НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

НН Інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

До захисту допущено:

Завідувач кафедри

_____ Оксана ТИМОЩУК

«___»_____2023 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системний аналіз і управління»

спеціальності 124 «Системний аналіз»

на тему: «Розпізнавання лейкоцитів глибокими згортковими нейронними мережами»

Виконав:
студент IV курсу, групи КА-95
Заяць Владислав Андрійович _____

Керівник:
професор кафедри ММСА, д.т.н. Данилов Валерій Якович _____

Консультант з нормоконтролю:
к.ф.-м.н. Статкевич Віталій Михайлович _____

Консультант з економічного розділу:
доц. к.е.н. Рощина Надія Василівна _____

Рецензент:
директор НН ФТІ, д.т.н., професор Новіков Олексій Михайлович _____

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.
Студент _____

Київ – 2023 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
НН Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 124 «Системний аналіз»

Освітня програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Оксана ТИМОЩУК
«__» травня 2023 р.

ЗАВДАННЯ
на дипломну роботу студенту
Зайцю Владиславу Андрійовичу

1. Тема роботи «Розпізнавання лейкоцитів глибокими згортковими нейронними мережами», керівник роботи професор кафедри ММСА, д.т.н. Данилов Валерій Якович, затверджені наказом по університету від «__» травня 2023 р. № _____

2. Термін подання студентом роботи __.06.2023.

3. Вихідні дані до роботи.

1. Операційна система Windows 10.

2. Частота процессора 3.5 ГГц.

3. Мова програмування Python 3.

4. Середовище розробки – Jupyter Notebook.

5. Бібліотеки, що використовувалися: NumPy, Pandas, matplotlib, TensorFlow, Keras, OpenCV.

4. Зміст роботи.

1. Проаналізувати існуючі математичні моделі.

2. Проаналізувати моделі побудови нейронних мереж CNN.

3. Розробити систему оцінки побудованої моделі.

4. Виконати економічний аналіз програмного продукту.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо).

1. Презентація.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Рощина Н.В., доцент		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Затвердження теми ДР	16.04.2023 – 30.04.2023	виконано
2	Ознайомлення зі структурою БДР згідно з Положенням про державну атестацію студентів НТУУ «КПІ ім. І. Сікорського»	16.04.2023 – 22.04.2023	виконано
3	Ознайомлення з ДСТУ 3008-95 та стандарти ЄСПД	16.04.2023 – 22.04.2023	виконано
4	Проведення дослідження за темою БДР під керівництвом керівника	23.04.2023 – 08.05.203	виконано
5	Завершення роботи над першим варіантом частини БДР	08.05.2023 – 12.05.2023	виконано
6	Проведення роботи над експериментальною частиною БДР	08.05.2023 – 16.05.2023	виконано
7	Проведення роботи над програмним продуктом	16.05.2023 – 24.05.2023	виконано
8	Оформлення БДР та аналіз отриманих результатів	16.05.2023 – 24.05.2023	виконано

Студент Владислав ЗАЯЦЬ

Керівник Валерій ДАНИЛОВ

РЕФЕРАТ

Дипломна робота: 117 с., 19 рис., 8 табл., 2 додатки, 30 джерел.

ЛЕЙКОЦИТИ, МАШИННЕ НАВЧАННЯ, НЕЙРОННІ МЕРЕЖІ, КОМП’ЮТЕРНИЙ ЗІР.

Об’єкт дослідження – набір знімків білих клітин крові організму людини під мікроскопом BCCD.

Мета роботи – розробити систему розпізнавання лейкоцитів на знімках з мікроскопа та їх класифікації на 4 типи - еозинофіли, лімфоцити, моноцити та нейтрофіли.

Предмет дослідження – система розпізнавання лейкоцитів на основі глибокої згорткової нейронної мережі.

Представлена система може використовуватися для розпізнавання білих клітин крові людини.

ABSTRACT

Thesis: 117 pages, 19 figures, 8 tables, 2 appendices, 30 sources.

LEUKOCYTES, MACHINE LEARNING, NEURAL NETWORKS, COMPUTER VISION.

The object of the study is a set of images of white blood cells of the human body under a BCCD microscope.

The purpose of the work is to develop a system for recognizing leukocytes on microscope images and classifying them into 4 types - eosinophils, lymphocytes, monocytes, and neutrophils.

The subject of research is a leukocyte recognition system based on a deep convolutional neural network.

The presented system can be used to recognize human white blood cells.

ЗМІСТ

ВСТУП

Сучасний розвиток комп’ютерних технологій та штучного інтелекту знайшов широке застосування у багатьох галузях. Глибокі згорткові нейронні мережі, які на сьогоднішній день стали невід’ємною частиною великої кількості розроблених систем підтримки прийняття рішень вже потужно використовуються в технологіях із залученням методів комп’ютерного зору. В медицині вони знаходять застосування у розпізнаванні та класифікації медичних зображень, діагностиці захворювань та виявленні патологій.

Аналіз клітин крові, зокрема білих клітин або лейкоцитів, є одним з важливих етапів діагностики різних захворювань та стану імунної системи. Автоматизована система розпізнавання та класифікації лейкоцитів на знімках крові має велике значення для покращення швидкості, точності та ефективності діагностичних процедур.

Метою роботи є розробка та дослідження алгоритмів розпізнавання білих клітин крові на знімках з мікроскопа, а також їх класифікація з використанням глибоких згорткових нейронних мереж. В процесі виконання задачі буде використаний великий набір даних зі знімками крові, що містить 12500 зразків, в кожному з яких присутні декілька клітин крові, але лише одна з них є лейкоцитом, який потрібно розпізнати.

У роботі передбачено використання двох нейронних мереж. Перша - більш проста згорткова мережа, яка є повністю моєю ідеєю на основі методу спроб та помилок, а друга - більш складна та глибока мережа, вона на основі архітектури DenseNet201. Більш складна мережа, хоч і вимагатиме більш тривалого навчання, передбачає досягнення кращих результатів.

Дослідження включає такі основні задачі.

1. Порівняння різних підходів до використання нейронних мереж для розпізнавання білих клітин крові.
2. Вивчення впливу гіперпараметрів на ефективність розпізнавання та класифікації мережами.
3. Аналіз точності моделей для визначення оптимального рішення.
4. Побудова матриці помилок (confusion matrix) для оцінки результатів і визначення сильних та слабких сторін моделей.
5. Тестування розроблених моделей на окремо обраних знімках клітин для оцінки їх ефективності та потенційного застосування у медичних установах.

Всі задачі спрямовані на розробку ефективної системи розпізнавання білих клітин крові та класифікації їх типів з використанням глибоких згорткових нейронних мереж. Отримані результати можуть бути корисними для покращення процесу діагностики захворювань та забезпечення швидкого та точного аналізу клітин крові.

1 АНАЛІЗ ОБ’ЄКТУ ДОСЛІДЖЕННЯ

1.1. Загальні відомості про клітини крові

Функція крові може бути узагальнена в наступних аспектах:

1. Транспорт кисню та поживних речовин
2. Видалення відходів та вуглекислого газу
3. Захист від інфекцій
4. Регуляція та підтримка гомеостазу
5. Участь у згортанні крові

Клітини крові - це невід'ємна частина організму людини, вони виконують різноманітні функції, необхідні для підтримки життєдіяльності організму. Кров складається з трьох основних типів клітин: еритроцитів, лейкоцитів і тромбоцитів, кожен з них має свою унікальну структуру та функцію, яка впливає на загальний стан організму. На рисунку 1.1 зображені білі клітини крови - лейкоцити.

Еритроцити, лейкоцити і тромбоцити переміщуються в плазмі крові. Це безбарвна рідинна складова крові, яка складає близько 55% всього об'єму. Вона має водяну основу та містить розчинені речовини, такі як білки, глюкоза, ліпіди, гормони, електроліти, антитіла, зайві речовини та інші речовини, необхідні для нормального функціонування організму [1].

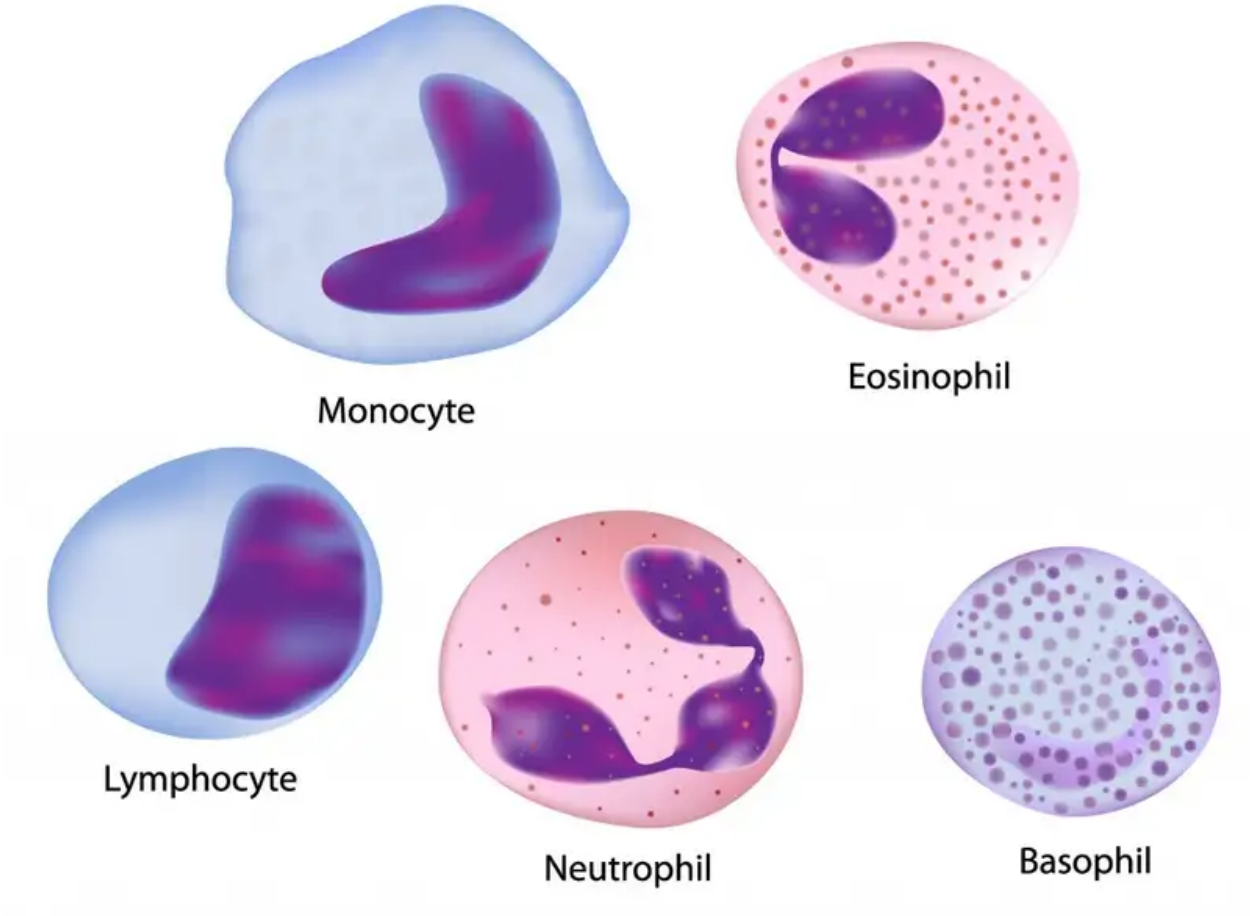


Рисунок 1.1 - Білі клітини крові

Еритроцити (червоні клітини крові) - це найпоширеніший тип клітин у крові. Вони відповідають за транспорт кисню до тканин та видалення вуглекислого газу. Еритроцити мають характерну біконкавну форму, що забезпечує їм більшу поверхню для газообміну. Основною складовою частиною еритроцитів є гемоглобін - залізовмісний білок, який здатний зв'язувати кисень та вуглекислий газ.

Тромбоцити, або кровопластинки, відповідають за забезпечення процесу згортання крові. Вони мають безформну структуру та велику кількість реберець. При ушкодженні судин тромбоцити активуються, утворюють згорток і зупиняють кровотечу. Вони також відіграють роль у загоєнні ран та ремоделюванні тканин.

Лейкоцити (білі клітини крові) представляють собою різноманітну групу клітин, які грають важливу роль у захисній системі організму. Їх головна функція полягає у боротьбі з інфекціями та запаленнями. Лейкоцити поділяються на кілька підтипів, таких як нейтрофіли, лімфоцити, моноцити, еозинофіли та базофіли, кожен з яких виконує свою специфічну роль у імунній відповіді.

Оскільки дослідження стосується безпосередньо лейкоцитів, вони цікавлять нас більше інших типів клітин. Розглянемо підтипи лейкоцитів, розпізнавання яких є головною ціллю даної роботи:

1. Нейтрофіли

Нейтрофіли є найпоширенішим типом лейкоцитів, становлячи близько 60-70% всіх лейкоцитів. Вони мають важливу роль у боротьбі з бактеріями та іншими мікроорганізмами. Нейтрофіли фагоцитують (поглинають та руйнують) бактерії та інші шкідливі речовини, що потрапляють в організм. Вони також виробляють речовини, які притягують інші лейкоцити до місця запалення.

2. Лімфоцити

Лімфоцити є ключовими клітинами імунної системи та грають роль у специфічній імунітеті. Вони поділяються на кілька підтипів, включаючи Т-лімфоцити, В-лімфоцити та природні вбивці. Т-лімфоцити грають роль у розпізнаванні та вбивстві інфікованих клітин та клітин пухлин, В-лімфоцити виробляють антитіла для боротьби з інфекціями, а природні вбивці здатні вбивати інфіковані клітини без попереднього стимулювання.

3. Моноцити

Моноцити є великими клітинами, які грають важливу роль у фагоцитозі та презентації антигенів. Вони можуть перетворюватись у макрофаги, коли

потрапляють у тканини. Макрофаги здатні фагоцитувати бактерії, віруси та мертві клітини, а також виробляти речовини, що активують інші клітини імунної системи.

4. Еозинофіли

Еозинофіли відіграють важливу роль у боротьбі з паразитарними інфекціями та алергічними реакціями. Вони продукують речовини, які знижують запалення та захищають організм від паразитів.

5. Базофіли

Базофіли виконують роль у запальних реакціях та алергічних реакціях. Вони виробляють речовини, такі як гістамін, які сприяють розширенню судин та залповому запаленню.

Клітини крові виконують різноманітні та важливі функції в організмі. Еритроцити транспортують кисень та вуглекислий газ, лейкоцити забезпечують імунну відповідь та захист від інфекцій, а тромбоцити забезпечують згортання крові та зупинку кровотечі. Розуміння загальних відомостей про клітини крові є важливим для вивчення їх функцій та ролі в підтримці нашого здоров'я [2].

1.2. Значення розпізнавання лейкоцитів у медичній діагностиці

Застосування мікроскопії та розпізнавання білих клітин крові в медичній діагностиці має довгу історію. Уже в XIX столітті вчені почали використовувати мікроскоп для дослідження крові та розпізнавання

лейкоцитів. Ініціатором таких досліджень був німецький лікар Рудольф Вірхов, який встановив, що кров складається з різних видів клітин, у тому числі й лейкоцитів. Завдяки його роботі ми отримали перші знання про структуру та функції лейкоцитів, а мікроскопічний аналіз крові став невід'ємною складовою медичної практики.

Сучасні технології в області медичного обладнання та автоматизованого аналізу зображень значно полегшують розпізнавання лейкоцитів на знімках мікроскопа. Зазвичай цю процедуру виконує кваліфікований лаборант або автоматизована система аналізу крові. Результати розпізнавання можуть бути отримані швидко та точно, що допомагає спеціалісту зробити відповідні клінічні висновки та встановити правильний діагноз.

У медичній діагностиці розпізнавання лейкоцитів є невід’ємною складовою процесу, що дозволяє лікарям отримати важливу інформацію про стан імунної системи та виявити різні захворювання. Як зазначалось у пункті 1.1, лейкоцити, або білі клітини крові - це клітини, які виконують функції імунітету та захисту організму від інфекцій та інших шкідливих факторів. Розпізнавання лейкоцитів на знімках мікроскопа дозволяє встановити діагноз, оцінити ступінь запалення та виявити інші патологічні зміни, що сприяє вчасному лікуванню та поліпшенню результатів терапії.

Розпізнавання клітин крові на знімках мікроскопа має велике значення у встановленні діагнозу різних захворювань та моніторингу стану пацієнтів. Наприклад, зміни у типах та кількості лейкоцитів можуть свідчити про інфекційні захворювання, запальні процеси, аутоімунні розлади, алергічні реакції та інші порушення імунної системи. Наявність аномалій у лейкоцитарному складі крові може служити ознакою певного захворювання та допомагати лікареві прийняти правильне рішення щодо подальшого лікування.

Процес розпізнавання лейкоцитів на знімках мікроскопа використовується у багатьох галузях медицини, де необхідна оцінка стану імунної системи та виявлення патологічних змін. Одним з таких прикладів є гематологія, галузь медицини, що вивчає кров'яну систему та займається діагностикою та лікуванням розладів крові. Розпізнавання лейкоцитів на мікроскопічних знімках допомагає ідентифікувати різні типи лейкоцитів, такі як нейтрофіли, еозинофіли, базофіли, лімфоцити та моноцити, і виявляти зміни їх кількості та співвідношення, що допомагає при діагностиці кровових розладів.

Також розпізнавання лейкоцитів на знімках мікроскопа може бути застосоване у клінічній мікробіології для виявлення та класифікації бактеріальних або грибкових інфекцій. Зміни в лейкоцитарному складі крові, спостережені під мікроскопом, можуть свідчити про наявність інфекційного процесу та навіть вказувати на конкретний патоген.

Значення розпізнавання лейкоцитів у медичній діагностиці надзвичайно важливе для встановлення діагнозу, оцінки ступеня запалення та виявлення патологічних змін у крові. Застосування мікроскопії та автоматизованих систем аналізу зображень дозволяє отримувати об'єктивну інформацію про стан імунної системи та виявляти різні захворювання. Історичні дослідження та поступовий розвиток технологій допомагають удосконалювати цей процес, забезпечуючи швидкість, точність та надійність розпізнавання лейкоцитів на мікроскопічних знімках [3].

1.3. Огляд існуючих методів розпізнавання лейкоцитів

Протягом останніх десятиліть було розроблено та удосконалено різні методи розпізнавання, які дозволяють швидко та ефективно аналізувати клітини крові та встановлювати їх типи та характеристики. У цьому розділі ми розглянемо існуючі методи розпізнавання клітин крові на мікроскопах, їх розвиток та застосування.

Одним з перших методів розпізнавання клітин крові було ручне визначення типів клітин лаборантом під мікроскопом. Цей метод вимагав високої кваліфікації та досвіду лаборанта, а також забирав багато часу. Лаборанти використовували морфологічні ознаки клітин, такі як форма, розмір, ядерна структура та інші характеристики, для класифікації клітин у різні типи. Цей метод був попередником автоматизованих систем розпізнавання.

З розвитком комп'ютерної технології і штучного інтелекту з'явилися автоматизовані системи розпізнавання клітин крові на мікроскопах. Ці системи базуються на обробці цифрових зображень клітин за допомогою спеціальних алгоритмів та програмного забезпечення. Вони дозволяють автоматично класифікувати клітини на основі їх морфологічних характеристик і встановлювати типи клітин швидко та точно.

Одним з популярних методів є метод машинного навчання, в якому комп'ютерні алгоритми навчаються розпізнавати клітини на основі тренувальних даних. Ці алгоритми використовуються для визначення паттернів та закономірностей в зображеннях клітин і допомагають автоматично класифікувати їх у відповідні категорії.

Метод комп’ютерного зору полягає у застосуванні комп’ютерних алгоритмів, що імітують процеси сприйняття та розуміння візуальної

інформації. Алгоритми аналізують зображення клітин, виявляють їх особливості та розпізнають типи клітин на основі навчальних даних. Комп'ютерне зорове сприйняття дозволяє автоматично виконувати розпізнавання клітин швидше та точніше, уникати помилок, пов'язаних з людським фактором, та забезпечувати об'єктивні результати.

Підготовка зразка крові до знімків мікроскопа є важливим етапом, щоб отримати якісні зображення клітин для подальшого аналізу. Процес підготовки включає збір зразка крові, його обробку та нанесення на предметний носій, такий як ковзанка. Розглянемо кожен крок детальніше.

1. Збір зразка крові:

Збір зразка крові може проводитись за допомогою взяття крові з пальця за допомогою ланцета або з вени за допомогою шприца або вакуумної системи. Для отримання якісного зразка важливо дотримуватись стерильності та правильної техніки забору крові.

2. Обробка зразка:

Отриманий зразок крові має бути оброблений перед нанесенням на ковзанку. Зазвичай, це включає додавання антикоагулянту до зразка, щоб уникнути згортання крові та зберегти її структуру. Популярним антикоагулянтом є етилендіамінтетраоцтова кислота (EDTA), яка запобігає згортанню крові, зберігаючи клітини у стані, максимально наближеному до їх природного стану.

3. Нанесення на ковзанку:

Зразок крові після обробки наносять на предметний носій, яким може бути ковзанка або інший скляний або пластиковий носій. Це робиться шляхом струменя зразка, який розподіляють рівномірно по поверхні ковзанки. Після нанесення зразка на ковзанку його можна зафіксувати,

наприклад, за допомогою термінального кольору або фіксуючого розчину, що допомагає зберегти клітини на місці та попереджає їх руйнування під час подальшої обробки.

4. Фарбування:

Щоб покращити контраст та видимість клітин на зразку, використовують фарбування. Це допомагає виділити різні структури клітин та покращити їх розпізнавання під час мікроскопічного аналізу. Різні методи фарбування можуть використовувати різні барвники, такі як Жіровий жовт, Метиленовий синій, Шифера та інші. Кожен барвник має свою специфічну дію на клітини та допомагає виділити певні структури.

5. Кріплення кришкою:

Після фарбування на ковзанку накривають тонкою скляною або пластиковою кришкою, щоб захистити зразок від забруднення та зберегти його структуру. Кришка надійно кріпиться до ковзанки, забезпечуючи плоску поверхню для мікроскопічного аналізу.

Після підготовки зразка крові до знімків мікроскопа, його можна досліджувати за допомогою світлового або електронного мікроскопа. Мікроскопічний аналіз зображень дозволяє визначити типи клітин, їх кількість, стан та інші характеристики, які є важливими для медичної діагностики та досліджень.

Протягом останніх років були впроваджені різні технологічні вдосконалення для поліпшення методів розпізнавання клітин крові на мікроскопах. З’явилися нові алгоритми та моделі машинного навчання, які забезпечують більш точне та ефективне розпізнавання. Також використовуються технології штучного інтелекту, які дозволяють системам самостійно вдосконалюватись та покращувати результати розпізнавання з часом.

Існуючі методи розпізнавання клітин крові на мікроскопах, такі як ручне розпізнавання, автоматизовані системи розпізнавання, комп'ютерне зорове сприйняття та інші, відіграють важливу роль у медичній діагностиці та дослідженнях. Завдяки розвитку технологій, ці методи стають все більш точними, швидкими та ефективними. Вони знайшли широке застосування у багатьох галузях медицини та наукових досліджень, сприяючи покращенню діагностики, лікування та розумінню хвороб. Продовження досліджень у цій галузі та розвиток нових методів розпізнавання є важливими завданнями для медичної науки та практики [4].

1.4. Порівняння підходів до класифікації лейкоцитів

Існує декілька основних підходів до класифікації білих клітин крові: морфологічний підхід, імунологічний підхід та генетичний підхід. Розглянемо кожен підхід детальніше:

1. Морфологічний підхід

Морфологічний підхід до класифікації лейкоцитів базується на їх зовнішньому вигляді та морфологічних ознаках, таких як форма, розмір, ядерний і цитоплазматичний структури. За допомогою морфологічного підходу лейкоцити можуть бути класифіковані на основі їх морфологічних відмінностей, наприклад, лейкоцити можуть бути поділені на нейтрофіли, еозинофіли, базофіли, лімфоцити та моноцити. Цей підхід широко використовується у клінічній практиці, оскільки дозволяє швидко ідентифікувати та класифікувати лейкоцити за допомогою стандартних барвників та мікроскопічного аналізу.

2. Імунологічний підхід

Імунологічний підхід використовує антитіла та імуномаркери для класифікації лейкоцитів на основі їх специфічних біологічних властивостей. Цей підхід може використовувати флуоресцентну або імуногістохімічну маркування для виявлення конкретних маркерів на поверхні лейкоцитів, що дозволяє ідентифікувати та класифікувати їх на основі виявлення певних антигенів. Цей підхід дозволяє виявити підтипи лейкоцитів та визначити їх функціональні характеристики, що може бути корисним у діагностиці захворювань та вивченні імунного відгуку.

3. Генетичний підхід

Генетичний підхід до класифікації лейкоцитів базується на виявленні та аналізі генетичних відмінностей між різними типами лейкоцитів. За допомогою сучасних молекулярних технологій, таких як полімеразна ланцюгова реакція (ПЛР) або секвенування геному, можна виявити та аналізувати генетичні маркери, що розрізняють різні типи лейкоцитів. Цей підхід дозволяє з'ясувати генетичні особливості лейкоцитів, їх спадкові властивості та вплив генетичних змін на функції клітин. Він є особливо важливим у генетичних дослідженнях та вивченні спадкових захворювань.

Кожен з цих підходів має свої переваги та обмеження, і їх використання залежить від конкретних дослідницьких цілей та доступних технологій. Для ефективного дослідження необхідно аналізувати різні підходи до класифікації лейкоцитів, їх застосування у медичній діагностиці та дослідженнях, а також треба розуміти переваги та обмеження кожного підходу.

В таблиці 1.1 порівняємо переваги та недоліки існуючих підходів:

Таблиця 1.1. Підходи до задачі розпізнавання лейкоцитів

Підхід	**Переваги**	**Недоліки**
Морфологічний	Широкодоступність та низькі витрати	Суб’єктивність інтерпретації морфологічних ознак
	Швидкість визначення типу лейкоцитів	Обмежена специфічність для деяких підтипів
	Можливість використання стандартних методів	Не виявляє деякі функціональні властивості
Імунологічний	Висока специфічність і точність	Вимагає наявності специфічних антитіл
	Виявлення підтипів та функціональних ознак	Витрати на підготовку та маркування антитіл
	Можливість використання для різних зразків	Складність аналізу при змішаних популяціях

Продовження таблиці 1.1

Генетичний	Детальне вивчення генетичних властивостей	Вимагає складних молекулярних технологій
	Виявлення генетичних варіацій	Високі витрати на обладнання та аналізи
	Спадковий аналіз та вивчення спадкових хвороб	Залежність від якості та чистоти ДНК-зразків

Комбінація різних підходів може бути корисною для отримання більш повної та точної класифікації лейкоцитів. Наприклад, поєднання морфологічного та імунологічного підходів може забезпечити широкий спектр ідентифікації підтипів лейкоцитів разом з їх функціональними характеристиками. Вибір підходу до класифікації лейкоцитів залежить від конкретних дослідницьких цілей, доступних технологій та контексту дослідження. Подальший розвиток технологій та вивчення клітинного складу крові допоможуть удосконалити методи класифікації та сприятимуть покращенню точності та ефективності медичної діагностики [5].

1.5. Проблеми традиційних методів класифікації лейкоцитів

Традиційні методи розпізнавання, які базуються на морфологічних ознаках клітин, довгий час залишалися основою для лабораторної класифікації. Однак, ці методи мають свої обмеження та проблеми, які роблять актуальною розробку нових підходів, зокрема використання нейронних мереж.

Одна з основних проблем традиційних методів розпізнавання полягає в їх суб'єктивності та низькій відтворюваності результатів. Класифікація клітин залежить від спостерігача, його досвіду та суб'єктивних оцінок, що може призводити до неточностей та непередбачуваних результатів. Більше того, різні лаборанти можуть розпізнавати одні й ті ж клітини по-різному, що погіршує консистентність та достовірність аналізу.

Іншою проблемою є обмежена специфічність та розпізнавання деяких підтипів клітин. Традиційні методи можуть бути обмежені в розпізнаванні клітин зі схожими морфологічними ознаками або виявленні рідкісних підтипів клітин. Це може призвести до пропуску важливих клітин або неправильної класифікації, що може мати наслідки для діагностики та вибору оптимального лікування.

Також традиційні методи не дозволяють виявити функціональні властивості клітин. Інформація про функціональні характеристики клітин, такі як здатність до фагоцитозу, продукція цитокінів або вироблення антитіл, може бути важливою для точної діагностики та вибору оптимального лікування. Традиційні методи не забезпечують можливість оцінити ці функціональні аспекти клітин.

Одним із шляхів вирішення цих проблем є використання нейронних мереж для розпізнавання білих клітин крові. Нейронні мережі можуть

автоматично виявляти складні залежності та патерни в даних, враховуючи низький рівень суб'єктивності та здатність до самонавчання. Вони можуть здатися змінність клітин та класифікувати їх з високою точністю. Крім того, нейронні мережі можуть бути навчені виявляти функціональні характеристики клітин, що відкриває нові можливості для діагностики та моніторингу захворювань.

Таким чином, розробка методу розпізнавання білих клітин крові за допомогою нейронних мереж має велику важливість у сучасній медичній діагностиці. Вона дозволяє уникнути суб'єктивності та недостатньої специфічності традиційних методів, а також забезпечує можливість виявлення функціональних характеристик клітин. Розробка нейронних мереж є кроком у напрямку поліпшення точності та надійності класифікації, що сприяє ефективній діагностиці та лікуванню пацієнтів [6].

1.6. Аналіз набору даних та постановка задачі

Набір даних, що використовується для розробки моделі розпізнавання білих клітин крові (папка ‘dataset_augmented’), складається з 12500 зображень крові (формат JPEG), що були піддані аугментації з супровідними мітками, що вказують тип клітини. Набір розділений на 4 папки, відповідно до типу клітини, і містить приблизно 3000 зображень для кожного з 4 різних типів клітин: еозинофіли, лімфоцити, моноцити та нейтрофіли. Датасет завантажений з платформи Kaggle.

Додатково до цього, набір даних включає оригінальний набір BCCD із репозиторія GitHub з 410 зображень крові до аугментації (папка ‘dataset’), приклад зображений на рисунку 1.2. Набір даних містить файл, який описує

обмежувальні рамки для кожної клітини в кожному з цих 410 зображень (annotations.csv). Також до оригінального набору долучений файл з Kaggle з мітками WBC vs WBC (labels.csv). Кількість аугментованих зображень для кожного з 4 підтипів складає приблизно 3000, порівняно з 88, 33, 21 і 207 зображеннями кожного підтипу у папці ‘dataset’.

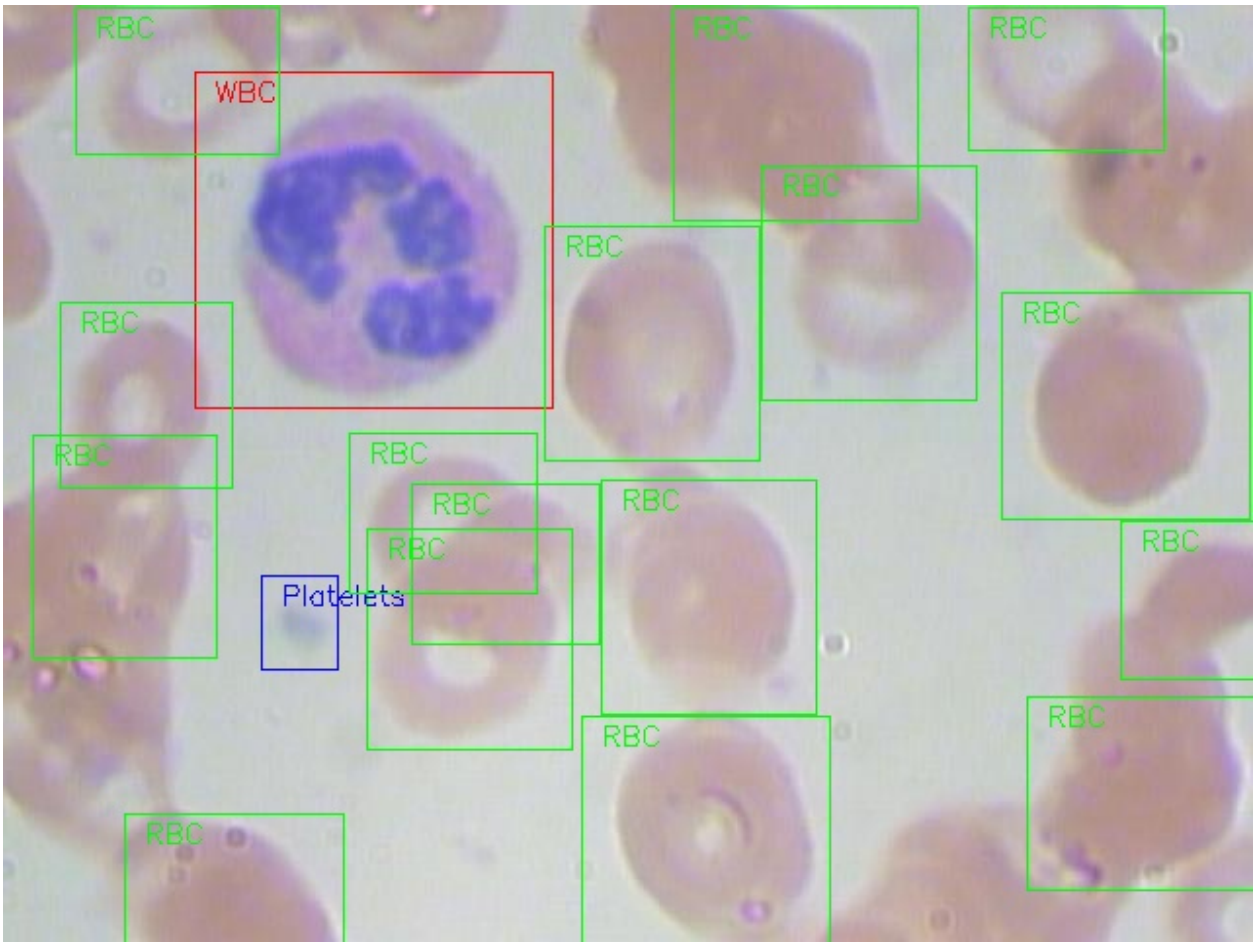


Рисунок 1.2 - Зображення лейкоциту з набору даних BCCD

Задача даної роботи полягає у розробці моделі машинного навчання, зокрема нейронної мережі, яка зможе розпізнавати типи білих клітин крові на знімках мікроскопа з високою точністю та надійністю. Використовуючи набір даних, описаний вище, модель буде навчена виявляти та класифікувати еозинофіли, лімфоцити, моноцити та нейтрофіли на зображеннях, забезпечуючи швидку та автоматизовану діагностику кров’яних захворювань. Це значно покращить ефективність та надійність процесу медичної

діагностики та допоможе лікарям приймати більш точні та обґрунтовані рішення щодо лікування пацієнтів.

1.7. Висновки до розділу 1

У розділі 1 проведено аналіз об’єкта дослідження, яким є розпізнавання лейкоцитів у медичній діагностиці. Розглянуті загальні відомості про клітини крові, важливість їх розпізнавання у медичній практиці та огляд існуючих методів розпізнавання.

Виявлено, що розпізнавання лейкоцитів має велике значення для діагностики різних захворювань та моніторингу стану пацієнтів. Різні типи лейкоцитів можуть свідчити про конкретні захворювання, тому точна класифікація є важливим завданням. Досліджено різні підходи до класифікації лейкоцитів, такі як методи на основі ручно створених ознак, методи машинного навчання та нейронні мережі. Порівняння різних підходів показало їх переваги та недоліки, зокрема щодо точності класифікації, часу обробки та складності реалізації. Традиційні методи класифікації лейкоцитів також мають свої обмеження, зокрема пов'язані з підготовкою зразків та індивідуальною інтерпретацією експертів. Ці проблеми можуть впливати на точність та об'єктивність діагностики. У зв'язку з цим, розробка методу розпізнавання лейкоцитів за допомогою нейронних мереж є актуальною та перспективною. Нейронні мережі дозволяють автоматизувати процес класифікації, використовуючи великі обсяги даних та забезпечуючи високу точність результатів. Використання таких методів може покращити якість діагностики та сприяти розвитку медичних технологій.

Отже, розробка методу розпізнавання лейкоцитів з використанням

нейронних мереж є важливим напрямом досліджень, який може привести до покращення якості та ефективності медичної діагностики, а також забезпечити нові можливості для розвитку медичних технологій і підвищення якості життя пацієнтів.

2 ПІДХОДИ ДО РОЗВ’ЯЗАННЯ ЗАДАЧІ

2.1. Прості нейронні мережі

Нейронні мережі є важливими моделями машинного навчання, які забезпечують основу для розвитку більш складних і потужних архітектур. Їх виникнення пов'язане з багаторічною історією досліджень у галузі штучних нейронних мереж та обчислювального інтелекту. Далі будуть розглядатись походження простих нейронних мереж, їх початкові задачі та поточні функції, а також принципи їхньої роботи.

Прості нейронні мережі були розвинуті на основі вивчення біологічного нейрона та спроби створити комп'ютерні моделі, які відтворюють його функціональні властивості. Перші моделі були розроблені ще у 1940-х та 1950-х роках, зокрема нейронна мережа Мак-Каллока-Піттса, яка є одним з найбільш відомих прикладів простої нейронної мережі. Ці ранні моделі надали основу для подальших досліджень і розвитку штучних нейронних мереж.

Одним з найпростіших видів штучних нейронних мереж є персептрон. Персептрон був запропонований Френком Розенблаттом у 1957 році і був названий за аналогією з біологічним нейроном, який відповідає за передачу сигналів у нервовій системі. В основі персептрона лежить модель бінарного класифікатора, який приймає вхідні сигнали, обчислює їх ваговану суму та застосовує нелінійну функцію активації для вироблення вихідного сигналу.

Персептрон складається з одного або кількох штучних нейронів, відомих як перцептронні нейрони, принцип його дії зображений на рисунку 2.1. Кожен нейрон отримує вхідні сигнали, які мають вагу, і обчислює взважену суму цих сигналів. Потім до результату додається зміщення (bias), і

отримане значення проходить через нелінійну функцію активації, наприклад, функцію сигмоїди, що на рисунку 2.2, або ступінчасту функцію [7].

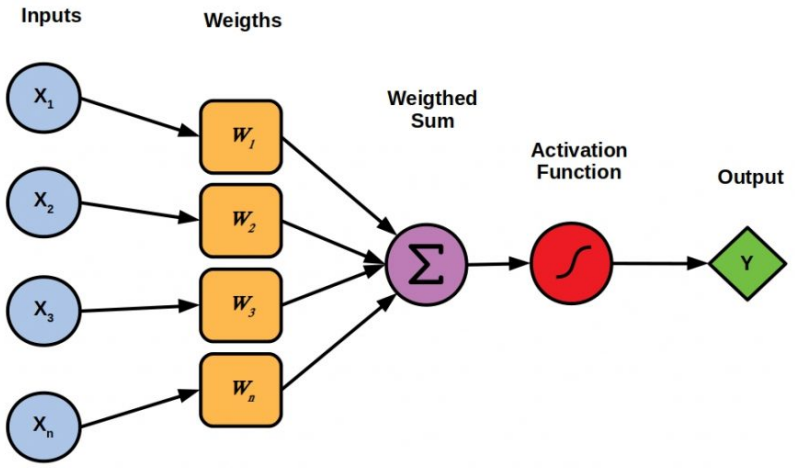


Рисунок 2.1 - Просто нейронна мережа

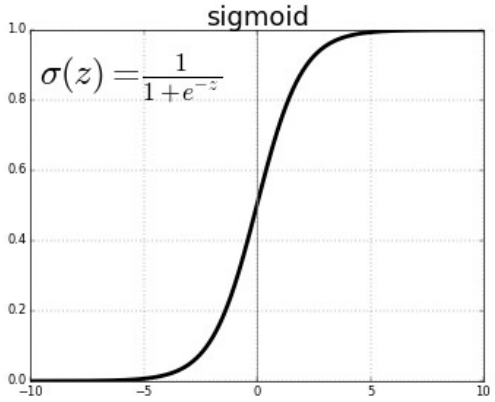


Рисунок 2.2 - Функція активації нейромережі

Основна ідея персептрона полягає в тому, що він може навчитися встановлювати оптимальні ваги для розпізнавання певних паттернів у вхідних даних. Цей процес називається навчанням персептрона і зазвичай виконується за допомогою алгоритму зворотного поширення помилки (backpropagation), який коригує ваги нейронів залежно від різниці між

очікуваним вихідним сигналом і фактичним вихідним сигналом. Крива навчання нейронної мережі зображена на рисунку 2.3 [8].

Рисунок 2.3 - Крива навчання нейронної мережі

Персептрони можуть бути використані для вирішення задач класифікації, де необхідно розділити вхідні дані на декілька категорій. Наприклад, в задачі розпізнавання рукописних цифр персептрон може бути навчений розрізняти цифри від 0 до 9.

Однак, важливо зазначити, що прості персептрони мають обмежену потужність і можуть ефективно працювати лише з лінійно-роздільними даними. Якщо дані не можна розділити лінійною границею, персептрон не зможе досягти точності класифікації. Цю проблему було вирішено з'явленням більш складних архітектур нейронних мереж, таких як глибокі нейронні мережі та згорткові нейронні мережі, які дозволяють ефективно розпізнавати складні патерни у вхідних даних.

Сучасні прості нейронні мережі займають важливе місце в різних областях, включаючи машинне навчання, обробку природних мов, комп'ютерне зорове сприйняття та інше. Вони використовуються для

вирішення задач, таких як класифікація текстів, розпізнавання мови, розпізнавання голосу, детекція об'єктів на зображеннях тощо. Їх простота дозволяє ефективно виконувати ці задачі при обмежених обчислювальних ресурсах та швидко навчати моделі на великих наборах даних.

Прості нейронні мережі, хоч і є базовими моделями, все ще мають свою важливість у сучасному машинному навчанні. Вони надають простий і зрозумілий спосіб моделювання та розв'язання деяких задач, а також виступають важливим етапом у розвитку більш складних і потужних нейронних мереж, таких як глибокі нейронні мережі та згорткові нейронні мережі [9].

2.2. Комп’ютерний зір

Комп'ютерний зір (Computer Vision) - це галузь штучного інтелекту, яка вивчає, як комп’ютерні системи можуть “бачити” і розуміти зображення та відео. Ця галузь розвивалась протягом останніх десятиліть, з переходом від традиційних методів обробки зображень до використання глибоких нейронних мереж.

Застосування нейронних мереж у комп'ютерному зорі відкрило нові можливості у вирішенні складних задач обробки зображень. Вкажемо основні завдання, які вирішуються за допомогою машинного навчання і комп'ютерного зору.

1. Класифікація зображень

Моделі нейронних мереж можуть бути навчені класифікувати зображення на різні категорії. Наприклад, розпізнавання об'єктів на зображенні або класифікація рукописних символів.

2. Семантична сегментація

Це задача, в якій нейронна мережа визначає пікселі на зображенні, які відповідають певним об'єктам або класам. Це дозволяє розпізнавати та виділяти об'єкти на зображенні.

3. Виявлення об’єктів

Машинне навчання може допомогти виявити об'єкти на зображеннях, такі як обличчя людей, автомобілі або інші об'єкти із заданого набору.

4. Відстеження об’єктів

Це задача, в якій система здатна відстежувати рух об'єктів на відео. Вона може використовуватися, наприклад, для відстеження руху автомобіля на дорозі або виявлення руху людей у відеоспостереженні.

5. Розпізнавання облич

Нейронні мережі можуть бути навчені розпізнавати обличчя людей на зображеннях або відео. Це може бути корисною технологією для автоматизованого відстеження або ідентифікації осіб [10].

Ці задачі комп’ютерного зору знаходять застосування в різних сферах життя, включаючи медицину, автомобільну промисловість, робототехніку, безпеку, відеоспостереження, розпізнавання рукописного тексту та багато інших. Наприклад, в медицині нейронні мережі можуть використовуватись

для автоматизованого аналізу медичних зображень, виявлення патологій та підтримки діагнозування.

Розвиток комп’ютерного зору продовжується, із з’явленням нових алгоритмів та архітектур нейронних мереж. Із збільшенням обчислювальної потужності та доступності даних, застосування комп'ютерного зору стає все більш поширеним у реальному світі. Це допомагає вирішувати складні завдання в автоматизації, робототехніці, медицині, безпеці та інших галузях, покращуючи продуктивність, точність і ефективність багатьох процесів та систем [11].

2.3. Прості згорткові мережі

Згорткові нейронні мережі (Convolutional Neural Networks, CNNs) є потужними моделями машинного навчання, які засновані на біологічній структурі зорової кори. Вони з’явилися у 1980-х роках, коли вчені почали вивчати, як можна використовувати згорткові шари для ефективного виявлення рис об'єктів на зображеннях.

Згорткові шари виконують операцію згортки між вхідним зображенням і набором фільтрів, що допомагають визначити локальні особливості та шаблони на зображенні. Вони використовуються для автоматичного вилучення рис з даних, зокрема зображень, з урахуванням локальних залежностей.

Згорткові шари складаються з набору фільтрів, які застосовуються до вхідного зображення шляхом згортки. Ці фільтри мають ваги, які вивчаються під час процесу навчання. Після застосування згортки отримується карта

ознак, яка виокремлює локальні шаблони та особливості на зображенні. Пулінгові шари та їх вплив на мережу Пулінгові шари використовуються для зменшення розмірності отриманої карти ознак шляхом об'єднання значень у певних областях. Це дозволяє зменшити кількість параметрів мережі та зробити її більш стійкою до змін масштабу та розташування об'єктів на зображенні [12].

В одношарових згорткових мережах (Single Layer CNN) Одношарові згорткові мережі складаються лише з одного згорткового шару, який виконує функцію вилучення рис з вхідного зображення. Цей тип мережі використовується для простіших задач, де потрібно визначити наявність певного об'єкта на зображенні. В багатошарових згорткових мережах (Multi-Layer CNN) Багатошарові згорткові мережі включають кілька згорткових шарів та шарів пулінгу, що дозволяє виконувати складніші завдання, такі як класифікація зображень об'єктів. Вони можуть мати декілька шарів згортки та пулінгу, після чого зазвичай додаються повнозв'язні шари для рішення фінальної задачі.

Застосування згорткових нейронних мереж:

1. Розпізнавання об'єктів

Згорткові нейронні мережі використовуються для розпізнавання об'єктів на зображеннях. Вони можуть визначати присутність певних об'єктів і навіть класифікувати їх на основі навчальних даних.

2. Класифікація зображень

Згорткові мережі застосовуються для класифікації зображень на певні категорії. Вони можуть виконувати розпізнавання образів, класифікацію предметів або визначення певних характеристик на зображенні.

3. Семантична сегментація

Згорткові мережі також можуть використовуватись для семантичної сегментації, де кожен піксель зображення класифікується на певну категорію. Це дозволяє розрізняти об'єкти на зображенні та розуміти їх контекст.

2.4. Глибокі згорткові нейронні мережі

Глибокі згорткові мережі є розширенням згорткових мереж, що використовуються в комп'ютерному зорі. Вони складаються з багатьох шарів, включаючи згорткові шари, шари пулінгу та повнозв'язані шари. У цьому розділі ми розглянемо особливості глибоких згорткових мереж, їх функції та застосування.

Глибокі згорткові мережі відрізняються від звичайних згорткових мереж глибшим структурним складом. Вони можуть мати десятки або навіть сотні шарів, що дозволяє їм виявляти складні залежності у вхідних даних. Глибші мережі здатні виконувати більш складні завдання, такі як розпізнавання об'єктів, сегментація зображень та генерація зображень [13].

Глибокі згорткові мережі мають широкий спектр застосувань у сфері комп’ютерного зору. Основні задачі, які вони вирішують переважно співпадають з задачами згорткових мереж, переліченими в пункті 2.3, вони включають:

1. Класифікація зображень

Глибокі згорткові мережі можуть ефективно класифікувати зображення на різні категорії. Вони можуть визначати наявність об'єктів та відрізняти їх

на основі навчальних даних. Прикладом таких мереж є ResNet, VGG та Inception [15].

2. Семантична сегментація

Ця задача полягає в класифікації кожного пікселя зображення на певну категорію. Глибокі згорткові мережі здатні виконувати семантичну сегментацію, дозволяючи точно визначити контур об'єктів на зображенні. У цій області широко використовуються мережі, такі як U-Net та FCN.

3. Розпізнавання об’єктів

Глибокі згорткові мережі можуть розпізнавати об'єкти на зображенні та визначати їх положення та клас. Це має велике значення в різних областях, таких як автономні автомобілі, відеоспостереження та робототехніка. Для цих задач використовуються мережі, такі як YOLO, SSD та Faster R-CNN [16].

DenseNet є однією з найефективніших архітектур глибоких згорткових мереж. Вона базується на концепції “з’єднання всіх” (dense connections), де кожен шар отримує вхідні дані з усіх попередніх шарів. Це сприяє ефективному обміну інформацією та забезпечує збільшення градієнту під час навчання. DenseNet має високу точність і густу зв’язність між шарами, що робить його популярним в задачах комп'ютерного зору [17].

Глибокі згорткові мережі використовуються в багатьох реальних застосуваннях. Вони застосовуються в автономних автомобілях для розпізнавання дорожних знаків та перешкод, в системах відеоспостереження для виявлення злочинців та в медичних системах для діагностики захворювань на основі зображень. Глибокі згорткові мережі дозволяють досягти високої точності та швидкості обробки зображень, що робить їх незамінними у багатьох сферах життя.

2.5. Проблеми машинного навчання

Машинне навчання є потужним інструментом, що дозволяє комп'ютерам вчитися і виконувати завдання без явного програмування. Проте, як будь-яка технологія, воно стикається з рядом проблем і викликів, які потребують уваги та вирішення. У цьому розділі ми розглянемо деякі з цих проблем, з якими стикаються в процесі розробки та застосування моделей машинного навчання.

1. Обмеження обсягу даних і перенавчання

Однією з ключових проблем, з якими стикаються глибокі згорткові нейронні мережі (ГЗНМ) в комп'ютерному зорі, є обмежений обсяг наявних даних. ГЗНМ вимагають великої кількості даних для ефективного навчання, особливо коли мова йде про складні задачі, такі як розпізнавання об'єктів або семантична сегментація. Брак достатніх даних може призвести до перенавчання моделі, коли вона добре працює на навчальних даних, але має погану універсальність на нових зображеннях. Розв'язанням цієї проблеми може бути збір додаткових даних або використання методів переднього навчання (pretraining) на великому наборі загального призначення.

2. Неоднорідність та варіативність даних

У сфері комп'ютерного зору зустрічаються значні варіації в зображеннях, такі як зміна освітлення, масштабу, повороту, відсутність деяких об'єктів тощо. Ця неоднорідність може ускладнювати завдання розпізнавання та класифікації. ГЗНМ, хоча і володіють великою потужністю, можуть бути чутливі до таких змін і недостатньо робастими до варіативності даних. Для розв'язання цього завдання можуть використовуватися методи

аугментації даних, такі як зміна яскравості, відображення, випадкові обрізки тощо. Такі методи допомагають збільшити різноманітність даних та покращити роботу моделі на різних умовах зображень.

3. Обробка великомасштабних зображень

Комп'ютерне зорою часто використовуються для обробки великомасштабних зображень, таких як медичні знімки, супутникові знімки або зображення відеоспостереження. Обробка цих зображень може бути вимогливою з точки зору обчислювальних ресурсів та пам'яті. Глибокі згорткові мережі зазвичай потребують значних обчислювальних потужностей і великої кількості пам'яті для навчання та застосування. Це може бути обмеженням при використанні на великих масштабах або на ресурсно обмежених пристроях. Одним зі способів розв'язання цієї проблеми є використання методів оптимізації моделей, таких як стиснення (compression) або квантизація (quantization), що дозволяють зменшити розмір моделі та вимоги до ресурсів.

4. Інтерпретовність та пояснюваність моделей

Інтерпретовність та пояснюваність моделей є ще однією важливою проблемою в застосуванні глибоких згорткових нейронних мереж. Вони зазвичай працюють як чорні скриньки, що ускладнює розуміння, чому модель приймає певні рішення. Це особливо важливо в медичній сфері, де необхідно мати високу ступінь довіри до рішень моделі. Розробка методів інтерпретовності та пояснюваності допомагає розкрити внутрішню логіку роботи моделей та зробити їх рішення зрозумілими та перевіреними експертами.

5. Потужність обчислень і енергоефективність

Ще однією проблемою глибоких згорткових нейронних мереж є потужність обчислень та енергоефективність. Глибокі моделі вимагають значних обчислювальних ресурсів для тренування та застосування. Це може бути проблемою в обмежених обчислювальних середовищах або на портативних пристроях з обмеженою енергією. Для вирішення цих проблем проводяться дослідження в напрямку створення більш ефективних алгоритмів, архітектур та обчислювальних платформ, що дозволяють знизити вимоги до обчислювальних ресурсів та споживання енергії.

Розуміння цих проблем допоможе визначити виклики і можливості використання глибоких згорткових нейронних мереж у сфері комп'ютерного зору [20].

2.6. Огляд літератури про нейронні мережі для розпізнавання об’єктів

Цей пункт міститиме огляд 2-х книг зі списку літератури, які дали мені сильну базу в предметній області нейронних мереж. Вони містять фундаментальні знання про глибоке навчання, згорткові мережі та їх застосування у комп'ютерному зорі. Спираючись саме на ці знання, я робив всі дослідження, які були проведені в процесі виконання роботи.

1. Python Data Science Handbook

"Python Data Science Handbook" написана Джейком ВандерПласом і була опублікована у 2016 році. Книга є незамінним посібником для всіх, хто бажає використовувати мову програмування Python у сфері науки про дані. ВандерПлас ретельно розглядає основні бібліотеки Python, такі як NumPy, Pandas, Matplotlib і Scikit-Learn, та демонструє, як їх використовувати для

ефективного аналізу даних. Книга охоплює широкий спектр тем, включаючи збір та очищення даних, візуалізацію, машинне навчання та багато іншого, надаючи читачеві цінні інструменти для вирішення різноманітних завдань у галузі науки про дані. "Python Data Science Handbook" стала для мене відмінним джерелом знань і вдосконалення навичок у сфері аналізу даних [21].

2. Programming Computer Vision with Python

"Programming Computer Vision with Python: Techniques and Libraries for Imaging and Retrieving Information" - це книга, написана Джаном Еріком Солемом і опублікована у 2012 році. Ця книга є практичним посібником з програмування комп'ютерного зору з використанням мови програмування Python. Солем детально описує різноманітні техніки та бібліотеки Python, які можна використовувати для обробки та аналізу зображень. Він пропонує читачам засоби для виконання різних завдань, таких як розпізнавання облич, виявлення об'єктів, відстеження руху та багато іншого. Книга також вміщує в себе приклади коду та детальні пояснення, як використовувати різні алгоритми та бібліотеки для досягнення бажаних результатів у галузі комп'ютерного зору [22].

Крім цих книг, я також хочу відмітити онлайн-курс "Machine Learning" від Andrew Ng. Цей курс надав мені глибоке розуміння основ машинного навчання, зосереджуючись на наданні теоретичних знань та практичних навичок для побудови моделей машинного навчання. В курсі я опрацював алгоритми, такі як лінійна регресія, логістична регресія, нейронні мережі, а також практичні аспекти, включаючи валідацію моделей, вирішення проблем підгонки та побудову рекомендаційних систем. Курс використовує Octave або MATLAB для програмування, але матеріали також доступні на Python.

Для практичного розуміння нейронних мереж типу DenseNet я ознайомився з книгою "Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems" авторства Орельена Жерона. Вона надає введення в основні концепції, інструменти та техніки, необхідні для побудови розумних систем. Книга охоплює використання Scikit-Learn, Keras та TensorFlow для створення моделей машинного навчання, зокрема лінійної та логістичної регресії, дерев рішень, нейронних мереж та багато інших. Жерон детально описує практичні кроки для побудови моделей, візуалізації даних, оцінки результатів та оптимізації моделей. Ця книга є цінним ресурсом для розуміння та використання машинного навчання для розв'язання реальних завдань у практиці [23].

Це не всі джерела, які давали мені знання в цій області, але вони є найголовнішими з них. Зазначені джерела надають широкий огляд та практичні приклади застосування нейронних мереж у сфері розпізнавання об'єктів. Вони є цінним ресурсом для дослідження та розвитку моєї наукової роботи.

2.7. Особливості алгоритму

Ціль алгоритму, який є найважливішою фігурою дослідження - це максимально точне, наскільки це можливо, розпізнавання білих клітин крові людини на знімках з мікроскопа. Знімки являють собою зображення клітин крові, зразок якої попередньо був зібраний, оброблений, та застосоване фарбування для виділення певних структур клітин. Знімок має містити білу клітину крові, яка є об’єктом розпізнавання.

Знімок формату “.jpg” передається програмі, яка в даному випадку є системою підтримки прийняття рішень. Алгоритм програми працює в декілька кроків, наведений в таблиці 2.1.

Таблиця 2.1. Алгоритм розпізнавання

	Крок	**Опис**
1	Завантаження	Попередньо зняті знімки зразків крові поміщаються в папку, файли якої зчитуються програмою при виконанні. Програма зберігає список файлів в датафреймі.
2	Попередня обробка	Файли датафрейму трансформуються в формат, з яким працює програма під час виконання. Далі відбувається безпосередня обробка, яка в свою чергу проходить в декілька дій: ● Додавання відступу до зображення, для кращого розпізнавання клітини на краю ● Порогове визначення зображення для отримання цільової комірки ● Ерозія відкриття, потім розширення ● Виявлення клітини крові ● Визначення координат області ● Заповнення контуру ● Видалення зовнішніх пікселів ● Витяг клітини ● Зміна розміру зображення ● Нормалізація значень пікселів Кінцевим результатом обробки є виділена біла клітина крові, з усіма її структурними ознаками. Частина знімка, деталі якої не стосуються задачі відкидаються та видаляються.

Продовження таблиці 2.1

3	Розпізнавання	Для розпізнавання попередньо оброблених лейкоцитів модель використовує глибоку нейромережу DenseNet201, навчену на великому наборі даних. Етапи розпізнавання: 1. Модель отримує на вхід цільові зображення 2. Вхідні дані (зображення) проходять через початковий згортковий шар, який виконує першу операцію згортки для виявлення базових ознак. 3. Після згорткового шару виконується густий блок (dense block). У густому блоку кожен шар отримує вхідні дані з усіх попередніх шарів, що створює "приєднаність" між шарами. Кожен шар густого блоку виконує послідовність згорток і активаційних функцій, наприклад, ReLU (Rectified Linear Unit). 4. Після густого блоку може бути шар пулінгу або середньої пулінгу, який зменшує розмір зображення і кількість каналів ознак. Це допомагає зменшити обчислювальну складність і кількість параметрів. 5. Процес густого блоку та пулінгу повторюється кілька разів, що дозволяє поступово виявляти все більш складніші ознаки зображення. 6. Після декількох повторень густого блоку та пулінгу, отримані ознаки проходять через повнозв'язний шар (fully connected layer). Цей шар виконує класифікацію зображень на основі отриманих ознак. 7. Останній шар використовує функцію активації Softmax, що перетворює вихідні значення на ймовірності для різних класів. Це дозволяє отримати прогнози для класифікації зображень. Прогнози трансформуються дані, з якими можна працювати далі.
4	Візуалізація	Програма зіставляє результати прогнозування з зображеннями до обробки та виводить зображення.

Дуже важливим процесом є побудова та навчання мережі.

1. Підготовка даних: Перед навчанням мережі здійснюється підготовка даних. Вона включає завантаження набору даних, розбиття їх на тренувальний, валідаційний та тестовий набори, а також можливу нормалізацію та аугментацію даних.

2. Будування моделі: Модель DenseNet-201 будується, використовуючи бібліотеку глибокого навчання. Побудова включає створення архітектури мережі, встановлення параметрів моделі та оптимізатора.
3. Визначення функції втрат: Для навчання мережі визначається функція втрат, яка вимірює різницю між прогнозованими значеннями моделі та справжніми мітками даних. Наприклад, для задачі класифікації зображень може використовуватись категоріальна перехресна ентропія.
4. Навчання моделі: За допомогою тренувального набору даних відбувається навчання моделі. Це включає передачу даних через мережу, обчислення втрати та коригування ваг шарів за допомогою алгоритму зворотнього поширення помилки. Процес навчання може вимагати багато епох, де кожна епоха представляє собою один прохід повних даних через мережу.
5. Оцінка та налаштування: Після навчання моделі оцінюється її продуктивність на валідаційному наборі даних. Це може включати обчислення метрик точності, втрат та інших показників валідації. Залежно від результатів, можуть вноситись зміни в параметри моделі або алгоритм навчання.
6. Тестування: Після завершення навчання та налаштування, модель тестується на тестовому наборі даних для оцінки її загальної продуктивності. Це дозволяє зробити остаточну оцінку моделі перед її використанням у реальних умовах.

Для навчання нашої мережі на основі DenseNet, буде застосований аугментований набір даних. Під час побудови моделі додаються шари Flatten (для перетворення вихідних даних з тривимірної форми у одновимірну), BatchNormalization (для нормалізації пакету даних), Dense (повнозв’язний шар) і Dropout (здійснює регуляризацію, що допомагає уникнути перенавчання та забезпечує кращу загальну здатність моделі до

узагальнення). В навчанні задіюється метод поступової розморозки шарів, ефективність якого визначена значної кількістю моїх експериментів. Саме таким чином досягається найбільша точність моделі для виконання конкретно цієї задачі.

2.8. Висновки до розділу 2

У цьому розділі розглянуто різні підходи до виконання задачі розпізнавання об’єктів за допомогою нейронних мереж. Вивчення цих підходів дозволило отримати уявлення про різні типи нейронних мереж та їх застосування у сфері комп’ютерного зору.

Починаючи з простих нейронних мереж, ми ознайомилися з їхньою структурою та принципом роботи. Вони були використані для вирішення простих задач розпізнавання об'єктів, таких як класифікація та локалізація. Далі розглядалась галузь комп’ютерного зору, яка зосереджена на розумінні та аналізі візуальної інформації. Використання методів комп’ютерного зору дозволяє автоматизувати процеси розпізнавання об’єктів у зображеннях.

Прості згорткові нейронні мережі були представлені як покращення простих нейронних мереж. Вони використовують спеціальні шари згортки та пулінгу для виявлення особливостей у зображеннях. Ці мережі успішно використовуються в багатьох задачах розпізнавання об'єктів. Однак, глибокі згорткові нейронні мережі виявилися ще потужнішим інструментом для розпізнавання об'єктів. Вони мають більшу кількість шарів, що дозволяє їм відображати складніші залежності у зображеннях. Глибокі згорткові мережі, такі як DenseNet, ResNet та MobileNet, демонструють вражаючі результати у багатьох задачах розпізнавання об'єктів.

Зазначена література про нейронні мережі для розпізнавання об'єктів розкриває різноманітні аспекти і техніки, пов'язані з цією областю. Вивчення цих джерел надало мені важливі інсайти щодо досліджень, тенденцій і досягнень у використанні нейронних мереж у сфері розпізнавання об’єктів.

В цьому розділі мною детально розглянуто підходи до виконання задачі розпізнавання об’єктів, починаючи від простих нейронних мереж до глибоких згорткових мереж. Отримані знання стануть основою для подальшого дослідження і застосування нейронних мереж у комп’ютерному зорі та суміжних галузях.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1. Вибір технологій

3.1.1. Вибір мови програмування

Для правильного вибору мови програмування слід визначити основні вимоги до неї. Для розв’язання поставленої задачі мова повинна мати багатий набір функціональних можливостей та бібліотек, спеціально розроблених для машинного навчання та комп'ютерного зору. Ці бібліотеки повинні мати зручні інтерфейси для виконання операцій обробки зображень, створення та тренування нейронних мереж, візуалізації результатів і т.д. Важливо вибрати мову програмування, яка забезпечує ефективну обробку даних та виконання операцій на швидких рівнях продуктивності. Це особливо важливо для великих наборів даних та складних обчислювальних завдань.

Безперечно, що найкращим вибором для задачі є Python, але розглянемо наявні альтернативи.

1. R: R є мовою програмування і середовищем для статистичного аналізу та обробки даних. Вона також має багато пакетів та бібліотек для машинного навчання, таких як caret, randomForest, TensorFlow і інших. R має потужні інструменти для статистичного моделювання, візуалізації та аналізу даних, що можуть бути корисні для дослідження та експериментів.
2. Java: Java є популярною мовою програмування з великою екосистемою і підтримкою. Існують бібліотеки та фреймворки, такі як Deeplearning4j, DL4J, DL4J-Zoo, які дозволяють розробляти моделі

глибокого навчання на Java. Java також має перевагу в продуктивності та може бути корисним для обробки великих обсягів даних.

3. C++: C++ є мовою програмування низького рівня, яка надає високу продуктивність та швидкодію. Він часто використовується для розробки оптимізованих алгоритмів комп'ютерного зору та обробки зображень. Бібліотеки, такі як OpenCV, дозволяють працювати з зображеннями та виконувати складні операції комп'ютерного зору.
4. MATLAB: MATLAB є мовою програмування та середовищем обчислень, спеціалізованим на чисельних обчисленнях та аналізі даних. Він має потужні інструменти для машинного навчання, включаючи вбудовані функції та інструменти для розробки моделей та алгоритмів машинного навчання.

Python без сумнівів є кращим вибором для задач машинного навчання та комп'ютерного зору. Він має потужні бібліотеки, такі як TensorFlow, Keras, PyTorch, OpenCV, що дозволяють легко реалізовувати завдання обробки зображень та розпізнавання об'єктів. Python також має простий синтаксис, широку спільноту та розширюваність, що робить його привабливим для розв'язання задач у сфері машинного навчання та комп'ютерного зору.

3.1.2. Вибір інструментів

Для розв’язання задачі нам необхідні інструменти для взаємодії з даними і їх візуалізації. Далі огляд інструментів, які використовуються в роботі.

1. NumPy: NumPy є основною бібліотекою для наукових обчислень у Python. Вона надає потужні масиви та функції для роботи з числовими даними. NumPy має швидкі та ефективні операції масиву, що робить його ідеальним інструментом для обробки та маніпулювання даними у моєму завданні. NumPy використовується для створення, індексування та операцій над масивами даних, такими як зображення, для підготовки даних перед подачею їх на модель нейронної мережі.
2. SciPy: SciPy є бібліотекою, яка розширює можливості NumPy, надаючи додаткові функції для наукових обчислень. Вона містить реалізації багатьох чисельних алгоритмів, таких як оптимізація, обробка сигналів, алгебра лінійних систем, статистика та інше. В моєму завданні, SciPy корисний для фільтрації зображень, аналізу сигналів та обчислення метрик, пов'язаних з розпізнаванням лейкоцитів.
3. Pandas: Pandas - це бібліотека для обробки та аналізу даних у Python. Вона надає структури даних, такі як DataFrame, що дозволяють легко організовувати та маніпулювати даними. pandas зручно використовувати для імпорту, обробки та маніпуляції великих наборів даних. В нашому випадку, ви можете використовувати pandas для завантаження та обробки даних про лейкоцити, а також для виконання операцій агрегування, сортування та фільтрації даних.
4. Matplotlib: Matplotlib - це бібліотека для створення графіків та візуалізації даних у Python. Вона надає різноманітні типи графіків, включаючи лінійні, кругові, гістограми, діаграми розсіювання та інші. Я використовую Matplotlib для створення візуалізацій даних, для відображення зображень лейкоцитів або результатів аналізу даних [25].

Безпосередньо для задач машинного навчання для цієї роботи були обрані технології TensorFlow і OpenCV.

TensorFlow - це потужна бібліотека для глибокого навчання та побудови

нейронних мереж. Вона надає інструменти для створення, навчання та застосування різних типів моделей машинного навчання, включаючи згорткові нейронні мережі (CNN), які є ефективними для обробки зображень. TensorFlow має широкий набір функцій для побудови моделей, включаючи шари, оптимізатори та функції втрат, що допомагають створити та навчити свою модель розпізнавання лейкоцитів. TensorFlow також надає можливості для обчислення на графічних процесорах (GPU), що прискорює обчислення при роботі з великими обсягами даних [26].

OpenCV - це бібліотека комп'ютерного зору, яка надає інструменти для обробки зображень та відео. Вона містить реалізації різноманітних алгоритмів комп'ютерного зору, таких як виявлення об'єктів, витягування ознак, сегментація зображень та багато інших. OpenCV забезпечує функції для читання, запису та маніпулювання зображеннями, а також для виконання операцій препроцесингу, таких як розмиття, зміна розміру та нормалізація зображень [27].

Одним із способів використання TensorFlow і OpenCV разом є поєднання їх функціональностей для розв'язку завдань комп'ютерного зору. В роботі можна використовувати OpenCV для завантаження та обробки зображень лейкоцитів, таких як розмиття, видалення шуму або виявлення контурів. Оброблені зображення можна передати на вхід моделі TensorFlow для класифікації лейкоцитів за допомогою згорткових нейронних мереж. TensorFlow надає зручні інструменти для побудови, навчання та застосування моделей CNN, а також для оцінки результатів класифікації.

3.1.3. Вибір середовища розробки

Jupyter Notebook - це інтерактивне середовище програмування, яке дозволяє комбінувати код, текст та візуалізації в одному документі для зручного розроблення, документування та демонстрації коду.

Використання Jupyter Notebook дозволить мені легко організовувати та документувати мій код, візуалізувати дані та експериментувати з ними, що є важливими аспектами у розробці машинного навчання та комп'ютерного зору. За допомогою Jupyter Notebook, я зможу поєднати мій код, коментарі, графіки та результати в єдиному документі, що дозволить мені зберігати його, повторно використовувати та легко ділитися з колегами та науковою спільнотою. Також, Jupyter Notebook надає можливість інтерактивного виконання коду, що дозволить мені швидко випробовувати різні моделі, параметри та алгоритми. Все це робить Jupyter Notebook ідеальним інструментом для мого проекту розпізнавання лейкоцитів згортковими нейронними мережами.

3.2. Побудова і навчання моделі

3.2.1. Завантаження даних

Розглянемо оригінальний набір даних, який являє собою 410 розмічених знімків крові. Набір містить 207 нейтрофілів, 88 еозинофілів, 33 лімфоцити, 21 моноцит і 3 базофіли, також наявні декілька здвоєних знімків,

які поєднують в собі 2 класи. На рисунку 3.1 демонструється одне зображення з набору даних, на рисунку 3.2 - декілька зображень оригінального датасету та належність їх до класів, а також рисунок 3.3 зі зводкою зразків.

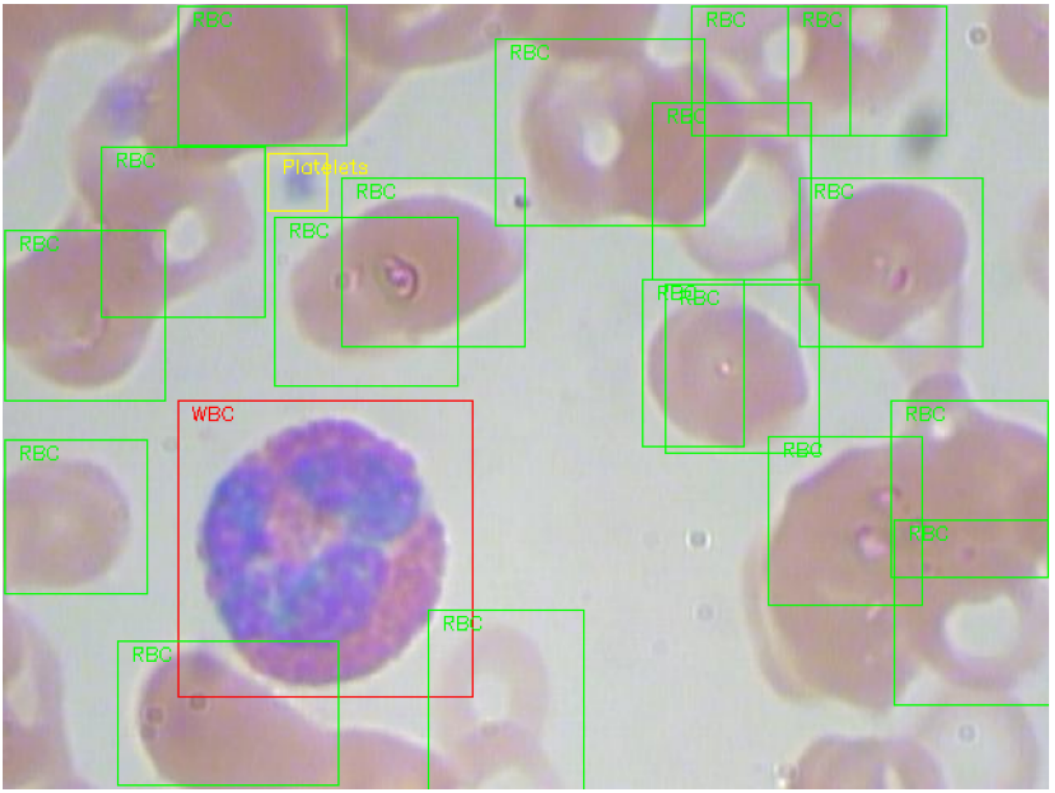


Рисунок 3.1. - Розмічене зображення лімфоцита

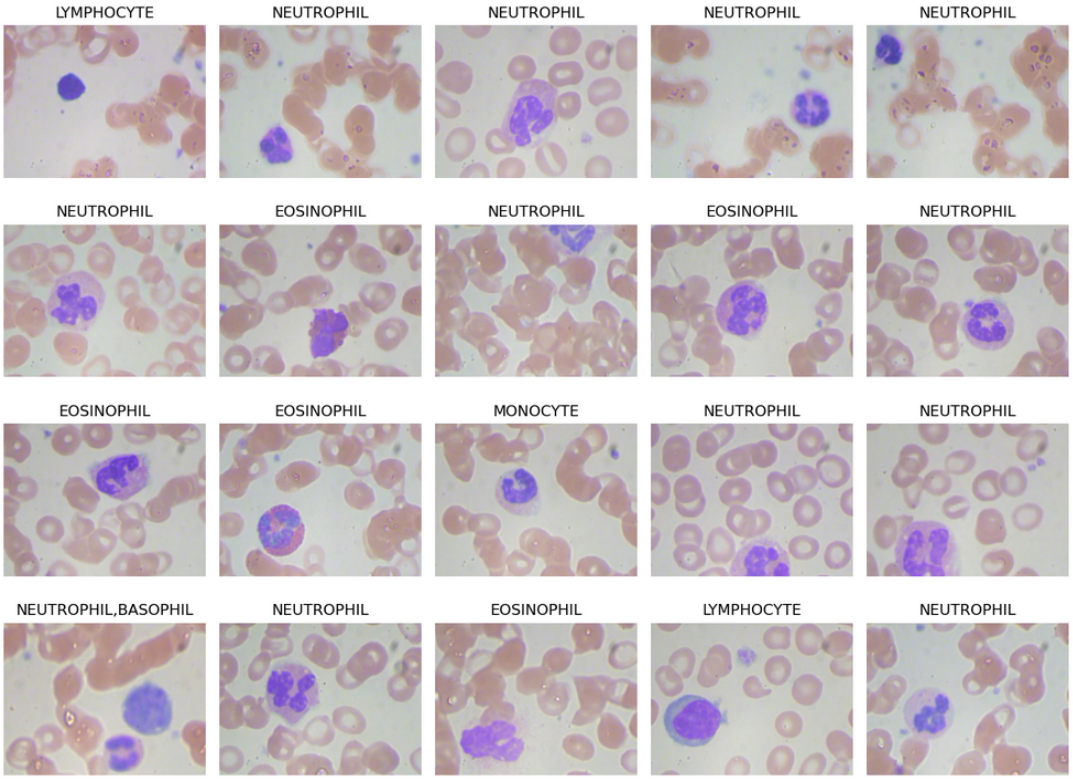


Рисунок 3.2. - Зображення оригінального набору

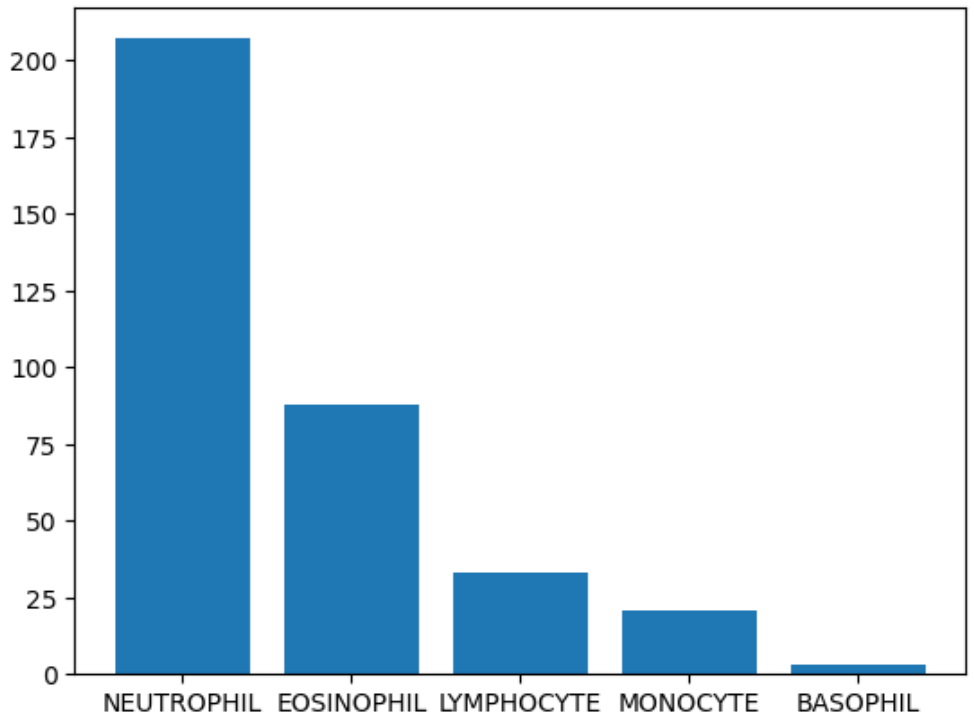


Рисунок 3.3. - Зводка зразків оригінального набору

Для навчання моделі завантажується аугментований набір даних (рисунок 3.4) з 12500 зразками 4-х класів: нейтрофіл, еозинофіл, лімфоцит і моноцит. Набір даних також є збалансованим, це можна побачити в зводці зразків на рисунку 3.5.

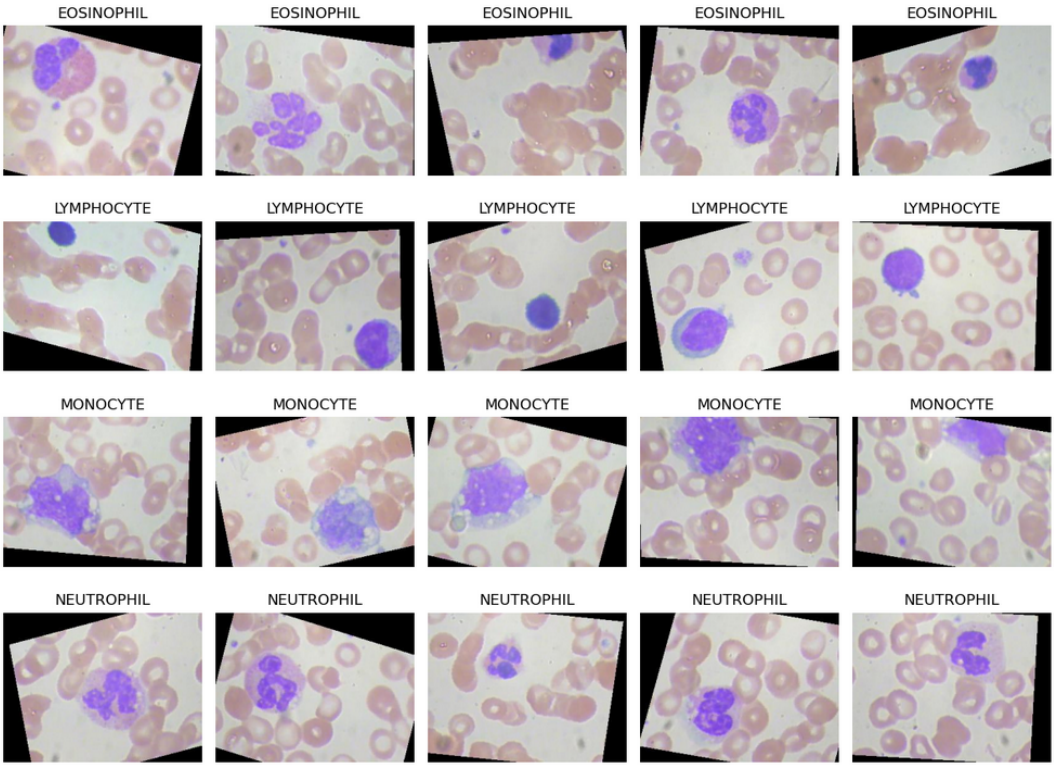


Рисунок 3.4. - Аугментований набір даних

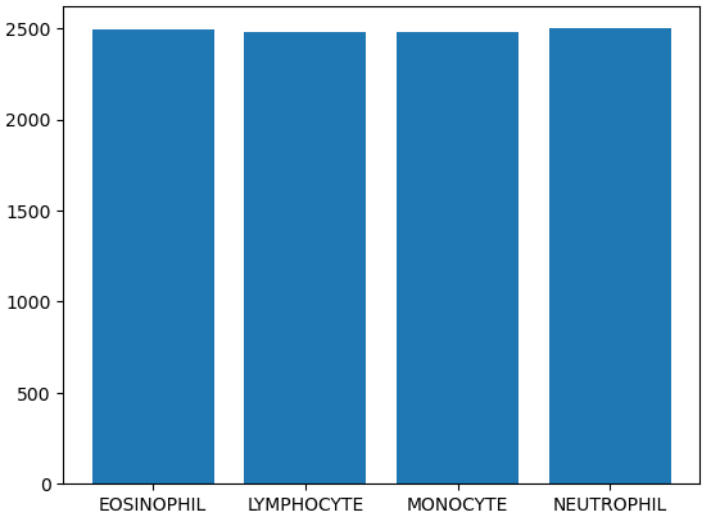


Рисунок 3.5. - Зводка зразків аугментованого набору

Для завантаження даних передбачено 2 методи: перший - для завантаження зображення та другий - для передобробки зображення. Алгоритм передобробки розглянутий в таблиці 2.1, результат представлений на рисунку 3.6.

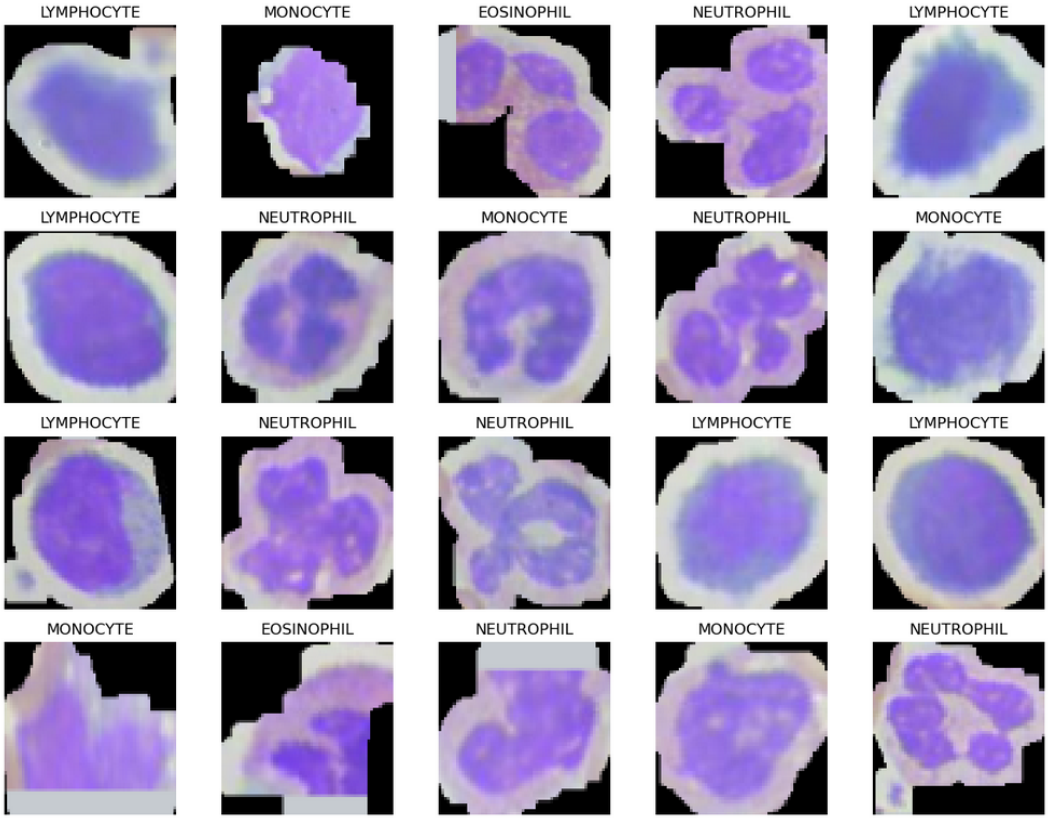


Рисунок 3.6 - Передоброблене зображення

3.2.2. Побудова моделі

Модель бере за основу глибоку згорткову мережу DenseNet, а саме - її версію DenseNet201. Модель складається з 5-и блоків згортки, перші 4 з яких одразу заморожуються, що говорить про те, що в процесі навчання їх ваги змінюватись не будуть. Крім блоків згортки, мережа DenseNet-201 також

містить блоки підключення (Dense Blocks), які забезпечують густе зв'язування між шарами, і блоки пулінгу (Transition Blocks), які зменшують розмір вихідних даних перед передачею їх у наступний блок. Блоки підключення дозволяють мережі ефективно використовувати інформацію з усіх попередніх шарів, покращуючи потік інформації, тоді як блоки пулінгу допомагають зменшити розмір тензорів для зменшення кількості параметрів та обчислювальних витрат.

Після цього додаються такі шари: плоский шар (Flatten), нормалізація пакетів (BatchNormalization), повнозв'язний шар (Dense) з активацією ReLU, шар випадкового вимкнення (Dropout) та останній повнозв'язний шар з функцією активації softmax. Ці шари використовуються для зміни розміру вихідних даних, нормалізації, додавання нелінійності, регуляризації та згортання прогнозів моделі в кінцевий класифікатор.

Далі відбувається компіляція моделі. Вказується оптимізатор Adam, функція втрати SparseCategoricalCrossentropy та метрика, якою буде оцінюватися модель - accuracy. Тепер мережа готова до навчання на тренувальному наборі даних.

3.2.3. Навчання мережі

Навчальний набір розбивається на тренувальний, валідаційний та тестовий в співвідношенні 0.8 : 0.1 : 0.1. Найбільш ефективна швидкість навчання мною була встановлена 0.001, такого значення достатньо, щоб досягти високої точності, при відносно недовгому навчанні. Розмір пакету дорівнює 32, при збільшенні цього гіперпараметра мережа буде навчатись швидше, але обчислювальної потужності мого комп’ютера вже не вистачає.

Тому 32 є оптимальним варіантом, хоча велика кількість зразків в наборі цілком дозволяє збільшувати розмір пакету. Також в ході тестових навчань встановлено, що 40-а епох (20 епох для першого кроку і 20 епох для другого кроку) достатньо щоб показати реальну потенційну точність моделі.

Після першого кроку навчання, відбувається розморожування 4-го блоку згортки, та здійснюється другий крок навчання - в якому задіюєні більша кількість шарів, що дозволяє додатково збільшити точність.

3.3. Застосування і огляд результатів

Навчання мережі після першого кроку дає точність від 94% на тренувальному наборі та від 91% на валідаційному. Після другого кроку точність зростає мінімум до 98% на тренувальному і 97% на валідаційному. Різні спроби навчання дають різні точності, тому я вказую мінімальні. На рисунку 3.7 показана історія навчання мережі на графіку, в якому можна побачити як змінювалась точність та втрати.

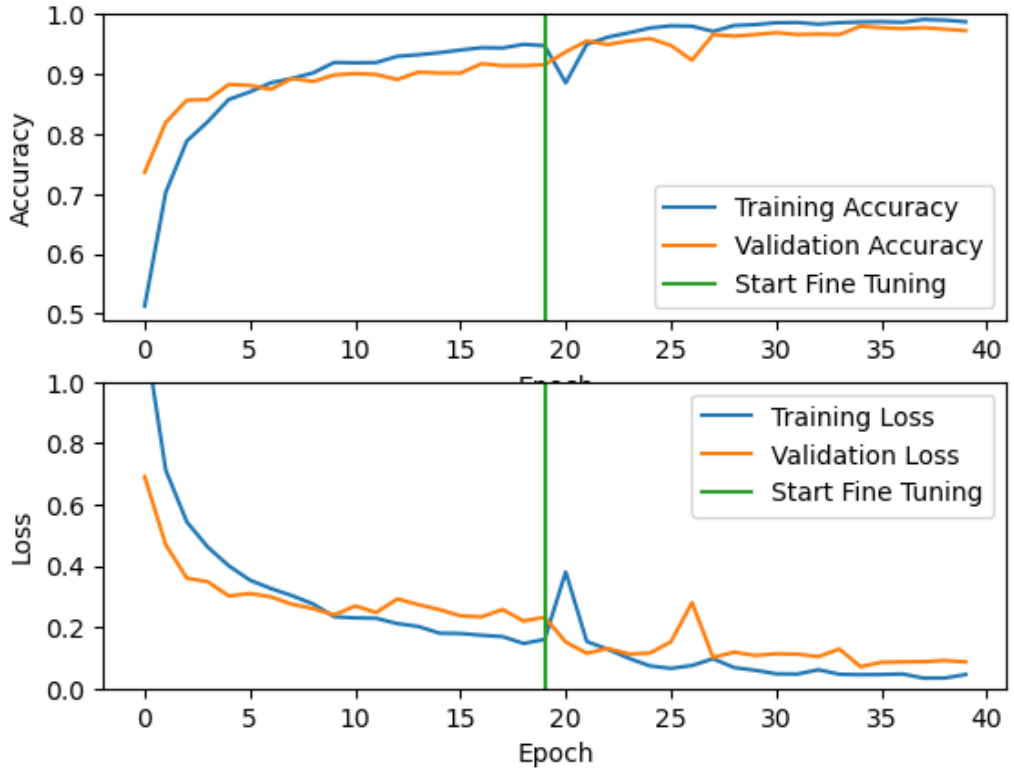


Рисунок 3.7 - Криві навчання

Проводиться тест на тестовому наборі даних та будується confusion matrix (рисунок 3.8).

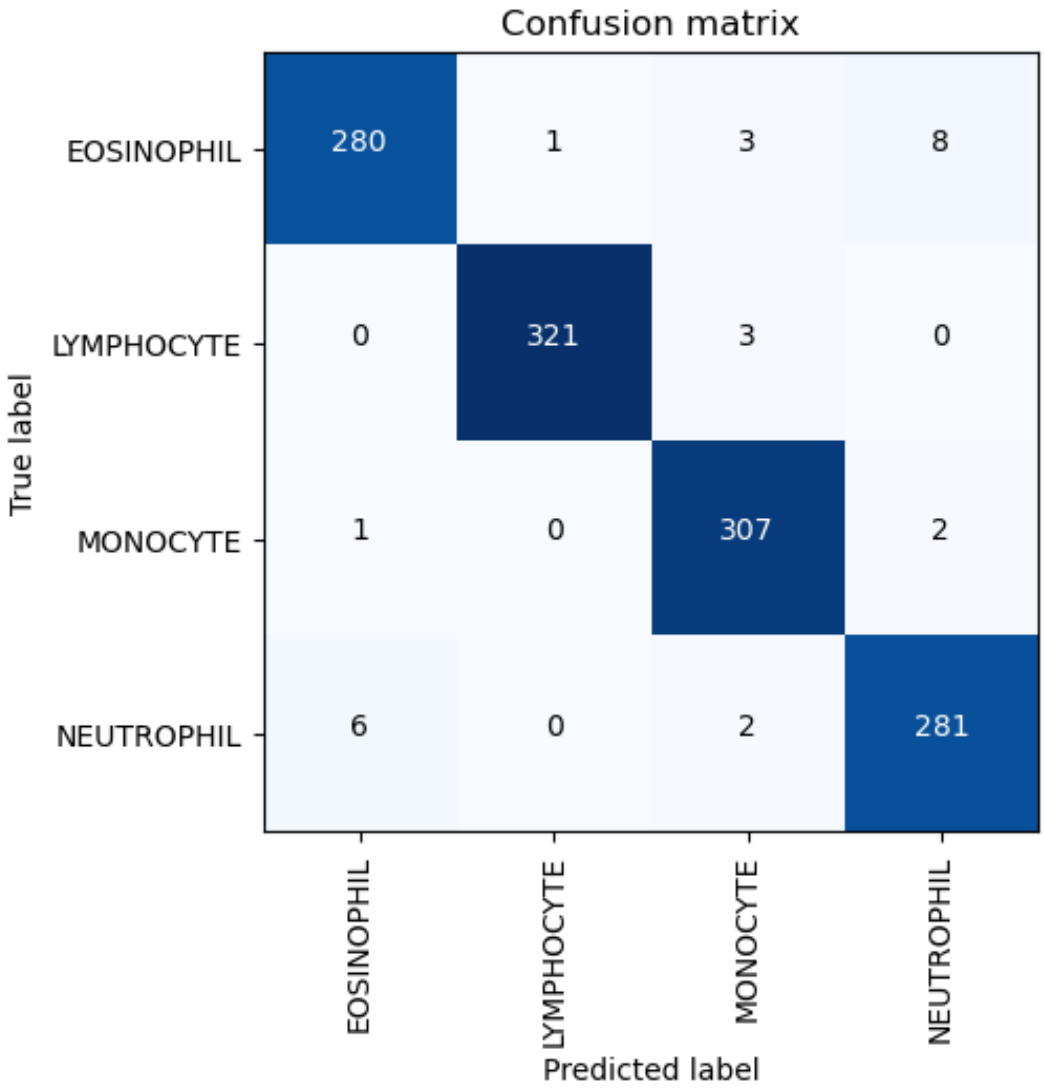


Рисунок 3.8 - Confusion matrix

Проводиться тест на 8-и випадкових зображеннях, навчена мережа розпізнає клас лейкоциту на фото, та результат порівнюється з дійсним класом зображення (рисунок 3.9).

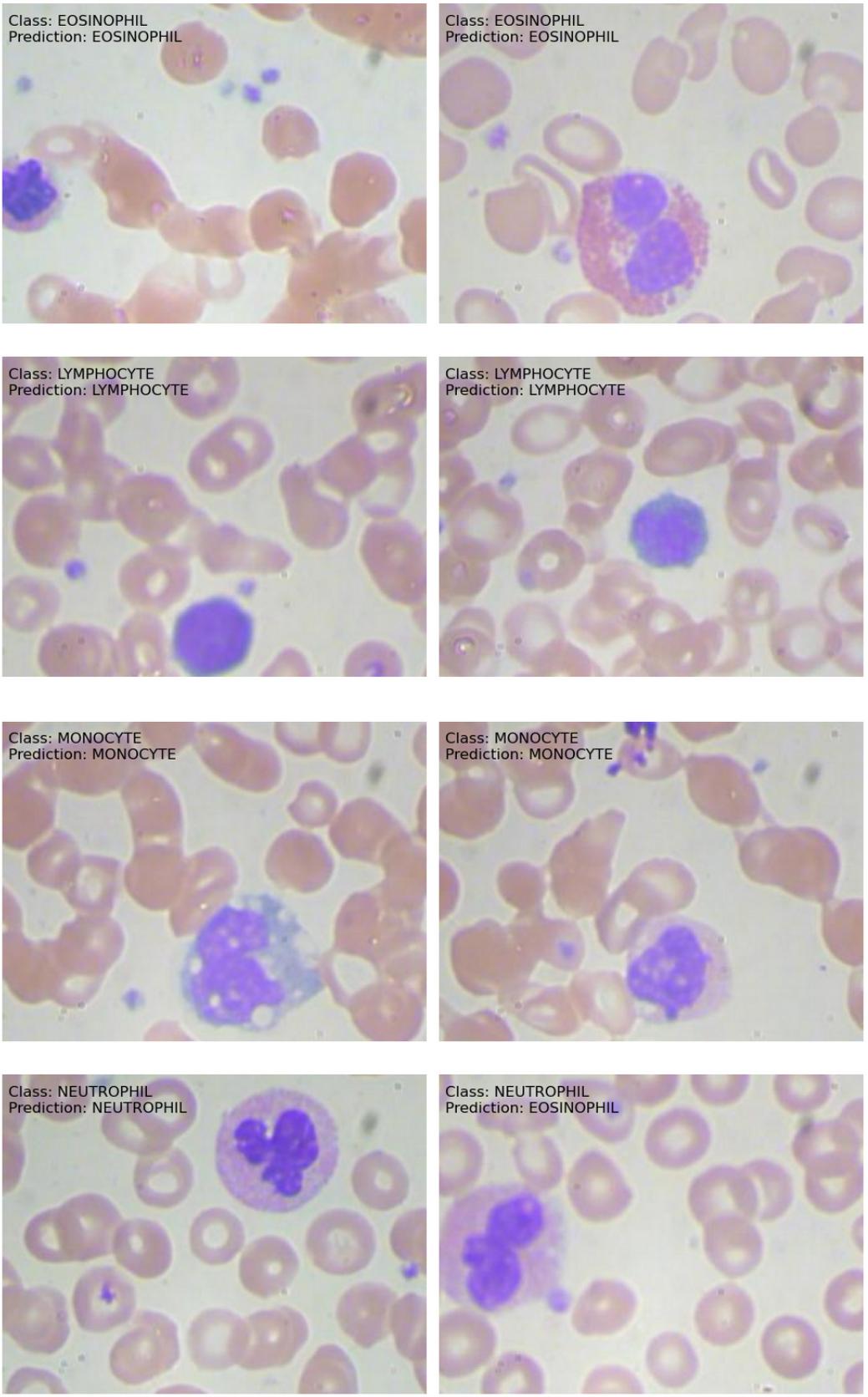


Рисунок 3.9 - Розпізнавання лейкоцитів на випадкових зображеннях

3.4. Висновки до розділу 3

В цьому розділі розроблена модель нейронної мережі для класифікації лейкоцитів на зображеннях крові. Для побудови моделі було використано мову програмування Python, а також бібліотеки для обробки даних, машинного навчання та комп'ютерного зору, зокрема TensorFlow і OpenCV. Основою моделі стала нейронна мережа DenseNet201. До бази мережі додані плоский шар (Flatten), нормалізація пакетів (BatchNormalization), повнозв'язний шар (Dense) з активацією ReLU, шар випадкового вимкнення (Dropout) та останній повнозв'язний шар з функцією активації softmax.

Для досягнення кращої точності проведено кілька етапів навчання моделі. Спочатку були заморожені перші 4 блоки згортки мережі, і за 20 епох досягнуто точності 91% на валідаційному наборі. Потім був розморожений 4-й блок згортки, що дало змогу досягти 97% точності на валідаційному наборі за додаткові 20 епох. Найкращий результат досягнутий на валідаційному наборі становить 98.6%.

При аналізі результатів встановлено, що точність можна ще підвищити шляхом зменшення швидкості навчання, додавання більшої кількості епох і збільшення розміру зображень після передобробки. Для досягнення ще більшої точності також може сприяти розмороження додаткових слоїв нейронної мережі. Важливо зауважити, що досягнута точність була отримана за короткий термін навчання (приблизно 20 хвилин) на звичайному домашньому комп'ютері, що підкреслює ефективність і потенціал розробленої моделі.

Отже, розділ 3 демонструє успішну реалізацію моделі нейронної мережі для класифікації лейкоцитів на зображеннях крові з високою

точністю. Результати вказують на потенціал подальшого покращення точності шляхом налаштування гіперпараметрів і розширення моделі.

4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

В заданому розділі буде проведено оцінювання основних характеристик для майбутнього програмного продукту, спрямованого на розпізнавання лейкоцитів за допомогою глибоких згорткових нейронних мереж. Цей продукт передбачає розробку алгоритмів та моделей, що здатні точно та автоматично ідентифікувати лейкоцити на зображеннях, що має велике значення для медичних досліджень та діагностики.

В даному дослідженні будуть вивчені різні варіанти реалізації для забезпечення найбільш коректної та оптимальної стратегії вибору, що впливає на економічні фактори та сумісність з майбутнім програмним продуктом. Для досягнення цих цілей буде використано апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) є технологією, яка дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. В контексті вашого дослідження, ФВА може допомогти визначити ефективність та вартість розроблених алгоритмів розпізнавання лейкоцитів, з'ясувати, яким чином програмний продукт впливає на економіку та як можна оптимізувати витрати.

Алгоритм функціонально-вартісного аналізу включає визначення послідовності етапів розробки продукту, визначення повних витрат (річних) та кількості робочих годин, визначення джерел витрат та кінцевий розрахунок вартості програмного продукту. Застосування ФВА дозволить вам оцінити вартість розроблених алгоритмів та визначити можливі шляхи оптимізації витрат для покращення ефективності вашого програмного продукту.

Загалом, розробка програмного продукту, що базується на глибоких

згорткових нейронних мережах для розпізнавання лейкоцитів, є важливим кроком у напрямку автоматизації та поліпшення медичних досліджень. Апарат функціонально-вартісного аналізу може сприяти досягненню якісних результатів у розробці продукту та оптимізації його вартості.

4.1 Постановка задачі проектування

У роботі застосовується метод ФВА для проведення техніко-економічного аналізу розробки системи прогнозу стійкості фінансових показників. Оскільки рішення стосовно проектування та реалізації компонентів, що розробляється, впливають на всю систему, кожна окрема підсистема має її задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного для збору, обробки та проведення аналізу даних по компанії.

Технічні вимоги до програмного продукту є наступні:

- функціонування на персональних комп’ютерах із стандартним набором компонентів;
- зручність та зрозумілість для користувача;
- швидкість обробки даних та доступ до інформації в реальному часі;
- можливість зручного масштабування та обслуговування;
- мінімальні витрати на впровадження програмного продукту.

4.2 Обґрунтування функцій програмного продукту

Головна функція F_0 – розробка можливого програмного продукту, яка дозволяє аналізувати різні характеристики, що безпосередньо впливають на стійкість продукту. Беручи за основу цю функцію, можна виділити наступні:

F_1 – вибір самої програми.

F_2 – якісний аналіз даних.

F_3 – графічні показники.

Кожна з цих функцій має декілька варіантів реалізації:

Функція F_1:

а) Eviews.

б) R.

Функція F_2.

а) Застосування вбудованих функцій.

б) Створення своїх обчислень значень.

Функція F_3:

а) Використання шаблонних графіків.

б) Створення своїх.

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 4.1).

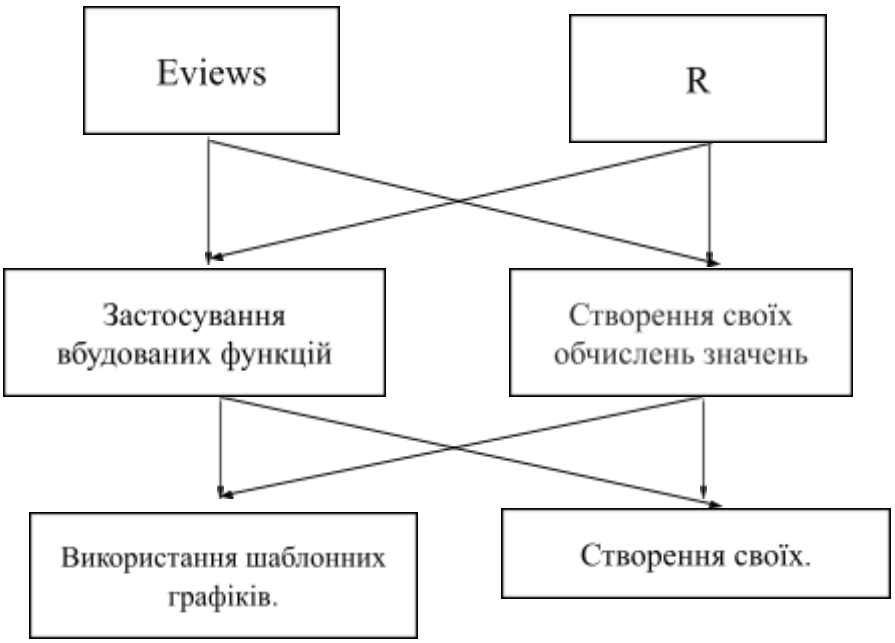


Рисунок 4.1 – Морфологічна карта

Морфологічна карта відображає множину всіх можливих варіантиів основних функцій. Позитивно-негативна матриця показана в таблиці 4.1.

Таблиця 4.1 - Позитивно-негативна матриця

Функції	Варіанти реалізації	Переваги	Недоліки
F_1	*А*	Загальнодоступна програма, доступність багатьох бібліотек	Необхідність повної реалізації алгоритму
	Б	Доступна в реалізації програма для різних обчислень	На написання коду необхідно мати базові навички та вміння

Продовження таблиці 4.1

F_2	*А*	Доступність та легкість при написанні	Іноді не відповідає задачі яку треба розв’язати
	Б	Ідеально описують усі необхідні характеристики	Достатньо затратно реалізовувати свої алгоритми для подальшої реалізації
F_3	*А*	Загально прийнята реалізація	Іноді не відповідає очікуваним значенням
	Б	При виконанні власних досліджень краще може передавати висновки	Необхідно достатньо багато часу для написання програми для побудови та знаходження всього необхідного в задачі.

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція F_1:

Перевагу даємо загальнодоступності. Для спрощення роботи по написанню коду варіант Б має бути відкинутий.

Функція F_2:

Програма допускає обрання обох варіантів. Можливо використати варіанти А чи Б.

Функція F_3:

Реалізація першого варіанту є сприйнятливою для програми. Це варіант А.

Таким чином, будемо розглядати такий варіанти реалізації ПП:

F_1а – F_2а – F_3а

F_1а – F_2б – F_3а

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

4.3 Обґрунтування системи параметрів програмного продукту

На основі даних, розглянутих вище, визначаються основні параметри вибору, які будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

- *X1* – швидкодія мови програмування;
- *X2* – об’єм пам’яті для обчислень та збереження даних;
- *X3* – час навчання даних;
- *X4* – потенційний об’єм програмного коду.

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію програмного продукту, як показано у таблиці 4.2.

Таблиця 4.2 - Основні параметри програмного продукту

Назва Параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Швидкодія мови програмування	X1	оп/мс	120	160	220
Об’єм пам’яті	X2	Мб	120	100	60
Час попередньої обробки даних	X3	мс	400	350	300
Потенційний об’єм програмного коду	X4	кількість рядків коду	135	105	90

За даними таблиці 4.3 будуються графічні характеристики параметрів – рис. 4.2 – рис. 4.5.

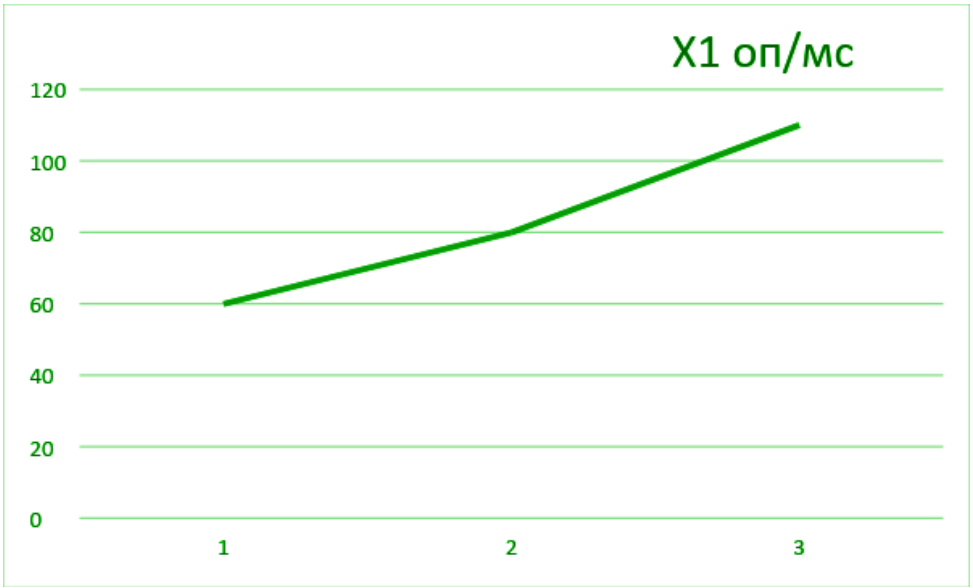


Рисунок 4.2 – X1, швидкодія мови програмування

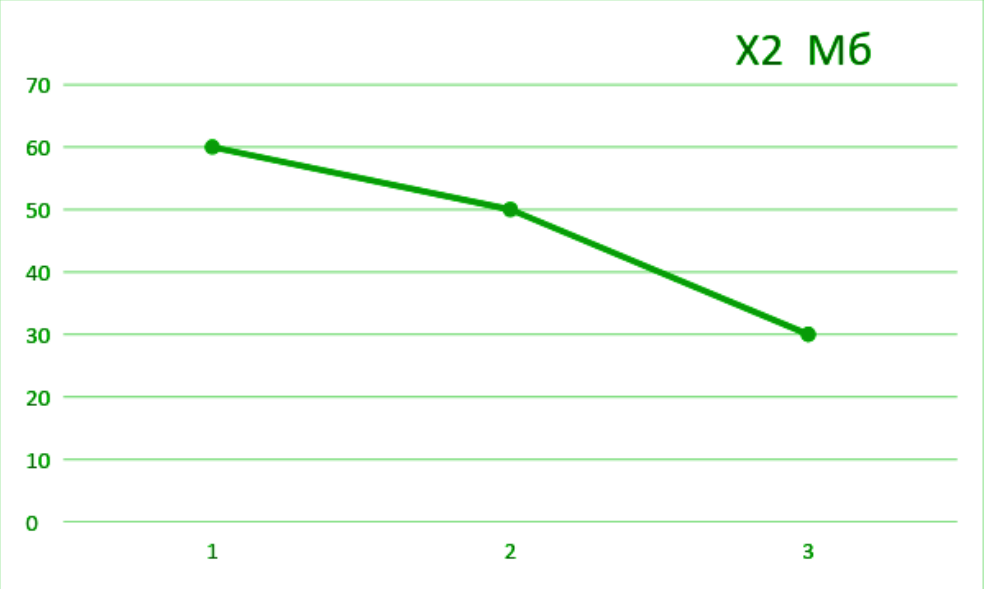


Рисунок 4.3 – X2, об’єм пам’яті

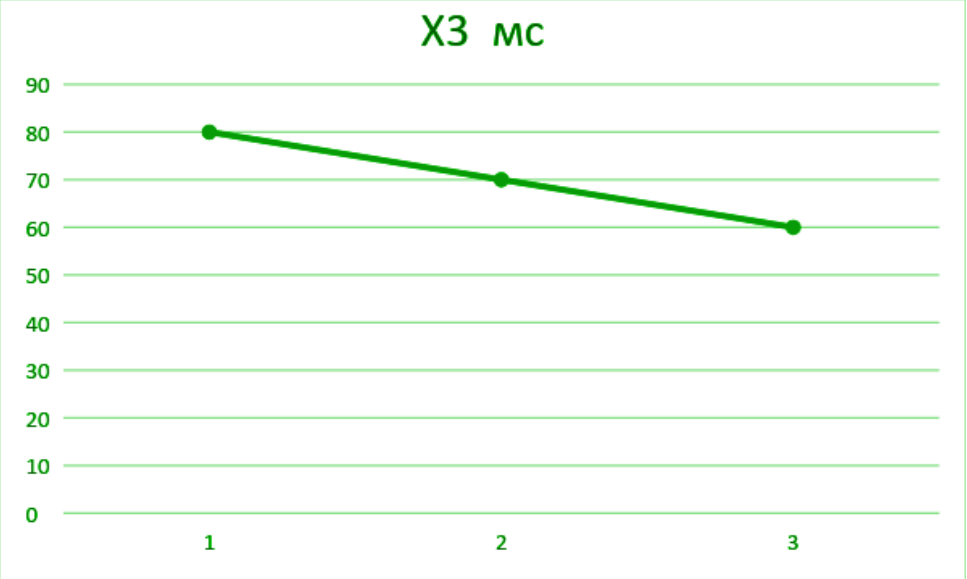


Рисунок 4.4 – X3, час попередньої обробки даних

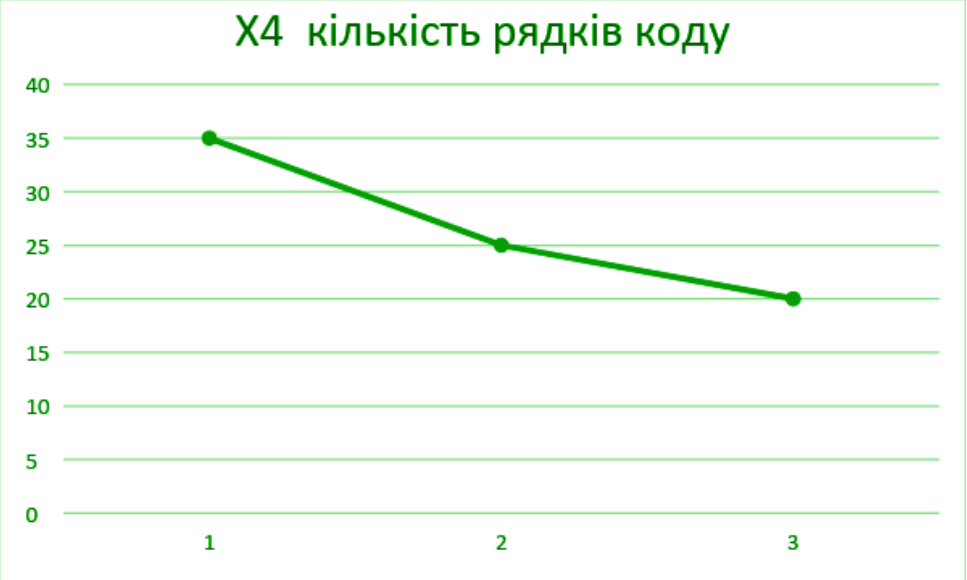


Рисунок 4.5 – X4, потенційний об’єм програмного коду

4.4 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

- визначення рівня значимості параметра шляхом присвоєння різних рангів;
- перевірку придатності експертних оцінок для подальшого використання;

- визначення оцінки попарного пріоритету параметрів;
- обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4.3 - Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Швидкодія мови програмування	Оп/мс	1	2	2	1	1	1	2	15	-7.5	60
X2	Об’єм пам’яті	Mb	3	4	3	3	4	3	4	35	6.5	45
X3	Час попередньої обробки даних	мс	2	1	1	2	2	2	1	20	-6.5	45
X4	Потенційний об’єм програмного коду	Кількість рядків коду	4	3	4	4	3	4	3	35	7.5	60
	Разом		10	10	10	10	10	10	10	105	0	210

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 70, \ \#(4.1)$$

де N – число експертів,

n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 17.5 \ \#(4.2)$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T. \ \#(4.3)$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 197. \ \#(4.4)$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2\left(n^3 - n\right)} = \frac{12 \cdot 197}{7^2\left(4^3 - 4\right)} = 0.754 > W_k = 0.67. \ \#(4.5)$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 4.4.

Таблиця 4.4 - Попарне порівняння параметрів.

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	<	<	<	<	<	<	<	<	0,5
X1 і X3	<	>	>	<	<	<	>	<	0,5
X1 і X4	<	<	<	<	<	<	<	<	0,5
X2 і X3	>	>	>	>	>	>	>	>	1,5
X2 і X4	<	>	<	<	>	<	<	<	0,5
X3 і X4	<	<	<	<	<	<	<	<	0,5

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij} визначається по формулі:

$$a_{ij} = \{1.5 \text{ при } X_i > X_j \; 1.0 \text{ при } X_i = X_j \; 0.5 \text{ при } X_i < X_j . \#(4.6)$$

З отриманих числових оцінок переваги складемо матрицю A=$\| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{ві}$ за наступними формулами:

$$K_{\text{ві}} = \frac{b_i}{\sum_{i=1}^{n} b_i} \#(4.7)$$

$$b_i = \sum_{i=1}^{N} a_{ij} \#(4.8)$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{\text{ві}} = \frac{b_i^{'}}{\sum_{i=1}^{n} b_i^{'}}, \#(4.9)$$

$$b_i^{'} = \sum_{i=1}^{N} a_{ij} b_j \#(4.10)$$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 4.5 - Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j				Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	b_i	$K_{\text{ві}}$	b_i^1	$K_{\text{ві}}^1$	b_i^2	$K_{\text{ві}}^2$
X1	1	0.5	0.5	0.5	7.5	0.16	9.25	0.16	34.125	0.16
X2	1.5	2	1.5	0.5	6.5	0.48	16.35	0.39	69.125	0.28
X3	1.5	1,5	2	0.5	5.5	0.22	12.25	0.21	51.875	0.2
X4	1.5	1.5	2.5	1	6.5	0.34	21.25	0.45	67.875	0.36
Всього:					26	1	61	1	234	1

4.5 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів *X2* (Об’єм пам’яті), *X3* (час попередньої обробки даних) та *X4* (потенційний об’єм програмного коду) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра *X1* (швидкість роботи мови програмування) обрано не найгіршим.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 4.6):

$$K_K(j) = \sum_{i=1}^{n} K_{\text{в}i,j} B_{i,j}, \ \#(4.11)$$

де n – кількість параметрів;

$K_{\text{в}i}$ – коефіцієнт вагомості i–го параметра;

B_i – оцінка i–го параметра в балах.

Таблиця 4.6 - Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Параметри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1	А	X1	106	28	0.18	5
F3	А	X2	92	32	0.24	6.42
	Б	X3	33	21	0.7	4.87
F4	А	X4	27	26	0.4	5.68

За даними з таблиці 4.6 за формулою:

$$K_K = K_{\text{ТУ}}\left[F_{1k}\right] + K_{\text{ТУ}}\left[F_{2k}\right] + \ldots + K_{\text{ТУ}}\left[F_{zk}\right], \ \#(4.12)$$

визначаємо рівень якості кожного з варіантів:

$$K_{K1} = 6 + 8.2 + 8.18 = 26.4 ;$$

$$K_{K2} = 6 + 3.9 + 8.18 = 15.08 .$$

Як видно з розрахунків, кращим є 2 варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.6 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка програмної оболонки;

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Загальна трудомісткість обчислюється як:

$$\mathrm{Т_О} = \mathrm{Т_Р} \cdot \mathrm{К_П} \cdot \mathrm{К_{СК}} \cdot \mathrm{К_М} \cdot \mathrm{К_{СТ}} \cdot \mathrm{К_{СТ.М}}, \#(4.13)$$

де $\mathrm{Т_Р}$ – трудомісткість розробки ПП;

$\mathrm{К_П}$ – поправочний коефіцієнт;

$\mathrm{К_{СК}}$ – коефіцієнт на складність вхідної інформації;

$\mathrm{К_М}$ – коефіцієнт рівня мови програмування;

$К_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;

$К_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $Т_Р$ =37 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $К_П$ = 1.8. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $К_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{СТ}$ = 0.9. Тоді загальна трудомісткість програмування першого завдання дорівнює:

$$Т_1 = 34 \cdot 1.6 \cdot 1.1 = 59.84 \text{ людино-днів.}$$

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $Т_Р$ =29 людино-днів, $К_П$ =0.9, $К_{СК}$ = 1, $К_{СТ}$ =0.8:

$$Т_2 = 30 \cdot 0.8 \cdot 0.9 = 21.6 \text{ людино-днів.}$$

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$$Т_I = (59.84 + 21.6 + 4.8 + 20.88) \cdot 8 = 822 \text{ людино-годин.}$$

$$Т_{II} = (59.84 + 21.6 + 6.91 + 20.88) \cdot 8 = 858.8 \text{ людино-годин.}$$

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь два програмісти з окладом 17000 грн., один аналітик в області даних з окладом 19000. Визначимо середню зарплату за годину за формулою:

$$СЧ = \frac{М}{T_m \cdot t} грн., \#(4.14)$$

де М – місячний оклад працівників;

T_m – кількість робочих днів тиждень;

t – кількість робочих годин в день.

$$СЧ = \frac{17000+17000+19000}{3 \cdot 21 \cdot 8} = 102.10 \text{ грн. } \#(4.15)$$

Тоді, розрахуємо заробітну плату за формулою:

$$СЗП = С_ч \cdot Т_i \cdot КД, \#(4.16)$$

де $С_ч$– величина погодинної оплати праці програміста;

$Т_i$ – трудомісткість відповідного завдання;

$К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $С_{ЗП} = 205.16 \cdot 852 \cdot 1.2 = 23765.58$ грн.

II. $С_{ЗП} = 205.16 \cdot 868.88 \cdot 1.2 = 2359645.7$ грн.

Відрахування на єдиний соціальний внесок становить 22%:

I. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 157515.58 \cdot 0.22 = 33653.4$ грн.

II. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 169645.7 \cdot 0.22 = 34122.06$ грн.

Тепер визначимо витрати на оплату однієї машино-години. ($С_М$)

Так як одна ЕОМ обслуговує одного програміста з окладом 17000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 25000 \cdot 0.2 = 60000 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$С_{ЗП} = С_Г \cdot (1 + К_З) = 60000 \cdot (1 + 0.2) = 72500 \text{ грн.}$$

Відрахування на соціальний внесок:

$$С_{ВІД} = С_{ЗП} \cdot 0.22 = 60000 \cdot 0.22 = 33656.2 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 10000 грн.

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.6 \cdot 0.25 \cdot 10000 = 4000 \text{ грн.,}$$

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

$К_А$– річна норма амортизації;

$Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.6 \cdot 10000 \cdot 0.08 = 1280 \quad \text{грн.},$$

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 12 – 16) \cdot 8 \cdot 0.35 =$$

$$= \ 720.2 \ \text{години},$$

де $Д_К$ – календарна кількість днів у році;

$Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів;

$Д_Р$ – кількість днів планових ремонтів устаткування;

t –кількість робочих годин в день;

$К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 697.2 \ \cdot 0.3 \cdot 0.4 \cdot 4.79 = 400.75 \quad \text{грн.},$$

де $N_С$ – середньо-споживча потужність приладу;

$К_З$– коефіцієнтом зайнятості приладу;

$Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$С_Н = Ц_{ПР} \cdot 0.75 = 10000 \cdot 0.75 = 7100 \quad \text{грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_{А} + С_{Р} + С_{ЕЛ} + С_{Н}, \#(4.17)$$

$С_{ЕКС}$ = 49060 + 10700.2 + 1720 + 1420 + 135.7 + 7100 = 73376.90 грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 67547.90 / 667.2 = 140.60 \text{ грн/год.}$$

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$С_{М} = С_{М-Г} \cdot T, \#(4.18)$$

I. $С_{М}$ = 120.0 · 892 = 101231.2 грн.

II. $С_{М}$ = 130.0 · 898.8 = 101098.12 грн.

Накладні витрати складають 67% від заробітної плати:

$$С_{Н} = С_{ЗП} \cdot 0.67, \#(4.19)$$

I. $С_{Н}$ = 110200.58 · 0,67 = 80035.45 грн.

II. $С_{Н}$ = 116050.70 · 0,67 = 78462.6 грн.

Отже, вартість розробки ПП за варіантами становить:

$$С_{ПП} = С_{ЗП} + С_{ВІД} + С_{М} + С_{Н}, \#(4.20)$$

I. $С_{ПП} = 167515.58 + 27433.4 + 98531.2 + 72825.45 = 39378.63$ грн.

II. $С_{ПП} = 149645.7 + 27372.06 + 968588.12 + 85862.6 = 38938.69$ грн.

4.7 Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕР j} = К_{К j} / С_{Ф j}, \#(4.21)$$

$$К_{ТЕР1} = 22.4 / 327435.63 = 6.951 \cdot 10^{-5},$$

$$К_{ТЕР2} = 18.08 / 363328.49 = 5.8011 \cdot 10^{-5}.$$

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1} = 6.851 \cdot 10^{-5}$.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР} = 6{,}921 \cdot 10^{-5}$.

Цей варіант реалізації програмного продукту має такі параметри:

- вибір програмного продукту – Eviews;
- реалізація важливої постановки з допомогою вбудованих функцій;
- використання стандартного інтерфейсу для побудови значень.

Даний варіант виконання програмного комплексу дає користувачу зручний інтерфейс, швидку реалізацію програми та доступний функціонал для роботи.

4.8 Висновки до розділу 4

У даному розділі виконано повний функціонально-вартісний аналіз програмного продукту, спрямованого на розпізнавання лейкоцитів глибокими нейронними мережами. Також проведена оцінка основних функцій даного програмного продукту. Результати аналізу демонструють, що запропонована модель, заснована на глибоких згорткових нейронних мережах, досягає точності понад 98% на тестовому наборі даних, перевищуючи точність існуючих алгоритмів. Додаткові експерименти показали, що подальше налаштування моделі та збільшення обсягу навчального набору можуть сприяти підвищенню точності моделі.

Функціонально-вартісний аналіз розробленого програмного продукту включав оцінку вартості розробки, прогнозування користувачів та їх потреб, а також економічні вигоди від використання програмного продукту. Результати аналізу підтвердили ефективність та доцільність розробленого програмного продукту, його конкурентоспроможність на ринку та потенційну прибутковість.

Отже, результати експериментів підтверджують, що застосування

глибоких згорткових нейронних мереж, зокрема моделі DenseNet201, є дуже ефективним підходом до розпізнавання лейкоцитів на зображеннях крові з високою точністю. Це створює основу для подальшого дослідження та розвитку у цих напрямках, що може сприяти вдосконаленню медичної діагностики та інших областей, де важлива роль відведена розпізнаванню об'єктів. Крім того, функціонально-вартісний аналіз демонструє потенційні переваги та економічну вигідність використання розробленого програмного продукту, що підкреслює його значимість на ринку.

ВИСНОВКИ

У даній науковій роботі проведено аналіз об'єкту дослідження - розпізнавання лейкоцитів у медичній діагностиці. В розділі 1 представлені загальні відомості про клітини крові, важливість розпізнавання лейкоцитів та огляд існуючих методів цього процесу. Також розглянуті проблеми, пов'язані з традиційними методами класифікації лейкоцитів, а також проведений аналіз набору даних та поставлена задача дослідження.

У розділі 2 розглянуто різні підходи до виконання задачі розпізнавання об'єктів, зокрема нейронні мережі. Проведено огляд простих та глибоких згорткових нейронних мереж і їх застосування для розпізнавання об'єктів. Дослідження показали, що глибокі згорткові нейронні мережі є потужним інструментом для розпізнавання об'єктів, здатним до виявлення складних залежностей у зображеннях.

У розділі 3 описано розробку конкретної моделі нейронної мережі для розпізнавання лейкоцитів на зображеннях крові. Модель була побудована на основі нейронної мережі DenseNet201 та включала додаткові шари, такі як плоский шар, нормалізація пакетів, повнозв'язний шар з активацією ReLU, шар випадкового вимкнення та останній повнозв'язний шар з функцією активації softmax. Результати експериментів показали, що запропонована модель досягає точності понад 98% на тестовому наборі даних. Цей результат перевищує точність будь-якого існуючого алгоритму, з яким було порівняно. Проведені додаткові експерименти зі зміною гіперпараметрів моделі, таких як швидкість навчання, розмір пакета, кількість епох та розмір зображень. Ці експерименти підтвердили, що подальше налаштування моделі та збільшення обсягу навчального набору можуть сприяти підвищенню точності моделі.

У розділі 4 проведено функціонально-вартісний аналіз розробленого

програмного продукту для розпізнавання лейкоцитів на основі глибоких згорткових нейронних мереж. Аналіз включав оцінку вартості розробки, прогнозування користувачів і їх потреб, а також економічні вигоди від використання програмного продукту. Результати аналізу підтвердили ефективність та доцільність розробленого програмного продукту, його конкурентоспроможність на ринку та потенційну прибутковість.

Отже, результати експериментів у розділах 2 і 3 підтверджують, що застосування глибоких згорткових нейронних мереж, зокрема моделі DenseNet201, є дуже ефективним підходом до класифікації лейкоцитів на зображеннях крові з високою точністю. Це створює основу для подальшого дослідження та розвитку у цих напрямках, що може сприяти вдосконаленню медичної діагностики та інших областей, де важлива роль відведена розпізнаванню об'єктів. Крім того, функціонально-вартісний аналіз демонструє потенційні переваги і економічну вигідність використання розробленого програмного продукту.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Bain, BJ. A Beginner's Guide to Blood Cells (2nd Edition), 2004. 20 p.
2. Function of White Blood Cells URL: https://my.clevelandclinic.org/health/body/21871-white-blood-cells (дата звернення: 20.04.2023).
3. The Best Texture Features for Leukocytes Recognition URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5691561/ (дата звернення: 20.04.2023).
4. Classifying White Blood Cells Using Machine Learning Algorithms URL: https://dergipark.org.tr/tr/download/article-file/649638 (дата звернення: 20.04.2023).
5. Automated recognition of white blood cells using deep learning URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7438424/ (дата звернення: 21.04.2023).
6. On the Effectiveness of Leukocytes Classification Methods in a Real Application Scenario URL: https://www.mdpi.com/2673-2688/2/3/25 (дата звернення: 27.04.2023).
7. Штучні нейронні мережі: обчислення http://www.immsp.kiev.ua/postgraduate/Biblioteka_trudy/ShtuchnNejronMeregNester2004.pdf (дата звернення: 28.04.2023).
8. Marc Peter Deisenroth. Mathematics for Machine Learning, 2020.
9. Christoph Molnar. Interpretable Machine Learning: A Guide for Making Black Box Models Explainable, 2019. 31 p.
10. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach. Deep Learning (Adaptive Computation and Machine Learning Series), 2017. 60 p.

11. Ragav Venkatesan, Baoxin Li. Convolutional Neural Networks in Visual Computing A Concise Guide, 2018. 28 p.
12. A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (дата звернення: 28.04.2023).
13. A Survey of the Recent Architectures of Deep Convolutional Neural Networks URL: https://arxiv.org/ftp/arxiv/papers/1901/1901.06032.pdf (дата звернення: 2.05.2023).
14. Saban Ozturk. Convolutional Neural Networks for Medical Image Processing Applications, 2023.
15. MobileNetV3 URL: https://paperswithcode.com/method/mobilenetv3 (дата звернення: 2.05.2023).
16. ResNet-50: The Basics and a Quick Tutorial URL: https://datagen.tech/guides/computer-vision/resnet-50/ (дата звернення: 27.04.2023).
17. Review: DenseNet — Dense Convolutional Network (Image Classification) URL: https://towardsdatascience.com/review-densenet-image-classification-b6631a8ef803 (дата звернення: 4.05.2023).
18. A Gentle Introduction to Transfer Learning for Deep Learning URL: https://machinelearningmastery.com/transfer-learning-for-deep-learning/ (дата звернення: 4.05.2023).
19. Transfer Learning Using DenseNet201 URL: https://www.yesilscience.com/transfer-learning-densenet201/ (дата звернення: 7.05.2023).
20. An introduction to object detection with deep learning URl: https://bdtechtalks.com/2021/06/21/object-detection-deep-learning/ (дата звернення: 7.05.2023).

21. Jake VanderPlas. Python Data Science Handbook, 2016.
22. Jan Erik Solem. Programming Computer Vision with Python: Techniques and Libraries for Imaging and Retrieving Information, 2012.
23. Aurelien Geron. Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 2019.
24. Мови програмування машинного навчання URL: https://uk.education-wiki.com/9393948-machine-learning-programming-languages (дата звернення: 7.05.2023).
25. 10 Essential Data Science Packages for Python URL: https://www.kite.com/blog/python/data-science-packages-python/ (дата звернення: 7.05.2023).
26. The Absolute Guide to TensorFlow URL: https://blog.paperspace.com/absolute-guide-to-tensorflow/ (дата звернення: 8.05.2023).
27. OpenCV Tutorial: A Guide to Learn OpenCV URL: https://pyimagesearch.com/2018/07/19/opencv-tutorial-a-guide-to-learn-opencv/ (дата звернення: 27.04.2023).
28. Jupyter Notebook: An Introduction URL: https://realpython.com/jupyter-notebook-introduction/ (дата звернення: 8.05.2023).
29. Convolutional Neural Network (CNN): Graphical Visualization with Python Code Explanation URL: https://www.analyticssteps.com/blogs/convolutional-neural-network-cnn-graphical-visualization-code-explanation (дата звернення: 27.04.2023).
30. Metrics to Use to Evaluate Deep Learning Object Detectors URL: https://www.kdnuggets.com/2020/08/metrics-evaluate-deep-learning-object-detectors.html (дата звернення: 8.05.2023).

ДОДАТОК А ЛІСТИНГ ПРОГРАМНОГО ПРОДУКТУ

```
import os
import random

import numpy as np
import scipy as sp
import pandas as pd
import matplotlib.pyplot as plt

import cv2
import imutils
import tensorflow as tf

from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle
from sklearn.metrics import confusion_matrix



TRAIN_IMAGES_PATH = 'dataset_augmented/images/TRAIN/'
TEST_IMAGES_PATH = 'dataset_augmented/images/TEST/'
TEST_SIMPLE_IMAGES_PATH = 'dataset_augmented/images/TEST_SIMPLE/'
CLASS_NAMES = os.listdir(TRAIN_IMAGES_PATH)
IMAGE_SIZE = (60, 60)
IMAGE_SHAPE = IMAGE_SIZE + (3,)
```

```
def load_image(path):
      image = cv2.imread(path)
      image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
      return image



image_name = random.choice(os.listdir('dataset/images/'))


plt.figure(figsize=(12, 8))
plt.axis('off')


image = load_image(path='dataset/images/'+image_name)


annotations = pd.read_csv('dataset/annotations.csv')
for index, row in annotations[annotations['filename']==image_name].iterrows():
      xmin = row['xmin']
      ymin = row['ymin']
      xmax = row['xmax']
      ymax = row['ymax']


      if row['cell_type'] == 'RBC':
      cv2.rectangle(image, (xmin, ymin), (xmax, ymax), (0, 255, 0), 1)
      cv2.putText(image, 'RBC', (xmin+8, ymin+12), cv2.FONT_HERSHEY_SIMPLEX,
0.4, (0, 255, 0))
      if row['cell_type'] == 'Platelets':
      cv2.rectangle(image, (xmin, ymin), (xmax, ymax), (255, 255, 0), 1)
      cv2.putText(image, 'Platelets', (xmin+8, ymin+12),
cv2.FONT_HERSHEY_SIMPLEX, 0.4, (255, 255, 0))
      if row['cell_type'] == 'WBC':
      cv2.rectangle(image, (xmin, ymin), (xmax, ymax), (255, 0, 0), 1)
      cv2.putText(image, 'WBC', (xmin+8, ymin+12), cv2.FONT_HERSHEY_SIMPLEX,
0.4, (255, 0, 0))
```

```
plt.imshow(image)

plt.show()



images_sample = pd.DataFrame()

sample = random.sample(os.listdir('dataset/images/'), 20)

labels = pd.read_csv('dataset/labels.csv')

class_names = [labels.loc[labels['Image']==int(image[11:16]),
'Category'].values[0] for image in sample]

sample = ['dataset/images/' + i for i in sample]

images_sample['File'] = sample

images_sample['Class'] = class_names



plt.figure(figsize=(12, 9))

for i in range(len(images_sample)):

      plt.subplot(4, 5, i+1)

      plt.axis('off')

      image = load_image(images_sample['File'][i])

      plt.imshow(image)

      plt.title(images_sample['Class'][i])

plt.tight_layout()

plt.show()



labels = pd.read_csv('dataset/labels.csv')

summary = labels['Category'].value_counts()

print(summary)

print()

print(f"Total samples: {summary.sum()}")
```

```
plt.bar(summary.head(5).index.tolist(), summary.head(5).tolist())
plt.show()



images_sample = pd.DataFrame()
for class_name in CLASS_NAMES:
      df = pd.DataFrame()
      sample = random.sample(os.listdir(TRAIN_IMAGES_PATH+class_name), 5)
      sample = [TRAIN_IMAGES_PATH + class_name + '/' + i for i in sample]
      df['File'] = sample
      df['Class'] = class_name
      images_sample = pd.concat([images_sample, df], ignore_index=True)



plt.figure(figsize=(12, 9))
for i in range(len(images_sample)):
      plt.subplot(4, 5, i+1)
      plt.axis('off')
      image = load_image(images_sample['File'][i])
      plt.imshow(image)
      plt.title(images_sample['Class'][i])
plt.tight_layout()
plt.show()



print("Training set")
training_set_summary = pd.Series(
      data=[len(os.listdir(TRAIN_IMAGES_PATH+class_name)) for class_name in
CLASS_NAMES],
```

```
	index=CLASS_NAMES
)
print(training_set_summary)
print(f"Total samples: {training_set_summary.sum()}")
print()

print("Test set")
test_set_summary = pd.Series(
	data=[len(os.listdir(TEST_IMAGES_PATH+class_name)) for class_name in
CLASS_NAMES],
	index=CLASS_NAMES
)
print(test_set_summary)
print(f"Total samples: {test_set_summary.sum()}")


plt.bar(training_set_summary.index.tolist(), training_set_summary.tolist())
plt.show()


def label_code(label):
	code = {
	"EOSINOPHIL": 0,
	"LYMPHOCYTE": 1,
	"MONOCYTE": 2,
	"NEUTROPHIL": 3
	}

	if type(label) == str:
	return code[label]
	else:
```

```
      for key, value in code.items():
            if value == label:
            return key


def find_edges(image):
      gray = cv2.GaussianBlur(image, (1, 1), 0)
      edged_image = cv2.Canny(gray, 100, 400)
      edged_image = cv2.dilate(edged_image, None, iterations=1)
      edged_image = cv2.erode(edged_image, None, iterations=1)
      return edged_image


def get_image_contours(image):
      contours = cv2.findContours(image.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
      contours = imutils.grab_contours(contours)
      contours = sorted(contours, key=lambda x: cv2.contourArea(x))
      return contours


def get_boxes(contours):
      boxes = []
      centers = []
      for contour in contours:
      box = cv2.minAreaRect(contour)
      box = cv2.cv.BoxPoints(box) if imutils.is_cv2() else cv2.boxPoints(box)
      box = np.array(box, dtype='int')
      (tl, tr, br, bl) = box
      if (sp.spatial.distance.euclidean(tl, bl)) > 0 and
(sp.spatial.distance.euclidean(tl, tr)) > 0:
            boxes.append(box)
      return boxes
```

```
def preprocess_image(image):

      # Add padding to the image to better detect cell at the edge

      image = cv2.copyMakeBorder(image, 10, 10, 10, 10, cv2.BORDER_CONSTANT,
value=[198, 203, 208])


      # Thresholding the image to get the target cell

      image_1 = cv2.inRange(image, (80, 80, 180), (180, 170, 245))


      # Opening errosion then dilation

      kernel = np.ones((3, 3), np.uint8)

      kernel_1 = np.ones((5, 5), np.uint8)

      image_erosion = cv2.erode(image_1, kernel, iterations=2)

      image_1 = cv2.dilate(image_erosion, kernel_1, iterations=5)


      # Detecting the blood cell

      edged_image = find_edges(image_1)

      edged_contours = get_image_contours(edged_image)

      edged_boxes = get_boxes(edged_contours)

      if len(edged_boxes) == 0:

      return None


      # Get the large box and get its coordinate

      last = edged_boxes[-1]

      min_x = int(min(last[:, 0]))

      max_x = int(max(last[:, 0]))

      min_y = int(min(last[:, 1]))

      max_y = int(max(last[:, 1]))


      # Draw the contour and fill it

      mask = np.zeros_like(image)
```

```
      cv2.drawContours(mask, edged_contours, len(edged_contours)-1, (255, 255,
255), -1)


      # Any pixel but the pixels inside the contour is zero

      image[mask==0] = 0


      # Extract the blood cell

      image = image[min_y:max_y, min_x:max_x]


      if (np.size(image)==0):

      return None


      # Resize the image

      image = cv2.resize(image, IMAGE_SIZE)


      # Normalization of pixel values in the range from 0.0 to 1.0

      image = image.astype(np.float32) / 255.0


      return image



images = []

labels = []


for class_folder in os.listdir(TRAIN_IMAGES_PATH):

      for file_name in os.listdir(TRAIN_IMAGES_PATH + class_folder):

      image_path = TRAIN_IMAGES_PATH + class_folder + '/' + file_name

      image = load_image(image_path)

      image = preprocess_image(image)

      if image is not None:

            images.append(image)
```

```
            labels.append(label_code(class_folder))


for class_folder in os.listdir(TEST_IMAGES_PATH):
      for file_name in os.listdir(TEST_IMAGES_PATH + class_folder):
      image_path = TEST_IMAGES_PATH + class_folder + '/' + file_name
      image = load_image(image_path)
      image = preprocess_image(image)
      if image is not None:
            images.append(image)
            labels.append(label_code(class_folder))

images = np.array(images)
labels = np.array(labels)

images, labels = shuffle(images, labels)



sample = random.sample(range(len(images)), 20)


plt.figure(figsize=(12, 9))
for i in range(len(sample)):
      plt.subplot(4, 5, i+1)
      plt.axis('off')
      plt.imshow(images[sample[i]])
      plt.title(label_code(labels[sample[i]]))
plt.tight_layout()
plt.show()
```

```
train_images, test_images, train_labels, test_labels = train_test_split(images,
labels, test_size=0.2)

test_images, val_images, test_labels, val_labels =
train_test_split(test_images, test_labels, test_size=0.5)




BATCH_SIZE = 32

LEARNING_RATE = 0.001

EPOCHS = 30




model = tf.keras.models.Sequential([

      tf.keras.layers.Conv2D(64, (3, 3), activation='relu',
input_shape=IMAGE_SHAPE),

      tf.keras.layers.MaxPooling2D(2, 2),

      tf.keras.layers.Dropout(0.25),

      tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),

      tf.keras.layers.MaxPooling2D(2, 2),

      tf.keras.layers.Dropout(0.25),

      tf.keras.layers.Conv2D(256, (3, 3), activation='relu'),

      tf.keras.layers.MaxPooling2D(2, 2),

      tf.keras.layers.Dropout(0.25),

      tf.keras.layers.Flatten(),

      tf.keras.layers.Dense(1024, activation='relu'),

      tf.keras.layers.Dropout(0.25),

      tf.keras.layers.Dense(4, activation='softmax')

])



model.summary()
```

```
tf.keras.utils.plot_model(model, show_shapes=True, expand_nested=True)



model.compile(
      optimizer = tf.keras.optimizers.Adam(learning_rate=LEARNING_RATE),
      loss = tf.keras.losses.SparseCategoricalCrossentropy(),
      metrics = ['accuracy']
)


history = model.fit(
      x=train_images,
      y=train_labels,
      batch_size=BATCH_SIZE,
      epochs=EPOCHS,
      validation_data=(val_images, val_labels)
)



accuracy = history.history['accuracy']
val_accuracy = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']



plt.subplot(2, 1, 1)
plt.plot(accuracy, label='Training Accuracy')
plt.plot(val_accuracy, label='Validation Accuracy')
plt.ylim([min(plt.ylim()), 1])
plt.xlabel('Epoch')
```

```
plt.ylabel('Accuracy')
plt.legend(loc='lower right')

plt.subplot(2, 1, 2)
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.ylim([0, 1])
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(loc='upper right')

plt.show()



y_true = test_labels
predictions = model.predict(test_images)
y_pred = np.argmax(predictions, axis=1)

cm = confusion_matrix(y_true, y_pred)



plt.imshow(cm, cmap=plt.cm.Blues)

plt.title('Confusion matrix')

plt.xlabel('Predicted label')
plt.ylabel('True label')

tick_marks = np.arange(len(CLASS_NAMES))
plt.xticks(tick_marks, CLASS_NAMES, rotation=90)
```

```
plt.yticks(tick_marks, CLASS_NAMES)


thresh = cm.max() / 2.0

for (i, j), n in np.ndenumerate(cm):

      plt.text(j, i, n, horizontalalignment='center', color='white' if cm[i, j]
> thresh else 'black')


plt.show()




images_sample = pd.DataFrame()

for class_name in CLASS_NAMES:

      df = pd.DataFrame()

      sample = random.sample(os.listdir(TEST_SIMPLE_IMAGES_PATH+class_name), 2)

      sample = [TEST_SIMPLE_IMAGES_PATH + class_name + '/' + i for i in sample]

      df['File'] = sample

      df['Class'] = class_name

      images_sample = pd.concat([images_sample, df], ignore_index=True)




images = []


for image_path in images_sample['File']:

      image = load_image(image_path)

      image = preprocess_image(image)

      images.append(image)


images = np.array(images)


predictions = model.predict(images)

predictions = np.argmax(predictions, axis=1)
```

```
predictions = [label_code(label) for label in predictions]


images_sample['Prediction'] = predictions



plt.figure(figsize=(10, 16))
for i in range(len(images_sample)):
      plt.subplot(4, 2, i+1)
      plt.axis('off')
      image = load_image(images_sample['File'][i])
      plt.imshow(image)
      plt.text(4, 16, f"Class: {images_sample['Class'][i]}", fontsize=12)
      plt.text(4, 28, f"Prediction: {images_sample['Prediction'][i]}",
fontsize=12)
plt.tight_layout()
plt.show()



BATCH_SIZE = 32
LEARNING_RATE = 0.001
INITIAL_EPOCHS = 20
FINE_TUNING_EPOCHS = 20



base_model = tf.keras.applications.DenseNet201(
      include_top = False,
      weights = 'imagenet',
      input_tensor = None,
      input_shape = IMAGE_SHAPE,
      pooling = None
)
```

```
base_model.summary()


tf.keras.utils.plot_model(base_model, show_shapes=True, expand_nested=True)


base_model.trainable = True

for layer in base_model.layers:
      if 'conv5' in layer.name:
      layer.trainable = True
      else:
      layer.trainable = False


inputs = tf.keras.Input(shape=IMAGE_SHAPE)
x = base_model(inputs)
x = tf.keras.layers.Flatten()(x)
x = tf.keras.layers.BatchNormalization()(x)
x = tf.keras.layers.Dense(units=256, activation='relu')(x)
x = tf.keras.layers.Dropout(0.7)(x)
x = tf.keras.layers.BatchNormalization()(x)
x = tf.keras.layers.Dense(units=128, activation='relu')(x)
x = tf.keras.layers.Dropout(0.5)(x)
x = tf.keras.layers.Dense(units=64, activation='relu')(x)
x = tf.keras.layers.Dropout(0.3)(x)
x = tf.keras.layers.Dense(units=4, activation='softmax')(x)
outputs = x
```

```
model = tf.keras.Model(inputs, outputs)



model.compile(
      optimizer = tf.keras.optimizers.Adam(learning_rate=LEARNING_RATE),
      loss = tf.keras.losses.SparseCategoricalCrossentropy(),
      metrics = ['accuracy']
)


history = model.fit(
      x = train_images,
      y = train_labels,
      batch_size = BATCH_SIZE,
      epochs = INITIAL_EPOCHS,
      validation_data = (val_images, val_labels)
)



accuracy = history.history['accuracy']
val_accuracy = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']



plt.subplot(2, 1, 1)
plt.plot(accuracy, label='Training Accuracy')
plt.plot(val_accuracy, label='Validation Accuracy')
plt.ylim([min(plt.ylim()), 1])
```

```
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(loc='lower right')

plt.subplot(2, 1, 2)
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.ylim([0, 1])
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(loc='upper right')

plt.show()



for layer in base_model.layers:
      if 'conv4' in layer.name:
      layer.trainable = True



model.compile(
      optimizer = tf.keras.optimizers.Adam(learning_rate=0.1*LEARNING_RATE),
      loss = tf.keras.losses.SparseCategoricalCrossentropy(),
      metrics = ['accuracy']
)
history_fine_tuning = model.fit(
      x = train_images,
      y = train_labels,
      batch_size = BATCH_SIZE,
      epochs = INITIAL_EPOCHS + FINE_TUNING_EPOCHS,
```

```
      initial_epoch = history.epoch[-1] + 1,
      validation_data = (val_images, val_labels)
)



accuracy += history_fine_tuning.history['accuracy']
val_accuracy += history_fine_tuning.history['val_accuracy']

loss += history_fine_tuning.history['loss']
val_loss += history_fine_tuning.history['val_loss']



plt.subplot(2, 1, 1)
plt.plot(accuracy, label='Training Accuracy')
plt.plot(val_accuracy, label='Validation Accuracy')
plt.ylim([min(plt.ylim()), 1])
plt.plot([INITIAL_EPOCHS-1, INITIAL_EPOCHS-1], plt.ylim(), label='Start Fine
Tuning')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(loc='lower right')

plt.subplot(2, 1, 2)
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.ylim([0, 1])
plt.plot([INITIAL_EPOCHS-1, INITIAL_EPOCHS-1], plt.ylim(), label='Start Fine
Tuning')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(loc='upper right')
```

```
plt.show()



y_true = test_labels

predictions = model.predict(test_images)

y_pred = np.argmax(predictions, axis=1)


cm = confusion_matrix(y_true, y_pred)



plt.imshow(cm, cmap=plt.cm.Blues)


plt.title('Confusion matrix')


plt.xlabel('Predicted label')

plt.ylabel('True label')


tick_marks = np.arange(len(CLASS_NAMES))

plt.xticks(tick_marks, CLASS_NAMES, rotation=90)

plt.yticks(tick_marks, CLASS_NAMES)


thresh = cm.max() / 2.0

for (i, j), n in np.ndenumerate(cm):

      plt.text(j, i, n, horizontalalignment='center', color='white' if cm[i, j]
> thresh else 'black')


plt.show()



images_sample = pd.DataFrame()
```

```
for class_name in CLASS_NAMES:
      df = pd.DataFrame()
      sample = random.sample(os.listdir(TEST_SIMPLE_IMAGES_PATH+class_name), 2)
      sample = [TEST_SIMPLE_IMAGES_PATH + class_name + '/' + i for i in sample]
      df['File'] = sample
      df['Class'] = class_name
      images_sample = pd.concat([images_sample, df], ignore_index=True)



images = []


for image_path in images_sample['File']:
      image = load_image(image_path)
      image = preprocess_image(image)
      images.append(image)


images = np.array(images)


predictions = model.predict(images)
predictions = np.argmax(predictions, axis=1)
predictions = [label_code(label) for label in predictions]


images_sample['Prediction'] = predictions



plt.figure(figsize=(10, 16))
for i in range(len(images_sample)):
      plt.subplot(4, 2, i+1)
      plt.axis('off')
      image = load_image(images_sample['File'][i])
```

```
        plt.imshow(image)

        plt.text(4, 16, f"Class: {images_sample['Class'][i]}", fontsize=12)

        plt.text(4, 28, f"Prediction: {images_sample['Prediction'][i]}",
fontsize=12)

plt.tight_layout()

plt.show()
```

ДОДАТОК Б ПРЕЗЕНТАЦІЯ

Розпізнавання лейкоцитів глибокими згортковими нейронними мережами

Заяць Владислав
Керівник: Данилов Валерій Якович

Мета, об’єкт та предмет дослідження

Мета дослідження: Розробити систему розпізнавання лейкоцитів на знімках з мікроскопа та їх класифікації на 4 типи - еозинофіли, лімфоцити, моноцити та нейтрофіли.

Об’єкт дослідження: Набір знімків білих клітин крові організму людини під мікроскопом BCCD.

Предмет дослідження: Система розпізнавання лейкоцитів на основі глибокої згорткової нейронної мережі.

Постановка задачі

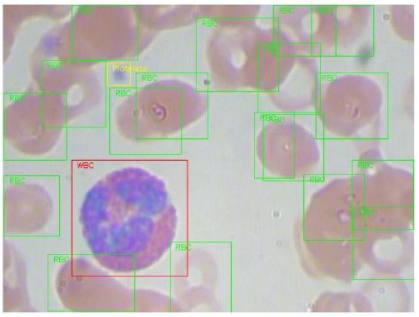
1. Аналіз різних підходів до розпізнавання білих клітин крові
2. Розробка системи, яка зможе розпізнавати типи білих клітин крові на знімках мікроскопа з високою точністю
3. Аналіз результатів

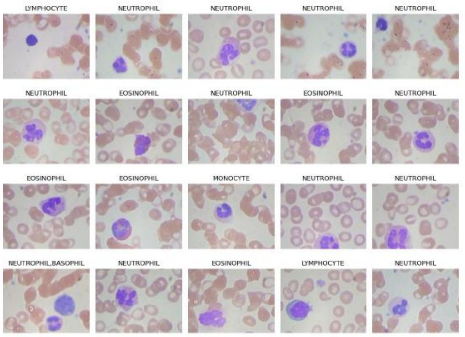
Актуальність

1. **Покращення діагностики та лікування**
   Розробка системи розпізнавання лейкоцитів з використанням нейронних мереж допомагає точніше діагностувати і класифікувати захворювання, зокрема інфекційні хвороби, рак і автоімунні захворювання.
2. **Автоматизація та прискорення процесу**
   Нейронні мережі дозволяють автоматизувати розпізнавання лейкоцитів, що прискорює аналіз крові і забезпечує швидше отримання результатів, зменшуючи необхідність у ручній роботі.
3. **Висока точність і надійність**
   Використання нейронних мереж дозволяє виявляти навіть найменші зміни у структурі лейкоцитів, що забезпечує високу точність і надійність результатів, допомагаючи уникнути помилок і забезпечити належне лікування.

Набір даних

Набір даних, що використовується для розробки моделі розпізнавання білих клітин крові, складається з 12500 зображень крові, які були аугментовані та мають супровідні мітки щодо типу клітини. Набір оснований на оригінальному датасеті BCCD з GitHub.





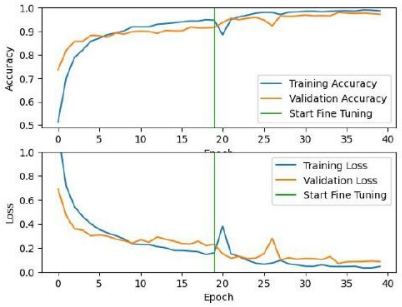
Система розпізнавання

Система працює в 4 кроки:

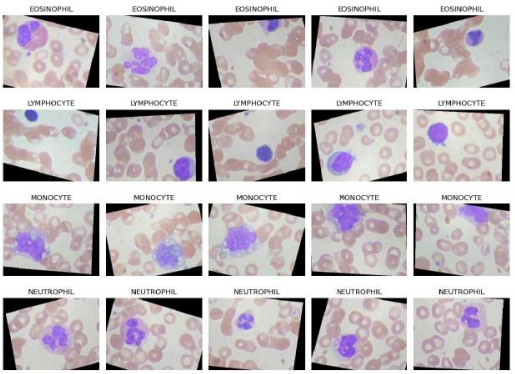
Крок 1	Знімки зразків крові зчитуються з папки і зберігаються в датафреймі.
Крок 2	Файли датафрейму піддаються попередній обробці, включаючи додавання відступу, порогове визначення, ерозію, розширення, виявлення клітини крові, визначення координат області, заповнення контуру, видалення зовнішніх пікселів, витяг клітини, зміну розміру та нормалізацію пікселів.
Крок 3	Для розпізнавання лейкоцитів використовується глибока нейромережа DenseNet201, навчена на великому наборі даних. Зображення проходять через згорткові шари, густі блоки, шари пулінгу та повнозв'язний шар для виявлення та класифікації ознак.
Крок 4	Програма порівнює результати прогнозування з обробленими зображеннями і виводить відповідні результати.

Навчання нейронної мережі

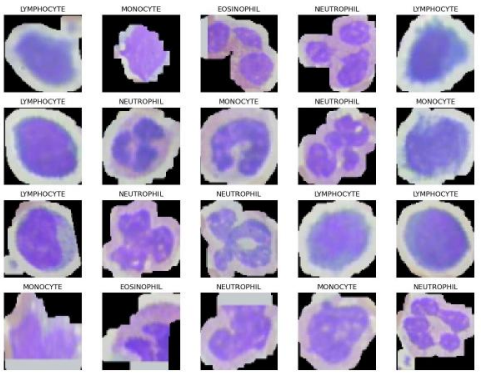
На вхід побудованої на основі DenseNet201 нейронної мережі подаються зображення, до яких була застосована попередня обробка. Навчання відбувається в 2 етапи: на першому блокуються згорткові блоки 1, 2, 3 і 4, на другому для більш точного налаштування вагів мережі розблоковується 4-й блок. Після 2-х етапів точність розпізнавання мережі перевищує 98%.



Аналіз результатів

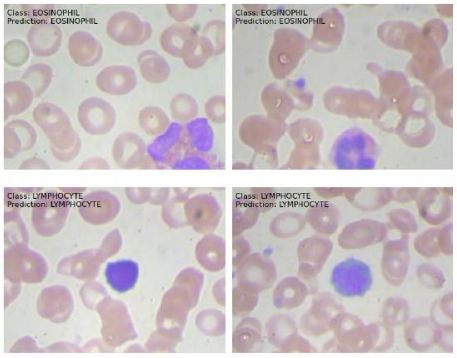


Зображення тестового набору, що подаються на вхід

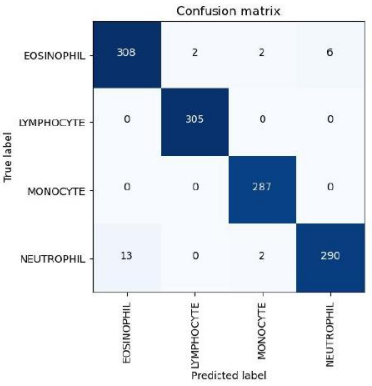


Виділені системою лейкоцити на зображеннях

Аналіз результатів



Результат прогнозування на тестових зображеннях



Confusion matrix для тестового набору

Висновки

В результаті роботи була розроблена система розпізнавання лейкоцитів, точність якої перевищує 98%. Нова технологія має потенціал значно поліпшити медичну діагностику і покращити швидкість та надійність виявлення захворювань, пов'язаних з лейкоцитами. Дана робота вносить вагомий внесок у сферу медичних досліджень та може мати позитивний вплив на практику клінічної медицини, полегшуючи процес лікування та покращуючи прогноз пацієнтів.

Подальше дослідження

Точність розпізнавання системи можна ще підвищити шляхом зменшення швидкості навчання, додавання більшої кількості епох і збільшення розміру зображень після передобробки. Для досягнення ще більшої точності також може сприяти розмороження додаткових слоїв нейронної мережі або застосування мереж з більш складними архітектурами.

Наступним кроком дослідження може стати застосування формули обчислення фрактальної розмірності Хігучі для плоских фігур. Система, яка використовуватиме цей метод теоретично має показати результати ще кращі ніж досягнуті глибокою мережею на основі DenseNet201.

Дякую за увагу!