Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно–аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА СИСТЕМИ ДЛЯ ЗАБЕЗПЕЧЕННЯ ПЕРЕНАВЧАННЯ НЕЙРОННИХ СТРУКТУР МОЗКУ ЩОДО ФУНКЦІОНУВАННЯ ШЛЯХІВ ПЕРЕДАЧІ ЗОБРАЖЕННЯ ВІД ЗОРОВОГО НЕРВА
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ–20–1
Гема О.Г.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо–професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник проф. Кузьомін О.Я.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно–аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові _____Гемі Олександру Геннадійовичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка системи для забезпечення перенавчання нейронних структур мозку щодо функціонування шляхів передачі зображення від зорового нерва

затверджена наказом університету від 20 травня 2024 року № 464 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 02 травня 2024 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, бібліотека комп'ютерного зору з відкритим кодом OpenCV, програмне забезпечення для розробки та навчання моделей глибокого навчання Tensorflow, інтерфейс нейронних мереж для TensorFlow Keras, фреймворк для розробки вебзастосунків Flask.

4. Перелік питань, що потрібно опрацювати в роботі

1. Огляд предметної області зорового шляху людини.

2. Огляд можливостей використання штучного інтелекту у аналізі зорового шляху людини.

3. Розроблення системи для аналізу зорового шляху людини.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми розробки та вдосконалення систем штучного інтелекту, які дозволять навчити нейронні мережі відтворювати та аналізувати процеси передачі зображення від зорового нерву, обробки зображень, діаграма DFD вебзастосунку, графіки точності та втрат під час навчання моделей, тестові зображення фундусу ока.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	08.04.2024	
2	Аналіз завдання, підбір літератури	08.04.24-15.04.24	
3	Аналіз літератури з досліджуваної проблеми	16.04.24-18.04.24	
4	Аналіз технічних засобів	19.04.24-25.04.24	
5	Розробка методу	26.04.24-14.05.24	
6	Програмна реалізація	15.05.24-23.05.24	
7	Оформлення пояснювальної записки	24.05.24-26.05.24	
8	Перевірка на плагіат	27.05.24	
9	Рецензування	28.05.24	
10	Підготовка презентації та доповіді	29.05.24-02.06.24	
11	Занесення роботи в електронний архів	03.06.24	
12	Попередній захист кваліфікаційної роботи	12.06.24	

Дата видачі завдання 8 квітня 2024 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Кузьомін О.Я.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 84 с., 37 рис., 44 джерела.

НЕЙРОННІ МЕРЕЖІ, ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, ГЕНЕРАТИВНО-ЗМАГАЛЬНІ МЕРЕЖІ, МОЗКОВІ СТРУКТУРИ, ШЛЯХИ ПЕРЕДАЧІ ЗОБРАЖЕННЯ, ЗОРОВИЙ НЕРВ, ПЕРЕНАВЧАННЯ НЕЙРОННИХ СТРУКТУР, ШТУЧНИЙ ІНТЕЛЕКТ, ФУНКЦІОНУВАННЯ ЗОРОВОЇ СИСТЕМИ, ОПТИМІЗАЦІЯ ПЕРЕДАЧІ ІНФОРМАЦІЇ, ІМІТАЦІЯ МОЗКОВИХ ПРОЦЕСІВ, АНАЛІЗ ЗОБРАЖЕНЬ.

Об’єктом роботи є розробка системи для забезпечення перенавчання нейронних структур мозку щодо функціонування шляхів передачі зображення від зорового нерву. Це охоплює дослідження і аналіз функціональних шляхів зорової системи, виявлення патологій та створення алгоритмів для їх діагностики.

Метою роботи є створення системи, яка за допомогою класифікації та аналізу зорового шляху людини може допомогти у поліпшенні розуміння функціонування шляхів передачі зображення від зорового нерву та навчання нейронних структур мозку оптимальним шляхам передачі зображення від зорового нерву.

Результатом роботи є створення функціональної системи, яка може ефективно аналізувати проблеми зорового шляху людини.

NEURAL NETWORKS, CONVOLUTIONAL NEURAL NETWORKS, GENERATIVE ADVERSARIAL NETWORKS, BRAIN STRUCTURES, IMAGE TRANSMISSION PATHS, OPTIC NERVE, NEURAL STRUCTURE RETRAINING, ARTIFICIAL INTELLIGENCE, FUNCTIONING OF THE VISUAL SYSTEM, INFORMATION TRANSMISSION OPTIMIZATION, BRAIN PROCESS IMITATION, IMAGE ANALYSIS.

The object of the work is the development of a system to facilitate retraining of neural brain structures regarding the functioning of pathways for image transmission from the optic nerve.

The aim of the work is to create a system that through classification and analysis of the human visual pathway, can aid in improving understanding of the functioning of pathways for image transmission from the optic nerve and training neural brain structures on optimal pathways for image transmission from the optic nerve.

The result of the work is the creation of a functional system capable of effectively analyzing issues with the human visual pathway.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ШІ − штучний інтелект

CNN – Convolutional Neural Network (згорткова нейронна мережа)

RNN – Recurrent Neural Network (рекурентна нейронна мережа)

PCA – Principal Component Analysis (метод головних компонент)

GAN – Generative Adversarial Network (генеративно–змагальна мережа)

LGN – Lateral Geniculate Nucleus (латеральне колінчасте ядро)

CPU – Central Processing Unit (центральний процесор)

GPU – Graphics Processing Unit (графічний процесор)

API – Application Programming Interface (інтерфейс прикладного програмування)

IDRiD – Indian Diabetic Retinopathy Image Dataset (Індійський набір даних зображень діабетичної ретинопатії)

HRF – High-Resolution Fundus (високоякісне зображення очного дна)

MVC – Model-View-Controller (модель-вид-контролер)

ВСТУП

Розумова система людини залишається однією з найбільш складних і загадкових областей науки. Функціонування шляхів передачі зображення від зорового нерву, зокрема, привертає увагу дослідників з різних галузей науки, включаючи нейробіологію, штучний інтелект та інженерію систем.

Розробка систем для забезпечення перенавчання нейронних структур мозку має потенціал стати ключовим напрямком досліджень у майбутньому. Подальші вдосконалення цих систем можуть принести нові світлі ідеї та рішення, що допоможуть покращити якість життя людей та розвинути сучасні технології до рівня, недосяжного раніше.

Попри значний прогрес у дослідженнях зорового нерву та штучного інтелекту, багато аспектів механізмів передачі зорової інформації залишаються невивченими або недостатньо зрозумілими. Наприклад, процеси обробки та аналізу зорової інформації все ще викликають багато питань, особливо в контексті відновлення функцій зору після травм або захворювань.

Дослідження у цьому напрямку мають великий потенціал не тільки для науки, а й для медицини та технологічного розвитку. Подальші відкриття у цій галузі можуть відкрити двері до нових методів лікування та реабілітації, а також до створення більш ефективних та просунутих систем штучного інтелекту для обробки зорової інформації.

Актуальність роботи полягає у необхідності розробки та вдосконалення алгоритмів та систем штучного інтелекту, які дозволять навчити нейронні мережі відтворювати та аналізувати процеси передачі зображення від зорового нерву. Вивчення цих механізмів не тільки сприятиме кращому розумінню принципів роботи мозку, але й відкриє нові можливості для розвитку методів діагностики та лікування різних захворювань зору, таких як глаукома, діабетична ретинопатія та деякі форми сліпоти.

1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Теоретичний аналіз ключових понять та концепцій

Нейронні мережі, або штучні нейронні мережі, є математичними моделями, які намагаються імітувати функціонування нейронних мереж у біологічних системах. Вони складаються зі штучних нейронів, які сполучені між собою і можуть передавати сигнали один одному.

Однією з ключових особливостей нейронних мереж є їхні здатності до самостійного навчання з даних. Це означає, що вони можуть адаптуватися до нових вхідних даних та покращувати свою продуктивність з часом. Величезна кількість параметрів нейронних мереж дає їм потенціал для моделювання складних залежностей у даних.

Одним з ключових напрямків розвитку нейронних мереж є збільшення їхньої глибини та складності. Глибокі нейронні мережі здатні автоматично визначати та аналізувати складні шаблони у даних, що призводить до покращення їхньої ефективності та точності.

Застосування нейронних мереж стало широко поширеним у багатьох галузях, зокрема в обробці зображень, розпізнаванні мови, природній мові, медицині, фінансах та інших. Нейронні мережі відкривають нові можливості в аналізі даних та автоматизації процесів, що раніше вважалися складними або неможливими для комп’ютерів.

Особливо важливе значення нейронні мережі мають в обробці зображень. Завдяки своїй здатності визначати шаблони та властивості у великих обсягах вхідних даних, вони використовуються для задач таких як розпізнавання обличь, класифікація об’єктів на зображеннях, автоматична обробка медичних знімків та багато інших.

1.1.1 Архітектури нейронних мереж для аналізу зображень

Аналіз зображень став невід’ємною частиною сучасних технологій, зокрема в областях комп’ютерного зору, медичної діагностики, автономних транспортних засобів та багатьох інших. Для ефективного аналізу зображень широко використовуються різні архітектури нейронних мереж, які спеціалізуються на різних аспектах обробки та розпізнавання зображень.

Два найбільш популярних типи це згорткові нейронні мережі (CNN) та рекурентні нейронні мережі (RNN).

Згорткові нейронні мережі є потужним інструментом для аналізу зображень. Вони здатні автоматично визначати та екстрагувати важливі характеристики з вхідних зображень за допомогою загорткових та пулінгових шарів. Основна ідея полягає в тому, щоб розпізнати локальні шаблони на зображенні та поступово комбінувати їх для розпізнавання більш складних об'єктів.

Згорткові шари (Convolutional Layers): ці шари використовуються для локального згортання вхідних даних з допомогою фільтрів, що дозволяє визначати важливі характеристики на зображенні, такі як краї та форми.

Пулінгові шари (Pooling Layers): ці шари використовуються для зменшення розмірності отриманих карт характеристик, зберігаючи при цьому важливі особливості. Найпоширенішим методом пулінгу є максимальне пулювання, яке вибирає максимальне значення з кожного регіону зображення.

Повнозв’язані шари (Fully Connected Layers): ці шари використовуються для зведення вихідних карт характеристик до вектору перед подачею на вихідний шар для класифікації або регресії.

Рекурентні нейронні мережі також можуть використовуватися для аналізу зображень, зокрема в області обробки послідовностей зображень або зображень з контекстуальними залежностями. Їхня здатність до врахування контексту та послідовності робить їх ефективними у вирішенні завдань, де

важлива історія або контекст зображення. Ці мережі використовують рекурентні шари (Recurrent Layers). Ці шари дозволяють моделі зберігати та оновлювати внутрішні стани в кожен момент часу на основі вхідних даних та попередніх станів. Це дозволяє моделі враховувати послідовність даних при прийнятті рішень.

Обидва типи архітектури, CNN і RNN, мають свої унікальні переваги та застосування у аналізі зображень. Вони використовуються для різних завдань, від простого класифікації до складних задач, що потребують розуміння контексту та послідовності [1].

1.1.2 Перенавчання нейронних мереж

Перенавчання є серйозною проблемою у навчанні нейронних мереж, яка може виникати, коли модель стає занадто специфічною для даних тренувального набору і втрачає здатність узагальнювати свої знання на нові дані.

Одна з основних причин перенавчання – недостатня кількість даних для тренування. Якщо навчальний набір даних маленький або не репрезентативний для всього простору можливих вхідних даних, модель може навчитися неправильним або неповним уявленням про дані. Це може призвести до того, що модель буде робити невірні прогнози на нових даних, які відрізняються від тих, які вона бачила під час тренування.

Надмірна складність моделі є іншою причиною перенавчання. Якщо модель має занадто багато параметрів або шарів, вона може стати занадто гнучкою та здатною до адаптації до випадкових шумів або нерелевантних шаблонів у даних. Це може призвести до того, що модель просто «запам’ятовує» тренувальний набір даних, замість того, щоб вивчати загальні закономірності, які можуть застосовуватися до нових даних.

Додатковим фактором може бути дисбаланс у класах даних. Якщо певні класи представлені дуже мало, модель може схильна до надмірної уваги до більш представлених класів, що може призвести до недооцінки менш представлених класів.

Перенавчання може суттєво погіршити точність моделі на нових даних. Це означає, що модель може давати дуже високу точність на даних для тренування, але показувати погану здатність до узагальнення на нових, реальних даних. Це може призвести до неправильних прогнозів або класифікацій у реальних ситуаціях, де важлива точність [2].

Існує кілька методів для запобігання перенавчанню нейронних мереж. Один з них – регуляризація. Цей метод включає в себе додавання штрафу до функції втрат для великих значень параметрів моделі. Це допомагає уникнути перенавчання шляхом зменшення різких коливань у вагах моделі.

Також використовуються методи зменшення розмірності даних, такі як метод головних компонентів (PCA) або метод випадкового відбору (Dropout). Ці методи допомагають зменшити складність моделі та підвищити її узагальнюючу здатність.

Крім того, використання валідаційної вибірки для налаштування гіперпараметрів моделі є ефективним методом контролю перенавчання. Деякі досвідчені методики також використовують менш складні моделі з меншою кількістю параметрів, що дозволяє їм краще узагальнювати на нові дані та уникати перенавчання.

Остаточно, комбінація цих методів може допомогти забезпечити надійну та ефективну нейронну мережу, яка добре працює на нових, реальних даних.

1.2 Функціонування шляхів передачі зображення від зорового нерву

Зоровий шлях та візуальна система людини складаються з складної мережі анатомічних структур та шляхів, які забезпечують сприйняття та передачу зорової інформації від очей до мозку. Цей процес включає в себе кілька ключових етапів, від фізіології сітківки до обробки зорових стимулів у візуальних центрах кори головного мозку.

Зоровий шлях починається з фоторецепторів на сітківці, які перетворюють світлові сигнали на електричні сигнали. Фоторецептори розділяються на центральну ділянку сітківки, відому як макула, та периферійні області. Активність фоторецепторів передається гангліозним клітинам сітківки, які утворюють зоровий нерв (рис. 1.1).

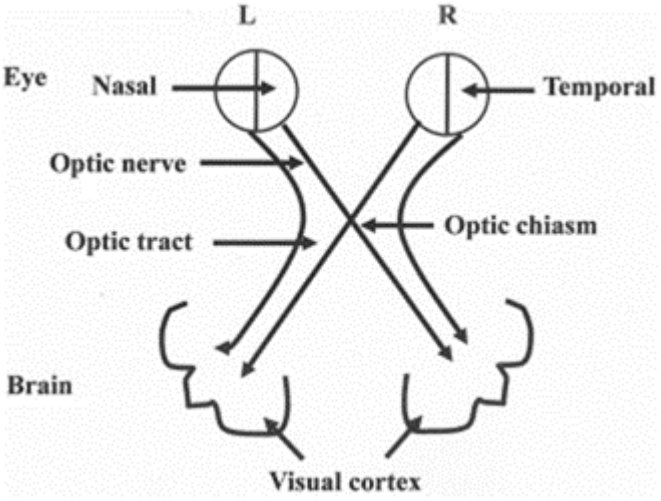

Рисунок 1.1 – Зоровий шлях

Зоровий нерв складається з нервових волокон, які збираються разом на диску зорового нерву перед виходом з ока через орбітальні кістки. Нервові волокна з різних ділянок сітківки стають більш організованими, коли вони проходять по зоровому нерву. Зорові нерви кожного ока зустрічаються в зоровому перехресті, де нервові волокна перетинаються, щоб утворити зоровий тракт, який направляє сигнали до мозку через ядро (LGN), що розташоване в середньому мозку. В LGN відбувається певна обробка

електричних сигналів перед тим, як вони передаються до зорової кори в задній частині потиличної долі головного мозку [3].

У мозку зорова інформація обробляється в різних областях візуальної кори. Ці області відповідають за аналіз та інтерпретацію зорових стимулів, формуючи наше зорове сприйняття навколишнього світу. Поля зору визначаються на основі проекції сітківки на кору головного мозку і відображають мапу зорового простору.

Візуальна система і зоровий шлях є важливими компонентами сприйняття світу навколо нас. Розуміння їх функцій та структур допомагає в дослідженні сприйняття та обробки зорової інформації у мозку людини.

1.2.1 Порушення та захворювання зорового шляху

Порушення та захворювання зорового шляху можуть виникати з різних причин і впливати на різні частини візуальної системи, включаючи сітківку, зоровий нерв, таламус та візуальні кортекси мозку. Надаймо огляд деяких з найбільш поширених порушень та захворювань.

Глаукома: це захворювання, яке зазвичай пов'язане з підвищеним тиском всередині очей. Воно може призвести до ураження зорового нерву та втрати поля зору. Глаукома може бути вродженою або розвиватися протягом життя людини і часто не супроводжується симптомами на ранніх стадіях, тому діагноз часто ставиться пізно.

Діабетична ретинопатія: це ускладнення діабету, яке виникає через ураження судин сітківки. Пошкодження судин може призвести до кровотеч, відкладення жирових відкладень та навіть відшаровування сітківки. У зрозумілій формі це може призвести до важких втрат зору.

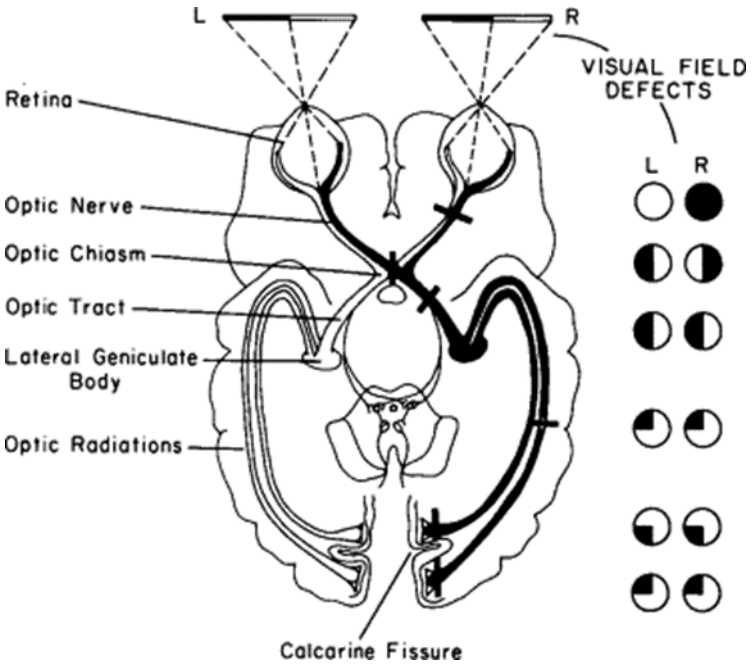

Рисунок 1.2 – Зорові шляхи з ділянками ураження та пов’язаними з цим дефектами поля зору

Макулярна дегенерація: це захворювання, яке впливає на макулу, центральну ділянку сітківки, відповідальну за деталізоване бачення. Існують два типи макулярної дегенерації: волога та суха. Волога макулярна дегенерація може призвести до швидкої втрати зору через кровотечі та відшаровування сітківки.

Ретиніт: це загальна назва для різних запальних захворювань сітківки. Ретиніт може бути спричинений інфекціями, автоімунними захворюваннями або іншими причинами. Вона може призвести до втрати зору, особливо якщо вона впливає на центральну ділянку сітківки.

Оптична невропатія: це ураження зорового нерву, яке може бути спричинене травмою, вірусними інфекціями, захворюваннями або іншими факторами. Оптична невропатія може призвести до втрати зору або навіть сліпоти, особливо якщо вона не лікується належним чином.

Діагноз та лікування цих захворювань вимагають спеціалізованої медичної допомоги. Лікарі можуть використовувати різноманітні методи, включаючи офтальмоскопію, ангіографію, оптичну кохлеарну томографію та інші обстеження для діагностики порушень зорового шляху. Лікування може

включати медикаментозну терапію, хірургічні втручання, фізіотерапію або інші методи, залежно від природи та важкості захворювання.

Дефекти зору, що виникають внаслідок переривання шляху від сітківки до кори, описуються в термінах поля зору, а не сітківки.

Щоб краще зрозуміти, як ураження вздовж зорового тракту впливає на поля зору, слід враховувати маршрут сигналу (рис. 1.2). Волокна з скроневої половини сітківки рухаються в тому ж напрямку, тоді як волокна з носової половини перетинаються в зоровому перехресті.

Таким чином, скроневі волокна сітківки відповідають за носову половину поля зору, а носові волокна – за скроневу половину поля зору. Наприклад, якщо ураження перериває як скроневий, так і носовий шлях волокон лівої півкулі головного мозку, виникає дефект поля зору, який називається правобічною гомонімною геміанопсією. Це означає, що не видно все поле правого зору, тобто втрачається периферійний зір праворуч.

Це пояснюється тим, що скроневі волокна правого ока бачать носову половину правого поля зору, а носові волокна лівого ока бачать скроневу половину правого поля зору. Інші типи дефіциту поля зору відображаються у відповідних місцях ураження в зоровому шляху [4].

1.3 Застосування нейронних мереж у захворюваннях зорового шляху

Застосування нейронних мереж у аналізі зображень зорового шляху відіграє важливу роль у виявленні патологій, діагностиці та розробці нових методів лікування. Ця сфера досліджень стверджується численними науковими роботами, які демонструють потужний потенціал нейронних мереж у різних аспектах аналізу зорового шляху.

Так існує низка досліджень з використанням ШІ для скринінгу діабетичної ретинопатії (ДР) [5] з метою підвищення ефективності виявлення та направлення пацієнтів на лікування. Так, наприклад, у однієї з робіт [6]

оцінено ефективність платформи штучного інтелекту EyeArt для скринінгу діабетичної ретинопатії (ДР) на основі аналізу зображень зорового дна. Виявлено, що штучний інтелект мав загальну узгодженість у 79% з медичним працівником на місці та ретинологом. Для діагностики ДР, точність системи штучного інтелекту становила 81%, а для визначення необхідності направлення на лікування – 83%. Точність системи у виявленні ДР складала 74% для чутливості та 87% для специфічності. Отже, робота демонструє високу ефективність та потенціал штучного інтелекту у скринінгу та діагностиці ДР, хоча залишаються певні обмеження у використанні в масовому скринінгу. Інша робота [7] спрямована на оцінку точності та надійності алгоритму штучного інтелекту (AI) DAIRET® для автоматизованої діагностики діабетичної ретинопатії (ДР). Дослідження використовувало зображення сітківки пацієнтів, які були класифіковані як офтальмологом, порівняно з класифікацією алгоритму DAIRET®. Результати показали, що DAIRET® мав високу чутливість у виявленні важливих форм ДР і достатню специфічність, щоб служити альтернативою ручній класифікації.

У той же час існує дослідження [8], яке демонструє, що алгоритми ШІ вже застосовуються та отримали регуляторне схвалення для автоматизованого виявлення ДР, демонструючи клінічно прийнятну діагностичну ефективність. На прикладі ДР були розглянуті фактори, які потрібно враховувати при впровадженні алгоритмів ШІ на практиці. Серед них – реальна оцінка безпеки, ефективності та справедливості (врахування упередження); вплив на результати пацієнтів; етичні, організаційні та регуляторні аспекти. Подібна стаття [9] також визнає можливість впровадження систем глибокого навчання у рутинний скринінг діабетичної ретинопатії, що може сприяти зменшенню глобального навантаження непопереджуваною сліпотою. Однак зазначає – оцінка таких інструментів в клінічних перспективних дослідженнях залишається обмеженою.

Розглядаються також й сучасні методи телеофтальмології для скринінгу ДР [10], що включають застосування фундусних камер, ультра широкої поля і навіть смартфонів. Використання алгоритмів ШІ сприяє автоматизації процесу оцінки зображень сітківки та робить можливим проведення скринінгу навіть у домашніх умовах.

Окрім цього нейронні мережі знаходять застосування для діагностування інших захворювань зорового шляху. Так недавно ШІ був широко вивчений для багатьох автоматизованих завдань у сфері макулярних захворювань [11], таких як скринінг захворювань, сегментація макулярних шарів, кількісне визначення біомаркерів, прогнозування витрат на лікування та результатів тощо. Є також успіхи із діагностуванням ретинопатії недоношених (ROP) [12]. Важливим кроком є досягнення у використанні штучного інтелекту для покращення догляду за пацієнтами з дегенерацією сітківки у передчасно народжених дітей [13], незважаючи на перешкоди в медико–правових та етичних питаннях, включаючи регулювання, які треба подолати перед впровадженням у реальному світі.

Узагальнюючи, недавні досягнення в галузі ШІ відкривають захоплюючі можливості для поліпшення медичного обслуговування пацієнтів.

1.4 Постановка задачі

Таким чином розробка системи для забезпечення перенавчання нейронних структур мозку щодо функціонування шляхів передачі зображення від зорового нерву є надзвичайно важливою в контексті розвитку сучасних технологій обробки зображень та медичної діагностики. За останні десятиліття нейронні мережі та інші методи машинного навчання здійснили значний прогрес у сфері аналізу медичних зображень, допомагаючи виявляти хвороби та виконувати різні завдання автоматизації, що раніше вимагали б

великої кількості людських ресурсів та часу. Одним з основним способів для забезпечення та допомоги у питанні перенавчання нейронних структур мозку щодо функціонування шляхів передачі зображення від зорового нерву є використання штучного інтелекту для аналізу та класифікації функціонування шляхів передачі зображення від зорового нерву.

Об’єктом роботи є дослідження функціонування шляхів передачі зображення від зорового нерву

Для досягнення цієї мети необхідно реалізувати наступні завдання:

– зібрати дані зображень, які стосуються функціонування шляхів передачі зображення від зорового нерву, такі як зображення здорових і хворих зорових нервів. Підготувати дані зображення, перетворивши їх у відповідний формат і розділивши на навчальні та тестові набори:

– вибрати відповідні методи такі як нейронні мережі, згорткові нейронні мережі (CNN) або генеративні змагальні мережі (GAN);

– впровадити загорткову нейронну мережу (CNN), щоб краще отримувати характеристики зображень зорового нерву;

– перевірка та оптимізація моделі**:** перевірити розроблені моделі на тестові дані та оптимізувати їх шляхом налаштування гіперпараметрів і методів доповнення даних;

– інтегрувати розроблену систему в прикладне середовище, таке як вебінструмент або мобільний застосунок, для підтримки в клінічній практиці.

2 ТЕОРЕТИЧНІ ТА МАТЕМАТИЧНІ АСПЕКТИ СТВОРЮВАНОЇ СИСТЕМИ

2.1 Огляд згорткових нейронних мереж (CNN)

Згорткова нейронна мережа, також відома як CNN або ConvNet, є потужним інструментом для обробки даних з сітчастою топологією, зокрема зображень. Цей клас нейронних мереж виявився надзвичайно ефективним у розпізнаванні та класифікації об’єктів на зображеннях, а також у вирішенні багатьох інших завдань обробки зображень, таких як виявлення захворювань на медичних зображеннях, визначення рукописного тексту та інше.

Основна ідея згорткових нейронних мереж полягає в тому, щоб вони могли автоматично виявляти важливі ознаки або шаблони в наборі даних. Це досягається за рахунок застосування фільтрів, які скользять по вхідному зображенню, виконуючи операцію згортки. Після цього застосовується функція активації для введення нелінійності. Потім можуть застосовуватися шари підвибірки, щоб зменшити розмір карти ознак і забезпечити інваріантність до малих зміщень вхідних даних [14].

Що стосується архітектури згорткової нейронної мережі, вона зазвичай складається з кількох згорткових шарів, чергових шарів підвибірки (Pooling Layers) та повністю зв’язаних шарів. Така структура дозволяє мережі автоматично вивчати ієрархічні ознаки вхідних даних, починаючи з простих форм та текстур і закінчуючи складними візуальними концепціями (рис. 2.1).

Важливо також зазначити, що згорткові нейронні мережі можуть бути навчені з використанням великої кількості даних, що зазвичай вимагається для досягнення високих результатів у задачах комп’ютерного зору. Однак існують і методи для ефективного навчання CNN з обмеженими обсягами даних, такі як трансферне навчання та збільшення даних.

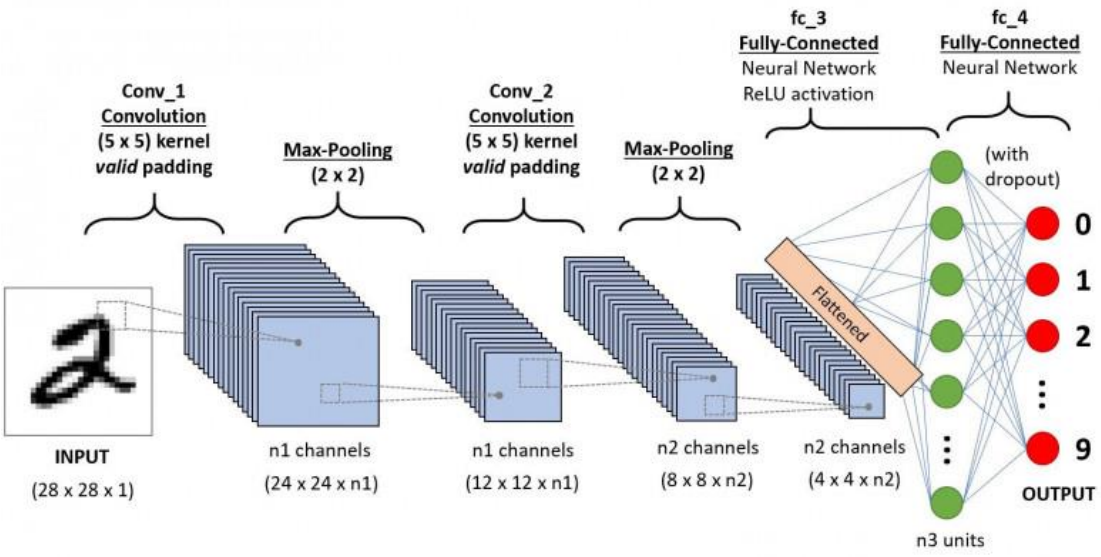

Рисунок – 2.1 Структура CNN

2.1.1 Пулінгові шари

Пулінгові шари є важливою частиною згорткових нейронних мереж (CNN), допомагаючи зменшити розмірність карти ознак, що є результатом згорткових операцій. Основна функція пулінгу – зменшення кількості параметрів моделі та обробка інформації з карти ознак в більш абстрактній формі, зберігаючи при цьому важливі патерни та ознаки.

У згорткових нейронних мережах найчастіше використовуються два типи пулінгу: максимальний пулінг та середній пулінг.

У максимальному пулінгу для кожного фрагмента карти ознак обирається максимальне значення.

Цей підхід допомагає зберігати найважливіші ознаки та положення об'єктів на зображенні, зменшуючи при цьому розмірність карти ознак.

У середньому пулінгу для кожного фрагмента карти ознак обчислюється середнє значення.

Цей підхід менш схильний до перенавчання, оскільки він робить агрегацію інформації на основі середнього значення.

Пулінгові шари допомагають зменшити розмірність карти ознак, зменшуючи кількість параметрів моделі, що робить навчання більш ефективним і запобігає перенавчанню. Вони також допомагають забезпечити інваріантність до зміщень та змін в масштабі вхідних даних, що підвищує стійкість моделі.

У комбінації з згортковими шарами, пулінгові шари утворюють основну архітектуру згорткової частини CNN. Ця комбінація забезпечує автоматичне виявлення та ієрархічну абстракцію важливих ознак на зображеннях, що є ключовим для успішного вирішення багатьох завдань обробки зображень.

2.1.2 Повністю зв’язані шари

Повністю зв’язані шари, також відомі як «повністю зв’язані шари» або «Fully Connected Layers», є одним з основних видів шарів у штучних нейронних мережах. Вони використовуються для створення моделей глибокого навчання в багатьох областях, включаючи комп'ютерне зорове розпізнавання, обробку мови та аналіз даних.

У повністю зв'язаному шарі кожен нейрон пов'язаний з кожним нейроном попереднього та наступного шару. Це означає, що кожен вихідний сигнал з попереднього шару враховується кожним нейроном наступного шару.

Ця архітектура часто застосовується в задачах класифікації, де модель повинна встановити зв'язок між вхідними даними та відповідними категоріями або мітками. Наприклад, у задачі класифікації зображень нейронна мережа може використовувати повністю зв'язані шари для розпізнавання особливостей об'єктів на зображеннях та призначення їм відповідних класів [15].

Кожен нейрон у повністю зв’язаному шарі виконує операції лінійної комбінації вхідних сигналів за допомогою ваг та зсувів, а потім подає результат через нелінійну функцію активації, таку як сигмоїдальна функція або ReLU.

Повністю зв’язані шари в нейронних мережах грають ключову роль у процесі виявлення взаємозв'язків між різними частинами зображення зорового нерву. Ці шари дозволяють мережі аналізувати кожен піксель зображення, ураховуючи його вплив на решту пікселів та загальну структуру зображення.

У сучасних дослідженнях зі штучного інтелекту та медичного зображення повністю зв’язані шари використовуються для виявлення навіть найтонших деталей у зображеннях зорового нерву, що раніше могли бути пропущені. Це допомагає у підвищенні точності та надійності діагностики захворювань та у вчасному виявленні патологічних змін.

2.1.3 Згорткові шари

Згорткові шари є ключовими складовими у згорткових нейронних мережах для обробки зображень, зокрема для аналізу зорового нерву. Ці шари використовують фільтри або ядра для згортання над вхідним зображенням, що дозволяє виділяти різноманітні шаблони та ознаки. Фільтри здійснюють складання невеликих частин зображення, щоб визначити їхню важливість та присутність у різних частинах зображення.

У загорткових шарах кожен фільтр реагує на певні характеристики, такі як краї, текстури або форми. Після проходження кожного фільтру через вхідне зображення, створюється карта ознак, яка показує активацію цих характеристик у різних частинах зображення.

Однією з основних переваг загорткових шарів є їх здатність враховувати просторову інформацію у зображенні. Завдяки цьому здатності

мережа може виявляти патерни та ознаки незалежно від їхнього розташування на зображенні, що робить її більш адаптивною та ефективною у різних ситуаціях.

Крім того, згорткові шари дозволяють зменшити кількість параметрів моделі, що зменшує обчислювальну складність та ризик перенавчання. Це досягається за рахунок використання тих самих фільтрів для всіх частин зображення, що дозволяє спільно використовувати параметри та підвищує роботу мережі.

Також важливою особливістю загорткових шарів є їхній внесок у виявлення ієрархічних ознак у зображеннях. Починаючи з найпростіших функцій, таких як різкість країв, мережа поступово аналізує більш складні характеристики, що дозволяє їй розрізняти складніші об’єкти та структури на зображенні.

2.1.4 Активаційні функції

Функції активації в нейронних мережах відіграють важливу роль у процесі передачі сигналів через штучні нейрони. Вони визначають, як сильно активується нейрон після обчислення вагованих вхідних сигналів. Без функції активації нейрони можуть просто виконувати лінійні операції, що обмежує їхню здатність до моделювання складних взаємозв’язків у даних.

Функції активації мають важливу властивість нелінійності, що дозволяє нейронним мережам виявляти складні нелінійні залежності між вхідними та вихідними даними. Це дозволяє нейронним мережам бути більш гнучкими та адаптивними до різноманітних видів даних [17].

Сигмоїдальна функція приймає будь-яке вхідне значення і стискає його до діапазону між 0 та 1. Вона часто використовується на виході бінарного класифікатора, де 0 відповідає одному класу, а 1 – іншому. Проте, вона може страждати від проблеми зниклого градієнту при глибокому навчанні.

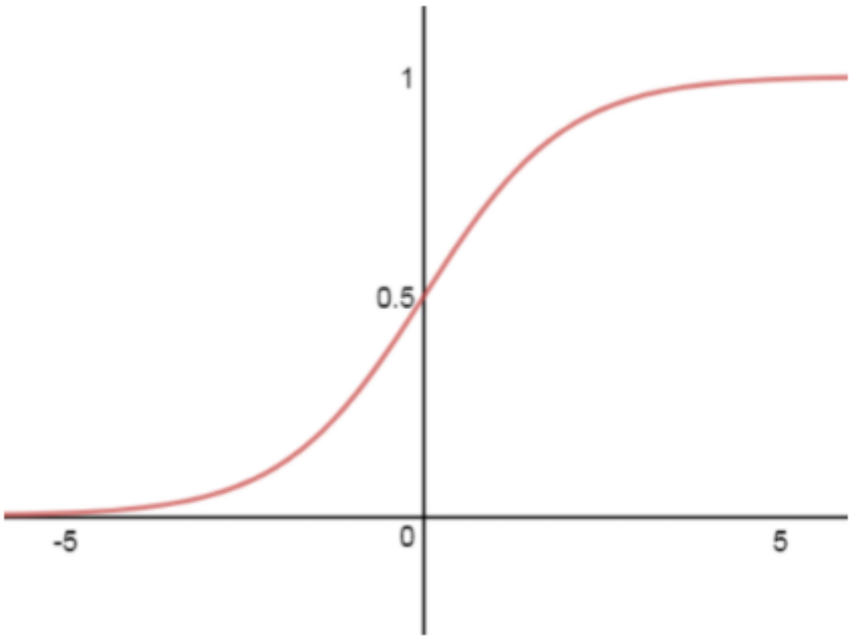

Рисунок 2.2 – Сигмоїдальна функція

Гіперболічна тангенс функція подібна до сигмоїдальної, але стискає вхід в діапазон між –1 та 1. Вона має центрований діапазон, що допомагає уникнути проблеми зниклого градієнту. Також, вона необхідна, коли потрібно вибрати від’ємне значення.

ReLU є однією з найпоширеніших функцій активації. Вона повертає 0 для від'ємних значень та відтворює вхідне значення для додатних значень. Це дозволяє здійснити швидше навчання, але може призвести до «мертвих» нейронів, коли вони завжди видають нульовий вихід.

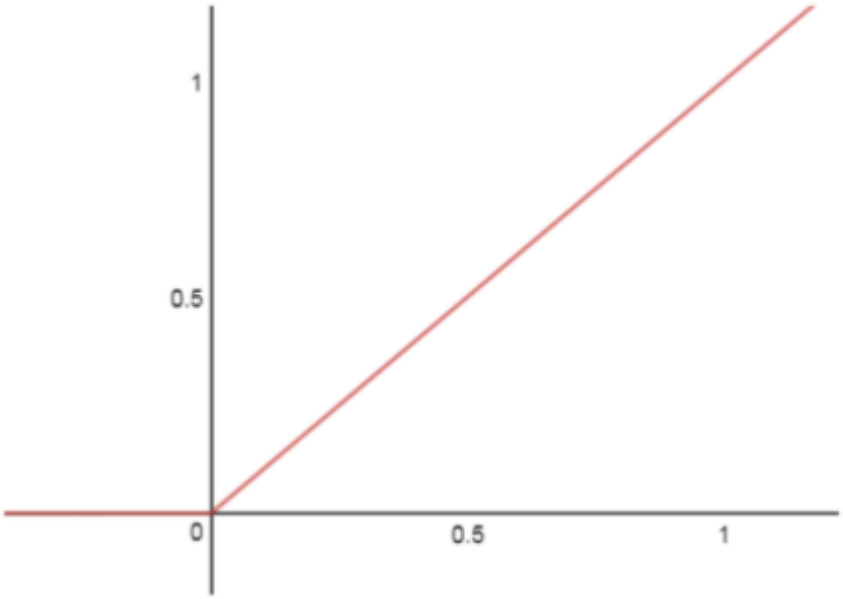

Рисунок 2.3 – Функція ReLU

Leaky ReLU є вдосконаленням ReLU, яке вирішує проблему «мертвих» нейронів. Вона додає невеликий нахил для від’ємних значень, що дозволяє градієнту проникнути і активувати нейрони.

Навіть наявність різних функцій активації відображається на результаті нейронних мереж. Обрання правильної функції активації може вплинути на швидкість навчання, точність прогнозування та стійкість до перенавчання. Тому вибір функції активації є важливим етапом при проєктуванні та навчанні нейронної мережі.

2.1.5 Функції втрат

Функція втрат (Loss function) є ключовою складовою при навчанні нейронних мереж, включаючи згорткові нейронні мережі (CNN). Вона визначає різницю між прогнозованими значеннями моделі та справжніми мітками у навчальному наборі даних. Мета функції втрат – зменшити цю різницю шляхом налаштування параметрів моделі під час навчання.

Функція втрат – це спосіб вимірювати ефективність та точність моделі машинного навчання. У цьому випадку вона діє як керівник для навчання у межах моделі або алгоритму машинного навчання.

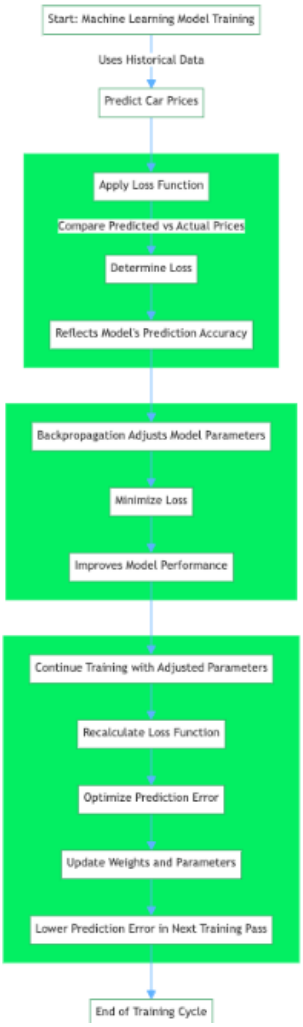

Рисунок 2.4 – Робота функцій втрат

Роль функції втрат надзвичайно важлива у навчанні моделей машинного навчання.

Вимірювання продуктивності: функції втрат пропонують чітку метрику для оцінки продуктивності моделі, кількісно визначаючи різницю між прогнозами та фактичними результатами.

Напрямок для покращення: функції втрат направляють покращення моделі, спрямовуючи алгоритм налаштування параметрів (ваг) ітеративно для зменшення втрат та покращення прогнозів.

Балансування зміщення та варіантності: ефективні функції втрат допомагають збалансувати зміщення моделі (недостатню узагальненість) та варіантність (перенавчання), що є важливим для узагальнення моделі до нових даних.

Вплив на поведінку моделі: деякі функції втрат можуть впливати на поведінку моделі, наприклад, бути більш стійкими до викидів даних або надавати пріоритет певним типам помилок.

Функція втрат, також називається функцією помилок, є критичним компонентом у машинному навчанні, яка кількісно визначає різницю між передбачуваними виходами алгоритму машинного навчання та фактичними цільовими значеннями. Наприклад, у випадку регресійної задачі для передбачення цін на автомобілі на основі історичних даних, функція втрат оцінює прогноз мережі на основі навчального зразка з навчального набору даних. Функція втрат кількісно визначає різницю або помилкову межу цінового прогнозу, яку зробила мережа в порівнянні з фактичною ціною [18].

Результатом є значення втрати, яке відображає точність прогнозів моделі. Під час навчання алгоритм навчання, такий як алгоритм зворотного поширення помилок, використовує градієнт функції втрати відносно параметрів моделі для їх налаштування та мінімізації втрат, ефективно покращуючи продуктивність моделі на наборі даних.

Функція витрат, іноді називається функцією цілі, є середнім значенням функції втрати всього навчального набору, що містить кілька навчальних прикладів. Функція витрат кількісно визначає продуктивність моделі на всьому навчальному наборі даних.

Хоча існують різні типи функцій втрат, вони всі функціонують за принципом кількісної оцінки різниці між передбаченнями моделі та фактичним цільовим значенням у наборі даних. Офіційний термін для цього числового визначення – це помилка передбачення. Алгоритм навчання та механізми в машинній моделі оптимізовані для мінімізації помилки передбачення, отже, це означає, що після обчислення значення функції

втрати, яке визначається помилкою передбачення, алгоритм навчання використовує цю інформацію для проведення оновлень ваг та параметрів, що наступного разу під час навчання призводить до зменшення помилки передбачення.

Функції втрат є ключовою складовою у машинному навчанні, особливо в задачах класифікації, де ми намагаємося визначити, до якого класу або категорії належить кожен зразок даних. Ці функції допомагають моделі оцінювати, наскільки добре вона працює, порівнюючи її прогнози з фактичними значеннями.

Бінарна перехресна ентропія та категоріальна перехресна ентропія використовуються для бінарної та багатокласової класифікації відповідно. Вони оцінюють різницю між розподілом ймовірностей моделі та фактичним розподілом класів.

Логарифмічна втрата (Log Loss) широко використовується у задачах бінарної класифікації та оцінює точність моделі за допомогою порівняння фактичних та прогнозованих значень.

Ці функції втрат допомагають моделям покращувати свої прогнози та адаптуватися до нових даних, що робить їх важливим інструментом для розв'язання різноманітних задач класифікації у машинному навчанні.

Summary				
Problem type	**Last layer output nodes**	**Hidden layer activation**	**Last layer activation**	**Loss function**
Binary classification	1	ReLU (first choice)	Sigmoid	Binary Cross Entropy
				Weighted Cross Entropy
			Tanh	Hinge Loss
Multi-class, single label classification	Number of classes		SoftMax	Categorical Cross Entropy
				Sparse Categorical Cross Entropy
				KullBack Leiber Divergence Loss
Multi-class, multi label classification	Number of classes		Sigmoid (one for each class)	Binary Cross Entropy

Рисунок 2.5 – Робота функцій втрат

2.2 Generative Adversarial Networks (GAN)

Генеративно-змагальні мережі (GANs) викликали значний інтерес у галузі штучного інтелекту не лише через їхні інноваційні методи створення та покращення даних, але й через їхню важливу роль у таких застосуваннях, як створення фотореалістичних зображень, трансформація стилів на зображеннях і генерація реалістичних облич людей. Представлені вперше на конференції NeurIPS у 2014 році Іаном Гудфеллоу та його командою, GAN вирізняються як системи машинного навчання, що можуть відтворювати конкретні розподіли даних.

Генеративно-змагальні мережі (GANs) завдяки своїм здібностям можуть знайти застосування у розробці системи для перенавчання нейронних структур мозку щодо функціонування шляхів передачі зображення від зорового нерву. Це може включати в себе створення синтетичних зображень, які імітують вхідні дані, поданим зоровим нервом, для подальшого використання в процесі навчання нейронних мереж. Ці синтетичні зображення допомагають в розвитку та вдосконаленні нейронних структур, що моделюють обробку візуальної інформації в мозку, дозволяючи вивчати їх функціональні властивості та механізми передачі зображення [19].

2.2.1 Архітектура GAN

Generative Adversarial Networks (GAN) складаються з двох нейронних мереж: генератора та дискримінатора, які одночасно навчаються через змагальне навчання.

Генератор: ця мережа отримує випадковий шум як вхід і генерує дані (наприклад, зображення). Її завдання – створити дані, які якомога більше схожі на реальні дані.

Дискримінатор: ця мережа отримує як вхід справжні дані і дані, створені Генератором, і намагається відрізнити одне від одного. Вона видає ймовірність того, що задані дані є реальними.

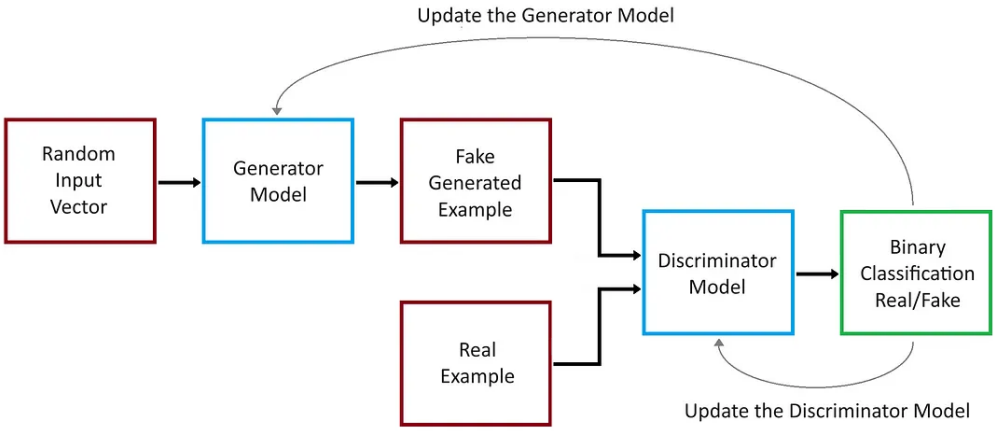

Рисунок 2.6 – Архітектура GAN

Під час навчання Генератор намагається створити дані, які Дискримінатор не може відрізнити від реальних даних, тоді як Дискримінатор намагається стати кращим у відрізненні реальних даних від фальшивих. Обидві мережі змагаються між собою: Генератор має створити переконливі фальшиві дані, а Дискримінатор – відрізнити справжні дані від фальшивих. Цей змагальний процес призводить до того, що Генератор створює все кращі дані з часом.

2.2.2 Навчання моделі GAN

Під час навчання GAN отримує два входи: випадковий шум і дані на вході без міток. Використовуючи ці два входи, вона генерує дані, що нагадують вхідні дані. Оскільки всі дані для GAN не мають міток, GAN є типом ненаглядного навчання.

У GAN є дві нейронні мережі, які змагаються між собою. Мета генератора – обдурити дискримінатор, а мета дискримінатора – правильно ідентифікувати, чи є вхід реальним чи фальшивим [20].

Головна роль генератора – генерувати дані. Спочатку ці дані ймовірно будуть випадковим шумом, оскільки Генератор починає без великої кількості знань про розподіл справжніх даних. З часом, навчаючи GAN, генератор навчається створювати дані, що наближаються до розподілу справжніх даних.

Під час тренування генератор намагається обдурити дискримінатор, продукуючи дані, які дискримінатор не може відрізнити від справжніх даних. генератор оновлює свої ваги на основі відгуку від дискримінатора, вдосконалюючи свою здатність створювати переконливі фальшиві дані.

Генератор видає дані як свій вихід. У випадку GAN, призначеного для генерації зображень, вихід – це зображення. Далі дані передаються дискримінатору для класифікації як справжні або фальшиві.

Дискримінатор відповідає за розрізнення між реальними та штучно створеними даними. Якщо ми говоримо про генеративно–змагальні мережі (GAN) у контексті зображень (що є поширеним застосуванням), дискримінатор намагається відрізнити справжні зображення від фальшивих, що згенеровані генератором.

На вхід дискримінатора надходять зразки даних, які можуть бути як реальними, так і згенерованими.

На виході дискримінатор видає скалярне значення від 0 до 1, яке представляє ймовірність того, що вхідний зразок є реальним. Значення, близьке до 1, вказує на те, що зразок ймовірно є реальним, тоді як значення, близьке до 0, свідчить про ймовірність того, що зразок штучний.

Архітектура дискримінатора часто відображає традиційні згорткові нейронні мережі (CNN), але з деякими коригуваннями. Зазвичай вона складається з наступних компонент:

– згорткові шари: ці шари є ключовими у обробці зображень і допомагають витягти ознаки з вхідних зображень. Кількість згорткових шарів може варіюватися в залежності від складності даних;

– нормалізація пакетів: іноді використовується між шарами для стабілізації навчання, нормалізуючи вхід до шару;

– функції активації: часто використовується протікання ReLU (Leaky ReLU) для функцій активації в дискримінаторі. Це дозволяє зберігати градієнт під час навчання;

– Шари пулінгу: деякі архітектури використовують шари пулінгу (наприклад, максимальний пулінг) для поступового зменшення просторових розмірів вхідних даних;

– Повністю зв’язані шари: в кінці мережі використовуються повністю зв'язані шари для обробки ознак, витягнутих згортковими шарами, що завершується вихідним шаром.

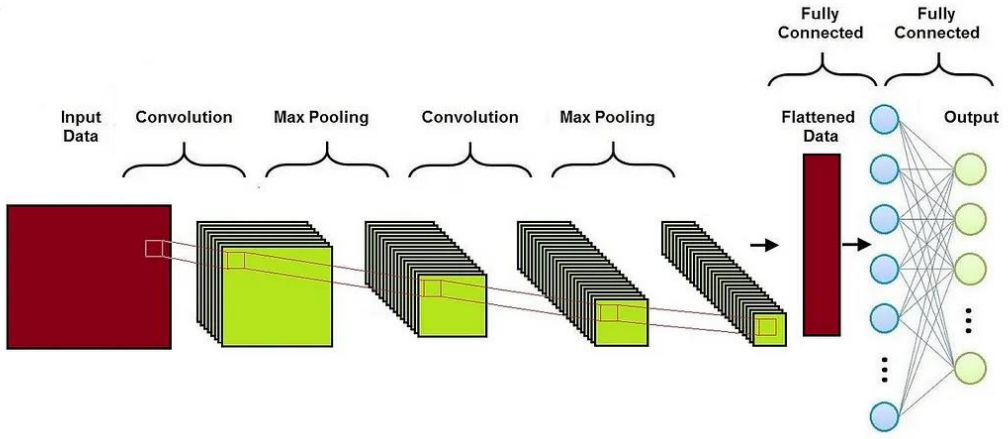

Рисунок 2.7 – Архітектура дискримінатора

Хоча фактична архітектура може суттєво відрізнятися залежно від конкретної проблеми та набору даних.

Основна різниця між класичною архітектурою CNN і архітектурою дискримінатора GAN полягає в декількох особливостях:

– функція втрат: у CNN використовується багато різних функцій втрат, тоді як дискримінатор GAN завжди використовує функцію втрати бінарного перехресного ентропії, оскільки він розрізняє між двома класами (реальними та фальшивими)

– зворотний зв’язок: у CNN немає зворотного зв'язку з іншою мережею під час тренування, і вона навчається ізольовано за допомогою маркованого набору даних, у Дискримінатора GAN відбувається взаємодія з Генератором. Дискримінатор надає зворотний зв'язок Генератору, дозволяючи йому покращувати свої можливості генерації зображень;

– функція активації виходу: залежно від завдання, CNN можуть використовувати або не використовувати активацію softmax, тоді як Дискримінатор GAN зазвичай використовує функцію активації сигмоїду на вихідному шарі для надання ймовірності того, що зображення є реальним чи фальшивим;

– глибина та складність: дискримінатор GAN часто є простішим і менш глибоким, ніж традиційні CNN, але, звичайно, складність залежить від конкретної архітектури GAN та набору даних, який використовується.

Під час навчання дискримінатор оновлює свої ваги таким чином:

– він отримує справжні дані і навчається виходити на значення, близькі до 1;

– він отримує фальшиві дані (згенеровані генератором) і навчається виходити на значення, близькі до 0;

– якщо дискримінатор стає занадто сильним занадто швидко, він може завжди давати дуже впевнені результати (близько до 0 або 1) для будь-якого вводу, що ускладнює навчання генератора.

З іншого боку, якщо дискримінатор занадто слабкий, генератор може не отримувати значущого зворотного зв’язку для покращення. Баланс між генератором і дискримінатором під час навчання є важливим для успіху GAN.

У випадку розробки моєї системи, коли обсяг даних стосовно рідких або складних захворювань зорового шляху обмежений, GAN може бути використаний для генерації нових зображень, які відповідають певним характеристикам захворювань, що дозволяє розширити обсяг доступних даних для навчання класифікатора та покращити репрезентування даних.

2.3 Підготовка даних

Підготовка даних у машинному навчанні – це процес очищення, обробки та перетворення сирої інформації з метою забезпечення точних прогнозів за допомогою алгоритмів машинного навчання.

Підготовка даних має велике значення для успішного розвитку моделей машинного навчання, оскільки вона допомагає уникнути недоліків та покращити якість прогнозів. Завдяки підготовці даних, моделі можуть бути більш точними та ефективними.

Основні етапи підготовки даних включають розуміння проблеми, збір даних, профілювання та дослідження даних, очищення та валідацію даних, форматування даних, покращення якості даних, інженерію ознак та вибір, а також розділення даних на навчальний та тестовий набори.

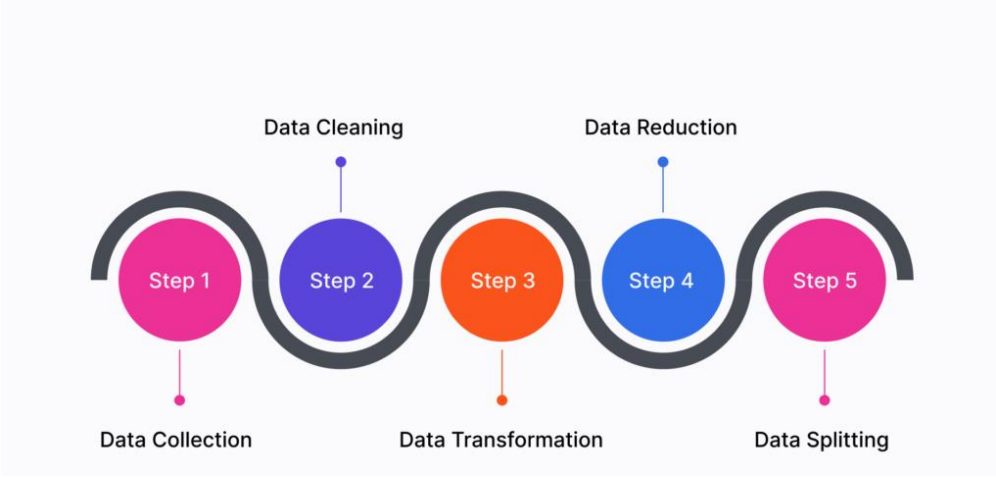

Рисунок 2.8 – Кроки підготовки даних

Крок 1. Збір даних.

Перший крок у підготовці даних для машинного навчання – це збір даних, які знадобляться для моделі.

Важливо переконатися, що зібрані дані мають відношення до проблеми. Нерелевантні або низькоякісні дані можуть призвести до поганої продуктивності моделі, тому будьте вибірковими та спрямованими в своїх зусиллях щодо збору даних.

Крок 2. Очищення даних.

Треба виявити та обробити будь-які відсутні значення в даних, використовуючи методи заповнення, видалення або імпутації.

Виявити та видалити дублікати, щоб уникнути перекосів у результатах аналізу. Провести аналіз аномалій та виявити незвичайні значення, які можуть впливати на точність моделі. У деяких випадках, особливо коли відсутні дані можуть призвести до викривлення, краще буде видалити ці рядки.

Викиди – це дані, які значно відрізняються від решти даних. Ці викиди можуть спотворити аналіз та призвести до неправильних висновків.

Один з способів виявлення викидів – це нормалізація збіжності за оцінкою z. Нормалізація збіжності за оцінкою z – це статистичний метод, який розраховує, на скільки стандартних відхилень дана точка даних відхиляється від середнього значення набору даних. Простими словами, вона допомагає зрозуміти, наскільки «аномальним» є певний пункт даних порівняно з середнім. Значення z-оцінки вище 3 або нижче –3 зазвичай вказує на викид.

Після виявлення викидів за допомогою нормалізації збіжності за оцінкою z є кілька варіантів дій. Можна видалити їх, щоб вони не спотворювали модель, або обмежити їх до певного значення, щоб зменшити їх вплив.

Неузгодженості в даних можуть викликати непорозуміння аналізу та призвести до неправильної інформації.

Для виправлення цього можна використовувати правила, специфічні для домену, які стандартизують найменування або метрики для виправлення цих неузгодженостей.

Також тут можуть бути корисними техніки перевірки даних, які дозволяють виявляти неузгодженості або аномалії та вносити їхні корективи, щоб вони не впливали на аналіз.

Витративши час на вирішення цих неузгодженостей, покращується якість поточного аналізу та створюються передумови для більш точних і проникливих аналізів у майбутньому.

Крок 3. Трансформація даних.

Трансформація ваших даних є ключовим кроком, оскільки те як будуть підготовані дані, безпосередньо впливає на те, наскільки добре модель може вивчити їх.

Трансформація даних – це процес перетворення очищеної інформації в формат, придатний для алгоритмів машинного навчання. Це часто включає масштабування ознак та кодування, серед інших технік.

Крок 4. Зменшення даних.

Спрощення даних допомагає моделі машинного навчання легше виявляти патерни, що пропонує швидку і точну маркетингову інформацію для своєчасних рішень.

Зменшення даних – це процес спрощення вашої інформації без втрати її сутності. Техніки зменшення даних можуть зробити набори даних більш керованими та прискорити алгоритми машинного навчання, не втрачаючи продуктивності моделі.

Одним з поширених методів є зменшення розмірності, яке зменшує кількість цільових змінних у повному наборі даних, зберігаючи його ключові характеристики.

Крок 5. Розбиття даних.

Останнім кроком у підготовці даних для машинного навчання є розбиття їх на різні набори: навчальний, валідаційний та тестовий.

Правильне розбиття ваших даних забезпечує те, що модель машинного навчання може добре узагальнюватися до нових даних.

Зазвичай використовується практика використання відношення 70-30 або 80-20 для навчального та тестового наборів. Навчальний набір використовується для навчання моделі, а тестовий – для її оцінки. Деякі також використовують валідаційний набір. Валідаційний набір – це окремий набір даних, який не використовується для навчання моделі, але служить для оцінки її ефективності після навчання. Використовується для уникнення перенавчання та оцінки готовності моделі до реального використання.

Важливо забезпечити, щоб кожен набір відображав загальні дані.

3 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Обґрунтування вибору середовища програмної реалізації

Одним з критично важливих етапів під час розробки застосунку є вибір технологій, які будуть використані у роботі. Після ретельного вивчення сучасних технологій та мов програмування прийнято рішення про використання TensorFlow, Keras, OpenCV та Flask. Використано середовища розробки PyCharm для написання коду та експериментів з моделями машинного навчання.

TensorFlow – це потужна бібліотека для машинного навчання, розроблена компанією Google. Вона забезпечує високу продуктивність та гнучкість при розробці і навчанні моделей нейронних мереж. TensorFlow підтримує як навчання моделей на CPU, так і на GPU, що дозволяє значно пришвидшити процес обробки великих обсягів даних. Використання TensorFlow було обґрунтоване потребою в потужному інструменті для навчання моделей глибокого навчання, які використовуються для класифікації зображень зорового шляху.

Keras – це високорівнева API, яка працює поверх TensorFlow, що значно спрощує процес створення та тренування моделей нейронних мереж. Keras пропонує зручний та інтуїтивно зрозумілий інтерфейс для швидкого прототипування моделей, забезпечуючи при цьому гнучкість та можливості для налаштування. Використання Keras дозволило скоротити час розробки та зосередитися на експериментах з архітектурою моделей для досягнення найкращих результатів.

OpenCV – бібліотека комп’ютерного зору з відкритим вихідним кодом, що забезпечує широкий набір інструментів для обробки зображень. Вона використовується для попередньої обробки зображень, таких як масштабування, обрізка та нормалізація, що є важливим етапом перед передачею зображень на аналіз моделям нейронних мереж. Завдяки своїй

універсальності та великій спільноті розробників, OpenCV стала незамінним інструментом для роботи з зображеннями.

Flask – мікрофреймворк для створення вебзастосунків на Python. Він забезпечує просту інтеграцію моделей машинного навчання з веб–інтерфейсом, що дозволяє користувачам взаємодіяти з системою через зручний веб–інтерфейс. Flask підтримує модульну структуру, що дозволяє розширювати функціонал застосунку та легко інтегрувати нові компоненти. Використання Flask було обрано через його легкість у використанні та можливість швидкої реалізації RESTful API для інтеграції моделей з вебзастосунком.

Для розробки та налагодження коду було використано PyCharm, потужне середовище розробки на Python. PyCharm надає широкий набір інструментів для розробки, включаючи інтегровані термінали, підтримку системи контролю версій та відладчики, що полегшило процес розробки та забезпечило безперешкодну інтеграцію та оптимізацію робочих процесів.

Вибір цих інструментів та технологій обумовлений їх потужністю, гнучкістю та зручністю використання. Вони забезпечили швидкий та ефективний процес розробки системи для аналізу зображень зорового шляху, дозволяючи досягти високих результатів у класифікації захворювань зорового шляху та інтеграції моделей з вебзастосунком для зручного використання кінцевими користувачами.

3.2 Створення датасету

Однією з ключових задач в рамках розробки системи для аналізу проблем та функціонування зорового шляху людини є вибір відповідного датасету. Враховуючи специфіку проєкту, було прийнято рішення використовувати зображення зорового дна (fundus image). Це рішення обґрунтоване рядом важливих причин.

По-перше, зображення зорового дна забезпечують детальну візуалізацію внутрішньої частини ока, включаючи сітківку, зоровий нерв та інші важливі структури. Фундусні зображення можуть показати ознаки пошкодження зорового нерву, такі як зміни у формі та кольорі диска зорового нерву. Такі зображення є ключовими для діагностики широкого спектра офтальмологічних захворювань, включаючи діабетичну ретинопатію, глаукому, катаракту та інші патології, які можуть впливати на зоровий шлях.

По-друге, фундусні зображення дозволяють виявити ранні ознаки захворювань, що можуть не проявлятися симптоматично на початкових етапах. Це особливо важливо для профілактики та своєчасного лікування офтальмологічних проблем, що безпосередньо впливають на якість життя пацієнтів.

Хоча існують інші технології для оцінки зорового нерву та нервового волокна, Фотографія очного дна була основним методом документування зорового нерву до появи комп’ютерних методів візуалізації.

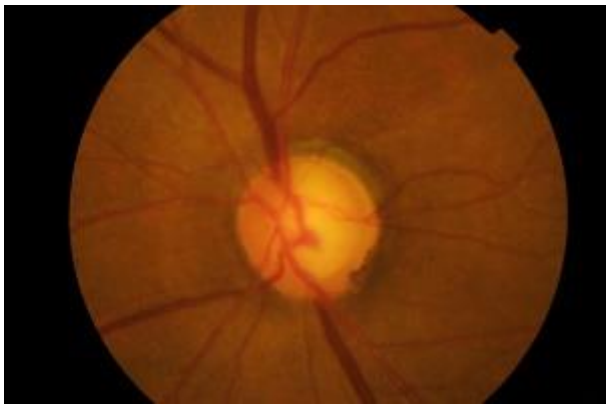

Рисунок 3.1 – Фотографія лівого диску зорового нерву

Таким чином, вибір зображень зорового дна для датасету дозволяє реалізувати ефективний інструмент для діагностики та моніторингу стану зорового шляху. Використання таких зображень надає можливість створити модель, здатну аналізувати та класифікувати різні патологічні стани, забезпечуючи високий рівень точності та надійності.

3.2.1 Збір даних

У процесі створення системи для аналізу проблем та функціонування зорового шляху людини, критично важливим етапом стало зібрання та підготовка високоякісного датасету. Для цього було використано різні джерела, що спеціалізуються на зображеннях зорового дна.

IDRiD (Indian Diabetic Retinopathy Image Dataset) [41]: IDRiD є одним з найбільш відомих та широко використовуваних датасетів для дослідження діабетичної ретинопатії. Він включає високоякісні зображення сітківки з різними стадіями діабетичної ретинопатії, а також нормальні зображення.

Ocular Recognition Databases: цей ресурс містить різноманітні зображення зорового дна для дослідження та розвитку технологій розпізнавання зображень. Він охоплює широкий спектр патологій, включаючи катаракту та глаукому.

HRF (High–Resolution Fundus) [43]: HRF є високоякісним датасетом з зображеннями зорового дна, призначеним для дослідження кровоносних судин сітківки. Він включає зображення як нормальних, так і патологічних станів.

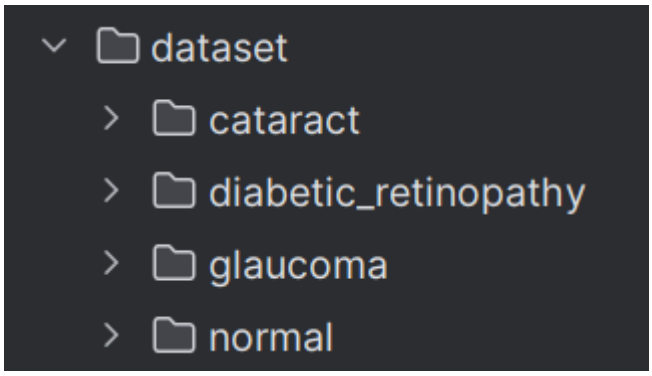

Рисунок 3.2 – Датасет зображення зорового дна

Зібрані дані з різних джерел були об’єднані в єдиний, збалансований датасет, який містить зображення нормального зорового дна та різних патологічних станів. Кінцевий датасет включає приблизно 4000 зображень,

рівномірно розподілених між чотирма класами: Нормальні зображення (Normal), Діабетична ретинопатія (Diabetic Retinopathy), Катаракта (Cataract) та Глаукома(Glaucoma) (рис. 3.2).

Рисунок 3.2 – Приклад зображення з датасету

3.2.2 Попередня обробка датасету

Для забезпечення високої якості та надійності моделей глибокого навчання, було необхідно ретельно підготувати датасет. Цей процес включав декілька ключових етапів: збирання даних, їх фільтрація, анотація, попередня обробка та аугментація.

Для зручності обробки та підготовки зображень цього у файлі ProcessData.py було розроблено клас EyeDiseaseDataset. Цей клас відповідає за збирання шляхів до зображень, створення анотацій, розділення датасету на тренувальні, валідаційні та тестові вибірки, а також збереження цих вибірок у відповідних директоріях.

Лістинг 3.1 Реалізація класу EyeDiseaseDataset:

```
class EyeDiseaseDataset:
    def __init__(self, dataDir):
        self.data_dir = dataDir

    def dataPaths(self):
        filepaths = []
        labels = []
        folds = os.listdir(self.data_dir)
        for fold in folds:
            foldPath = os.path.join(self.data_dir, fold)
            filelist = os.listdir(foldPath)
            for file in filelist:
                fpath = os.path.join(foldPath, file)
                filepaths.append(fpath)
                labels.append(fold)
        return filepaths, labels

    def dataFrame(self, files, labels):
        Fseries = pd.Series(files, name='filepaths')
        Lseries = pd.Series(labels, name='labels')
        return pd.concat([Fseries, Lseries], axis=1)

    def split_(self):
        files, labels = self.dataPaths()
        df = self.dataFrame(files, labels)
        strat = df['labels']
        trainData, dummyData = train_test_split(df, train_size=0.8,
shuffle=True, random_state=42, stratify=strat)
```

```
        strat = dummyData['labels']
        validData, testData = train_test_split(dummyData, train_size=0.5, shuffle=True, random_state=42, stratify=strat)
        return trainData, validData, testData

    def save_split_data(self, trainData, validData, testData, save_dir):
        os.makedirs(save_dir, exist_ok=True)
        train_dir = os.path.join(save_dir, 'train')
        valid_dir = os.path.join(save_dir, 'valid')
        test_dir = os.path.join(save_dir, 'test')

        for folder in [train_dir, valid_dir, test_dir]:
            os.makedirs(folder, exist_ok=True)

        for index, row in trainData.iterrows():
            shutil.copy(row['filepaths'], os.path.join(train_dir, row['labels']))

        for index, row in validData.iterrows():
            shutil.copy(row['filepaths'], os.path.join(valid_dir, row['labels']))

        for index, row in testData.iterrows():
            shutil.copy(row['filepaths'], os.path.join(test_dir, row['labels']))
```

Датасет було розділено на тренувальну, валідаційну та тестову вибірки у співвідношенні 80:10:10, що дозволило забезпечити адекватну кількість даних для кожної стадії навчання та оцінювання моделей.

3.2.3 Аугментація даних

Аугментація даних є важливим етапом, що дозволяє збільшити різноманітність тренувального датасету шляхом застосування різних трансформацій до зображень, таких як обертання, горизонтальне та вертикальне відображення, зміна яскравості тощо. Це допомагає моделям стати більш стійкими до різноманітних змін у вхідних даних та покращує їх загальну продуктивність.

Функція augment_data відповідає за створення генераторів зображень з аугментацією для тренувальної, валідаційної та тестової вибірок. Вона приймає такі параметри, як тренувальні, валідаційні та тестові дані, розмір зображень та розмір батча.

Аугментація тренувальних даних:

– обертання: випадкове обертання зображень на кут до 30 градусів;

– горизонтальне та вертикальне відображення: випадкове відображення зображень по горизонталі та вертикалі;

– зміна яскравості: зміна яскравості зображень у діапазоні від 0.5 до 1.5;

– нормалізація: масштабування значень пікселів до діапазону [0, 1] шляхом ділення на 255.

Реалізацію функції augment_data наведено у лістингу 3.2.

Лістинг 3.2 Функція augment_data

```
def augment_data(train_df, valid_df, test_df, batch_size=16, img_size=(256, 256)):
    train_datagen = tf.keras.preprocessing.image.ImageDataGenerator(
        rotation_range=30,
        horizontal_flip=True,
        vertical_flip=True,
```

```
    brightness_range=[0.5, 1.5],
    rescale=1./255
)

valid_test_datagen = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_dataframe(
    train_df,
    x_col='filepaths',
    y_col='labels',
    target_size=img_size,
    color_mode='rgb',
    batch_size=batch_size,
    shuffle=True,
    class_mode='categorical'
)

valid_generator = valid_test_datagen.flow_from_dataframe(
    valid_df,
    x_col='filepaths',
    y_col='labels',
    target_size=img_size,
    color_mode='rgb',
    batch_size=batch_size,
    shuffle=True,
    class_mode='categorical'
)

test_generator = valid_test_datagen.flow_from_dataframe(
```

```
    test_df,
    x_col='filepaths',
    y_col='labels',
    target_size=img_size,
    color_mode='rgb',
    batch_size=batch_size,
    shuffle=False,
    class_mode='categorical'
)

return train_generator, valid_generator, test_generator
```

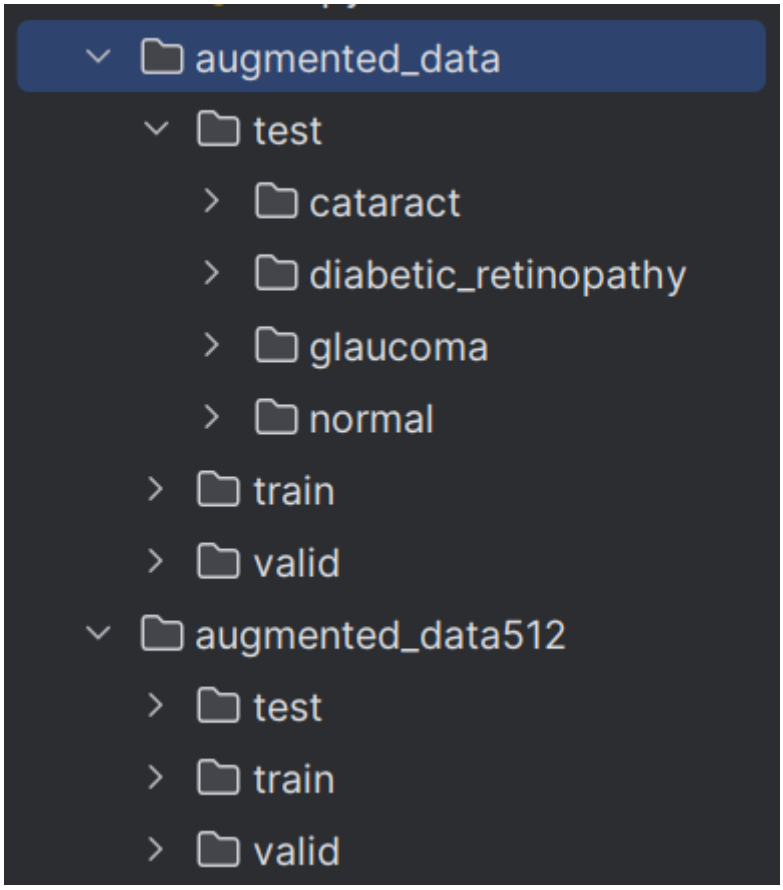

Рисунок 3.3 – Збережені аугментовані дані

Аугментовані дані були збережені у відповідні директорії для забезпечення можливості повторного використання та аналізу. Це також

дозволяє уникнути повторної обробки даних при кожному запуску моделі, що економить час та ресурси.

Для експериментів з навчанням було обрано два різні розміри зображень: 256x256 та 512x512 пікселів. Це дозволило провести порівняльний аналіз ефективності моделей на різних розмірах зображень, враховуючи такі фактори, як швидкість навчання, використання пам’яті та точність класифікації.

Розмір 256x256: менші зображення дозволяють зменшити час навчання моделей та використання пам’яті, що є важливим фактором для швидкого прототипування та експериментів.

Розмір 512x512: більші зображення забезпечують вищу деталізацію, що може покращити точність моделей, особливо для завдань, де важлива висока роздільна здатність зображень.

3.3 Реалізація системи

3.3.1 Вибір архітектури глибоких нейронних мереж

Для розробки системи класифікації захворювань зорового шляху було обрано використання глибоких згорткових нейронних мереж (CNN). Це обумовлено їхньою здатністю ефективно впоратися з обробкою зображень та автоматично виявляти складні залежності в даних.

Було розроблено три різні архітектури CNN з різними конфігураціями шарів згортки, пулінгу, повнозв’язних шарів та функцій активації. Кожна з них мала свої особливості та переваги.

ResNet50 є однією з перших та найуспішніших глибоких згорткових нейронних мереж для обробки зображень. Основна перевага ResNet50 полягає в її глибокій архітектурі, яка дозволяє ефективно вирішувати проблему зникання градієнту під час навчання нейронних мереж. Ця

архітектура використовує блоки зі з'єднанням «skip connections», що дозволяють передавати інформацію безпосередньо з одного шару до іншого, уникнувши втрати важливої інформації. Для задачі класифікації захворювань зорового шляху, ResNet50 відмінно підходить через свою здатність до навчання складних залежностей у даних та здатність до ефективної візуалізації ознак на зображеннях.

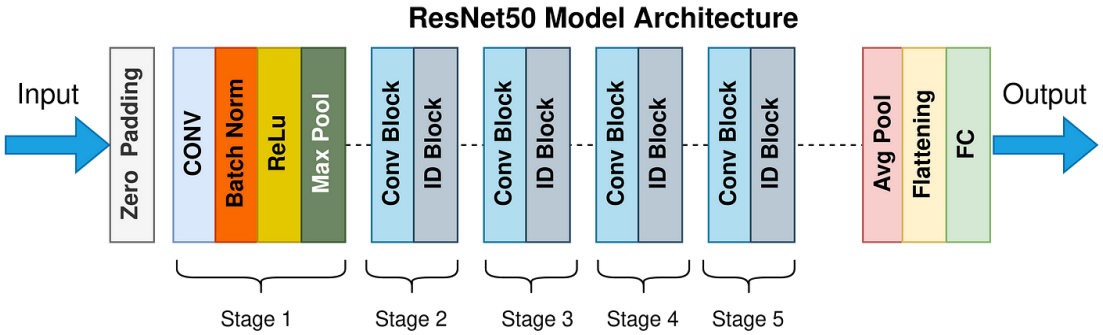

Рисунок 3.4 – Архітектура ResnNet50

EfficientNetB3 є однією з найефективніших архітектур для класифікації зображень, особливо в умовах обмежених обчислювальних ресурсів. Основна перевага EfficientNetB3 полягає в її здатності до автоматичного масштабування глибини, ширини та роздільної здатності мережі. Ця архітектура підтримує оптимальний баланс між кількістю параметрів та швидкодією обчислень, що робить її особливо підходящою для задач класифікації зорового шляху. Крім того, EfficientNetB3 має високу точність класифікації, що дозволяє отримати надійні результати на великих обсягах даних.

Базова модель CNN, побудована на основі класичних архітектур глибоких нейронних мереж, відома своєю простотою та ефективністю. Вона має менше параметрів та ваг, що робить її ефективною для роботи з невеликими обсягами даних, таких як датасети медичних зображень. Основна перевага базової моделі CNN полягає в її здатності до класифікації зорового шляху з високою точністю та швидкодією обчислень. Вона може

бути особливо корисною для використання у випадках, коли обчислювальні ресурси обмежені або коли потрібно швидко вирішити задачу класифікації зорового шляху.

Для їх реалізації та зручного навчання, тестування та застосування було розроблено клас CNNModels. Його реалізація наведена у лістингу 3.3.

Лістинг 3.3 Клас CNNModels

```
class CNNModels:
    def __init__(self, img_size=(256, 256, 3), num_classes=4, learning_rate=0.001):
        self.img_size = img_size
        self.num_classes = num_classes
        self.learning_rate = learning_rate

    def build_resnet50(self):
        base_model = ResNet50(weights='imagenet', include_top=False, input_shape=self.img_size)
        x = base_model.output
        x = GlobalAveragePooling2D()(x)
        x = Dense(512, activation='relu')(x)
        predictions = Dense(4, activation='softmax')(x)
        model = Model(inputs=base_model.input, outputs=predictions)

        for layer in base_model.layers:
            layer.trainable = False

        model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
        return model
```

```
        def build_efficientnetb3(self):
            base_model = EfficientNetB3(weights='imagenet', include_top=False, input_shape=self.img_size)
            x = base_model.output
            x = GlobalAveragePooling2D()(x)
            x = Dense(512, activation='relu')(x)
            predictions = Dense(4, activation='softmax')(x)
            model = Model(inputs=base_model.input, outputs=predictions)

            for layer in base_model.layers:
                layer.trainable = False

            model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
            return model

        def build_cnn_model(self):
            model = Sequential([
                Conv2D(32, (3, 3), activation='relu', input_shape=self.img_size),
                MaxPooling2D((2, 2)),
                Conv2D(64, (3, 3), activation='relu'),
                MaxPooling2D((2, 2)),
                Conv2D(128, (3, 3), activation='relu'),
                MaxPooling2D((2, 2)),
                Conv2D(128, (3, 3), activation='relu'),
                MaxPooling2D((2, 2)),
                Flatten(),
                Dense(512, activation='relu'),
                Dense(self.num_classes, activation='softmax')
            ])
```

```
        model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
        return model
    def train_model(self, model, train_generator, valid_generator, epochs=10, checkpoint_path='best_model.h5'):
        callbacks = [
          ModelCheckpoint(filepath=checkpoint_path, save_best_only=True, monitor='val_loss', mode='min'),
          EarlyStopping(monitor='val_loss', patience=5, mode='min', restore_best_weights=True)
        ]
        history = model.fit(train_generator,
                            epochs=epochs,
                            validation_data=valid_generator,
                            callbacks=callbacks,
                           )
        return history

      def save_model(self, model, model_path='model.h5'):
        model.save(model_path)
```

ResNet50 та EfficientNetB3: в останньому повнозв’язаному шарі використовується функція активації softmax, оскільки вона добре підходить для класифікації задач з багатьма класами.

У прихованих повнозв’язаних шарах використовується функція активації relu, яка допомагає уникнути проблеми з вибухом градієнту та прискорює навчання моделі.

Базова модель CNN: у всіх повнозв’язаних шарах використовується також функція активації relu для полегшення навчання та покращення нелінійності моделі. У останньому повнозв’язаному шарі також

використовується функція активації softmax для отримання ймовірностей класів.

Використання функції активації relu дозволяє моделі швидше та ефективніше навчатися за рахунок видалення негативних значень та уникнення затухання градієнту. Функція активації softmax в останньому шарі допомагає моделі вибрати найбільш ймовірний клас для кожного вхідного зображення.

Всі ці моделі компілюються з використанням оптимізатора Adam та функції втрати категоріальної крос-ентропії. Кожна модель готується до навчання з фіксованими вагами основної частини передвченої моделі, адже вони вже мають високі рівні представлення, що можна використовувати для розпізнавання захворювань зорового шляху.

Функція train_model використовує два колбеки: ModelCheckpoint та EarlyStopping. ModelCheckpoint дозволяє зберегти модель з найкращими результатами в процесі навчання. EarlyStopping припиняє навчання, якщо показник втрат на валідаційному наборі не покращується протягом певної кількості епох, щоб уникнути перенавчання та заощадити час.

3.3.2 Навчання моделей CNN

Для навчання моделей використовувалися генератори даних для завантаження зображень з відповідних директорій. Це дозволило ефективно працювати з великим обсягом даних, забезпечуючи належну нормалізацію та аугментацію зображень.

Навчимо моделі на зображеннях розміром 256x256 та з кількістю епох 15.

Лістинг 3.4 файл TrainModel.py

```
import ProcessData
```

```
import Models
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore', category=FutureWarning)
warnings.filterwarnings('ignore', category=DeprecationWarning)
train_dir = 'augmented_data/train'
valid_dir = 'augmented_data/valid'
test_dir = 'augmented_data/test'
train_generator = ProcessData.load_normalized_data_from_directory(train_dir)
valid_generator = ProcessData.load_normalized_data_from_directory(valid_dir)
test_generator = ProcessData.load_normalized_data_from_directory(test_dir)
img_size = (256, 256, 3)
cnn_models = Models.CNNModels(img_size=img_size)

# Створюємо модель
model = cnn_models.build_efficientnetb3()


# Навчаємо модель
history = cnn_models.train_model(model, train_generator, valid_generator, epochs=15,
                                 checkpoint_path='app/models/best_random_model.h5')


cnn_models.save_model(model, model_path='app/models/random_model.h5')
```

```
test_loss, test_accuracy = model.evaluate(test_generator)
print(f'Test loss: {test_loss}, Test accuracy: {test_accuracy}')

def plot_training_history(history):
    plt.figure(figsize=(10, 5))

    # Графік точності
    plt.subplot(1, 2, 1)
    plt.plot(history.history['accuracy'], label='Training Accuracy')
    plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.title('Training and Validation Accuracy')
    plt.legend()

    # Графік втрат
    plt.subplot(1, 2, 2)
    plt.plot(history.history['loss'], label='Training Loss')
    plt.plot(history.history['val_loss'], label='Validation Loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.title('Training and Validation Loss')
    plt.legend()

    plt.tight_layout()
    plt.show()

# Виведення графіків
plot_training_history(history)
```

```
211/211 [==============================] - 143s 680ms/step - loss: 0.2075 - accuracy: 0.9208 - val_loss: 0.3806 - val_accuracy: 0.8720
Epoch 7/15
211/211 [==============================] - 143s 677ms/step - loss: 0.1686 - accuracy: 0.9336 - val_loss: 0.4384 - val_accuracy: 0.8697
Epoch 8/15
211/211 [==============================] - 144s 680ms/step - loss: 0.1821 - accuracy: 0.9277 - val_loss: 0.3423 - val_accuracy: 0.8791
Epoch 9/15
211/211 [==============================] - 144s 681ms/step - loss: 0.1465 - accuracy: 0.9398 - val_loss: 0.4244 - val_accuracy: 0.8531
27/27 [==============================] - 16s 587ms/step - loss: 0.2742 - accuracy: 0.8957
Test loss: 0.2741864025592804, Test accuracy: 0.8957346081733704
```

Рисунок 3.5 – Результат навчання моделі ResNet50

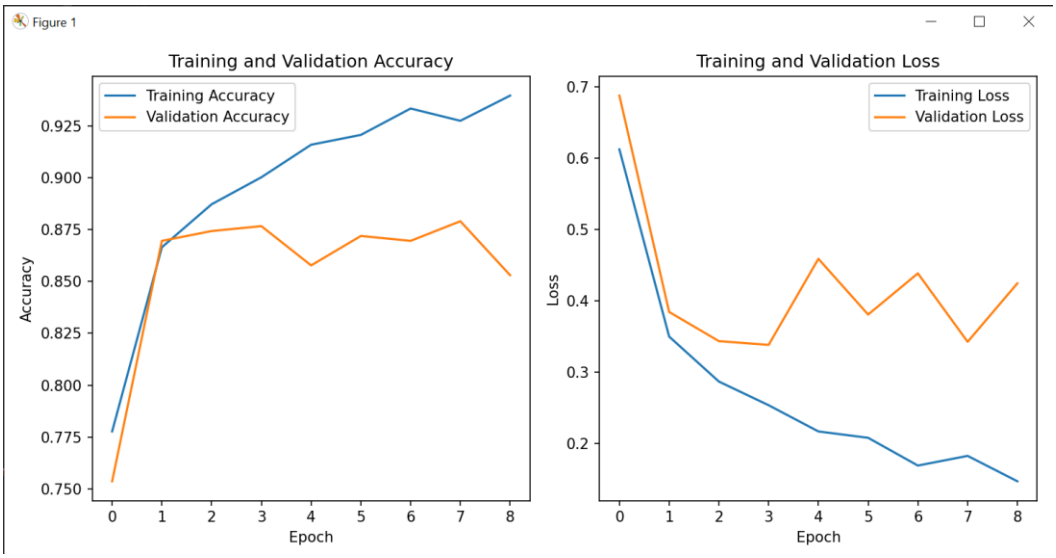

Рисунок 3.6 – Графік точності та втрат під час навчання моделі ResNet50

Характеристика навчання моделі ResNet50:

Модель ResNet50 була навчена на підготовленому датасеті, причому процес навчання завершився на 9-й епосі (рис. 3.5) завдяки використанню механізму EarlyStopping, який зупинив навчання через відсутність покращення валідаційних втрат, що що допомогло запобігти перенавчанню моделі та зберегти найкращі ваги моделі. Протягом перших декількох епох спостерігається значне зниження втрат та підвищення точності на тренувальних даних, що свідчить про ефективне навчання моделі. Початкова точність зросла з 77.76% до 93.98% на тренувальних даних. Валідаційні втрати та точність також демонструють загальну позитивну динаміку, хоча і з певними коливаннями. Найвища валідаційна точність досягла 87.91%, що свідчить про гарну здатність моделі до узагальнення на нових даних. На

епохах з 5 по 9 спостерігається деяке збільшення валідаційних втрат, що може свідчити про початок перенавчання моделі. Проте, загальна точність залишається високою. Оцінка на тестових даних показала точність 89.57%, що підтверджує, що модель має високу здатність до узагальнення і ефективно класифікує зображення на тестовому наборі.

Характеристика навчання моделі EfficientNetB3:

Рисунок 3.7 – Результат навчання моделі EfficientNetB3

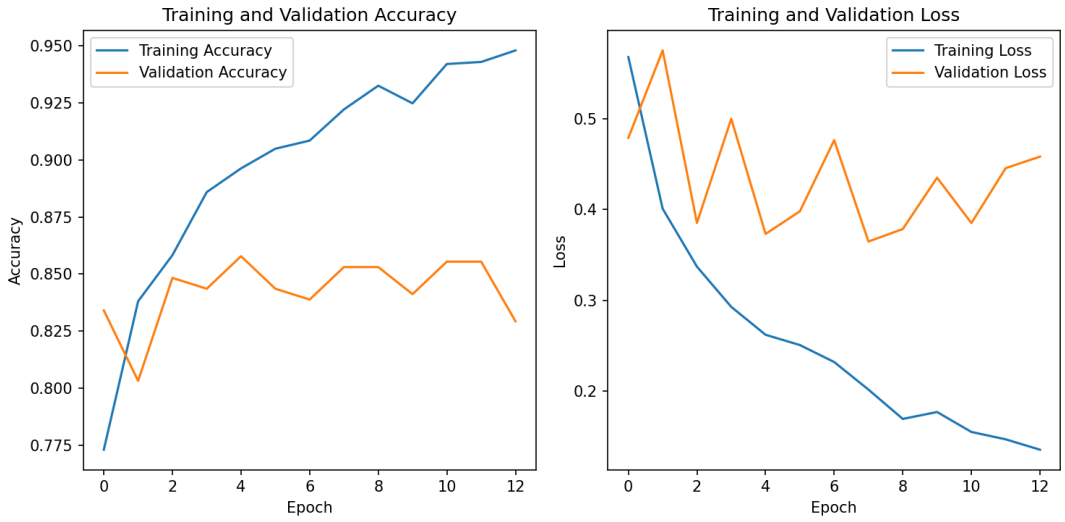

Рисунок 3.8 – Графік точності та втрат під час навчання моделі EfficientNetB3

Модель EfficientNetB3 була навчена на підготовленому датасеті, причому процес навчання завершився на 13-й епосі завдяки використанню механізму EarlyStopping, який зупинив навчання через відсутність покращення валідаційних втрат, що допомогло запобігти перенавчанню моделі та зберегти найкращі ваги моделі. Зниження втрат та підвищення

точності: Протягом перших декількох епох спостерігається значне зниження втрат та підвищення точності на тренувальних даних, що свідчить про ефективне навчання моделі. Початкова точність зросла з 77.32% до 94.78% на тренувальних даних. Валідаційні втрати та точність також демонструють загальну позитивну динаміку, хоча і з певними коливаннями. Найвища валідаційна точність досягла 85.78%, що свідчить про гарну здатність моделі до узагальнення на нових даних. На епохах з 10 по 13 спостерігається деяке збільшення валідаційних втрат, що може свідчити про початок перенавчання моделі. Проте, загальна точність залишається високою. Оцінка на тестових даних показала точність 87.44%, що підтверджує, що модель має високу здатність до узагальнення і ефективно класифікує зображення на тестовому наборі.

Тобто навіть незважаючи на те що на тренувальному наборі даних модель EfficientNetB3 (точність 94.78%) показувала себе дещо краще ніж модель ResNet50 (точність 93.98%), модель ResNet50 забезпечує більш високу точність на тестових даних.

Характеристика навчання базової моделі CNN:

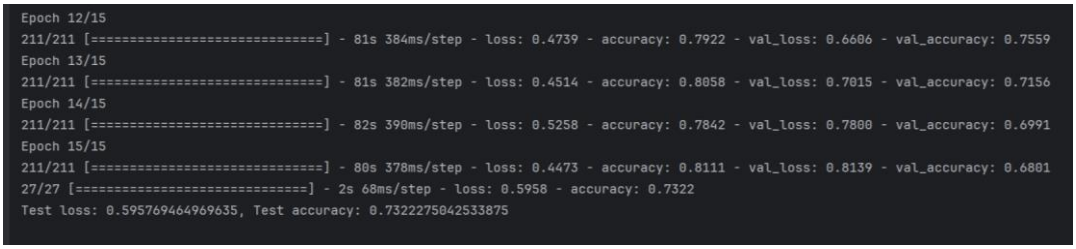

Рисунок 3.9 – Результат навчання базової моделі CNN

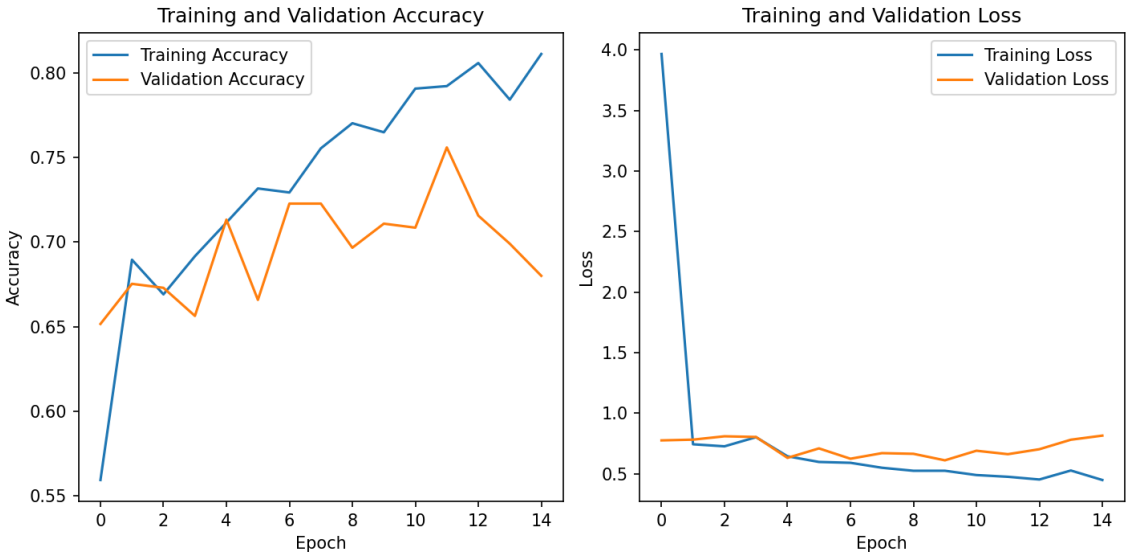

Рисунок 3.10 – Графік точності та втрат під час навчання базової моделі CNN

Модель базової CNN була навчена на підготовленому датасеті. Процес навчання тривав повні 15 епох. Протягом перших кількох епох спостерігається значне зниження втрат і підвищення точності на тренувальних даних, що свідчить про ефективне навчання моделі. Початкова точність зросла з 55.94% до 81.11% на тренувальних даних. Валідаційні втрати та точність демонструють загальну позитивну динаміку, хоча і з певними коливаннями. Найвища валідаційна точність досягла 75.59%, що свідчить про гарну здатність моделі до узагальнення на нових даних. На останніх епохах спостерігається деяке збільшення валідаційних втрат, що може свідчити про початок перенавчання моделі. Проте, загальна точність залишається досить високою. Оцінка на тестових даних показала точність 73.22%, що підтверджує, що модель має достатню здатність до узагальнення і ефективно класифікує зображення на тестовому наборі.

Тож як бачимо серед навчених моделей, модель архітектури ResNet50 найбільшу точність на тестових даних (89.57%). Незважаючи на це, дві інші моделі також мають досить високий показник точності та можуть застосовуватись до класифікації зображень зорового дна.

Спробуємо збільшити точність моделі, навчаючи її на більш деталізованих зображеннях розміром 512x512. Однак це може значно збільшити час навчання та використання пам'яті.

Характеристика навчання моделі ResNet50 для формату зображень 512x512:

```
Epoch 4/15
211/211 [==============================] - 147s 698ms/step - loss: 0.2567 - accuracy: 0.8971 - val_loss: 0.3869 - val_accuracy: 0.8483
Epoch 5/15
211/211 [==============================] - 147s 698ms/step - loss: 0.2337 - accuracy: 0.9087 - val_loss: 0.3552 - val_accuracy: 0.8768
Epoch 6/15
211/211 [==============================] - 148s 701ms/step - loss: 0.2143 - accuracy: 0.9185 - val_loss: 0.3559 - val_accuracy: 0.8483
Epoch 7/15
211/211 [==============================] - 167s 793ms/step - loss: 0.2129 - accuracy: 0.9167 - val_loss: 0.4009 - val_accuracy: 0.8412
Epoch 8/15
211/211 [==============================] - 166s 788ms/step - loss: 0.1638 - accuracy: 0.9363 - val_loss: 0.3998 - val_accuracy: 0.8507
27/27 [==============================] - 21s 757ms/step - loss: 0.3296 - accuracy: 0.8673
Test loss: 0.32956960797309875, Test accuracy: 0.8672986030578613
```

Рисунок 3.11 – Результат навчання ба моделі ResNet50 для формату зображень 512x512

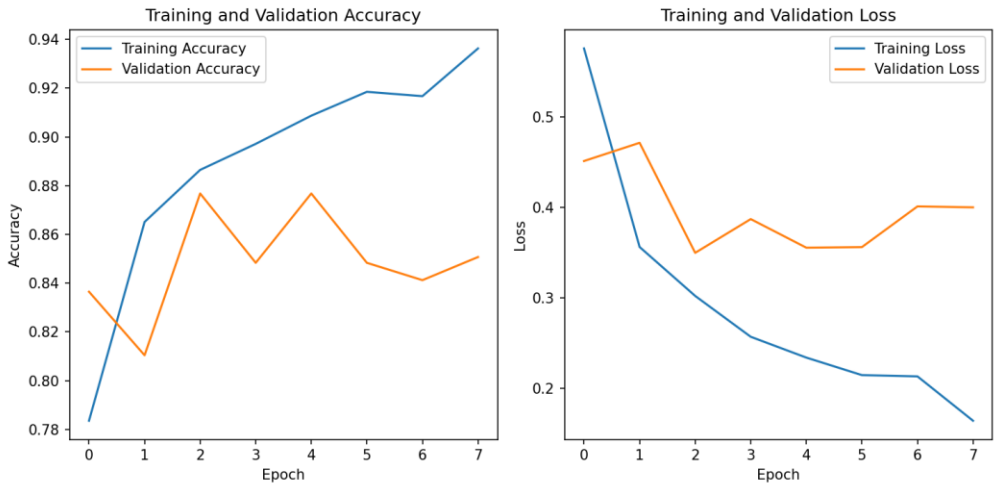

Рисунок 3.12 – Графік точності та втрат під час навчання моделі ResNet50 для формату зображень 512x512

Модель ResNet50 була навчена на зображеннях розміром 512x512 протягом 15 епох. Протягом усіх епох спостерігається зниження втрат і підвищення точності на тренувальних даних, але після деякої кількості епох цей приріст стає меншим, що може свідчити про збільшення складності завдання для моделі. Валідаційні втрати та точність також демонструють

загальну позитивну динаміку, з аналогічними коливаннями. Модель демонструє здатність до узагальнення на нових даних. Оцінка на тестовому наборі показала досить високу точність 86.73%, що свідчить про добре функціонування моделі на нових даних.

Як бачимо точність моделі майже не відрізняється від точності моделі для зображень розміром 256x256, тому навчання моделей на зображеннях розміром 512x512 не є доцільним щодо оптимального використання пам’яті.

3.4 Проведення тестування моделей

Після успішного навчання моделей на тренувальних даних та їх валідації, ми переходимо до етапу тестування, щоб оцінити їхню ефективність та загальну продуктивність. У цьому розділі ми застосовуємо розроблені моделі до тестового набору даних та аналізуємо їхні результати.

Для цього ми використовуємо діаграми розподілу ймовірностей та техніку Grad-CAM для візуалізації того, які області зображення модель вважає найбільш важливими при класифікації. Це дозволяє нам краще розуміти, як модель приймає рішення та які області зображення є вирішальними для її класифікації. Grad-CAM (Gradient–weighted Class Activation Mapping) – це техніка візуалізації та інтерпретації нейронних мереж, яка дозволяє визначити, які області зображення найбільш важливі для прийняття рішення моделлю. Ця техніка дозволяє зрозуміти, які частини зображення впливають на класифікаційний вихід моделі.

Основна ідея Grad-CAM полягає в тому, щоб здійснювати обчислення вагованих сум градієнтів активацій останнього шару передвідомого класу з приводу ваги цих активацій. Це дозволяє визначити вплив кожного пікселя на підсумковий вихід моделі.

Основна ідея Grad-CAM полягає в тому, щоб здійснювати обчислення вагованих сум градієнтів активацій останнього шару передвідомого класу з

приводу ваги цих активацій. Це дозволяє визначити вплив кожного пікселя на підсумковий вихід моделі.

Техніка Grad-CAM може бути корисною для аналізу зображень зорового шляху та виявлення патологічних областей. Вона дозволяє візуалізувати, які області зображення були найбільш важливими для прийняття рішення моделлю.

Застосування Grad-CAM може допомогти лікарям та дослідникам в ідентифікації та візуалізації патологічних змін у зображеннях зорового шляху, таких як діабетична ретинопатія, глаукома тощо. Використовуючи цю техніку, фахівці можуть отримати більш детальне розуміння того, які області зображення підтримують прийняте класифікаційне рішення, що може бути корисним для розробки та вдосконалення алгоритмів автоматичного виявлення патологій.

Застосування Grad-CAM у контексті аналізу зображень зорового шляху може покращити розуміння патологічних змін та допомогти в ранньому виявленні та лікуванні захворювань.

Результати роботи:

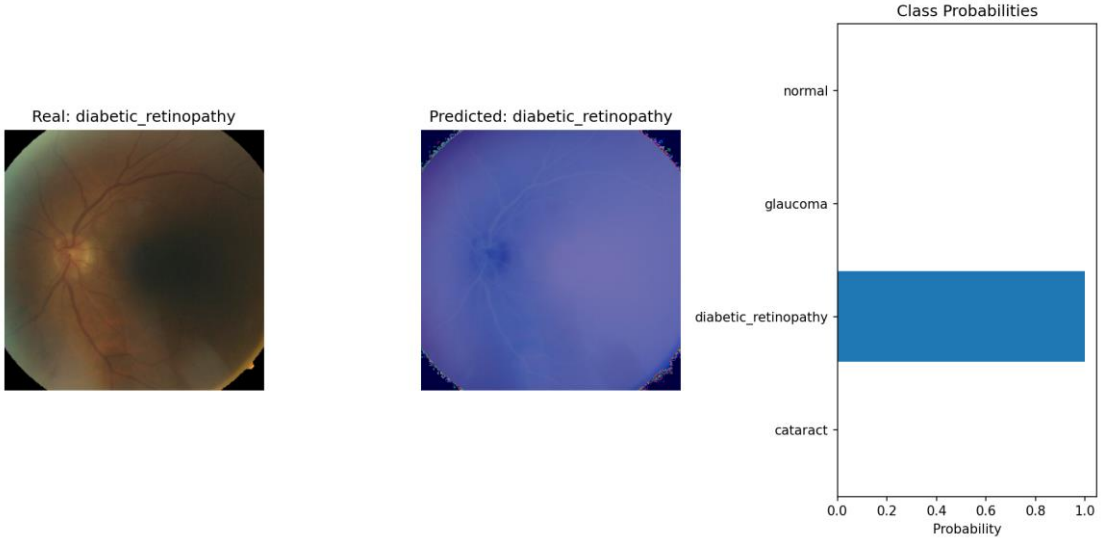

Рисунок 3.13 – Результат роботи базової CNN

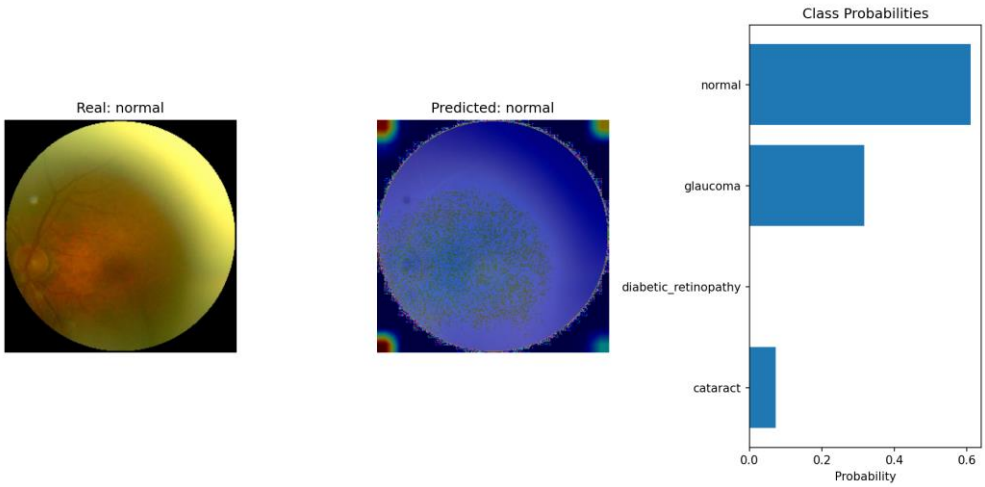

Рисунок 3.14 – Результат роботи базової CNN

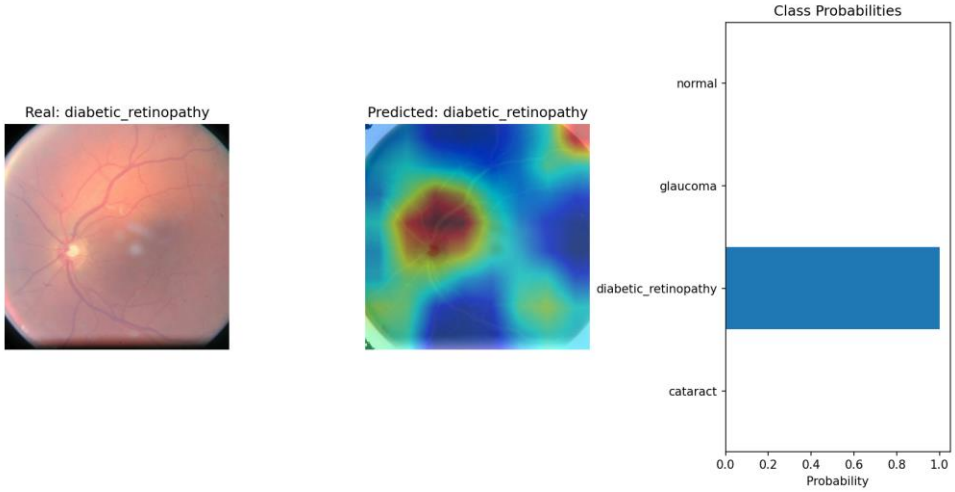

Рисунок 3.15 – Результат роботи базової EfficientNetB3

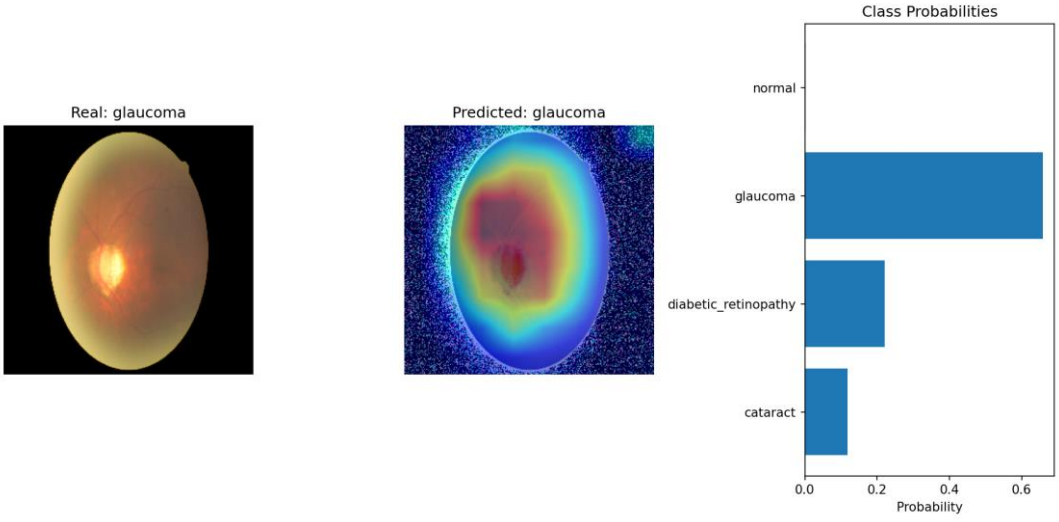

Рисунок 3.16 – Результат роботи базової EfficientNetB3

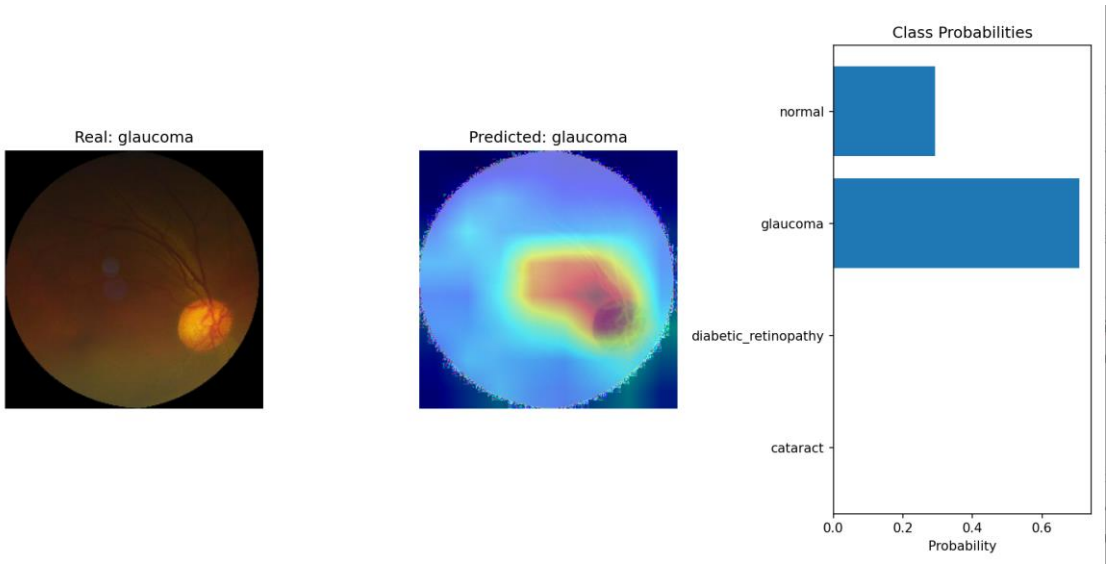

Рисунок 3.17 – Результат роботи базової ResNet50

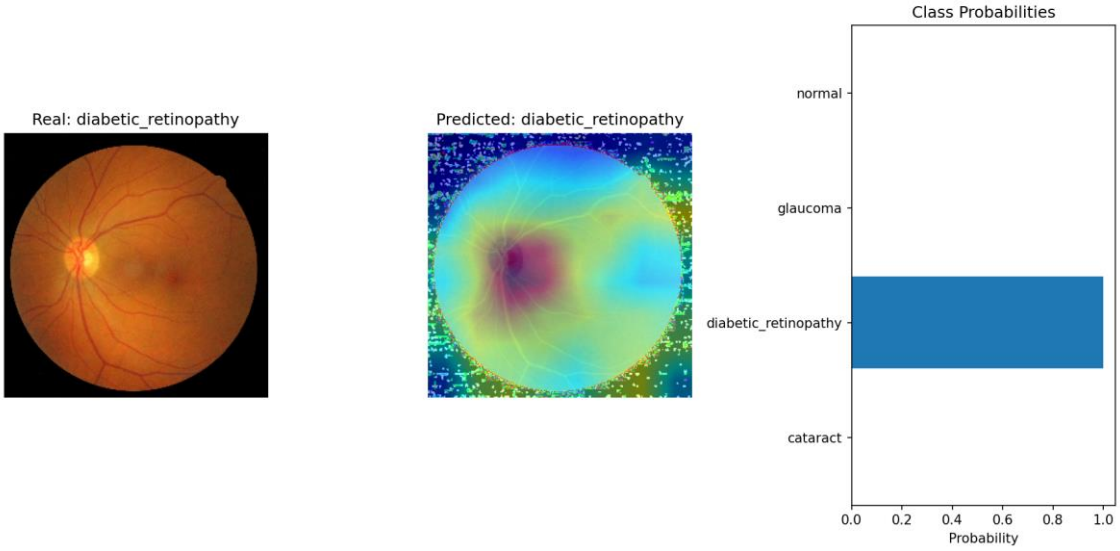

Рисунок 3.18 – Результат роботи базової ResNet50

Тестування моделей показало, що як ResNet, так і EfficientNetB3 перевершують базову модель CNN за точністю та функцією втрат. Використання Grad-CAM допомогло візуалізувати важливі області зображень, що робить моделі більш інтерпретованими та прозорими. Це, в свою чергу, підвищує довіру до моделей та їх використання в реальних медичних застосуваннях для виявлення патологічних областей зорового шляху.

3.5 Інтеграція системи з вебзастосунком

Інтеграція моделі для класифікації зображень зорового шляху в вебзастосунок є ключовим етапом для забезпечення зручного доступу до системи для користувачів. Розглянемо процес інтеграції, включаючи розгортання моделі, створення інтерфейсу користувача та налаштування сервера для обробки запитів.

Для інтеграції моделі машинного навчання з вебзастосунком було обрано Flask – легкий мікрофреймворк для Python. Flask був обраний через свою простоту у використанні, гнучкість та можливість швидкого прототипування, що є критично важливим для дослідницьких проектів та невеликих систем.

3.5.1 Архітектура вебзастосунку

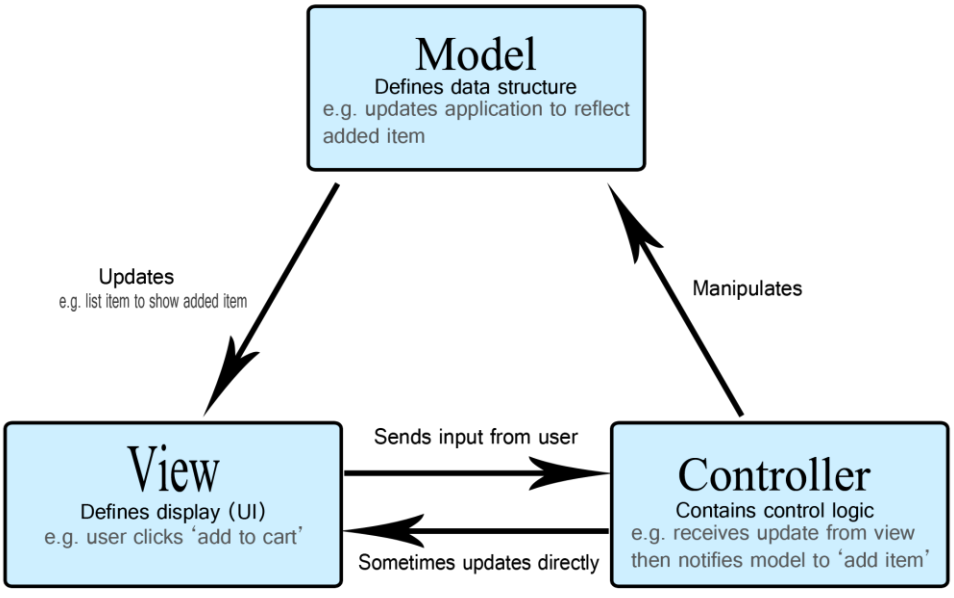

Рисунок 3.19 – Архітектура MVC

Архітектура MVC – це популярний шаблон проєктування програмного забезпечення, який розділяє додаток на три основні компоненти: Model, View і Controller. Це розділення полегшує розробку, тестування та підтримку коду.

Model (Модель) – це рівень, що відповідає за управління даними додатка. Він відповідає за логіку бізнес-процесів, взаємодію з базами даних та обробку даних.

View (Представлення) – це рівень, що відповідає за відображення даних користувачу. View отримує дані від Model і відображає їх у відповідному форматі, наприклад, у вигляді вебсторінки.

Controller (Контролер) – це рівень, що відповідає за обробку запитів від користувача. Controller отримує вхідні дані від View, передає їх Model для обробки і повертає результати назад у View для відображення.

Вебзастосунок побудований за спрощеною архітектурою MVC (Model-View-Controller), де:

– Model: моделі машинного навчання, які оброблюють зображення та робить прогнози;

– View: HTML-сторінка, яка відображає інтерфейс користувача для завантаження зображень та перегляду результатів;

– Controller: код Flask, який приймає запити від користувача, обробляє зображення за допомогою моделі та повертає результати у вигляді відповіді.

Вебзастосунок складається з двох основних компонентів:

– клієнтська частина (frontend): Проста HTML-сторінка з формою для завантаження зображень та функціоналом для відправки даних на сервер та отримання результату;

– серверна частина (backend): Сервер на Flask, який приймає завантажені зображення, обробляє їх за допомогою моделі машинного навчання та повертає результати користувачу.

Рисунок 3.20 – Діаграма потоку даних(DFD) застосунку

На рисунку 3.20 наведено діаграму потоку даних застосунку для кращого розуміння процесв системи та взаємодії між різними компонентами.

3.5.2 Розроблення серверної частини

Для розроблення серверної частини нашого вебзастосунку ми використали Flask, легкий і гнучкий фреймворк для веброзробки на Python. Основними завданнями серверної частини є прийом зображень від користувачів, обробка їх за допомогою моделей машинного навчання, створення теплових карт (Grad-CAM) та повернення результатів назад користувачу.

Створюється Flask-застосунок, який буде приймати HTTP-запити та обробляти їх. Це включає імпорт необхідних бібліотек, таких як Flask для

веб–розробки, OpenCV для обробки зображень та TensorFlow для роботи з моделями машинного навчання.

```python
def create_app():
    app = Flask(__name__)
    from .controllers.main_controller import main
    app.register_blueprint(main)
    return app
```

Рисунок 3.21 – Ініціалізація Flask–застосунку

```python
@main.route('/')
def index():
    from ..views.main_views import render_index
    return render_index()
```

Рисунок 3.22 – Маршрут для головної сторінки

Маршрут для головної сторінки (рис. 3.22) відображає головну HTML-сторінку, яка містить форму для завантаження зображень та вибору моделі. Цей маршрут повертає функцію render_index з main_views, яка відповідає за рендер головної сторінки.

Маршрут для прогнозування зображень ("/predict") реадізується наступним чином:

– сервер приймає зображення від користувача через HTTP POST запит. Перевіряється наявність файлу в запиті. Якщо файл відсутній, повертається помилка;

– декодування зображення: зображення декодується з буфера в формат, придатний для обробки;

– вибір моделі: вибір моделі для прогнозування на основі даних з форми. Це може бути базова модель, EfficientNet або ResNet50. Якщо обрана модель недійсна, повертається помилка;

– завантаження моделі: Завантажується обрана модель з відповідного файлу;

– прогнозування: виконується прогнозування за допомогою обраної моделі. Результати містять передбачений клас та ймовірності для кожного класу;

– генерація теплової карти: генерується теплова карта (Grad-CAM) для візуалізації областей зображення, які вплинули на прогноз моделі;

– накладання теплової карти: Теплова карта накладається на оригінальне зображення, створюючи візуалізацію, яка допомагає зрозуміти рішення моделі;

– кодування та повернення результату: отримане зображення з тепловою картою кодується у формат Base64 та повертається разом з результатами прогнозування у форматі JSON.

```python
@main.route( rule: '/recognition', methods=['POST'])
def recognition():
    if 'file' not in request.files:
        return jsonify({'error': 'No file provided'}), 400
    file = request.files['file']
    img = cv2.imdecode(np.frombuffer(file.read(), np.uint8), cv2.IMREAD_COLOR)
    if img is None:
        return jsonify({'error': 'Failed to decode image'}), 400
    is_fundus_result =is_fundus(img)

    return jsonify({
        'is_healthy': is_fundus_result
    })
```

Рисунок 3.23 – Маршрут для виявлення зображення зорового дна ("/recognition")

Маршрут для виявлення зображення зорового дна (рис. 3.23) реалізується наступним чином:

– прийом зображення: сервер приймає зображення від користувача через HTTP POST запит. Перевіряється наявність файлу в запиті. Якщо файл відсутній, повертається помилка;

– декодування зображення: зображення декодується з буфера в формат, придатний для обробки;

– аналіз зображення: виконується аналіз, чи є зображення зображенням зорового дна, за допомогою навченої CNN моделі для бінарної класифікації;

– повернення результату: результат класифікації (чи є зображення зображенням зорового дна) повертається у форматі JSON.

Таким чином, серверна частина на Flask забезпечує ефективний механізм для інтеграції моделі машинного навчання з вебзастосунком, дозволяючи користувачам взаємодіяти з моделлю через веб–інтерфейс без необхідності зберігати зображення на сервері.

3.5.3 Розроблення клієнтської частини

Клієнтська частина вебзастосунку відповідає за інтерфейс взаємодії користувача з серверною частиною. Вона складається з HTML-сторінки з формою для завантаження зображень, вибору моделі для прогнозування та кнопкою для відправки даних на сервер. Вона також містить функціонал для відображення результатів прогнозування.

```
</head>
<body>
    <div class="container">
        <div class="upload-block">
            <div class="image-container" id="image-container">
                <img id="uploaded-image" alt="Uploaded Image">
            </div>
            <div class="analysis-block">
                 <div class="model-selection-block">
            <label for="model-select">Обрати архітектуру моделі:</label>
            <select id="model-select">
                <option value="base">Base CNN</option>
                <option value="efficientnet">EfficientNetB3</option>
                <option value="resnet50">ResNet50</option>
            </select>
              </div>
                <input type="file" id="file-input" accept="image/*" required>
                <label for="file-input">Обрати зображення</label>
                <button id="analyze-button">Аналізувати</button>
            </div>
        </div>
        <div class="results" id="results"></div>
        <div id="heatmap-container">
            <div id="heatmap-stage-container"></div>
            <div id="legend">
                <h3>Legend</h3>
                <div><div><span class="color-box high"></span></div>High Activity</div>
                <div><div><span class="color-box medium"></span></div>Medium Activity</div>
                <div><div><span class="color-box low"></span></div>Low Activity</div>
            </div>
        </div>
```

Рисунок 3.23 – Частина коду HTML-структури

Основні компоненти клієнтської частини:

– HTML-структура: створює базову розмітку сторінки, форму для завантаження зображень, вибору моделі та місце для відображення результатів;

– CSS-стилі: відповідає за візуальне оформлення сторінки, забезпечуючи приємний і зручний для користувача інтерфейс;

– JavaScript: обробляє події на сторінці, зокрема завантаження зображень, відправку запитів на сервер і відображення результатів прогнозування.

Опис роботи скрипту JavaScript:

– завантаження зображення: після завантаження зображення користувачем скрипт відправляє запит на сервер, щоб перевірити, чи є це зображенням зорового дна. Якщо так, кнопка для аналізу стає доступною;

– аналіз зображення: коли користувач натискає кнопку «Аналізувати», зображення та обрана модель відправляються на сервер для прогнозування. Сервер повертає прогноз, який включає ймовірності для кожного класу та накладання теплової карти;

– відображення результатів: прогнозований клас та ймовірності відображаються на сторінці. Якщо ймовірність патологічного стану перевищує 50%, користувач може натиснути кнопку для перегляду теплових карт можливих патологічних областей;

– теплова карта: після натискання на кнопку для перегляду патологічних областей, відображається зображення з тепловою картою, яка вказує на ділянки з високою, середньою та низькою активністю.

Додаткові моменти:

– адаптивний дизайн: інтерфейс адаптивний та підходить для різних розмірів екрану завдяки використанню flexbox;

– валідація: скрипт перевіряє, чи було завантажено зображення перед відправкою на сервер для аналізу

– інтерфейс українською мовою: всі елементи інтерфейсу та повідомлення виведені українською мовою для зручності користувачів.

3.5.4 Робота вебзастосунку

Перед початком роботи вебзастосунку необхідно запустити серверну частину застосунку. Для цього використовується серверний фреймворк, такий як Flask. Після запуску сервера вебзастосунок стає доступним для користувачів через веббраузер.

Користувач відкриває вебзастосунок у своєму веббраузері, введенням URL–адреси локального сервера: http://127.0.0.1:5000 вебзастосунку у відповідне поле браузера. Після цього вебзастосунок завантажується на екран користувача і готовий до використання.

Користувач обирає зображення зорового дна для аналізу. Для цього він натискає на кнопку або посилання "Обрати зображення", після чого відкривається діалогове вікно для вибору файлу.

Після вибору зображення користувач завантажує його на сервер, натискаючи кнопку «Обрати зображення». Обране зображення пересилається на сервер для подальшої обробки та аналізу.

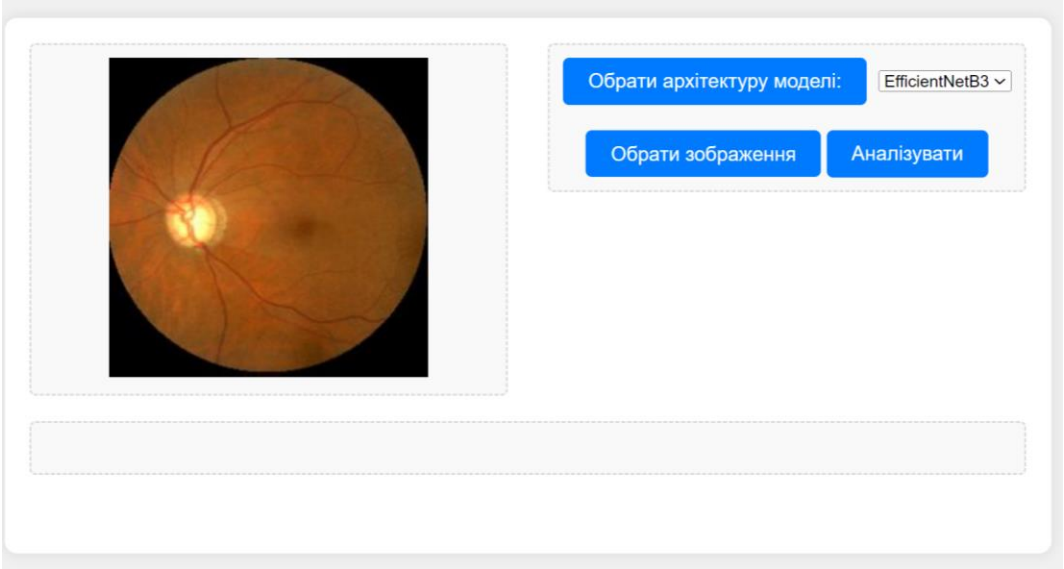

Рисунок 3.23 – Результат роботи вебзастосунку

Користувач обирає бажану модель для аналізу зображення, використовуючи випадаючий список. Він може вибрати одну з доступних моделей машинного навчання, які надаються вебзастосунком.

Після завантаження зображення та вибору моделі користувач запускає аналіз зображення, натискаючи кнопку «Аналізувати». На цьому етапі відбувається передача зображення та вибраної моделі на сервер для подальшого аналізу та обробки.

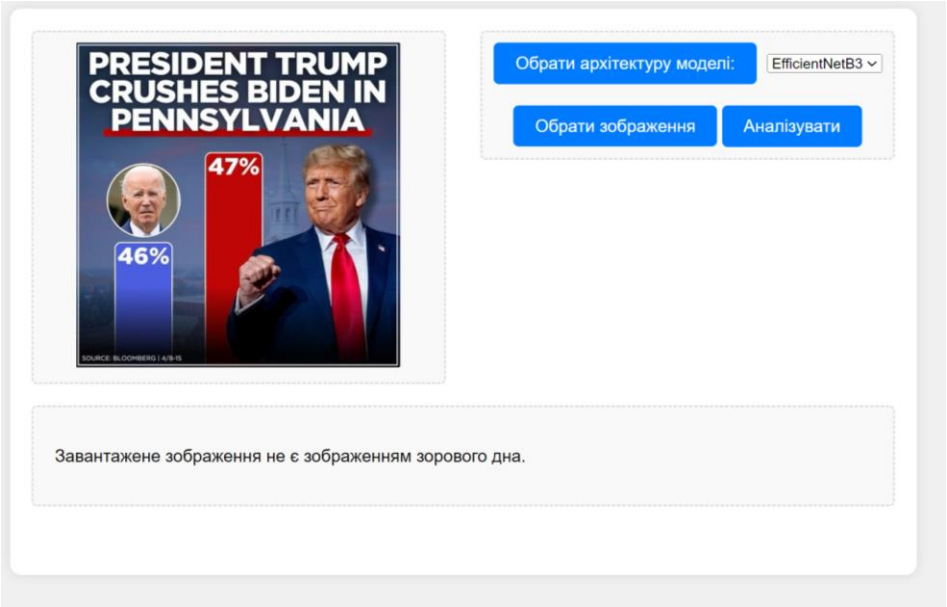

Рисунок 3.24 – Результат роботи вебзастосунку

Сервер отримує запит від користувача на обробку зображення та аналіз його за допомогою вибраної моделі машинного навчання.

Сервер обробляє отримане зображення та генерує прогнози за допомогою вибраної моделі машинного навчання.

Якщо результат аналізу показує наявність патологічних областей на зображенні, сервер генерує теплову карту. Ця карта візуалізує активні області на зображенні та показує, які саме області вплинули на результат аналізу.

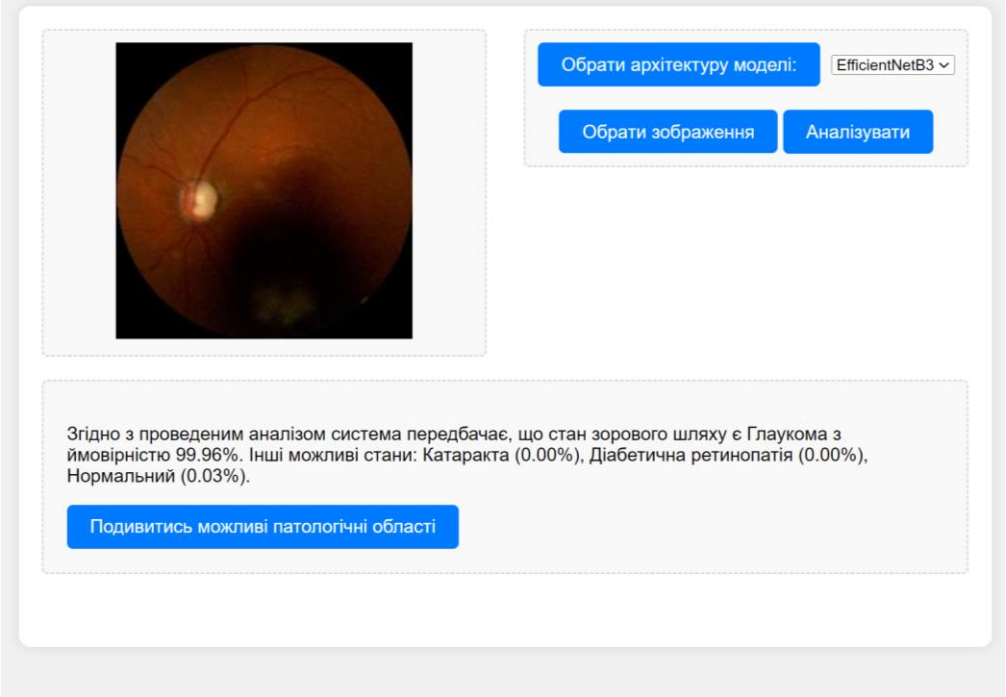

Рисунок 3.25 – Результат роботи вебзастосунку

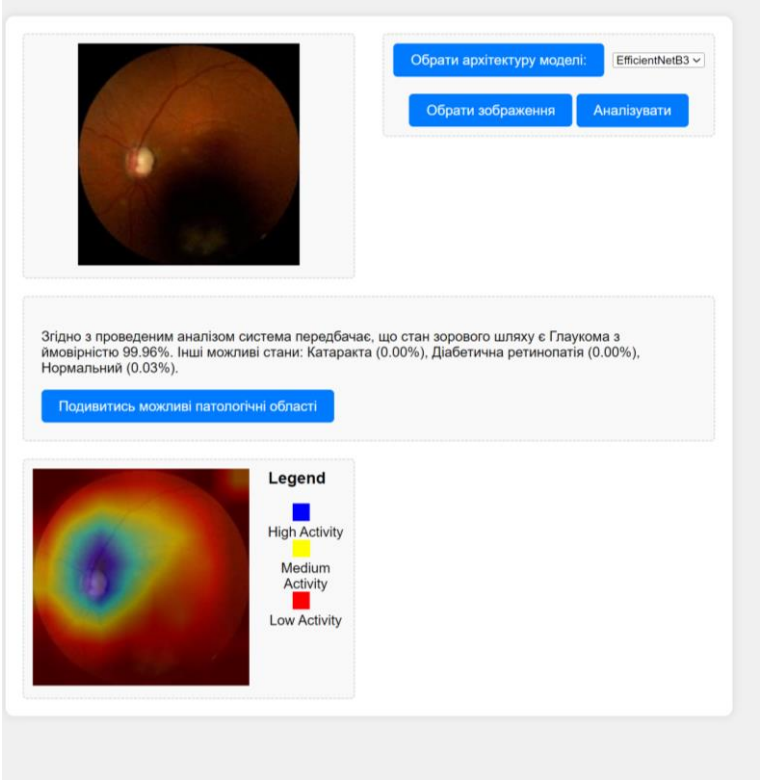

Рисунок 3.26 – Результат роботи вебзастосунку

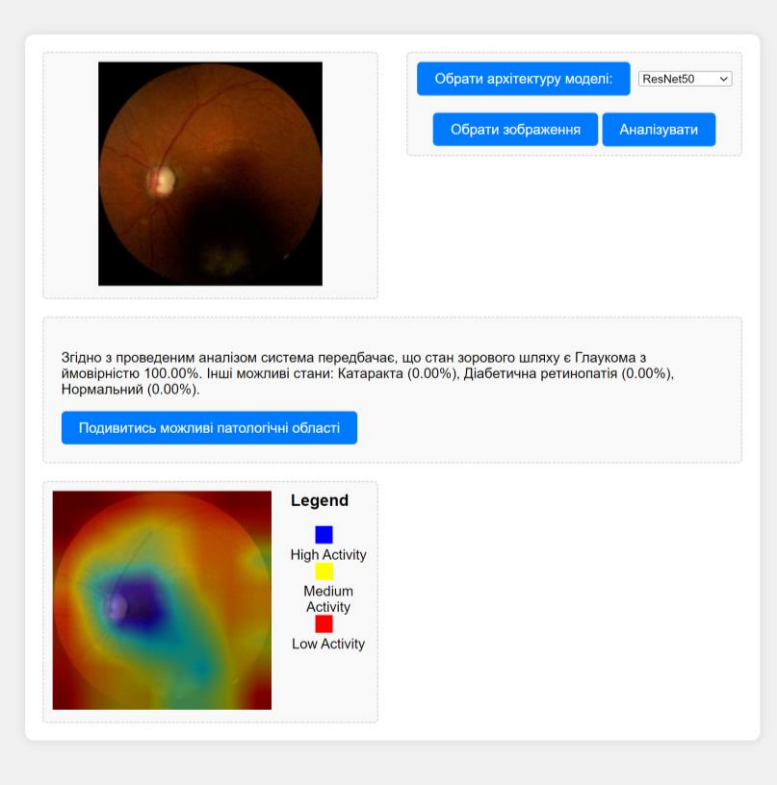

Рисунок 3.27 – Результат роботи вебзастосунку

Сервер надсилає результати аналізу користувачеві у вигляді текстової інформації, що можна побачити на рисунках 3.25, та можливості перегляду теплової карти (за необхідності) (рис.3.26). Ці результати відображаються на екрані вебзастосунку для подальшого ознайомлення користувача з ними.

Користувач отримує результати аналізу на екрані свого веббраузера, де відображаються прогнозований клас зорового захворювання, інші можливі стани, що були розглянуті моделлю, та, в разі необхідності, можливість перегляду теплової карти з відзначеними патологічними областями.

3.6 Перспективи подальшої роботи

Під час розробки були враховані поставлені завданням умови та створено функціональну систему, яка може ефективно аналізувати проблеми зорового шляху людини. Таким чином, виконано великий обсяг роботи, але

все ще залишаються напрямки, які потребують подальшої доробки. Для цього можуть бути використані різні підходи.

Таким підходом може бути розширення набору даних для навчання моделей може включати збільшення обсягу наявних даних або збір нових даних з різних джерел. Це дозволить моделям отримувати більше різноманітної інформації та покращити їх універсальність і точність прогнозування.

Іншим підходом може бути проведення експериментів з різними архітектурами нейронних мереж дозволить визначити оптимальну конфігурацію для конкретної задачі. Можливі варіації включають зміни у розмірі та кількості шарів мережі, використання різних типів шарів (згорткові) та застосування різних методів регуляризації.

Після вдосконалення моделей важливо провести детальний аналіз їх результатів та зворотний зв’язок. Це дозволить виявити слабкі сторони моделей та ідентифікувати області для подальшого вдосконалення.

Розширення функціональності вебзастосунку також є важливим кроком для поліпшення його користувацького досвіду та використання.

Розширення функціоналу для більш детального аналізу зображень може включати відображення додаткової інформації про розпізнані патології, такої як місце їх розташування, розмір та інші характеристики. А також додавання функціоналу для генерації та завантаження звітів про результати аналізу може стати корисним для подальшого аналізу та збереження результатів для майбутнього використання.

Головною метою подальшої роботи є вдосконалення системи для впровадження її у медичну практику, де вона може внести вклад у поліпшенні розуміння функціонування шляхів передачі зображення від зорового нерву та навчання нейронних структур мозку оптимальним шляхам передачі зображення від зорового нерву за допомогою аналізу зорового шляху. Для успішного впровадження системи у медичну практику важливо взаємодіяти з медичними працівниками та фахівцями офтальмології та

нейронауки щоб покращити результати її роботи в клінічному середовищі. Це включає аналіз точності діагностики, порівняння результатів з іншими методами діагностики, а також збір фідбеку від медичного персоналу та пацієнтів. Оцінка ефективності системи допоможе виявити її сильні та слабкі сторони та внести необхідні корективи для подальшого вдосконалення.

Перед впровадженням системи у медичну практику необхідно враховувати вимоги регуляторних органів щодо безпеки та ефективності медичних систем. Це включає виконання всіх необхідних сертифікаційних процедур та забезпечення відповідності стандартам і правилам безпеки даних. Після впровадження системи у медичну практику слід продовжити дослідження у напрямку оптимізації її роботи. Це може включати вивчення нових методів аналізу даних, дослідження впливу системи на практику клінічної діагностики та розробку нових моделей для покращення точності прогнозування патологій зорового шляху.

ВИСНОВКИ

У рамках кваліфікаційної роботи було розроблено та випробувано систему для аналізу зображень зорового шляху та виявлення патологій за допомогою машинного навчання. Починаючи зі збору даних та підготовки набору даних до розробки моделей машинного навчання, було пройдено усі етапи розробки програмного забезпечення, від створення моделей до інтеграції їх у вебзастосунок.

Використання методів машинного навчання, таких як згорткові нейронні мережі, дозволило створити ефективні моделі для автоматичного аналізу зображень та класифікації патологій зорового шляху. Для розробки моделей машинного навчання використовувалися бібліотеки TensorFlow та Keras. Для попередньої обробки зображень, такої як масштабування та аугментація, використовували бібліотеку OpenCV

Інтеграція цих моделей у вебзастосунок через фреймворк Flask створила зручне та доступне інструмент для медичних працівників та пацієнтів. Загальний архітектурний підхід включав в себе розробку та навчання моделей машинного навчання на локальному комп’ютері, після чого інтеграцію цих моделей у вебзастосунок через Flask. Користувачі могли завантажити свої зображення, які потім аналізувалися за допомогою розроблених моделей, а результати відображалися на вебсторінці в зручному форматі.

Розроблена система має потенціал для впровадження у медичну практику, де вона може стати корисним інструментом для ранньої діагностики та моніторингу захворювань зорового шляху. Подальші дослідження та вдосконалення системи є ключовими напрямками для подальшої роботи, а впровадження в клінічну практику може сприяти покращенню діагностики та лікування пацієнтів з захворюваннями зорового нерву.

Результати роботи мають значення для розвитку медичної діагностики зорового шляху та нейронауки з використанням штучного інтелекту. Результати роботи апробовано у вигляді тез доповіді під час Всеукраїнської студентської наукової конференції «НАУКОВИЙ ПРОСТІР: АНАЛІЗ, СУЧАСНИЙ СТАН, ТРЕНДИ ТА ПЕРСПЕКТИВИ».

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Aggarwal, C. C. (2018). Neural Networks and Deep Learning: A Textbook. Springer.

2. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

3. Armstrong, R. A., & Cubbidge, R. C. (2019). The Eye and Vision: An Overview. In Handbook of Nutrition, Diet, and the Eye (Second Edition), eds. V. R. Preedy & R. R. Watson (pp. 3–14). Academic Press.

4. Remington, L. A. (2016). Clinical Anatomy of the Visual System. Elsevier.

5. Ajay, Pandey, M. (2022). Artificial Intelligence in Healthcare: Diabetic Retinopathy. In: Gupta, D., Polkowski, Z., Khanna, A., Bhattacharyya, S., Castillo, O. (eds) Proceedings of Data Analytics and Management . Lecture Notes on Data Engineering and Communications Technologies, vol 90. Springer, Singapore.

6. Vought, R., Vought, V., Shah, M., et al. (2023). EyeArt artificial intelligence analysis of diabetic retinopathy in retinal screening events.

7. Piatti, A., Romeo, F., Manti, R. et al. Feasibility and accuracy of the screening for diabetic retinopathy using a fundus camera and an artificial intelligence pre–evaluation application. Acta Diabetol 61, 63–68 (2024).

8. Zafar, S., Mahjoub, H., Mehta, N. et al. Artificial Intelligence Algorithms in Diabetic Retinopathy Screening. Curr Diab Rep 22, 267–274 (2022).

9. Bellemo, V., Lim, G., Rim, T.H. et al. Artificial Intelligence Screening for Diabetic Retinopathy: the Real–World Emerging Application. Curr Diab Rep 19, 72 (2019).

10. Pieczynski, J., Kuklo, P. & Grzybowski, A. The Role of Telemedicine, In–Home Testing and Artificial Intelligence to Alleviate an Increasingly Burdened Healthcare System: Diabetic Retinopathy. Ophthalmol Ther 10, 445–464 (2021).

11. Ruamviboonsuk, P., Semturs, C., Raman, R., Nganthavee, V., Chotcomwongse, P. (2020). Artificial Intelligence in the Assessment of Macular

Disorders. In: Chang, A., Mieler, W.F., Ohji, M. (eds) Macular Surgery. Springer, Singapore.

12. Kumar, A., Menia, N.K., Agarwal, A. (2021). Artificial Intelligence in Retinal Diseases. In: Ichhpujani, P., Thakur, S. (eds) Artificial Intelligence and Ophthalmology. Current Practices in Ophthalmology. Springer, Singapore.

13. Campbell, J.P., Ostmo, S.R., Chiang, M.F. (2021). Artificial Intelligence for Retinopathy of Prematurity Diagnosis. In: Wu, WC., Lam, WC. (eds) A Quick Guide to Pediatric Retina. Springer, Singapore.

14. Convolutional networks. URL: https://cs231n.github.io/convolutional–networks (дата звернення 23.04.2024).

15. Fully connected layer. URL: https://medium.com/@vaibhav1403/fully–connected–layer–f13275337c7c (дата звернення 26.04.2024).

16. Understanding convolutional neural networks: A beginner's journey into the architecture. URL: https://medium.com/codex/understanding–convolutional–neural–networks–a–beginners–journey–into–the–architecture–aab30dface10 (дата звернення 26.04.2024).

17. Activation functions in deep learning: Sigmoid, tanh, ReLU. URL: https://artemoppermann.com/activation–functions–in–deep–learning–sigmoid–tanh–relu (дата звернення 27.04.2024).

18. Loss function in machine learning. URL: https://www.datacamp.com/tutorial/loss–function–in–machine–learning (дата звернення 28.04.2024).

19. Generative adversarial networks. URL: https://medium.com/@marcodelpra/generative–adversarial–networks–dba10e1b4424 (дата звернення 28.04.2024).

20. Generative adversarial network (GAN). URL: https://www.geeksforgeeks.org/generative–adversarial–network–gan (дата звернення 28.04.2024).

21. Kuzomin O. Agent–Based Model as a Research Too / Kuzomin O., Lyashenko V. l // International Journal of Academic Information Systems Research (IJAISR). – 2022. – Vol. 6(5). – pp. 17–21.

22. Kuzomin O. Methods analysis depict legen For diagnosis of COVID / VO Prokipets , O. Ya . Kuzyomin // INTERNATIONAL SCIENTIFIC JOURNALGRAIL OF SCIENCE / VO Prokipets , O. Ya . Kuzyomin . – Viden , 2022. pp. 356–361.

23. Kuzomin O. Rozrobka that investigation detector masks For exposing opencv , keras / tensorflow and deep _ navchannyam . G. S. Verkolab , O. Ya . Kuzyomin . Technologies and strategies for the implementation of scientific achievements: collection of additional materials _ "SCIENTIA" I Mizhnar . sc. – theory. conf. T. 2, herbs. 27, 2022. Stockholm, SWE: European Scientific Platform. pp. 78–83 .

24. Kuzomin O. Development and research of image segmentation using mask r– cnn , grabcut and opencv , International scientific journal "Grail of Science"/ Terebetskyi MA, Kuzomin OY 2022, 14/15, pp. 362–368.

25. O. Kuzomin . Decision support procedures for decision making in a COVID condition / Boboyorov Sardor Uchqun o'gli , O. Kuzomin , V. Lyashenko // Multidisciplinary Journal of Science and Technology, 3(2), 2023. R. 74–80.

26. Kuzomin O. Creation of intelligent systems for analyzing supermarket visitors to identify criminal elements / Berkovskyi D., Kuzomin O. Collection of Scientific Papers “SCIENTIA”, (May 5, 2023; Sydney, Australia), 113–118.

27. Kuzomin O. Modeling repetitions to understand in unbalanced flows of money / SE Kholodov , Kuzomino . Ya // Rozvitok scientific thoughts of the post–industrial marriage: current discourse: materials of the III International scientific conference, April 28, 2023, Lviv . – Vinnytsia : "European" Science Platform", 2023. – P. 112–119.

28. Kuzomin O. Spilny autoencoder From the threshold of detection anomalies sugloba on virobnichikh lines / AE Shustrova , Kuzomin O. // Y. Rozvitok scientific thoughts of the post–industrial matrimony: current discourse:

materials of the III International scientific conference, April 28, 2023, Lviv . – Vinnytsia : "European" Science Platform", 2023. – pp. 120–125.

29. Kuzomin O. Follow–up "diamond models" shodo vrahuvannya appointment link between motivation at zd i ysnenny hacker cyber attacks . Stebaev , D. _ UNIVERSUM, (2), (2023). P. 75–84.

30. Kuzomin O. Followup great models For translation Ukrainian movie h wikiristanyam piece intelligence . Kuzomin O., Stebaev , I. _ UNIVERSUM, (2), (2023). 85–94.

31. Kuzomin O. _ Follow–up methods neural languages models For translation Ukrainian movie h wikiristanyam piece intelligence / Popov I . , Kuzomin O. _ // UINVERSUM.2023. No. 2. S. 95–99.

32. Kuzomin O. Investigation and investigation segmentation picture for help mask r – cnn , grabcut and opencv / Terebetsky M., Kuzomin O.// Grail of Science. 2022. No. 14–15. pp. 362–368.

34. Kuzomin O. Follow–up methods reversals spillage program code, like based on static analysis program / Terebetsky M., Kuzomin O. // UINVERSUM. 2023. No. _ 2. P. 100–105.

35. O. Kuzomin ABOUT. I. _ Follow–up methods contextually – stale recognition of human activities behind I'll help you hybrid models deep navchannya / M. D'yachenko , O. Kuzomin // Theoretically that practical zastosuvannya results daily sciences: materials of the V International student scientific conference, June 27, 2023. – M. Rivne, Ukraine – p. 121–124.

36. O. Kuzomin . Usage of data science methodology in computer security usage of data science methodology in computer security / Valentyn Prokipets , Oleksandr Kuzomin // Digitalization of science and everyday life trends and its development: materials V International student scientific conference, m. Vinnytsia : LLC "UKRLOGOS Group", 2023. P. 270–272.

37. Kuzomin O. Methods analysis depicting legends for diagnostics COVID / Prokipets V. , Kuzomin O. // TECHNOLOGIES AND SYSTEMS.ARTICLE.

DOI 10.36074/grail–of–science.27.05.2022.063 International scientific journal "Grail of Science" | No. 14–15 (May, 2022). pp. 356 – 358

38. Kuzomin O. Composite autoencoder with a threshold for detecting anomalies on signal lines / AE Shustrova , Kuzomin O. Ya . // Development of the scientific thought of post–industrial marriage: current discourse: materials of the III International Scientific Conference, April 28, 2023, Lviv . – Vinnytsia : "European Science Platform", 2023. – P. 120–125.

39. Kuzomin O. Rozrobka will install with GSM alarm DV Medentsev , O. Ya . Kuzyomin – 2023 – Problems and prospects for the implementation and promotion of interdisciplinary scientific achievements: materials of the V International Scientific Conference, m. Ivano–Frankivsk, 9 Chernya , 2023 / International Center for Scientific Research. — Vinnytsia : European Science Platform, 2023. P.172 –175.

40. Eye diseases classification. URL: https://www.kaggle.com/datasets/gunavenkatdoddi/eye-diseases-classification (дата звернення 07.05.2024).

41. Indian Diabetic Retinopathy Image Dataset (IDRiD). URL: https://ieee-dataport.org/open-access/indian-diabetic-retinopathy-image-dataset-idrid (дата звернення 09.05.2024).

42. HRF. URL: https://paperswithcode.com/dataset/hrf (дата звернення 11.05.2024).

43. Ocular disease recognition (ODIR-5K). URL: https://www.kaggle.com/datasets/andrewmvd/ocular-disease-recognition-odir5k (дата звернення 12.05.2024).

44. Гема О.Г. Розробка системи для забезпечення перенавчання нейронних структур мозку щодо функціонування шляхів передачі зображення від зорового нерву. Науковий простір: аналіз, сучасний стан, тренди та перспективи: матеріали V Всеукраїнської студентської наукової конференції (м. Київ, 17 травня, 2024 рік) / ГО «Молодіжна наукова ліга». — Вінниця: ТОВ «УКРЛОГОС Груп», 2024. С. 398-401