моделі

### 4.2 Підбір параметрів та оцінювання моделі

Для створення ефективної моделі потрібно вказати **гіперпараметри** — це параметри моделі які не вивчаються під час навчання, а задаються заздалегідь та є основними важілями керування навчанням моделі. Після тривалого тестування було обрано кращу комбінацію яку вдалось віднайти.

Таблиця 4.1 – Результати налаштування параметрів моделі та їхній вплив на точність

| Параметр / Ітерація | Розмір батчу | Кількість нейронів у Dense-шарі | Значення Dropout | Кількість тренованих шарів | Точність моделі (accuracy) |
|---|---|---|---|---|---|
| 1 | 32 | 256 | 0.3 | 120 | 0.52 |
| 2 | 16 | 256 | 0.2 | 110 | 0.49 |

Кінець таблиці 4.1

| | | | | | |
|---|---|---|---|---|---|
| 3 | 16 | 512 | 0.2 | 110 | 0.54 |
| 4 | 16 | 512 | 0.2 | 125 | 0.48 |
| 5 | 16 | 512 | 0.3 | 105 | 0.42 |
| 6 | 32 | 512 | 0.3 | 110 | 0.60 |
| 7 | 32 | 512 | 0.15 | 100 | 0.52 |
| 8 | 32 | 512 | 0.2 | 70 | 0.33 |
| 9 | 32 | 512 | 0.2 | 120 | 0.52 |
| 10 | 32 | 512 | 0.2 | 110 | 0.65 |
| 11 | 32 | 256 | 0.2 | 110 | 0.52 |
| 12 | 32 | 512 | 0.1 | 110 | 0.57 |
| 13 | 32 | 512 | 0.15 | 110 | 0.42 |

Для повної оцінки навченої мережі було використано такі значення:

– **precision** — частка правильних передбачень серед усіх, які модель передбачила як цей клас;

– **recall** — частка всіх позитивних передбачень які правильно класифіковано серед прикладів цього класу;

– **f1-score** [32] **—** це середнє гармонійне між **precision** та **recall.** Метрика обчислює, скільки разів модель зробила правильний прогноз по всьому набору даних.

| | | | | |
|---|---|---|---|---|
| Age related Macular Degeneration | 0.55 | 0.79 | 0.65 | 52 |
| Cataract | 0.76 | 0.93 | 0.84 | 58 |
| Diabetes | 0.88 | 0.26 | 0.40 | 320 |
| Glaucoma | 0.57 | 0.84 | 0.68 | 56 |
| Hypertension | 0.48 | 0.52 | 0.50 | 25 |
| Normal | 0.66 | 0.83 | 0.73 | 569 |
| Other diseasesabnormalities | 0.43 | 0.45 | 0.44 | 136 |
| Pathological Myopia | 0.75 | 0.98 | 0.85 | 46 |
| accuracy | | | 0.65 | 1262 |
| macro avg | 0.64 | 0.70 | 0.64 | 1262 |
| weighted avg | 0.69 | 0.65 | 0.62 | 1262 |

Рисунок 4.8 – Результат навчання

Модель досягла загальної точності (**accuracy**) на рівні **65%**, що свідчить про помірний рівень ефективності при класифікації 8 класів захворювань очей. Загалом було оцінено 1262 зображення. Середнє значення F1-міри становить 0.62, що означає, що модель демонструє дещо кращу продуктивність для класів з більшою кількістю прикладів, таких як "**Normal**" і "**Diabetes**", але все ще має значні проблеми з менш представленими класами.

Хоча **precision** для класу **Diabetes** висока — 0.88, що означає, що більшість передбачених як Diabetes справді належать до цього класу, повернення **recall** становить лише 0.26. Це свідчить про критичну проблему: модель не виявляє більшість справжніх випадків **Diabetes**. Класи **Pathological Myopia** (F1 = 0.85) та **Cataract** (F1 = 0.84) продемонстрували найвищу точність класифікації. У випадку **Pathological Myopia**, recall сягнув 0.98, що означає, що майже всі справжні випадки були виявлені. Це свідчить про те, що модель має добру здатність розпізнавати характерні ознаки цього захворювання на зображеннях. Класи **Other diseases/abnormalities** та **Hypertension** мають найнижчі значення F1-міри: 0.44 та 0.50 відповідно. Це вказує на низьку здатність моделі розрізняти ці стани, ймовірно через недостатню кількість або візуальну схожість з іншими хворобами.

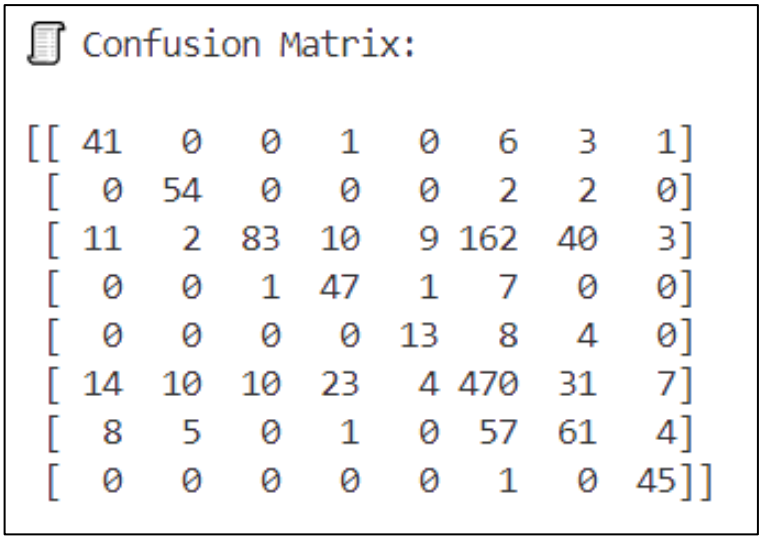


Рисунок 4.9 – Матриця плутанини

На матриці плутанини можна побачити що модель найчастіше плутає клас **Diabetes** та **Normal**, також є проблеми з класом **Other diseases/abnormalities**. З іншими класами глобальних проблем не відслідковується.

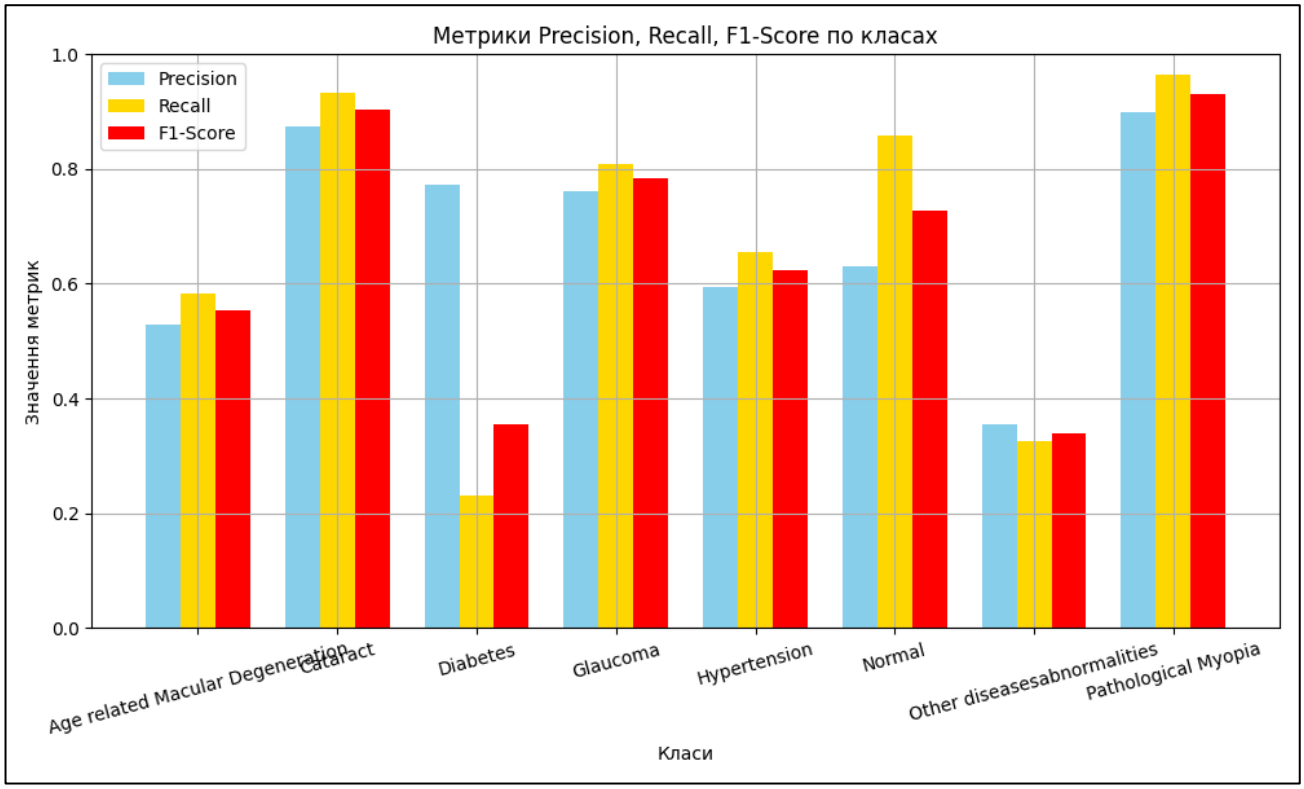


Рисунок 4.10 – Візуальне представлення метрик у вигляді діаграми

**Графік точності** демонструє зростання як навчальної, так і валідаційної точності, що є позитивним індикатором навчання. Навчальна точність стабільно покращується від приблизно 25% до понад 70%, а валідаційна — хоча й із більшими коливаннями — в кінці також сягає близько 65%. Хоча це гарні показники вони не є ідеальними.

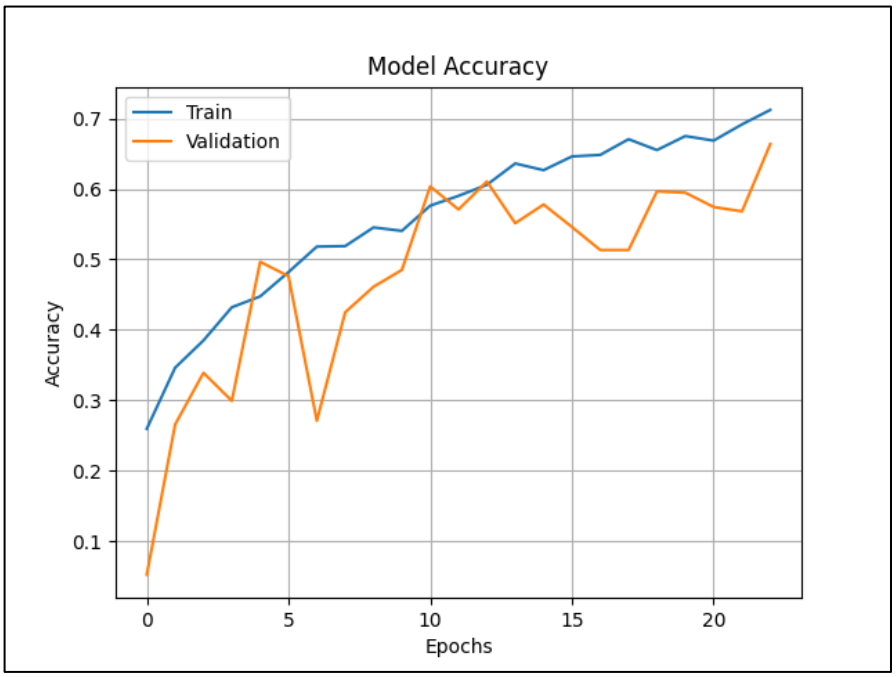


Рисунок 4.11 – Графік точності

**Графік втрат** демонструє іншу картину, на першій епосі значення валідаційної втрати дуже високе, після чого різко падає і далі коливається в межах 1 – 2. Це може свідчити про наявність одного або декількох викидів у валідаційних даних. Навчальні втрати поступово знижуються протягом усіх епох, що говорить про стабільне навчання без явного перенавчання. Однак певні коливання валідаційної втрати після 5-ї епохи можуть сигналізувати про чутливість моделі до змін у валідаційній вибірці.

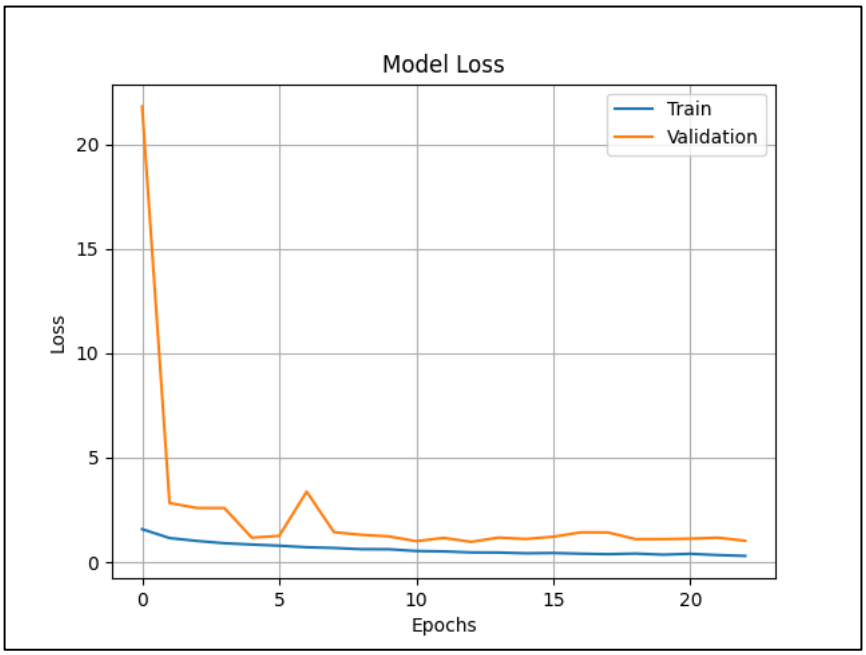


Рисунок 4.12 – Графік втрат

### 4.3 Інтерфейс та БД

Для взаємодії користувача з програмною системою було реалізовано графічний інтерфейс на основі бібліотеки **Gradio**. Основна мета інтерфейсу — надати зручний спосіб введення персональних даних пацієнта, завантаження зображень очей, автоматичної класифікації захворювань та перегляду результатів.

```
with gr.Blocks() as app:
    gr.Markdown("Класифікація хвороб ока")
```

Рисунок 4.13 – Компонент gr.Blocks

Інтерфейс побудовано за допомогою компонента *gr.Blocks*, який дозволяє групувати елементи в таби, рядки та колонки. У першій вкладці розміщено форму введення, яка містить такі поля:

– ім'я, прізвище, дата народження —текстові поля;

– стать — перемикач radio;
– скарги — багаторядкове текстове поле;
– завантаження зображень лівого та правого ока;
– кнопка для надсилання даних;
– поле для відображення результату.

```
with gr.Tab("Форма пацієнта"):
    with gr.Row():
        with gr.Column(scale=1):
            firstname = gr.Text(label="Ім'я")
            lastname = gr.Text(label="Прізвище")
            birthdate = gr.DateTime(label="Дата народження")
            gender = gr.Radio(["Чоловіча", "Жіноча"], label="Стать")
        with gr.Column(scale=3):
            complaints = gr.Textbox(label="Скарги", lines=14)
        with gr.Column(scale=1):
            result_output = gr.Textbox(label="Результат")
            submit_btn = gr.Button("Зберегти дані")
```

Рисунок 4.14 – Форма пацієнта

Зображення очей завантажуються через компоненти *gr.Image* з типом *PIL*, що дозволяє подальшу обробку зображень у Python. Кнопка надсилання викликає функцію *process_form*, яка відповідає за валідацію даних, класифікацію зображень та збереження результату.

```
with gr.Row():
    left_eye = gr.Image(type="pil", label="Зображення лівого ока")
    right_eye = gr.Image(type="pil", label="Зображення правого ока")
    submit_btn.click(process_form,
        inputs=[firstname, lastname, birthdate, gender, complaints, left_eye, right_eye],
        outputs=result_output)
```

Рисунок 4.15 – Завантаження зображень

Функція *process_form* містить логіку перевірки введених даних, викликає модель для класифікації кожного зображення окремо, а також зберігає результат у базу даних.

```python
def process_form(firstname, lastname, birthdate_str, gender, complaints, left_eye_img, right_eye_img):
    if not firstname.strip():
        return " Поле 'Ім'я' не може бути порожнім."
    if not lastname.strip():
        return " Поле 'Прізвище' не може бути порожнім."
    if not gender:
        return " Виберіть стать."
    if not left_eye_img:
        return " Завантажте зображення лівого ока."
    if not right_eye_img:
        return " Завантажте зображення правого ока."

    left_label, left_conf = classify_image(left_eye_img)
    right_label, right_conf = classify_image(right_eye_img)

    save_to_db(result)
    return f""" Дані збережено.
    Ліве око: {left_label} ({left_conf * 100:.2f}%)
    Праве око: {right_label} ({right_conf * 100:.2f}%)"""
```

Рисунок 4.16 – Перевірка на заповненість полів

У межах системи також реалізована **база даних** для збереження та подальшого пошуку інформації про пацієнтів. База даних представлена у вигляді звичайного JSON-файлу *database.json*, що зберігається в директорії *./data/*. Такий підхід є зручним, не потребує окремого серверного ПЗ і дозволяє легко зчитувати, редагувати або передавати дані.

У модулі бази даних реалізовано дві основні стандартні функції:

– *save_to_db(data)* — додає новий запис до JSON-файлу. Дані додаються у вигляді словника Python зі збереженням попередніх записів;

– *search_db(query)* — здійснює пошук за прізвищем пацієнта.

Цього функціоналу достатньо для базової роботи із збереженням результатів класифікації та пошуком даних у вкладці «Пошук пацієнтів».

```
[
    {
        "Ім'я": "Дмитро",
        "Прізвище": "Вартовий",
        "Дата народження": "1992-06-05",
        "Стать": "Чоловіча",
        "Скарги": "Блики в очах\nПідвищений тиск",
        "Дата обстеження": "2025-06-02 17:29:04",
        "Діагноз Лівого Ока": {
            "хвороба": "Normal",
            "впевненість": 0.8
        },
        "Діагноз Правого Ока": {
            "хвороба": "Cataract",
            "впевненість": 0.67
        }
    },
```


Рисунок 4.17 – Запис в json-файлі

## 4.3 Тестування

Готове головне вікно зображено на рис. 4.18.

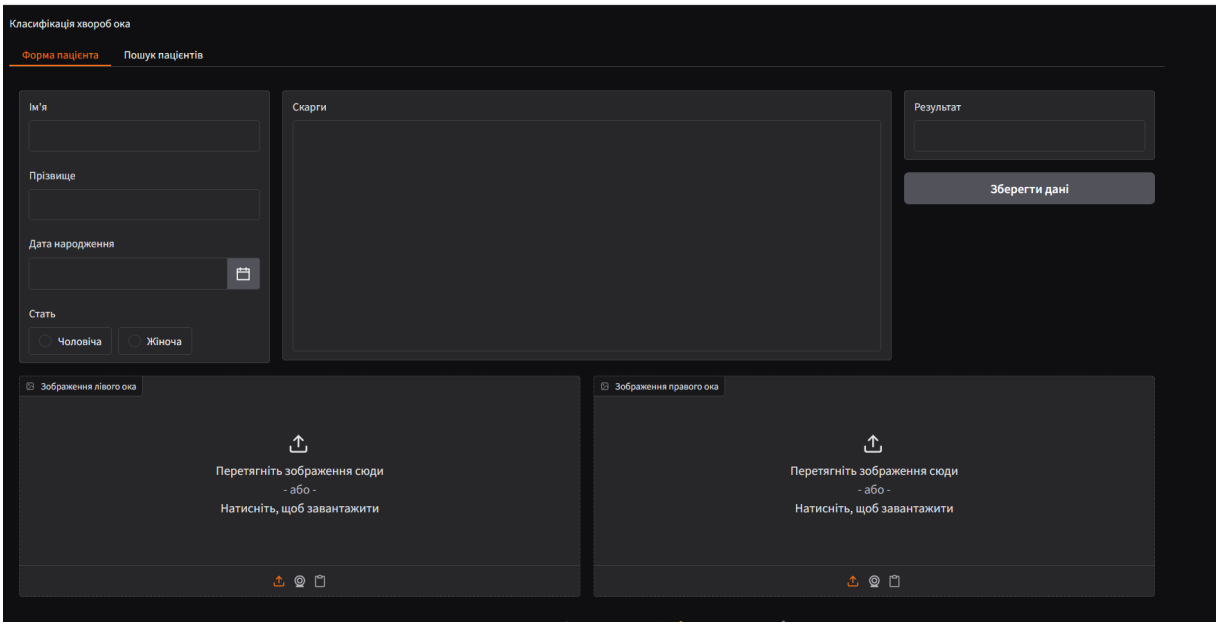


Рисунок 4.18 – Головне вікно

На головному вікні лікар вводить дані пацієнта: ПІБ, дату народження, стать та записує скарги для запису в базу даних пацієнтів. Після обирає зображення очей та натискає, зберегти дані.

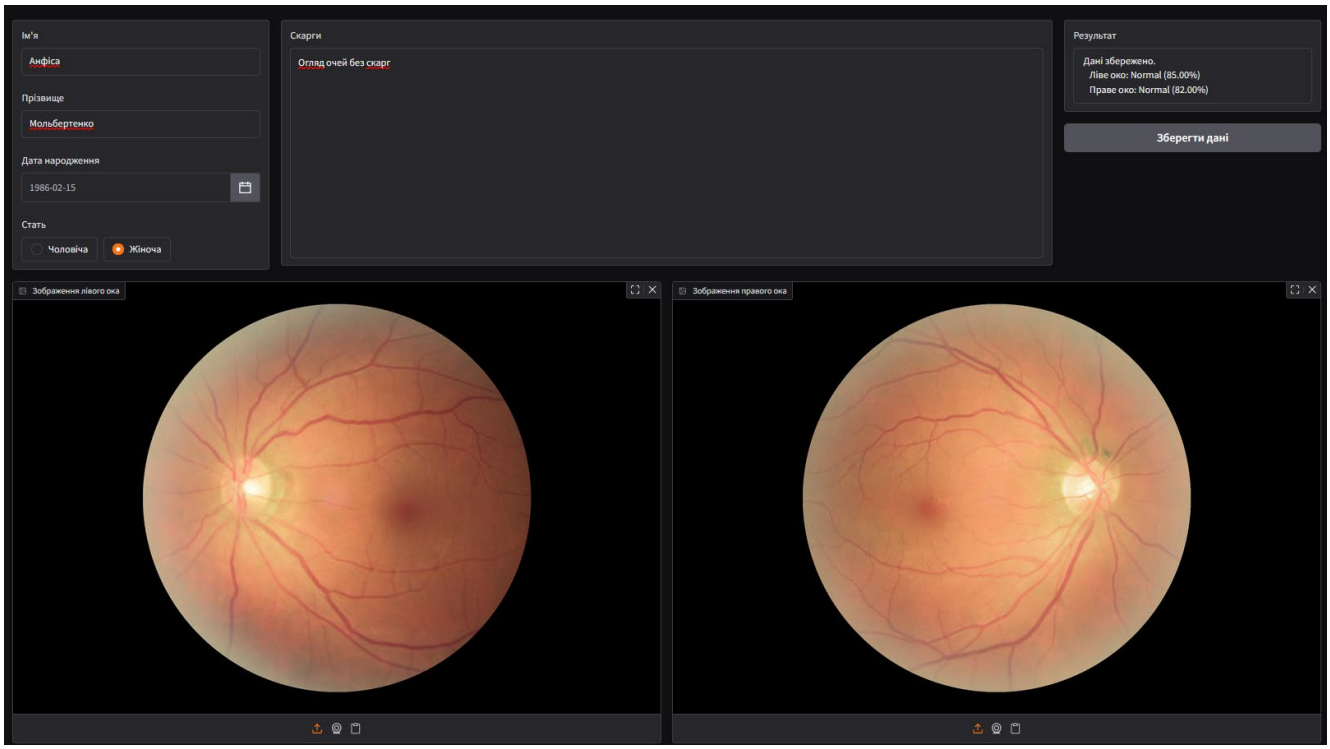


Рисунок 4.19 – Вікно з заповненими даними

Після натискання, викликається функція прогнозування та лікар може бачити діагноз та його вірогідність який передбачила інтелектуальна система. Вигляд повідомлення про незаповнене поле на рис. 4.20.

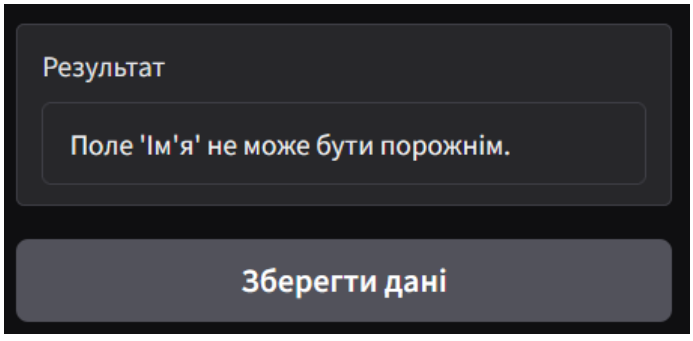


Рисунок 4.20 – Повідомлення про пусте поле

Друга вкладка призначена для пошуку пацієнтів за прізвищем. Вона містить поле введення запиту, кнопку пошуку та таблицю з результатами. При натисканні кнопки «пошук» викликається функція, яка звертається до бази даних та шукає відповідні результати.

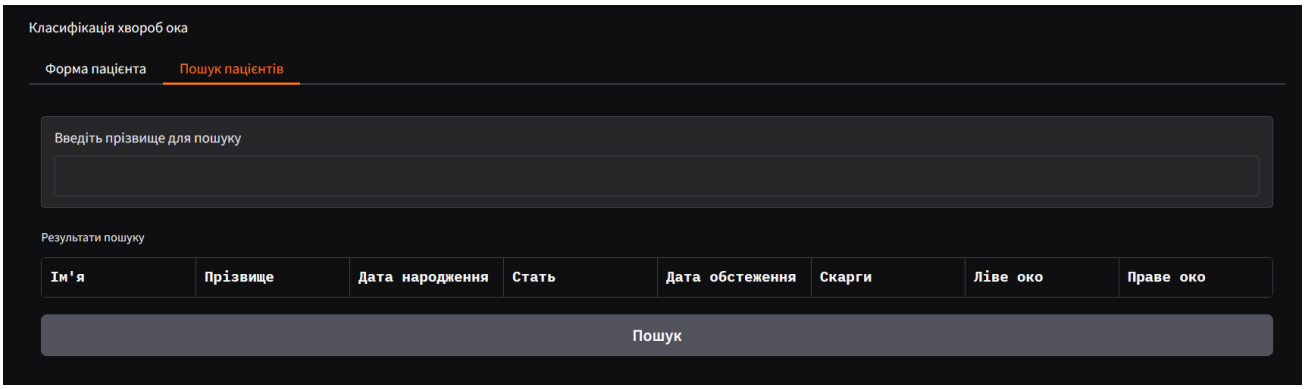


Рисунок 4.21 – Вікно пошуку пацієнтів

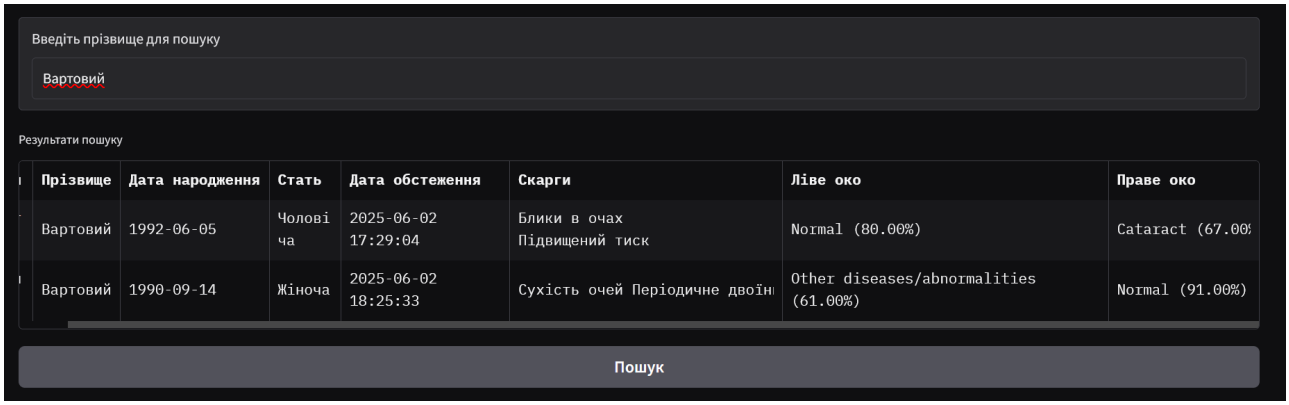


Рисунок 4.22 – Пацієнти з прізвищем «Вартовий»

Також бібліотке Gradio на якій і розроблений графічний інтерфейс дозволяє обрати між світлою та темною темою

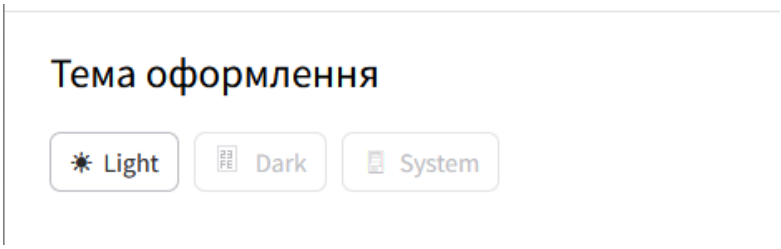


Рисунок 4.23 – Вибір теми оформлення

Під час тестування виявлено проблему з меню дати народження при використанні браузера «Firefox», а саме їх зберігання, при обиранні дати, вона не зберігається в рядок зліва, лише рукописний ввід. При використанні «Chrome»

такої проблеми не виявлено.

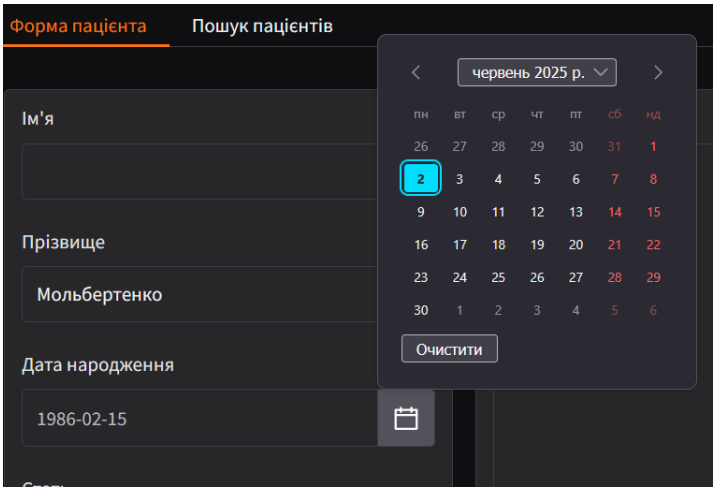


Рисунок 4.24 – Проблеми зі збереженням дати народження при використанні «Firefox»

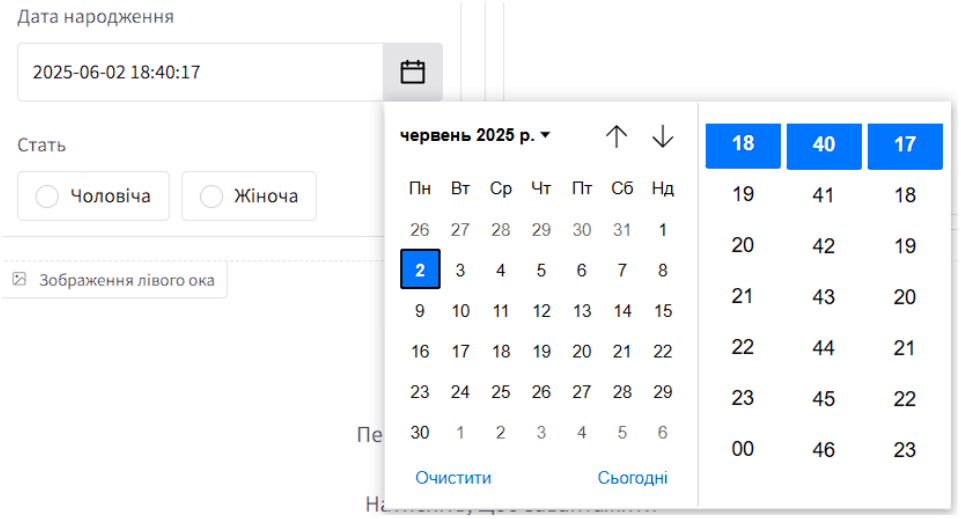


Рисунок 4.25 – Приклад правильної роботи у браузері «Chrome»

Отже, після тестування, можна сказати, що модель класифікації працює з помірною точністю, може надавати допомогу в розпізнанні та діагностуванні хвороб ока та через велику кількість та схожість деяких ідентифікованих хвороб може помилятись, тому не є самостійним медичним застосунком. Використання більш точного датасету та перенавчання моделі можуть покращити якість розпізнавання моделі

## Висновки до розділу 4

У цьому розділі було детально розглянуто процес програмної реалізації, навчання та тестування моделі глибокого навчання для класифікації зображень очного дна. Основою моделі стала попередньо натренована нейронна мережа ResNet50, яка була адаптована до задачі за допомогою тонкого перенавчання та додавання спеціалізованих шарів класифікації. Було застосовано низку технік для покращення ефективності: аугментація, балансування класів, Dropout та EarlyStopping, що дозволило досягти загальної точності класифікації на рівні 65%.

У ході гіперпараметричної оптимізації було випробувано різні конфігурації моделі, серед яких найкраща продемонструвала значення F1-міри 0.62. Найвищу точність виявлення продемонстрували класи ***Pathological Myopia*** та ***Cataract***, тоді як моделі було складно розрізняти зображення класів ***Other diseases/abnormalities*** та ***Hypertension***, що може бути пов’язано з їх недостатньою представленістю у вибірці або візуальною схожістю з іншими станами.

Також було реалізовано графічний інтерфейс користувача на основі Gradio, який забезпечує зручну взаємодію з системою: введення даних пацієнта, завантаження зображень, автоматичну класифікацію та збереження результатів у локальну JSON-базу даних. Це дає змогу використовувати систему в практичних умовах без необхідності розгортання серверної інфраструктури.

# ВИСНОВКИ

Під час виконання кваліфікаційної роботи було розроблено інформаційну систему для діагностики хвороб ока з використанням нейронних мереж.

На першому етапі було проведено дослідження предметної області, а саме, розглянуто найбільш розповсюджені захворювання очей їх симптоматику, методи лікування та діагностики. Досліджено сучасні системи діагностики з використанням нейронних мереж, та на основі цього визначено основні проблеми та завдання

Другий розділ присвячений опису математичних моделей та вибору технологій для створення системи та методів машинного навчання. Обґрунтовано вибір підходу навчання з учителем та обрано архітектуру ResNet50 яка поєднує глибоку структуру, залишкові з'єднання, високу точність класифікації та можливість трансферного навчання для класифікації хвороб ока.

У третьому розділі було розглянуто архітектуру, компоненти та основні етапи функціонування інформаційної, орієнтованої на обробку цифрових медичних зображень. Представлено діаграму послідовностей. Особливу увагу приділено аналізу та процесу попередньої обробки зображень, обрано датасет ODIR.

Четвертий етап охоплює процес програмної реалізації, навчання та тестування моделі глибокого навчання для класифікації зображень. У ході гіперпараметричної оптимізації було випробувано різні конфігурації моделі. Також було реалізовано графічний інтерфейс користувача та збереження результатів базу даних. Проведено тестування системи.

Загалом розроблена система демонструє функціональну готовність до використання та надає основу для подальшого вдосконалення — зокрема, шляхом розширення датасету, доопрацювання архітектури моделі та розширення інтерфейсу користувача.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. About Common Eye Disorders and Diseases. Vision and Eye Health. URL: https://www.cdc.gov/vision-health/about-eye-disorders/index.html (date of access: 15.04.2025).

2. At risk of diabetes-related vision loss?-Diabetic retinopathy - Symptoms & causes - Mayo Clinic. Mayo Clinic. URL: https://www.mayoclinic.org/diseases-conditions/diabetic-retinopathy/symptoms-causes/syc-20371611 (date of access: 15.04.2025).

3. NHS website. Glaucoma. nhs.uk. URL: https://www.nhs.uk/conditions/glaucoma/ (date of access: 15.04.2025).

4. øyesykdommer – Store medisinske leksikon. Store medisinske leksikon. URL: https://sml.snl.no/øyesykdommer (date of access: 15.04.2025).

5. Що таке глаукома. Офтальмологічний кабінет «Світло». URL: http://svitlomed.com.ua/shcho-take-glaukoma/ (дата звернення: 16.04.2025).

6. Cataract. American Optometric Association (AOA) | Doctors of Optometry. URL: https://www.aoa.org/healthy-eyes/eye-and-vision-conditions/cataract (date of access: 16.04.2025).

7. 葡萄膜炎并发性白内障怎么治疗?会自愈吗?这篇文章为你解答白内障手术 - 非常爱美. 眼科咨询排名_近视眼手术 - 非常爱美. URL: https://yanke.verym.com/baineizhang/5487.html (дата звернення: 16.04.2025).

8. Age-Related Macular Degeneration (AMD) | National Eye Institute. National Eye Institute | National Eye Institute. URL: https://www.nei.nih.gov/learn-about-eye-health/eye-conditions-and-diseases/age-related-macular-degeneration (date of access: 17.04.2025).

9. High blood pressure. British Heart Foundation. URL: https://www.bhf.org.uk/informationsupport/risk-factors/high-blood-pressure (date of access: 17.04.2025).

10. How to Manage Pathologic Myopia. Review of Ophthalmology – Monthly

Publication for Ophthalmologists. URL: https://www.reviewofophthalmology.com/article/how-to-manage-pathologic-myopia?utm_source=chatgpt.com (date of access: 18.04.2025).

11. IDx-DR - Healthvisors. Healthvisors. URL: https://www.healthvisors.com/en/idx-dr/ (date of access: 18.04.2025).

12. Vincent J. DeepMind’s AI can detect over 50 eye diseases as accurately as a doctor. The Verge. URL: https://www.theverge.com/2018/8/13/17670156/deepmind-ai-eye-disease-doctor-moorfields (date of access: 20.04.2025).

13. EyeArt. Eyenuk, Inc. ~ Artificial Intelligence Eye Screening. URL: https://www.eyenuk.com/en/products/eyeart/ (date of access: 21.04.2025).

14. Application of deep learning algorithms for diabetic retinopathy screening - PMC. PMC Home. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC9843336/ (date of access: 21.04.2025).

15. Explainable artificial intelligence for the automated assessment of the retinal vascular tortuosity / Á. S. Hervella et al. Medical & Biological Engineering & Computing. 2023. URL: https://doi.org/10.1007/s11517-023-02978-w (date of access: 21.04.2025).

16. Deep Transfer Learning-Based Automated Diabetic Retinopathy Detection Using Retinal Fundus Images in Remote Areas - International Journal of Computational Intelligence Systems. SpringerLink. URL: https://link.springer.com/article/10.1007/s44196-024-00520-w (date of access: 25.04.2025).

17. Субботін С. О. Нейронні мережі: теорія та практика : навч. посіб. Житомир : О. О. Євенок, 2020. 184 с.

18. Sciencedirect. Medical image identification methods: A review. URL: https://doi.org/10.1016/j.compbiomed.2023.107777 (date of access: 07.05.2025).

19. IBM. What are Convolutional Neural Networks? | IBM. IBM - United States. URL: https://www.ibm.com/think/topics/convolutional-neural-networks (date of access: 08.05.2025).

20. Які основні компоненти згорткової нейронної мережі (CNN) і як вони сприяють розпізнаванню зображень? - Академія EITCA. EITCA Academy. URL: https://uk.eitca.org/artificial-intelligence/eitc-ai-dltf-deep-learning-with-tensorflow/convolutional-neural-networks-in-tensorflow/convolutional-neural-networks-basics/examination-review-convolutional-neural-networks-basics/what-are-the-main-components-of-a-convolutional-neural-network-cnn-and-how-do-they-contribute-to-image-recognition/ (дата звернення: 10.05.2025).

21. API Documentatio|TensorFlow. URL: https://www.tensorflow.org/api_docs (date of access: 10.05.2024).

22. Keras documentation: The Model class. Keras: Deep Learning for humans. URL: https://keras.io/api/models/model/ (date of access: 10.05.2024)

23. Gradio Docs. Gradio. URL: https://www.gradio.app/main/docs/gradio/datetime (date of access: 11.05.2025).

24. GeeksforGeeks. VGG-16 | CNN model - GeeksforGeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/vgg-16-cnn-model/ (date of access: 12.05.2025).

25. VGG16 - Convolutional Network for Classification and Detection. Neurohive - Нейронні мережі. URL: https://neurohive.io/en/popular-networks/vgg16/ (date of access: 12.05.2025).

26. T A. N. Inception V3 Model Architecture. OpenGenus IQ: Learn Algorithms, DL, System Design. URL: https://iq.opengenus.org/inception-v3-model-architecture/ (date of access: 12.05.2025).

27. Ahmed A. Architecture of DenseNet-121. OpenGenus IQ: Learn Algorithms, DL, System Design. URL: https://iq.opengenus.org/architecture-of-densenet121/ (date of access: 12.05.2025).

28. Potrimba P. What is ResNet-50?. Roboflow Blog. URL: https://blog.roboflow.com/what-is-resnet-50/ (date of access: 12.05.2025).

29. Sapireddy S. R. ResNet-50: Introduction. Medium. URL: https://srsapireddy.medium.com/resnet-50-introduction-b5435fdba66f (date of access: 12.05.2025).

30. Maxym Z. Діаграма послідовності (Sequence Diagrams). Maxym Zosym. URL: https://www.maxzosim.com/sequence-diagrams/ (дата звернення: 14.05.2025).

31. Larxel. Ocular Disease Recognition. URL: https://www.kaggle.com/datasets/andrewmvd/ocular-disease-recognition-odir5k (date of access: 15.05.2025).

32. Класифікація: точність, повнота, влучність і пов’язані метрики | Machine Learning | Google for Developers. Google for Developers. URL: https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall?hl=uk (дата звернення: 20.05.2025).

# ДОДАТОК А

## Лістинг коду

main.py:

```
from interface import build_interface

if __name__ == "__main__":
    app = build_interface()
    app.launch()
```

database.py:

```
import json
import os
import uuid
import cv2
from datetime import datetime
DB_PATH = "./data/database.json"
SAVE_DIR = "./saved_inputs"
os.makedirs(SAVE_DIR, exist_ok=True)
def save_debug_image(img, suffix):
    filename =
f"{datetime.now().strftime('%Y%m%d_%H%M%S')}_{suffix}_{uuid.uuid4().hex[:6]
}.jpg"
    path = os.path.join(SAVE_DIR, filename)
    cv2.imwrite(path, img)
    return path
def save_to_db(data):
    if not os.path.exists(DB_PATH):
        with open(DB_PATH, "w") as f:
            json.dump([], f)
    with open(DB_PATH, "r+", encoding="utf-8") as f:
        db = json.load(f)
        db.append(data)
        f.seek(0)
        json.dump(db, f, indent=4, ensure_ascii=False)
def search_db(query):
    if not os.path.exists(DB_PATH):
        return []
    with open(DB_PATH, "r", encoding="utf-8") as f:
        db = json.load(f)
    return [
        entry for entry in db
        if query.lower() in entry.get("Прізвище", "").lower()
    ]
```

interface.py:

```
import gradio as gr
from datetime import datetime
```

```
from predictor import classify_image
from database import save_to_db, search_db
def process_form(firstname, lastname, birthdate_str, gender, complaints,
left_eye_img, right_eye_img):
    if not firstname.strip(): return " Поле 'Ім'я' не може бути порожнім."
    if not lastname.strip(): return " Поле 'Прізвище' не може бути
порожнім."
    if not gender: return " Виберіть стать."
    if not left_eye_img: return " Завантажте зображення лівого ока."
    if not right_eye_img: return " Завантажте зображення правого ока."
    left_label, left_conf = classify_image(left_eye_img, True)
    right_label, right_conf = classify_image(right_eye_img)
    result = {
        "Ім'я": firstname,
        "Прізвище": lastname,
        "Дата народження":
datetime.fromtimestamp(birthdate_str).strftime("%Y-%m-%d") if birthdate_str
else "",
        "Стать": gender,
        "Скарги": complaints,
        "Дата обстеження": datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
        "Діагноз Лівого Ока": {
            "хвороба": left_label,
            "впевненість": left_conf
        },
        "Діагноз Правого Ока": {
            "хвороба": right_label,
            "впевненість": right_conf
        }
    }
    save_to_db(result)
    return f""" Дані збережено.
    Ліве око: {left_label} ({left_conf * 100:.2f}%)
    Праве око: {right_label} ({right_conf * 100:.2f}%)"""
def build_interface():
    with gr.Blocks() as app:
        gr.Markdown("Класифікація хвороб ока")
        with gr.Tab("Форма пацієнта"):
            with gr.Row():
                with gr.Column(scale=1):
                    firstname = gr.Text(label="Ім'я")
                    lastname = gr.Text(label="Прізвище")
                    birthdate_str = gr.DateTime(label="Дата народження")
                    gender = gr.Radio(["Чоловіча", "Жіноча"],
label="Стать")
                with gr.Column(scale=3):
                    complaints = gr.Textbox(label="Скарги", lines=14)
                with gr.Column(scale=1):
                    result_output = gr.Textbox(label="Результат")
```

```
                    submit_btn = gr.Button("Зберегти дані")
            with gr.Row():
                left_eye = gr.Image(type="pil", label="Зображення лівого
ока")
                right_eye = gr.Image(type="pil", label="Зображення правого
ока")
                submit_btn.click(process_form,
                    inputs=[firstname, lastname, birthdate_str, gender,
complaints, left_eye, right_eye],
                    outputs=result_output)
        with gr.Tab("Пошук пацієнтів"):
            query = gr.Text(label="Введіть прізвище для пошуку")
            search_result = gr.Dataframe(
                headers=["Ім'я", "Прізвище", "Дата народження", "Стать",
"Дата обстеження",
                         "Скарги", "Ліве око", "Праве око"],
                col_count=(8),
                label="Результати пошуку",
                interactive=False
            )
            search_btn = gr.Button("Пошук")
            def _search(query):
                entries = search_db(query)
                rows = [
                    [   e.get("Ім'я", ""), e.get("Прізвище", ""),
e.get("Дата народження", ""), e.get("Стать", ""),
                        e.get("Дата обстеження", ""), e.get("Скарги", ""),
                        f"{e['Діагноз Лівого Ока']['хвороба']} ({e['Діагноз
Лівого Ока']['впевненість']*100:.2f}%)",
                        f"{e['Діагноз Правого Ока']['хвороба']}
({e['Діагноз Правого Ока']['впевненість']*100:.2f}%)"
]
                    for e in entries
                ]
                return gr.update(visible=True, value=rows)
            search_btn.click(fn=_search, inputs=query,
outputs=search_result)
    return app
predictor.py:

import numpy as np
import cv2
from PIL import Image
import tensorflow as tf
from database import save_debug_image
class_names = [
    'Age related Macular Degenerationmal', 'Cataract', 'Diabets',
'Glaucoma',
```

```
    'Hypertension', 'Normal', 'Other diseases/abnormalities', 'Pathological
myopia']
model_path = './models/ModelResNet5065.keras'
model = tf.keras.models.load_model(model_path)
def crop_black_borders(img, threshold=10):
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    _, thresh = cv2.threshold(gray, threshold, 255, cv2.THRESH_BINARY)
    coords = cv2.findNonZero(thresh)
    if coords is None:
        return None
    x, y, w, h = cv2.boundingRect(coords)
    return img[y:y+h, x:x+w]
def classify_image(image: Image.Image, is_left_eye: bool = False):
    img = np.array(image.convert("RGB"))
    img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)
    save_debug_image(img, "original")
    if is_left_eye:
        img = cv2.flip(img, 1)
        save_debug_image(img, "flipped")
    cropped = crop_black_borders(img)
    if cropped is None:
        return "Не вдалося обробити зображення", 0.0
    save_debug_image(cropped, "cropped")
    resized = cv2.resize(cropped, (224, 224))
    save_debug_image(resized, "resized")
    img_array = resized.astype(np.float32)
    img_array = np.expand_dims(img_array, axis=0)
    predictions = model.predict(img_array, verbose=0)[0]
    predicted_index = np.argmax(predictions)
    label = class_names[predicted_index]
    confidence = round(float(predictions[predicted_index]), 2)
    return label, confidence
```

test_model.py:

```
import os
import numpy as np
import tensorflow as tf
from sklearn.metrics import classification_report, confusion_matrix
model_path = './models/ModelResNet5065.keras'
data_dir = './sorted_images/'
img_size = (224, 224)
batch_size = 32
output_dir = './diagrams'
plot_filename = os.path.join(output_dir, 'class_metrics_plot.png')
print(f"Завантаження моделі з {model_path}")
if not os.path.exists(model_path):
    print(f"Файл моделі не знайдено: {model_path}")
    exit()
try:
```

```
    model = tf.keras.models.load_model(model_path)
except Exception as e:
    print(f"Помилка при завантаженні моделі: {e}")
    exit()
print("Завантаження валідаційного датасету...")
try:
    validation_ds = tf.keras.preprocessing.image_dataset_from_directory(
        data_dir,
        validation_split=0.2,
        subset="validation",
        seed=123,
        image_size=img_size,
        batch_size=batch_size
    )
except Exception as e:
    print(f"Помилка при завантаженні датасету: {e}")
    exit()
class_names = validation_ds.class_names
print(f"Класи: {class_names}")
print("Генерація передбачень...")
y_true, y_pred = [], []
for images, labels in validation_ds:
    preds = model.predict(images, verbose=0)
    y_pred.extend(np.argmax(preds, axis=1))
    y_true.extend(labels.numpy())
print("\nClassification Report:\n")
report = classification_report(y_true, y_pred, target_names=class_names,
digits=4)
print(report)
print("\nConfusion Matrix:\n")
print(confusion_matrix(y_true, y_pred))
```

train_model.py:

```
import os
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers, models, Sequential
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.optimizers import Adam
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.utils import class_weight
import matplotlib.pyplot as plt
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
data_dir = 'sorted_images/'
img_size = (224, 224)
batch_size = 32
epochs = 30
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir,
```

```
    validation_split=0.2,
    subset="training",
    seed=123,
    image_size=img_size,
    batch_size=batch_size)
validation_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir,
    validation_split=0.2,
    subset="validation",
    seed=123,
    image_size=img_size,
    batch_size=batch_size
)
class_names = train_ds.class_names
num_classes = len(class_names)
all_labels = np.concatenate([y.numpy() for x, y in train_ds])
class_weights = class_weight.compute_class_weight(
    class_weight='balanced',
    classes=np.unique(all_labels),
    y=all_labels
)
class_weights = dict(enumerate(class_weights))
data_augmentation = tf.keras.Sequential([
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.1),
    layers.RandomContrast(0.1),
    layers.RandomBrightness(0.1),
])
base_model = ResNet50(include_top=False, input_shape=(224, 224, 3),
pooling='avg', weights='imagenet')
base_model.trainable = True
fine_tune_at = 110
for layer in base_model.layers[:fine_tune_at]:
    layer.trainable = False
model = Sequential([
    data_augmentation,
    base_model,
    layers.Dense(256, activation='relu'),
    layers.Dropout(0.2),
    layers.Dense(num_classes, activation='softmax')
])
model.compile(
    optimizer=Adam(learning_rate=0.001),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
early_stop = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
patience=10, restore_best_weights=True)
history = model.fit(
```

```
    train_ds,
    validation_data=validation_ds,
    epochs=epochs,
    class_weight=class_weights,
    callbacks=[early_stop]
)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epochs')
plt.legend(['Train', 'Validation'])
plt.grid()
plt.savefig("accuracy_plot.png")
plt.clf()
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(['Train', 'Validation'])
plt.grid()
plt.savefig("loss_plot.png")
y_true, y_pred = [], []
for images, labels in validation_ds:
    preds = model.predict(images)
    y_pred.extend(np.argmax(preds, axis=1))
    y_true.extend(labels.numpy())
print("\nClassification Report:\n")
print(classification_report(y_true, y_pred, target_names=class_names))
print("\nConfusion Matrix:\n")
print(confusion_matrix(y_true, y_pred))
model.save('fine_tuned_model.keras')
```