**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО»**
**Навчально-науковий інститут системного аналізу**
**Кафедра штучного інтелекту**

До захисту допущено:
В.о. завідувач кафедри
_____ Олена ЧУМАЧЕНКО
(підпис)
“___”_____20__ р.

# Дипломна робота

**на здобуття ступеня бакалавра**
**за освітньо-професійною програмою «Системи та методи штучного інтелекту»**
**спеціальності 122 «Комп’ютерні науки»**
**на тему: «Сегментація медичних зображень на основі згорткових нейронних мереж»**

Виконала: студентка 4 курсу, групи КІ-91 _____
Фатнєва Катерина Геннадіївна

Керівник: д.т.н., професор Синєглазов Віктор Михайлович _____

Консультант з економічних питань: к.е.н., доцент _____
Рощина Надія Василівна

Консультант з нормоконтролю:
фахівець першої категорії кафедри ШІ
Гончарук Максим Миколайович _____

Рецензент: к.т.н., доцент Сергеєв Ігор Юрійович _____

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

Київ – 2023 року

**Національний технічний університет України**
**«Київський політехнічний інститут імені Ігоря Сікорського»**
**Навчально-науковий інститут системного аналізу**
**Кафедра штучного інтелекту**

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки»

Освітньо-професійна програма «Системи та методи штучного інтелекту»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Олена ЧУМАЧЕНКО
«___» _____ 2023 р.

**ЗАВДАННЯ**
**на дипломну роботу студентці**
**Фатнєвій Катерині Геннадіївні**

1. Тема роботи «Сегментація медичних зображень на основі згорткових нейронних мереж», керівник роботи Синєглазов Віктор Михайлович, д.т.н., професор, затверджені наказом по університету від «30» травня 2020 р. №2065-с.
2. Термін подання студентом роботи: 15.06.2023.
3. Вихідні дані до роботи: визначення патології хвороби раку легенів.
4. Зміст роботи: повний аналіз зображень у тренувальній вибірці та визначення певної закономірності у кожному з класів хвороби; спроба навчитись виявляти підвид хвороби за допомогою фото ураженої ділянки легенів, прогнозування наявності раку легенів.
5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): Ключові поняття в теорії нейронних мереж, навчання нейронних мереж: види, способи, методи. Ключові архітектури нейронних мереж та сучасні моделі.

Порівняння моделей, дослідження результатів навчання, створення власної моделі та порівняння кінцевих результатів.

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | Завдання видав | Завдання прийняв |
| Економічний | к.е.н., доц. Рощина Надія Василівна | | |

7. Дата видачі завдання _____

Календарний план

| № з/п | Назва етапів виконання дипломної роботи | Термін виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Огляд літератури за темою | 13.04.2023 | Виконано |
| 2 | Підготовка першого розділу | 22.04.2023 | Виконано |
| 3 | Підготовка другого розділу | 01.05.2023 | Виконано |
| 4 | Розробка програмного продукту | 15.05.2023 | Виконано |
| 5 | Підготовка третього розділу | 18.05.2023 | Виконано |
| 6 | Підготовка економічної частини | 30.05.2023 | Виконано |
| 7 | Оформлення розділів відповідно до нормоконтролю | 02.06.2023 | Виконано |
| 8 | Оформлення дипломної роботи | 04.06.2023 | Виконано |
| 9 | Підготовка презентації доповіді | 07.06.2023 | Виконано |

Студентка Катерина ФАТНЄВА

Керівник роботи Віктор СИНЄГЛАЗОВ

# РЕФЕРАТ

Дипломна робота: 101 с., 29 рис., 6 табл., 2 додатки, 19 джерел.

НЕЙРОННІ МЕРЕЖІ, МЕТОДИ ШТУЧНОГО ІНТЕЛЕКТУ, СЕГМЕНТАЦІЯ, КЛАСИФІКАЦІЯ, ДІГНОСТИКА РАКУ ЛЕГЕНІВ.

Об’єктом дослідження є цифрові КТ-зображення органів грудної клітини.

Предметом дослідження є алгоритми сегментації цифрових КТ-зображень органів грудної клітини.

Мета дослідження:

- розробка програмного продукту призначеного для автоматизації розрахунків та полегшення аналізу;
- дослідження існуючих методів та пропозиція власних покращень для цього процесу.

Теоретичною та методологічною основою дослідження є праці вітчизняних і зарубіжних вчених в галузі штучного інтелекту та нейронних мереж.

Результатом дипломної роботи є створення програмного продукту для передбачення хвороби та визначення діагнозу. Програмний продукт реалізовано за допомогою мови програмування Python та фреймворку JupyterLab.

# ABSTRACT

Diploma Thesis (Bachelor's Thesis): 101 p., 29 fig., 6 tabl., 2 annexes, 19 sources.

NEURAL NETWORKS, ARTIFICIAL INTELLIGENCE METHODS, SEGMENTATION, CLASSIFICATION, LUNG CANCER DIAGNOSIS.

The object of the research is digital CT images of organs in the chest.

The subject of the research is algorithms for segmenting digital CT images of organs in the chest.

The research objectives are:

- development of a software product for the automation of calculations and facilitation of analysis.
- study of existing methods and proposing improvements for this process.

The theoretical and methodological basis of the research consists of works by domestic and foreign scientists in the field of artificial intelligence and neural networks.

The outcome of the diploma work is the creation of a software product for disease prediction and diagnosis determination. The software product is implemented using the Python programming language and the JupyterLab framework.

ЗМІСТ

# ВСТУП

Люди роблять неймовірну роботу, коли дивляться на світ. Людина розпізнає об'єкти, навіть якщо у неї є лише одне зображення і вона не має змоги рухатися. Схожий метод, під назвою семантична сегментація, було вирішено застосувати у технологічному світі.

Проблема семантичної сегментації полягає в кластеризації частин зображення, що належать до одного класу об'єктів. Алгоритм класифікації за об'єктами має кілька практичних застосувань в багатьох областях, де потрібно розпізнавати елементи на зображеннях чи відео. Використання сегментації у сфері безпеки значно полегшує повсякденне життя, наприклад, виявлення злочинців на відео з камер спостереження чи розпізнавання номерів автомобілів на відео. У маркетингу алгоритм допомагає з аналізом зображень користувачів соціальних мереж для персоналізованої реклами. Кіноіндустрія значно покращилася в останні роки завдяки створенню неймовірних візуальних ефектів на основі семантичних алгоритмів. Виявлення різноманітних хвороб на рослинах допомагає в аграрному секторі. У сфері екології проводиться моніторинг змін клімату на основі аналізу зображень з супутників. Цей спосіб особливо корисно застосовується у медицині. Найбільш розповсюдженими прикладами є сегментація мозку на МРТ зображеннях та виявлення аномалій тканин і органів.

Розпізнавання об'єктів є завданням класифікації. Тому класи, на яких навчається алгоритм, є ключовим рішенням щодо його розробки [1]. Для досягнення якісних результатів у семантичній сегментації, необхідно відповідно вибирати класи об'єктів, на яких навчається алгоритм. Наприклад, якщо метою сегментації є виявлення пухлин, то відповідні класи повинні включати пікселі, які належать до пухлин і здорової тканини. З іншого боку, якщо метою сегментації є класифікація зображення з різними об'єктами, необхідно вибрати всі можливі класи об'єктів, які можуть бути

присутні на зображенні. Однак, вибір класів об'єктів не є єдиним визначальним чинником для досягнення точної сегментації. Іншими факторами є розмір даних, які використовуються для навчання алгоритму, кількість навчальних прикладів та ефективність використання нейронних мереж та інших алгоритмів машинного навчання. Всі ці фактори повинні бути взяті до уваги при розробці ефективного алгоритму семантичної сегментації [2].

Розподіл зображення на окремі семантичні області є однією з задач комп'ютерного зору, яку можна вирішувати за допомогою нейронних мереж. Зазвичай для семантичної сегментації використовують глибокі нейронні мережі, такі як Convolutional Neural Networks, які зазвичай складаються з декількох шарів. Кожен шар може виконувати різні завдання, наприклад, виділення рис та ознак зображення на попередніх шарах та класифікацію цих ознак на останньому шарі.

Нейронні мережі для семантичної сегментації можуть бути навчені за допомогою набору даних з позначкою, де кожному пікселю зображення призначається певний клас (наприклад, автомобіль, дерево, небо тощо). Після тренування нейронна мережа може бути використана для передбачення класів пікселів на нових зображеннях, тобто для розв'язання задачі семантичної сегментації [4].

Останні роки були виділені багатьма дослідженнями, які покращували результати семантичної сегментації за допомогою нейронних мереж. Наприклад, деякі з цих досліджень використовують більш складні архітектури мережі, включаючи додавання енкодера та декодера або використання рекурентних шарів. Крім того, використовуються різні підходи до передачі інформації між шарами мережі, включаючи пропускну здатність та підвищення роздільної здатності [5].

В даний час нейронні мережі широко використовуються в різних галузях медицини, включаючи пульмонологію. Вони стали незамінним інструментом для діагностики та лікування захворювань легень. Зокрема, нейронні мережі використовуються в електронному аналізі сигналів, обробці рентгенівських знімків та

медичному аналізі зображень. Ці технології значно полегшують виявлення та класифікацію різних захворювань легень, включаючи пневмонію, рак легень та хронічні обструктивні захворювання легень (ХОЗЛ).

Дослідження у цій області базуються на аналізі зображень легенів та розпізнаванні аномалій, що допомагає лікарям у виявленні патологічних змін, таких як пухлини, запалення або зміни в структурі тканини. Нейронні мережі можуть автоматично виділяти та класифікувати ці аномалії, допомагаючи лікарям приймати точні та швидкі рішення щодо діагностики та подальшого лікування. Використання нейронних мереж у медицині є перспективним напрямком, оскільки вони допомагають покращити точність та ефективність діагностики захворювань легень, зменшують час, необхідний для оцінки зображень та сприяють розробці нових методів дослідження та лікування.

Рак легенів є одним з найпоширеніших видів раку по всьому світу. Виявлення цього захворювання на ранніх стадіях грає вирішальну роль у підвищенні шансів на виживання пацієнтів. Проте ручне виявлення раку легенів вимагає високо кваліфікованих фахівців і не завжди є точним через різноманітність його типів та характеристик. Тому важливо розробити автоматизовану систему розпізнавання, яка забезпечує високу точність діагностики та класифікації цього захворювання.

Об’єктом дослідження є цифрові КТ-зображення органів грудної клітини.

Предметом дослідження є алгоритми сегментації цифрових КТ-зображень органів грудної клітини.

Методом дослідження є аналіз різних видів сегментації медичних зображень. Запропонований алгоритм був реалізований з використанням середовища JupyterLab.

Робота складається з чотирьох розділів.

Перший розділ дипломної роботи присвячений аналізу задачі сегментації, ознайомленню із обробкою зображень та основними проблемами, які можуть виникнути при обробці зображень. Проведено докладний аналіз актуальності

створення конкретної системи та розглянуті існуючі підходи до вирішення аналогічних проблем.

У другому розділі детально аналізуються методи навчання нейронних мереж. Крім цього, в цьому розділі проводиться детальний аналіз сучасних нейронних моделей і їх порівняння з метою визначення оптимальної моделі, яка може забезпечити точні результати протягом короткого проміжку часу і при низьких витратах ресурсів.

Третій розділ присвячений опису розробленого програмного продукту, включаючи його функціональні можливості. Надана детальна інструкція для користувача.

У четвертому розділі здійснюється економічно-вартісний аналіз розробленого продукту.

# РОЗДІЛ 1 ПРОБЛЕМА ОБРОБКИ ЗОБРАЖЕНЬ

## 1.1 Штучний інтелект в Україні

Штучний інтелект – одна з найбільш швидкозростаючих галузей в світі, і в Україні. Ця галузь почала розвиватися в 1960-х роках в Україні, коли створювалися перші прототипи систем штучного інтелекту. У 1972 році був створений перший відділ штучного інтелекту при Київському університеті, де працювали видатні вчені, такі як Євген Гнатюк, Олексій Іванченко та Олександр Іванченко.

### 1.1.1 Історія розвитку штучного інтелекту в Україні

У 1980-х роках ШІ в Україні переживав свій розквіт. На початку десятиріччя в Україні було створено відділи штучного інтелекту при більшості університетів, а також декілька наукових інститутів. В Україні було розроблено багато принципово нових методів та алгоритмів машинного навчання, які згодом були успішно використані у багатьох інших країнах.

У 1990-х роках в Україні почалися реформи в галузі науки, що призвело до зменшення фінансування досліджень у галузі ШІ. Проте, незважаючи на це, були створені нові наукові лабораторії та інноваційні підприємства, які займалися розробкою систем штучного інтелекту.

На сьогоднішній день в Україні ШІ продовжує активно розвиватися. У 2017 році була створена Національна стратегія розвитку штучного інтелекту в Україні, яка передбачає створення наукових лабораторій, інноваційних центрів та підприємств, спрямованих на розробку та впровадження штучного інтелекту в різних сферах економіки та соціуму.

### 1.1.2 Кращі практики використання штучного інтелекту в Україні

В останні роки в Україні з'явилось декілька стартапів, які займаються розробкою рішень на базі штучного інтелекту, а також кілька університетів, які викладають спеціальності зі штучного інтелекту та машинного навчання. Окрім того, у країні діють кілька організацій, які об'єднують фахівців зі штучного інтелекту, таких як Ukrainian Association for Artificial Intelligence, AI Ukraine та Data Science Community.

ШІ в Україні поступово розвивається та займає все більш важливе місце в інноваційному секторі. Одним з видатних успіхів українського штучного інтелекту є розробка відомої міжнародної компанії Grammarly, яка розробляє програму для автоматичної перевірки граматики та правопису текстів. Крім того, українські стартапи, такі як SoftServe та N-iX, активно займаються розробкою рішень на базі машинного навчання та інших технологій штучного інтелекту для клієнтів у США та Європі.

Іншим важливим напрямком застосування штучного інтелекту в Україні є медична діагностика та лікування. Використання алгоритмів машинного навчання і нейромереж в медицині значно поліпшує якість діагностики, зменшує кількість помилок та збільшує точність прогнозування. Наприклад, компанія "Інтернат Фармація" використовує інтелектуальну систему аналізу даних, щоб виявляти ознаки розвитку хвороб, зокрема онкологічних захворювань, на ранній стадії. Компанія "Юнілаб" створила систему "дистанційної діагностики", яка дозволяє лікарям швидко та точно діагностувати хвороби шляхом аналізу клінічних даних.

Ще одним напрямком застосування штучного інтелекту в Україні є підтримка прийняття рішень в різних галузях. Наприклад, використання аналітики даних та машинного навчання може допомогти в боротьбі зі злочинністю, управлінні міським транспортом, прогнозуванні погоди та інших важливих сферах. Крім того, ШІ може

бути використаний для автоматизації виробничих процесів та оптимізації ефективності бізнесу.

Війну між Росією та Україною можна розглядати як перший конфлікт, де в значному масштабі використовується програмне забезпечення з ШІ для розпізнавання облич. У березні 2022 року Міністерство оборони України почало використовувати програмне забезпечення для розпізнавання облич, яке було створене американською компанією Clearview AI. Це дозволяє Україні ідентифікувати загиблих військовослужбовців, виявляти російських нападників і боротися з дезінформацією.

Крім того, штучний інтелект грає важливу роль у електронній війні та шифруванні. Наприклад, американська компанія Primer застосувала свої інструменти штучного інтелекту для аналізу незашифрованих радіозв'язків Росії. Це ілюструє, як системи штучного інтелекту постійно переоснащуються та адаптуються.

Менш помітне, але важливе використання штучного інтелекту в Україні полягає в кібервійні. Ранній аналіз, проведений Microsoft, показує, що кіберзахист може виявитись відносно успішним завдяки швидкому розповсюдженню захисного програмного забезпечення до хмарних сервісів та інших комп'ютерних мереж.

Проте водночас, виникають питання щодо етики використання таких технологій та їхнього впливу на людське суспільство. Однак, залишається очевидним факт, що штучний інтелект є важливим інструментом для розвитку України та відкриває широкі можливості для вирішення складних завдань в різних сферах [7].

## 1.2 Проблема сегментації зображень

Глибоке навчання займає центральне місце у сучасних підходах до сегментації зображень. Воно показує високу ефективність у порівнянні з традиційними методами та відкриває нові можливості для точної та швидкої сегментації.

Проблема сегментації зображень полягає в розділенні зображення на окремі сегменти або об'єкти, з метою отримання детальної інформації про їх межі та

структуру. Недавні прориви в глибокому навчанні привели до використання нейронних мереж для розв'язання цієї задачі [9].

### 1.2.1 Огляд проблеми сегментації зображень

Розпізнавання об'єктів є завданням класифікації, тому вибір класів, на яких навчається алгоритм, є ключовим аспектом його розробки [1]. Для досягнення якісних результатів у семантичній сегментації необхідно правильно вибрати класи об'єктів, на яких алгоритм буде навчатися. Наприклад, якщо метою сегментації є виявлення пухлин, то необхідно включити класи, що представляють пікселі, належать до пухлин та здорової тканини. У разі, коли метою є класифікація зображення з різними об'єктами, варто вибрати всі можливі класи об'єктів, які можуть бути присутні на зображенні. Проте вибір класів об'єктів не є єдиним визначальним фактором для досягнення точної сегментації. Розмір даних, використаних для навчання алгоритму, кількість навчальних прикладів та ефективність використання нейронних мереж та інших алгоритмів машинного навчання також впливають на результат. Всі ці фактори повинні бути враховані при розробці ефективного алгоритму семантичної сегментації [2].

Протягом останніх двох років глибокі згорткові мережі перевершили себе в багатьох завданнях візуального розпізнавання. Хоча згорткові мережі існують вже давно, їх успіх був обмеженим через розмір наявних навчальних наборів та розмір розглянутих мереж. Прорив, досягнутий Алексом Крижевським в 2012 році, був завдяки навчанню під наглядом великої мережі з восьма шарами та мільйонами параметрів на наборі даних ImageNet з одним мільйоном зображень для навчання. З того часу були навчені ще більші та глибші мережі. Типове використання згорткових мереж – це завдання класифікації, де вихідним значенням для зображення є один клас. Однак у багатьох візуальних завданнях, особливо в обробці медичних зображень, вихідним значенням має бути локалізація, тобто клас повинен бути призначений для

кожного пікселя. Крім того, тисячі зображень для навчання зазвичай є недосяжними медичних сфері. Таким чином, Ден Сіресан вирішив навчати мережу в режимі ковзаючого вікна, щоб передбачити класові мітки кожного пікселя, надавши як вхід локальну область (патч) навколо цього пікселя. По-перше, ця мережа може локалізувати. По-друге, кількість навчальних даних у вигляді патчів значно перевищує кількість навчальних зображень. Отримана мережа виграла конкурс сегментації EM на ISBI 2012 з великим відривом [3].

### 1.2.2 Вхідні дані

Вибір набору даних для тренування має велике значення для досягнення високої якості сегментації. Розглядання різних наборів даних та їх відповідності до реальних сценаріїв допомагає забезпечити ефективну тренування моделей.

Доступні дані, які можуть використовуватись для виведення сегментації, різняться залежно від застосування. По-перше, це можуть бути відтінки сірого або кольорові. Відтінки сірого зображень часто використовуються в медичній діагностиці, такі як магнітно-резонансна томографія або ультразвукове дослідження, тоді як кольорові фотографії широко поширені.

Наступні дані включають в себе можливість врахування глибини. Тобто RGB-D дані доступні у робототехніці, автономних автомобілях і недавно також у споживчій електроніці. Крім того, є одне зображення, стереозображення та спільна сегментація. Сегментація за одним зображенням є найпоширенішим видом сегментації. Стереозображення можна розглядати як більш природний спосіб сегментації. Спільна сегментація – це задача знаходження однорідної сегментації для кількох зображень. Нарешті, варіації 2D та 3D вхідних даних. Сегментація зображень – це задача 2D сегментації, де найменшою одиницею є піксель. У 3D даних, таких як об'ємні рентгенівські CT-зображення, найменшою одиницею є воксель [1].

### 1.2.3 Послідовність обробки даних

Зазвичай, семантична сегментація виконується за допомогою класифікатора, який працює з фіксованими вхідними особливостями та застосовує підхід з ковзним вікном. Це означає, що класифікатор навчається на зображеннях фіксованого розміру. Потім навчений класифікатор вживається для аналізу прямокутних областей зображень, які називаються вікнами. Хоча класифікатор отримує фрагмент зображення, наприклад, 51×51 піксель оточуючого середовища, він може класифікувати лише центральний піксель або підмножину повного вікна. Ця послідовність сегментації показано на діаграмі нижче (рис. 1.1).

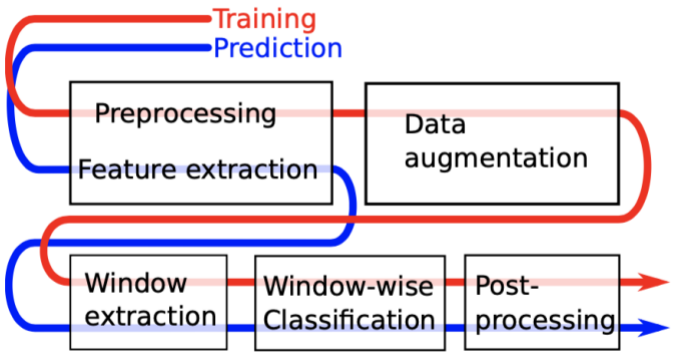



Рисунок 1.1 – Звичайна обробка зображення

Як бачимо, звичайна обробка сегментації отримує необроблені піксельні дані, застосовує техніки попередньої обробки, такі як масштабування, та витягує ознаки, наприклад, ознаки HOG. Під час навчання можуть застосовуватись техніки покращення даних, такі як обертання зображення. Для кожного окремого зображення вилучаються фрагменти зображення, які називаються вікнами, і ці вікна класифікуються. Отриману семантичну сегментацію можна вдосконалити за допомогою простих морфологічних операцій або більш складних підходів, таких як випадкові поля Маркова (MRFs) [1].

### 1.2.4 Традиційні підходи сегментації зображень

Алгоритми сегментації зображень, які використовують традиційні підходи і не застосовують нейронні мережі, і широко поширені в громаді комп'ютерного зору, використовують велику кількість доменних знань.

Нижче описано 7 основних підходів до сегментації зображень:

1. Підхід за кольором пікселів. Кольорові ознаки пікселів в різних кольорових просторах (наприклад, 3 ознаки для RGB, 3 ознаки для HSV, 1 ознака для сірого відтінку) є найбільш поширеними. Зазвичай зображення знаходиться в кольоровому просторі RGB, але в залежності від класифікатора і проблеми, інший кольоровий простір може давати кращі сегментації. Найпоширенішими виборами є RGB і HSI. Вибір RGB обумовлений простотою і підтримкою мов програмування. В той час як вибір простору кольорів HSI може спростити досягнення інваріантності класифікатора до освітлення.
2. Гістограма орієнтованих градієнтів (HOG). Ознаки гістограми орієнтованих градієнтів інтерпретують зображення як дискретну функцію $I$: $\mathbb{N}2 \rightarrow \{0, ..., 255\}$, яка відображає позицію $(x, y)$ в колір. Для кожного пікселя є два градієнти: часткова похідна за $x$ та $y$. Таким чином, вихідне зображення перетворюється на два мапи ознак однакового розміру, які представляють градієнт. Ці мапи ознак розбиваються на фрагменти, і для кожного фрагмента обчислюється гістограма напрямків.
3. Масштабоване відображення ключових точок (SIFT). Масштабоване відображення ключових точок (SIFT) описує ключові точки на зображенні. Береться зображення розміром 16×16 навколо ключової точки. Цей фрагмент розбивається на 16 різних частин розміром 4×4. Для кожної з цих частин обчислюється гістограма з 8 орієнтацій, подібна до ознак HOG. Це

призводить до вектора ознак розміром 128 для кожної ключової точки. Слід зауважити, що SIFT є глобальною ознакою для всього зображення.

4. Мішок з візуальних слов (BOV). Мішок з візуальних слів базується на векторній квантизації. Подібно до ознак HOG, ознаки BOV є гістограмами, які підраховують кількість входжень певних шаблонів у фрагменті зображення.
5. Позелети. Позелети ґрунтуються на ручно доданих додаткових ключових точках, таких як "праве плече", "ліве плече", "праве коліно" і "ліве коліно". Спочатку вони використовувалися для оцінки пози людини. Знаходження цих додаткових ключових точок легко виконується для відомих класів зображень, наприклад, людей. Проте, це складно для класів, таких як органи або клітини, де людські анотатори не знають ключові точки. Крім того, ключові точки повинні бути обрані для кожного окремого класу. Існують стратегії для вирішення цих проблем, такі як ключові точки, залежні від кута огляду.
6. Текстони. Текстони є мінімальними будівельними блоками зору. Текстони посилаються на фундаментальні мікроструктури в природних зображеннях (і відео) і вважаються атомами перед-увагового сприйняття людини. Незважаючи на це, термін "текстон" залишається неоднозначним поняттям в наукових дослідженнях через відсутність чіткої математичної моделі.
7. Зменшення розмірності. Високорозширені зображення мають велику кількість пікселів. Якщо для кожного пікселя використовувати одну або декілька ознак, кількість ознак може перевищувати мільйони. Це ускладнює тренування, тоді як вища роздільна здатність може не містити багато додаткової інформації. Простим підходом для вирішення цієї проблеми є зведення високорозширеного зображення до низькорозширеного варіанта. Інший спосіб зменшення розмірності – це аналіз головних компонентів (PCA). Ідея за PCA полягає в тому, щоб знайти гіперплощину, на яку можна

проектувати всі вектори ознак з мінімальною втратою інформації. Однак однією з проблем PCA є те, що він не розрізняє різні класи. Це означає, що може статися так, що ідеально лінійно роздільна множина векторів ознак стає зовсім нероздільною після застосування PCA. Існує багато інших технік зменшення розмірності [1].

### 1.2.5 Потенційні проблеми в даних для алгоритмів сегментації

Різні робочі процеси сегментації мають різні проблеми. Однак, є кілька випадків, які потрібно перевірити. Ці випадки можуть не часто зустрічатися у тренувальних даних, але все ж можуть відбуватися в робочій системі.

– Відблиск об'єктива (Lens Flare). Відблиск об'єктива – це ефект розсіювання світла в оптичній системі об'єктива камери (рис. 1.2).

Рисунок 1.2 – Відблиск на фотографії

– Винетування (Vignetting). Це ефект затемнення фотографії в кутах. Це може мати багато причин, наприклад, фільтри на камері, що блокують світло в кутах (рис. 1.3).

Рисунок 1.3 – Винетування фотографії

– Розмиті зображення. Фотографії можуть бути розмитими з різних причин: проблеми з механікою об'єктива, фокусуванням на неправильній точці, швидким рухом, димом або піною (рис. 1.4).

Рисунок 1.4 – Розмита фотографія машин на дорозі

– Часткові заслони. Системи сегментації, які використовують модель об'єктів, які повинні бути сегментовані, можуть страждати від часткових заслонів (рис. 1.5).

Рисунок 1.5 – Часткові загородження обличь

– Маскування. Деякі об'єкти, наприклад, тварини в дикій природі, активно намагаються сховатися. В інших випадках це може бути просто поганою удачею, коли об'єкти важко виявити для людей. Ця проблема має два цікаві аспекти. З одного боку, система сегментації може страждати від тих самих проблем, що й люди. З іншого боку, система сегментації може бути кращою, ніж люди, але вона змушена навчатися на зображеннях, позначених людьми. Якщо позначки неправильні, система змушена навчатися неправильно (рис. 1.6).

Рисунок 1.6 – Гепард в траві

– Півпрозорі заслони. Деякі об'єкти, наприклад, склянки для пиття, можуть бути видимими і все ж залишати об'єкт, що знаходиться за ними видимим. (рис. 1.7)

Рисунок 1.7 – Напівпрозора пляшка з водою

– Точки зору. Зміни точок зору можуть становити проблему, якщо вони не відображаються у тренувальних даних. Наприклад, система підписів до зображень, навчена на фотографіях професійних фотографів, може не мати фотографій з точки зору дитини (рис. 1.8) [1].

Рисунок 1.8 – Яблуко з різних ракурсів

Покращення точності сегментації можна досягти за допомогою прийомів переднавчання, додаткових функцій втрати та ансамблювання моделей. Ці підходи дозволяють збільшити стабільність та універсальність моделей для різних типів зображень [9].

## 1.3 Висновки до розділу 1

Отже, у даному розділі проведено огляд різних аспектів штучного інтелекту в Україні, включаючи його розвиток, досягнення та виклики. Розглянули різноманітні застосування штучного інтелекту в різних галузях, включаючи медицину, фінанси, транспорт та інші. Виявилося, що ШІ в Україні знаходиться на стадії активного розвитку та забезпечує розвиток багатьох сфер життя.

Також було розглянуто загальну проблему сегментації зображень, яка виникає при розділенні зображення на окремі об'єкти або регіони. Використання нейромереж виявилося дуже ефективним для розв'язання цієї проблеми, дозволяючи здійснювати точну та автоматизовану сегментацію зображень. Однак, існує також низка викликів, пов'язаних з недостатньою точністю, складністю навчання та обробкою великих обсягів даних.

# РОЗДІЛ 2 ВПРОВАДЖЕННЯ МЕТОДІВ НАВЧАННЯ НЕЙРОННИХ МЕРЕЖ

## 2.1 Нейроні мережі та їх класифікація

Штучні нейронні мережі – це класифікатори, які натхненні біологічними нейронами. Кожен окремий штучний нейрон має кілька вагованих вхідних сигналів, які сумуються. Потім нейрон застосовує так звану функцію активації до зваженої суми і дає вихідний сигнал. Ці нейрони можуть приймати на вхід або вектор ознак, або вихідні сигнали інших нейронів. Таким чином, вони утворюють ієрархію ознак.

Існує багато ідей щодо нейронних мереж, таких як регуляризація, кращі алгоритми оптимізації, автоматичне побудова архітектур, вибір активаційних функцій.

Одна з ключових ідей – це вдалий метод регуляризації, називаний "випадкове відключення" (dropout training), який випадковим чином встановлює вихідні сигнали нейронів на нуль під час тренування. Іншим важливим внеском було використання функції активації, що називається "релі імпульсна функція" (rectified linear unit) [1].

### 2.1.1 Історія розвитку нейронних мереж

Нейронні мережі, також відомі як штучні нейронні мережі (ШНМ), мають багату історію, яка починається з середини 20-го століття. Концепція нейронних мереж була початково натхненна структурою та функціонуванням біологічних нейронних мереж у людському мозку. Основу сучасних нейронних мереж було закладено у 1943 році, коли Воррен МакКаллох та Волтер Піттс запропонували обчислювальну модель нейронної мережі на основі бінарних нейронів. Протягом років у цій галузі було зроблено різні досягнення, включаючи розробку перцептрона Френка Розенблатта в 1957 році, що став одним з перших успішних втілень нейронних

мереж. Однак, нейронні мережі зіткнулися зі значними викликами та обмеженнями в наступні роки, що призвело до зниження зацікавленості [15].

Відродження нейронних мереж настало в 1980-х роках з введенням алгоритму зворотного поширення помилок Геоффрі Хінтоном, Девідом Румельхартом і Рональдом Вільямсом. Алгоритм зворотного поширення помилок революціонізував навчання нейронних мереж, дозволяючи ефективно коригувати ваги з'єднань між нейронами на основі сигналу помилки. Цей прорив прокладав шлях до глибших та більш складних архітектур нейронних мереж [16].

В останні роки галузь нейронних мереж спостерігає небачений розвиток, завдяки прогресу в обчислювальній потужності, великим обсягам даних та інноваціям в алгоритмах. Глибоке навчання, підгалузь нейронних мереж, здобуло значну увагу та досягло помітних успіхів у різних галузях, включаючи комп'ютерне зорове сприйняття, обробку природної мови та розпізнавання мови. Найновітніші моделі глибокого навчання, такі як згорткові нейронні мережі та рекурентні нейронні мережі, досягають рівня людини або навіть перевищують його у багатьох завданнях [17].

### 2.1.2 Огляд нейронних мереж

Нейронні мережі є одним з найпотужніших інструментів у галузі глибокого навчання та машинного навчання. Вони здатні моделювати складні зв'язки та розпізнавати складні патерни в даних. Встановлення зв'язків з мозком та нейронною структурою було ключовим для створення цих моделей [10].

### 2.1.3 Опис нейрона

Нейрон є основною будівельною одиницею в нейронних мережах, і він відіграє ключову роль у виконанні обчислень та розпізнаванні патернів в даних. Нейрон приймає вхідні сигнали, які відповідають вхідним даним, і обчислює зважену суму

цих сигналів з використанням вагових коефіцієнтів. Після обчислення зваженої суми, до неї застосовується нелінійна функція активації, яка визначає вихідний сигнал нейрона. Цей вихідний сигнал може бути переданий іншим нейронам у мережі як вхідний сигнал.

Описаний нейронний процес дозволяє нейронним мережам моделювати складні зв'язки і розпізнавати складні патерни в даних. Розуміння структури та функціонування нейронів є важливим для розвитку та застосування нейронних мереж у різних областях, включаючи комп'ютерне зору, обробку мови, розпізнавання мови та інші [10].

Однією з характеристик нейромереж є значна кількість зв'язків, які з'єднують окремі нейрони. В мозку людини нейрони групуються, щоб обробляти інформацію динамічним, взаємодіючим і самоорганізованим способом. Біологічні нейромережі складаються з мікроскопічних компонентів, розташованих у тривимірному просторі, і мають різноманітні зв'язки. Однак, при створенні штучних мереж є фізичні обмеження. Штучні нейрони, об'єднані в мережу, утворюють систему обробки інформації, яка забезпечує ефективну адаптацію моделі до постійних змін у зовнішньому середовищі. Процес функціонування мережі включає перетворення вхідних сигналів у вихідний вектор. Конкретний тип перетворення залежить від архітектури нейромережі, характеристик нейронів, засобів керування та синхронізації інформаційних потоків між нейронами. Для досягнення ефективності мережі важливо визначити оптимальну кількість нейронів і типи зв'язків, які пов'язують їх.

### 2.1.4 Архітектури нейронних мереж

Існує кілька основних архітектур нейромереж, кожна з яких має своє призначення та застосування. Серед відомих архітектурних рішень є група нейронних мереж зі слабкими зв'язками (рис. 2.1), де кожен нейрон має зв'язки лише зі своїми

сусідніми нейронами. У повнозв'язаних нейромережах (рис. 2.2) входи кожного нейрона пов'язані з виходами всіх інших нейронів.

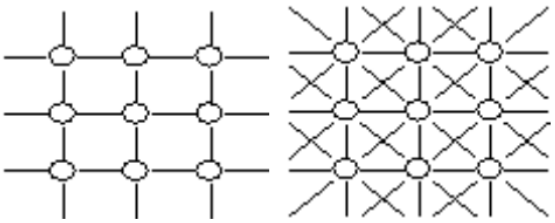


Рисунок 2.1 – Слабозв'язані нейромережі

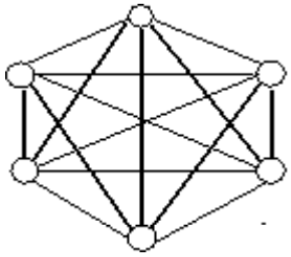


Рисунок 2.2 – Повнозв'язані нейромережі

Feedforward Neural Networks (FNN), також відомі як прямі нейронні мережі, є простою архітектурою нейромережі, в якій інформація передається в одному напрямку від входу до виходу через різні шари. Вони не мають циклів або зворотних зв'язків і можуть складатись з одного або кількох прихованих шарів.

Також наявні згорткові нейронні мережі (CNN), спеціально розроблені для обробки даних з просторовою структурою, таких як зображення. Вони використовують згорткові шари для автоматичного виявлення особливостей зображень, замість ручного інженерного проектування цих особливостей.

Ще одним варіантом архітектури нейромережі є рекурентні нейронні мережі (Recurrent Neural Networks), розроблені для обробки послідовних даних, таких як текст або часові ряди. Вони використовують зворотні зв'язки, що дозволяють зберігати інформацію з попередніх кроків. Це дозволяє їм ефективно працювати з даними, де контекст є важливим фактором.

Довга короткочасна пам'ять (Long Short-Term Memory) є варіацією рекурентної нейромережі, яка розв'язує проблему затухання градієнту, що виникає при навчанні традиційних RNN. LSTM використовує спеціальну структуру вузлів, яка дозволяє моделі "запам'ятовувати" або "забувати" інформацію протягом довгого періоду часу.

Мережі згорткового автоенкодера (Convolutional Autoencoders) представляють собою тип нейромереж, які навчаються стислому кодуванню вхідних даних, а потім відтворювати вихідні дані з цього кодування. Згорткові автоенкодери використовують згорткові шари для обробки просторових даних, таких як зображення, і зазвичай застосовуються для виявлення характеристик або реконструкції зображень.

Мережі генеративно-змагальних моделей (Generative Adversarial Networks) складаються з двох окремих нейромереж, які працюють у взаємодії: генератора, що створює синтетичні дані, та дискримінатора, який навчається розрізняти справжні дані від синтетичних. GAN-и широко використовуються для створення зображень, текстів та інших типів даних.

Мережі капсул (Capsule Networks, CapsNet) представляють собою один з типів згорткових нейромереж, що використовують спеціальні капсульні шари для збереження інформації про просторові взаємозв'язки між об'єктами на зображенні. Вони здатні краще усвідомлювати ієрархічну структуру даних і зазвичай виявляються ефективнішими, ніж традиційні згорткові мережі, у завданнях розпізнавання об'єктів.

Мережі з увагою (Attention Networks) представляють собою архітектуру, яка дозволяє моделям приділяти більше уваги значущим частинам вхідних даних. Механізм уваги зазвичай використовується у поєднанні з рекурентними або трансформаторними мережами, особливо в завданнях обробки природної мови, таких як машинний переклад або генерація тексту.

Спайкові нейронні мережі є класом мереж, що стараються більш точно моделювати динаміку спайкових нейронів біологічного мозку. Вони досягають цього шляхом введення темпоральної складової до активаційних функцій нейронів [11].

## 2.2 Огляд згорткової нейронної мережі для сегментації зображень

Для семантичної сегментації зображень часто використовують глибокі нейронні мережі, такі як Convolutional Neural Networks (CNNs). Такі мережі складаються з декількох шарів, кожен з яких виконує певні завдання. Наприклад, на попередніх шарах вони виділяють риси та ознаки зображення, а на останньому шарі проводять класифікацію цих ознак.

Нейронні мережі для семантичної сегментації можуть бути навчені за допомогою набору даних з позначкою, де кожному пікселю зображення призначається певний клас (наприклад, автомобіль, дерево, небо тощо). Після тренування нейронна мережа може бути використана для передбачення класів пікселів на нових зображеннях, тобто для розв'язання задачі семантичної сегментації [4].

Останні роки були виділені багатьма дослідженнями, які покращують результати семантичної сегментації за допомогою нейронних мереж. Наприклад, в деяких з цих досліджень використовуються більш складні архітектури мережі, які можуть включати енкодери та декодери або використовувати рекурентні шари. Крім того, використовуються різні підходи до передачі інформації між шарами мережі, такі як пропускна здатність та підвищення роздільної здатності [5].

Протягом останніх двох років глибокі згорткові мережі перевершили себе в багатьох завданнях візуального розпізнавання. Хоча згорткові мережі існують вже давно, їх успіх був обмеженим через розмір наявних навчальних наборів та розмір розглянутих мереж. Прорив, досягнутий Алексом Крижевським в 2012 році, був завдяки навчанню під наглядом великої мережі з восьма шарами та мільйонами параметрів на наборі даних ImageNet з одним мільйоном зображень для навчання. З того часу були навчені ще більші та глибші мережі. Типове використання згорткових мереж – це завдання класифікації, де вихідним значенням для зображення є один клас. Однак у багатьох візуальних завданнях, особливо в обробці медичних зображень,

вихідним значенням має бути локалізація, тобто клас повинен бути призначений для кожного пікселя. Крім того, тисячі зображень для навчання зазвичай є недосяжними медичних сфері. Таким чином, Ден Сіресан вирішив навчати мережу в режимі ковзаючого вікна, щоб передбачити класові мітки кожного пікселя, надавши як вхід локальну область (патч) навколо цього пікселя. По-перше, ця мережа може локалізувати. По-друге, кількість навчальних даних у вигляді патчів значно перевищує кількість навчальних зображень. Отримана мережа виграла конкурс сегментації EM на ISBI 2012 з великим відривом [3].

### 2.2.1 Модель архітектури згорткової нейронної мережі

Архітектура CNN складається з трьох основних шарів: шару згортки, шару пулінгу та повнозв'язаного шару. Може бути кілька шарів згортки та пулінгу. Чим більше шарів у мережі, тим більша складність і точність моделі машинного навчання. Кожен додатковий шар, який обробляє вхідні дані, збільшує здатність моделі розпізнавати об'єкти та патерни в даних [18].

### 2.2.2 Шар згортки

Шари згортки є основним будівельним блоком мережі, де відбувається більшість обчислень. Вони працюють, застосовуючи фільтр до вхідних даних для виявлення особливостей. Цей фільтр, відомий як виявлювач особливостей, перевіряє рецептивні поля зображення вхідних даних на наявність певної особливості. Ця операція називається згорткою.

Фільтр – це двовимірний масив вагів, який представляє частину двовимірного зображення. Фільтр зазвичай є матрицею розміром 3×3, хоча є інші можливі розміри. Фільтр застосовується до області всередині вхідного зображення і обчислює скалярний добуток між пікселями, який подається на вихідний масив. Фільтр

зсувається і повторює цей процес до тих пір, поки не охопить усе зображення. Кінцевим результатом всіх процесів фільтра є карта ознак.

Зазвичай CNN застосовує перетворення ReLU (Rectified Linear Unit) до кожної карти ознак після кожної згортки, щоб ввести нелінійність до моделі машинного навчання. Після шару згортки зазвичай слідує шар пулінгу. Разом шари згортки та пулінгу утворюють згортковий блок.

До першого блоку будуть додані додаткові згорткові блоки, створюючи ієрархічну структуру, де пізніші шари навчаються на основі ранніх шарів. Наприклад, модель CNN може навчатися виявляти автомобілі на зображеннях. Автомобілі можна розглядати як суму їх складових, включаючи колеса, багажник та лобове скло. Кожна ознака автомобіля відповідає низькорівневому патерну, виявленому нейронною мережею, яка потім поєднує ці частини, щоб створити високорівневий патерн [18].

### 2.2.3 Шари пулінгу

Шари пулінгу, також відомі як шари зниження розмірності, дозволяють зменшити розмір вхідних даних. У порівнянні з операцією згортки, пулінгові операції використовують фільтр для просканування всього вхідного зображення, проте без використання ваг. Фільтр застосовує агрегаційну функцію для заповнення вихідного масиву на основі значень рецептивного поля.

Існують два основних типи пулінгу:

1. Середній пулінг (Average pooling): фільтр обчислює середнє значення рецептивного поля під час сканування вхідних даних.
2. Максимальний пулінг (Max pooling): фільтр вибирає піксель з найбільшим значенням для заповнення вихідного масиву. Цей підхід є більш поширеним, ніж середнє пулінг.

Шари пулінгу є важливими, оскільки вони допомагають зменшити складність і підвищити ефективність згорткових мереж, незважаючи на те, що в

процесі може втрачатися деяка інформація. Крім того, вони допомагають зменшити ризик перенавчання моделі [18].

### 2.2.4 Повнозв'язаний шар

Останнім шаром є повнозв'язаний шар. Повнозв'язаний шар виконує завдання класифікації, використовуючи особливості, які були виявлені попередніми шарами та фільтрами. У протилежність до функцій ReLU, в повнозв'язаному шарі зазвичай використовується функція softmax, яка здійснює більш адекватну класифікацію вхідних даних і генерує ймовірнісний результат у межах від 0 до 1 [18].

### 2.2.5 Типи алгоритмів згорткової нейронної мережі

LeNet – це інноваційна згорткова нейронна мережа (CNN), спеціально розроблена для розпізнавання рукописних символів. Авторами її створення є Янн ЛеКун, Леон Боту, Йошуа Бенджіо та Патрік Гаффнер, які пропонували цю модель наприкінці 1990-х років. LeNet складається з послідовних шарів згортки та пулінгу, а також повнозв'язаного шару та softmax-класифікатора. Ця нейронна мережа була одним із перших успішних застосувань глибокого навчання в галузі комп'ютерного зору. Банки використовували її для розпізнавання чисел, написаних на чеках, у зображеннях з відтінками сірого кольору (рис. 2.3).

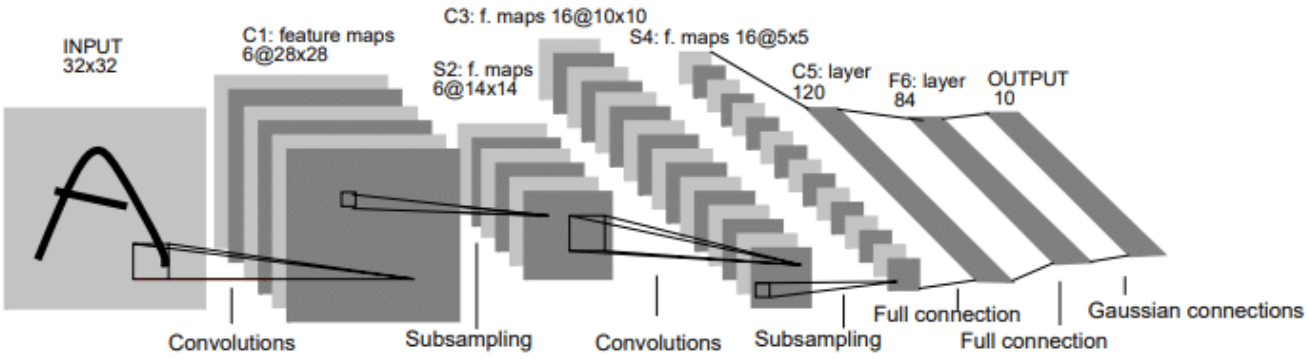



Рисунок 2.3 – Архітектура LeNet-5

Наступним алгоритмом є AlexNet. Він отримав перемогу в змаганні ImageNet Large Scale Visual Recognition Challenge (ILSVRC) у 2012 році, продемонструвавши значне зменшення помилок порівняно з попередніми найкращими моделями – з 26% до 15%. Алгоритм був розроблений Алексом Кріжевським, Ілєю Суцкевером і Джеффрі Хінтоном в Університеті Торонто. Архітектура AlexNet складається з п'яти згорткових шарів з максимальним пулінгом та трьох повнозв'язаних шарів з активаційною функцією ReLU. Крім фільтрів розміром 3×3, використовуються також згорткові фільтри розміром 5×5 та 11×11. Мережа використовує метод відключення (dropout) для зменшення перенавчання та використовує методи аугментації даних для збільшення ефективного обсягу тренувального набору. Успіх AlexNet продемонстрував потужність методів глибокого навчання в галузі комп'ютерного зору та призвів до бурхливого інтересу до цієї галузі. Він також популяризував використання графічних процесорів (GPU) для прискорення тренування глибоких нейронних мереж (рис. 2.4).

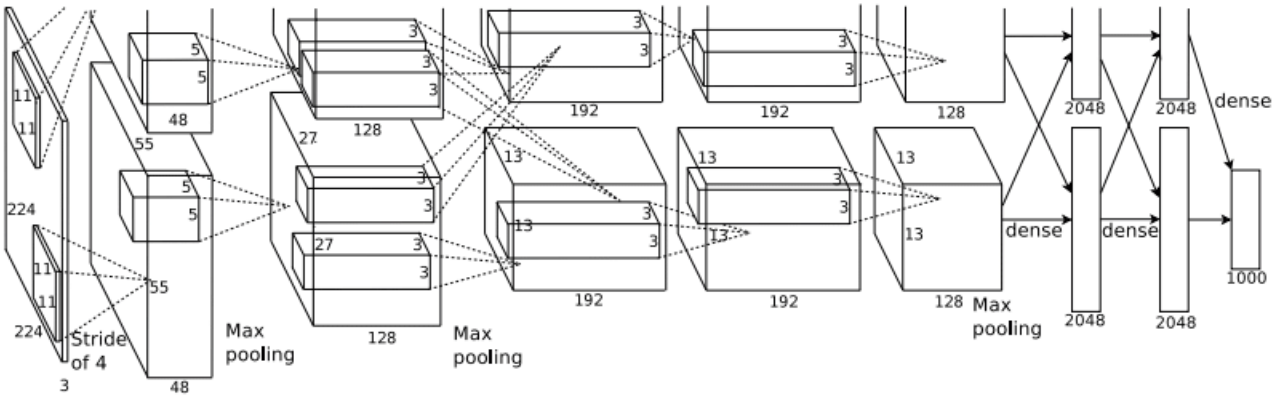



Рисунок 2.4 – Архітектура AlexNet

MobileNet – це інше сімейство згорткових нейронних мереж, спеціально розроблених для мобільних і вбудованих пристроїв (рис. 2.5). Вони оптимізовані для ефективності і швидкості, при цьому досягають високої точності в завданнях, таких як розпізнавання зображень. Однією з ключових особливостей MobileNet є

використання точкових (pointwise) та глибинних (depthwise) згорток. Глибинні згортки використовують один фільтр для кожного вхідного каналу незалежно, після чого йде точкова згортка, яка комбінує вихідні канали від глибинних згорток за допомогою згорток розміром 1×1. Цей підхід значно зменшує кількість обчислень та параметрів, не втрачаючи при цьому значущих особливостей вхідних даних. Це робить моделі MobileNet набагато меншими та швидшими, ніж традиційні згорткові нейронні мережі, що робить їх ідеальними для використання в обмежених ресурсах середовищах, таких як смартфони, дрони та пристрої Інтернету речей (IoT).

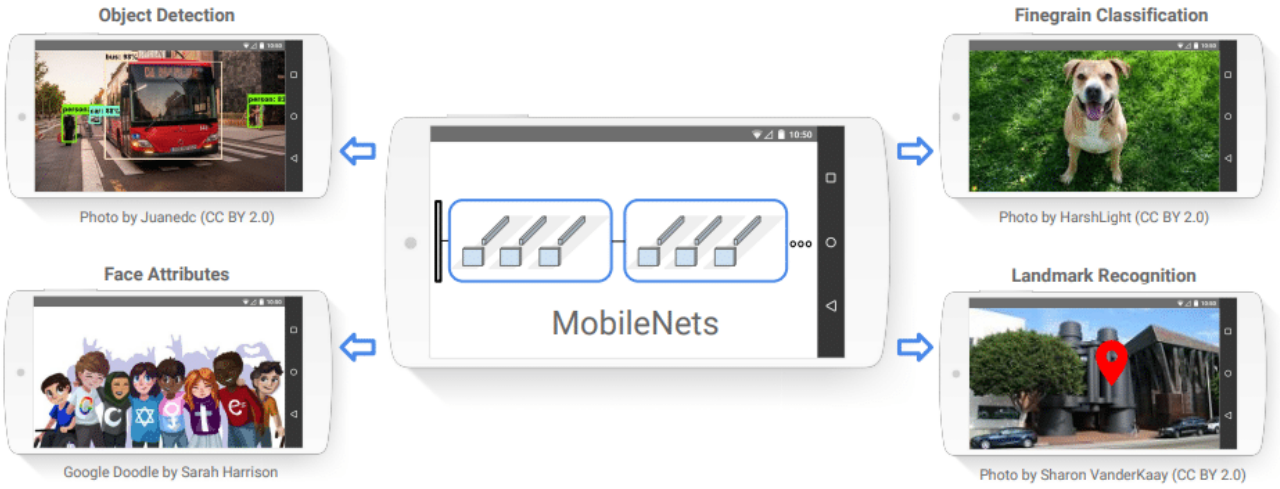



Рисунок 2.5 – Модель MobileNet

R-CNN (Region-based Convolutional Neural Network) був представлений у 2013 році і є двохетапним підходом до виявлення об'єктів. Спочатку вхідне зображення сегментується на області інтересу (ROIs) за допомогою алгоритму вибіркового пошуку, який генерує пропозиції областей на основі текстури, кольору та інших ознак зображення. Потім, кожна ROI проходить через згорткову нейронну мережу (CNN), яка генерує вектор ознак фіксованої довжини. Зазвичай у R-CNN використовується попередньо навчена мережа, така як AlexNet. Ці вектори ознак використовуються для класифікації об'єкту, що міститься в ROI, і передбачення його координат обмежувальної рамки. Незважаючи на те, що R-CNN досяг значних результатів на

момент свого запровадження, навчання та тестування було дуже повільним через необхідність обробки тисяч пропозицій областей для кожного зображення. R-CNN є частиною серії моделей виявлення об'єктів, разом з Fast R-CNN і Faster R-CNN, які були розроблені дослідниками з Університету Каліфорнії в Берклі та Microsoft Research і представляють суттєве вдосконалення порівняно з попередніми методами виявлення об'єктів (рис 2.6).

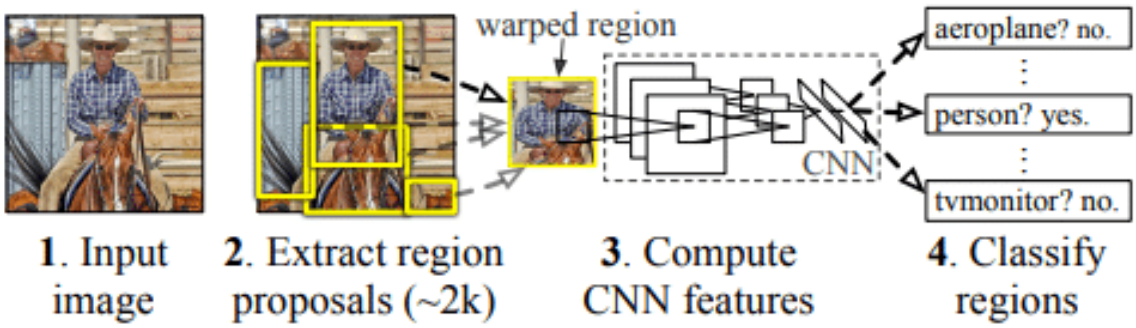



Рисунок 2.6 – Модель R-CNN

Fast R-CNN, представлений у 2015 році, вдосконалив R-CNN, усунувши необхідність окремого кроку вилучення ознак для кожної пропозиції області. Замість цього, вся картинка проходить через згорткову нейронну мережу (CNN), що генерує карти ознак, які використовуються всіма пропозиціями областей. Кожна ROI потім пулінгується в карту ознак фіксованого розміру за допомогою операції під назвою "Пулінг регіону інтересу (ROI Pooling)", і проходить через послідовність повнозв'язаних шарів для отримання кінцевих прогнозів обмежувальних рамок та класифікації. Цей підхід є набагато швидшим і точнішим, ніж R-CNN, і також дозволяє тренування всієї моделі від початку до кінця (end-to-end training) (рис. 2.7).

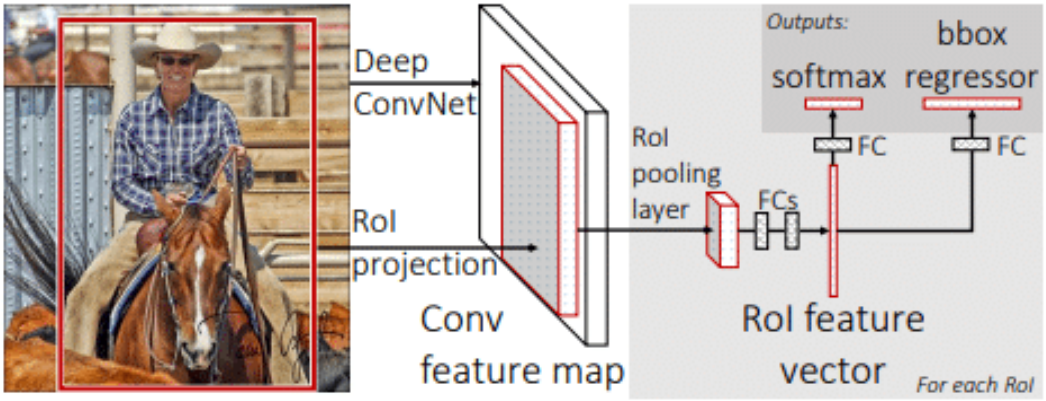



Рисунок 2.7 – Архітектура Fast R-CNN

Faster R-CNN, представлений у 2015 році, ще більше вдосконалив Fast R-CNN, замінивши вибірковий пошук на RPN (Region Proposal Network), яка є невеликою згортковою мережею, що генерує пропозиції областей безпосередньо з карти ознак, яку створює CNN (рис. 2.8). RPN використовує ті ж згорткові шари, що й мережа виявлення об'єктів, що робить усю модель більш ефективною та легшою в тренуванні. RPN передбачає оцінки об'єктності та координати обмежувальної рамки для кожної якорної точки на карті ознак, а ці пропозиції потім вдосконалюються та класифікуються мережею виявлення об'єктів. Цей підхід є ще швидшим і точнішим, ніж Fast R-CNN, і став стандартом для виявлення об'єктів у багатьох застосуваннях.

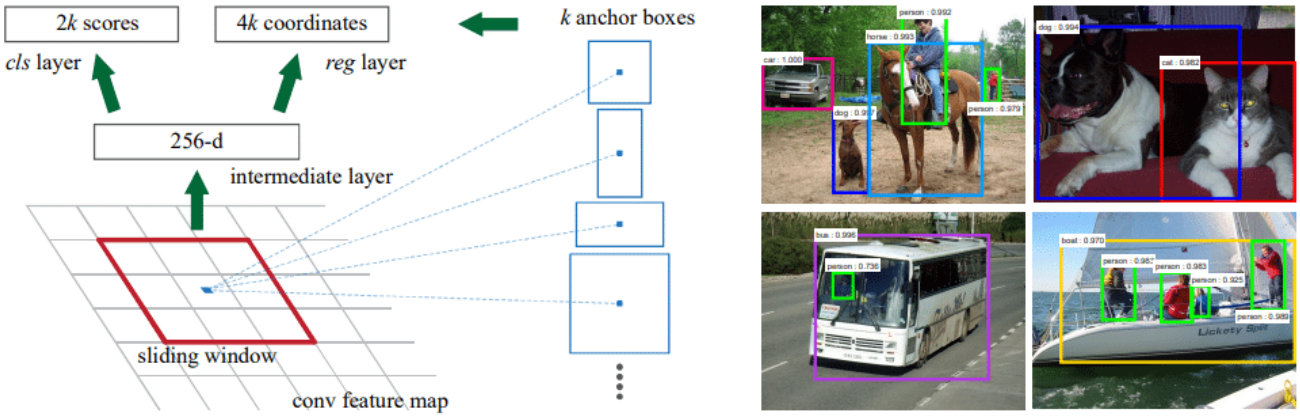



Рисунок 2.8 – Модель Faster R-CNN

Також існує модель VGGNet. VGG (Visual Geometry Group) – це дослідницька група в межах Відділу інженерних наук університету Оксфорд. Група VGG відома своєю роботою в галузі комп'ютерного зору, зокрема в області згорткових нейронних мереж (CNN). Одним з найвідоміших внесків від групи VGG є модель VGG, також відома як VGGNet. Модель VGG – це глибока нейронна мережа, яка досягла передових результатів у виклику ImageNet Large Scale Visual Recognition Challenge у 2014 році і широко використовується як основа для класифікації зображень та виявлення об'єктів.

Модель VGG характеризується використанням невеликих згорткових фільтрів (3×3) та глибокою архітектурою (до 19 шарів), що дозволяє їй вчитися все складніших ознак з вхідних зображень. Модель VGG також використовує шари максимального пулінгу для зменшення просторового розміру карт ознак і збільшення поле зору, що може покращити її здатність розпізнавати об'єкти різних масштабів та орієнтацій. Дана модель надихнула багато наступних дослідницьких робіт у глибокому навчанні, включаючи розробку ще глибших нейронних мереж та використання залишкових з'єднань для покращення потоку градієнтів та стабільності тренування [18].

## 2.3 Види медичних зображень

Медичні зображення можна класифікувати на дві основні групи незалежно від способу їх отримання, з урахуванням їх різноманітності:

- аналогове зображення;
- цифрове (матричне) зображення.

Стосовно аналогових зображень можна вказати ті, що містять неперервну інформацію. Наприклад, це можуть бути зображення на звичайних рентгенограмах або термограмах. Аналогові сигнали є безперервними сигналами, які часто містять зайву інформацію.

Цифрові (матричні) зображення, натомість, створюються за допомогою комп'ютера. Вони базуються на матриці (растрі), яка зберігається в пам'яті комп'ютера. Растр складається з великої кількості пікселів або, у випадку тривимірного зображення, вокселів. Чим більше пікселів міститься у растрі, тим вища якість зображення. Однак, обробка цих зображень може спричиняти деформацію, зокрема, зміну розміру. Це може призводити до зернистості та втрати деталей у зображенні. В рентгенології такі явища можуть спостерігатися при спробах створити паперові копії цифрових флюорографій та комп'ютерних томограм.

Створення матричного зображення відбувається шляхом електронного сканування по рядках. Це надає можливість отримувати зображення в реальному часі. Для відтворення зображення використовується вбудований дисплейний процесор, який з'єднаний з основним комп'ютером за допомогою зв'язкового інтерфейсу. Це дозволяє використовувати всю площу екрана дисплея як матрицю, що складається з пікселів. Щоб отримати більшу роздільну здатність на системі відображення, необхідно розбити площу дисплея на більшу кількість пікселів (рис. 2.9).

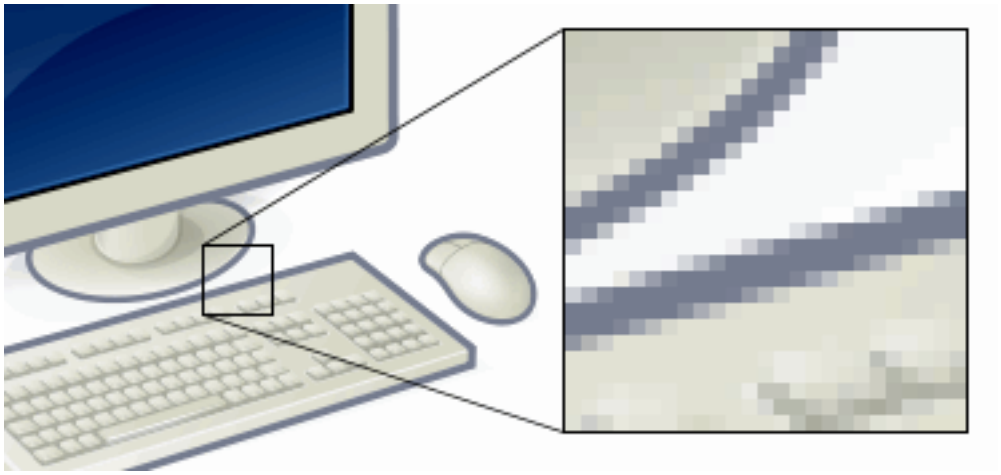


Рисунок 2.9 – Зображення в піксельному вигляді

У матричних зображеннях кожному елементу матриці відповідає окрема область пам'яті, яка може бути доступна. У медичній діагностиці, екранна площа дисплея зазвичай представлена у вигляді різних матриць, таких як 64×64, 128×128,

256×256, 512×512, 1024×1024, 2048×2048, 4096×4096 пікселів. Збільшення розміру матриці сприяє більш детальному зображенню. Однак, зі зростанням якості зображення також збільшується обсяг адресованої пам'яті. Це вимагає додаткових апаратних і програмних ресурсів. Тому на практиці обирають оптимальний розмір матриці, який забезпечує баланс між якістю та продуктивністю. Наприклад, в радіонуклідній візуалізації (ПЕТ), яка в основному передає функціональну інформацію, використовуються великі матриці, такі як 128×128 і 256×256. Таким чином, це віддаляє оперативну пам'ять комп'ютера від необхідності займатися відображенням зображення, дозволяючи їй вільно виконувати складні параметричні розрахунки та побудови [13].

У комп'ютерній та магнітно-резонансній томографії, цифровій рентгенографії, які вирішують переважно структурні завдання діагностики, використовуються більш детальні матриці, наприклад, 512×512, 1024×1024. Крім розміру матриці, самі растрові зображення можуть мати різну структуру пікселів. Кожен піксель зображення, як відомо, представляється в пам'яті за допомогою різної кількості бітів, від 1 до 24. Збільшення кількості бітів, що представляють кожен піксель зображення, призводить до більшої зорової якості, але вимагає більше обчислювальних ресурсів. Тому в променевій діагностиці застосовуються різні глибини пікселів. Наприклад, ультразвукова діагностика, яка в основному вирішує функціональні завдання або розпізнавання грубих морфологічних структур, частіше використовує 6-бітні пікселі, що представляють 64 відтінки сірого або 8-бітні пікселі з 256 відтінками сірої шкали. Для створення таких зображень в пам'яті комп'ютера потрібно від 1 до 5 МБ пам'яті [14].

При побудові об'ємних зображень та створенні 4-вимірної графіки потрібно залучати ще більше ресурсів. Наприклад, для виконання одного дослідження з використанням потокових та тривимірних файлів у кольоровій шкалі кодування даних сучасним комп'ютерним томографом може знадобитися до 5 ГБ оперативної пам'яті комп'ютера.

У деяких системах, які використовуються для отримання медичних зображень, використовується структура побудови та інтерпретації у воксельному (об'ємному) форматі (рис. 2.10). Вокселі є стандартним інструментом для візуалізації та аналізу медичних та наукових даних. Крім цього, ця технологія застосовується в деяких прогресивних програмах тривимірного моделювання і комп'ютерних іграх для створення складних зображень. Аналогічно до пікселів, вокселі зазвичай не містять інформації про своє просторове розташування. Замість цього, координати вокселів визначаються на основі їх розташування у структурі даних, що утворює одне цілісне просторове зображення.

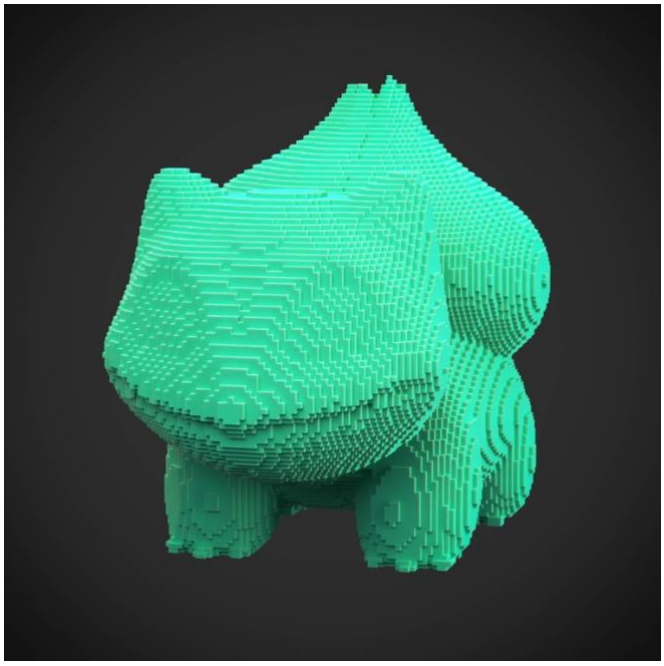


Рисунок 2.10 – Воксельне зображення

Зображення медичних об'єктів можна також поділити на наступні категорії:

- тверді фрагменти, такі як кістки, які мають стабільну форму;
- деформовані фрагменти, які включають структури м'якої тканини, які можуть змінювати свою форму;
- статичні фрагменти, наприклад череп, які залишаються незмінними протягом часу;

- динамічні фрагменти, такі як серце або рухомі з'єднання, які мають змінну форму або рухаються.

Розмірність воксельної матриці відповідає розмірності піксельної матриці, наприклад, 256×256, 512×512 і так далі. У кольорових зображеннях використовується трибайтний піксель, що може кодувати 16,7 мільйонів кольорів [14]. Однак, такі зображення вимагають більшого обсягу пам'яті комп'ютера. Тому для зображень часто використовується індексований колір, який містить 256 кольорів. Це має свої переваги, такі як менший обсяг пам'яті, швидше та простіше передавання через лінії зв'язку. Важливо також зазначити, що аналогові зображення можуть бути перетворені на матричні, а матричні зображення – на аналогові. Для досягнення цього використовують спеціальні пристрої, такі як АЦП (аналогово-цифрові перетворювачі) та ЦАП (цифро-аналогові перетворювачі).

## 2.4 Висновки до розділу 2

У даному розділі було розглянуто визначення нейронних мереж, різноманітні архітектури нейромереж та детально розглянуто згорткову нейронну мережу, яка часто використовується для семантичної сегментації зображень. Також було оглянуто види медичних даних.

Можна зробити висновок, що нейромережі дозволяють вирішувати складні проблеми, адаптуючись до незвичайних та нелінійних даних та аналізуючи їх у глибину. Це забезпечує здатність систем з нейромережами до розв'язання задач, які традиційними методами можуть бути важкими або неможливими. Крім того, нейромережі відрізняються багатогранністю своїх застосувань, оскільки вони можуть ефективно працювати з різними типами даних та адаптуватися до різних задач. Ця універсальність робить їх потужним інструментом для розв'язання різноманітних завдань у різних галузях.

Загалом, нейромережі стали необхідним елементом сучасних технологій, що відкривають нові можливості та дозволяють розв'язувати складні задачі шляхом адаптації та навчання.

# РОЗДІЛ 3 АНАЛІЗ АРХІТЕКТУРИ ТА ІНТЕРФЕЙСУ ПРОГРАМНОГО ПРОДУКТУ

## 3.1 Загальний огляд програмного продукту

Під час роботи був створений та втілений програмний продукт, що сприяє прогнозуванню раку легенів – хвороби, що вражає легені, а також виявленню типу хвороби. Програмний продукт є потужним інструментом для сегментації КТ-зображень легенів та аналізу їх стану за допомогою штучного інтелекту. Він використовує передові методи згорткових нейронних мереж для автоматичного розпізнавання та виділення легеневих областей на зображеннях.

Основні функції програмного продукту включають:

1. Сегментація легеневих областей: система використовує згорткові нейронні мережі для автоматичної сегментації легеневих тканин на КТ-зображеннях. Це дозволяє точно виділяти легені та відокремлювати їх від інших структур.
2. Аналіз стану легенів: після сегментації система проводить аналіз стану легенів. Вона використовує зібрані дані про форму, розмір, густину та інші характеристики легеневих областей для визначення потенційних ознак хвороби. Це допомагає виявляти патологічні зміни, такі як припухлість, маси, виразки або пухирі, які можуть свідчити про захворювання легенів.
3. Класифікація хвороб: система також здатна класифікувати хвороби легенів на основі аналізу отриманих ознак. За допомогою навчених моделей машинного навчання вона може визначати, чи є патологічні зміни позитивними ознаками конкретної хвороби, такої як рак легенів, пневмонія або інші захворювання. Це може допомогти лікарям і фахівцям здійснити швидку та точну діагностику.

4. Візуалізація результатів: програмний продукт надає зручні інтерфейси для відображення сегментованих легеневих областей та результатів аналізу. Візуалізація може включати позначення виявлених патологічних змін, що допомагає лікарям швидко оцінити стан легенів пацієнта.

Програмний продукт є потужним інструментом для автоматизації процесу сегментації КТ-зображень легенів та аналізу їх стану. Він може забезпечити швидку, точну та об'єктивну оцінку легеневих патологій, що полегшує роботу медичних фахівців та покращує діагностичні процедури.

## 3.2 Вибір мови програмування та фреймворків

Вибір мови програмування та фреймворку є важливими аспектами при розробці програмного продукту, особливо у сфері штучного інтелекту. Одним із найпопулярніших виборів для розробки програм, пов'язаних із штучним інтелектом, є мова програмування Python та фреймворк JupyterLab.

Python є однією з найпопулярніших мов програмування у сфері штучного інтелекту та машинного навчання. Вона має чистий та простий синтаксис, що робить її дуже зручною для розробки і розуміння коду. Python також має велику кількість сторонніх бібліотек та пакетів, таких як TensorFlow, PyTorch, Keras та інші, які надають потужні інструменти для розробки штучного інтелекту.

JupyterLab – це інтерактивне середовище для розробки програмного забезпечення, яке надає зручний спосіб створення та виконання коду, візуалізацію даних та взаємодію з результатами. Це робить його чудовим інструментом для експериментування, демонстрації та спільної роботи над проектами штучного інтелекту. JupyterLab підтримує Python та інші мови програмування, тому його можна використовувати для розробки різноманітних проектів.

Якщо виникають помилки або потрібно відлагоджувати код, Python та JupyterLab надають зручні інструменти для налагодження. Вони дозволяють зупиняти

виконання коду, перевіряти значення змінних та аналізувати помилки, що допомагає знайти й усунути проблеми у розробці.

Python та JupyterLab мають велику та активну спільноту розробників. Це означає, що є багато документації, питань та відповідей, туторіалів та онлайн-курсів, які можна використовувати для вивчення та розвитку. Якщо виникають питання або проблеми, є багато ресурсів, де можна знайти допомогу.

Таким чином, використання Python та JupyterLab у розробці програми штучного інтелекту, що сегментує КТ-зображення легенів, є вигідним вибором. Вони надають потужні інструменти, легкість використання та налагодження, а також доступ до великої спільноти та ресурсів.

## 3.3 Перелік використаних бібліотек та модулів

У даній програмі використовуються наступні бібліотеки та модулі:

- os: ця бібліотека надає функції для взаємодії з операційною системою. У даному випадку, вона використовується для роботи з файлами та шляхами, зокрема для зчитування та запису зображень;
- numpy: це потужна бібліотека для обчислення чисел у вигляді багатовимірних масивів або матриць. У даному випадку, вона використовується для роботи з масивами зображень та їх обробки перед подачею на модель;
- matplotlib.pyplot: ця бібліотека використовується для візуалізації даних та побудови графіків. У коді вона використовується для відображення деяких зображень та їх відповідних класів;
- tensorflow: це одна з найпопулярніших відкритих бібліотек для машинного навчання та глибокого навчання. У даному випадку, вона використовується для створення та навчання згорткової нейронної мережі для сегментації КТ-зображень легенів;

- tensorflow.keras.layers: цей модуль містить різні шари нейронної мережі, такі як Conv2D (шар згортки), MaxPooling2D (шар пулінгу), Dense (повнозв'язний шар) та інші. У коді використовуються ці шари для побудови архітектури згорткової нейронної мережі;
- sklearn.model_selection.train_test_split: ця функція з бібліотеки scikit-learn використовується для розбиття набору даних на тренувальний та валідаційний набори. У даному випадку, вона застосовується для розділення набору зображень та відповідних класів на тренувальні та валідаційні набори;
- PIL.Image: цей модуль надає функціональність для роботи з зображеннями. У коді він використовується для завантаження зображень, перетворення їх у відтінки сірого та зміни розміру;
- warnings: цей модуль використовується для керування попередженнями. У коді він використовується для приховування попереджень, які можуть виникати під час виконання програми.

Ці бібліотеки та модулі допомагають зчитувати дані, побудовувати та навчати нейронну мережу на основі моделі згорткової нейронної мережі, проводити нормалізацію зображень, завантажувати та обробляти дані тренувального, валідаційного та тестового наборів, а також виводити прогнози моделі та порівнювати їх з реальними класами.

## 3.4 Аналіз та попередня обробка даних для дослідження

Аналіз та попередня обробка даних у даному дослідженні відіграють важливу роль у підготовці даних для моделі машинного навчання, яка реалізує програму штучного інтелекту для сегментації КТ-зображень легенів та аналізу виявлення хвороби.

Зображення мають формат JPG або PNG, щоб відповідати моделі. Дані містять 3 типи раку легень: аденокарцинома (adenocarcinoma), великоклітинний карцинома (large cell carcinoma) та плоскоклітинний карцинома (squamous cell carcinoma), а також 1 папку для нормальних клітин (normal).

Папка "dataset" є головною папкою, яка містить усі підпапки. У цій папці розташовані підпапки "test" і "train". "Test" представляє набір для тестування моделі. "Train" представляє набір для навчання моделі (рис. 3.1).

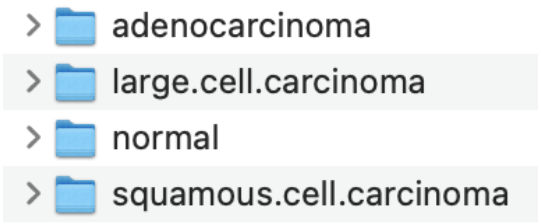



Рисунок 3.1 – Файлова структура папки test

Аденокарцинома легенів є найпоширенішою формою раку легень, що становить близько 30% всіх випадків раку легень і приблизно 40% всіх випадків недрібноклатинного раку легень. Діагностується у людей, які ніколи не курять або курять мало. Аденокарциноми зустрічаються в декількох типових раках органів, включаючи рак молочної залози, передміхурову залозу та товсту кишку. Аденокарциноми легенів розташовані в зовнішній частині легенів у залозах, які виділяють слиз і допомагають людям дихати. Симптоми включають кашель, хрипоту, втрату ваги та слабкість.

Великоклітинна недиференційована карцинома – ще один вид раку легенів, який швидко зростає і поширюється, і може бути знайдений в будь-якій частині легенів. Цей тип раку легень зазвичай становить 10-15% усіх випадків NSCLC (недрібноклатинного раку легенів). Він характеризується великими неправильними клітинами, які можуть мати різні структури та форми. Часто вони розташовуються в центральній частині легенів, але також можуть зустрічатися в периферійних ділянках. Великоклітинна недиференційована карцинома є агресивним типом раку, який

швидко поширюється та може давати вторинні ураження інших органів. Симптоми можуть включати кашель, задишку, болі у грудях, кровохаркання, втрату ваги, втому та інші ознаки, пов'язані з захворюванням легень. Важливо зазначити, що кожен випадок цього раку легенів є унікальним, і прогноз та результати лікування можуть варіюватися в залежності від багатьох факторів.

Плоскоклітинна карцинома – цей тип раку легень знаходиться у центральній частині легенів, де більші бронхи з'єднуються з трахеєю до легенів або в одній з головних гілок дихальних шляхів. Плоскоклітинний рак легень відповідає приблизно за 30% всіх випадків недрібноклатинного раку легень і, як правило, пов'язаний з курінням. Симптоми плоскоклітинної карциноми легенів можуть включати постійний кашель, кров'яний мокротиння, хрипіння, біль у грудях, проблеми з диханням та втрату ваги. Як і більшість типів раку легень, плоскоклітинна карцинома може поширюватися на інші частини тіла, такі як лімфатичні вузли, печінка, кістки або мозок [19].

Остання папка містить зображення нормальних КТ-сканувань легенів.

У процесі аналізу даних розглядаються основні характеристики набору даних, такі як розмір, тип та формат зображень. Також проводиться оцінка розподілу класів, які визначають стан легень – хворі чи здорові. Це допомагає зрозуміти розподіл даних та баланс між класами, що може впливати на налаштування моделі та процес навчання.

В даному програмному продукті проводяться наступні кроки аналізу та попередньої обробки даних:

1. Зчитування та обробка зображень:
   - Зображення завантажуються з відповідних папок за допомогою модулю os.
   - Зображення перетворюються в відтінки сірого за допомогою методу convert('L') з модулю PIL.Image.

- Розмір зображень змінюється на задану висоту та ширину за допомогою методу resize() з модулю PIL.Image.
- Зображення перетворюються в масиви numpy за допомогою np.array().

2. Кодування класів:
   - Класи зображень представлені у вигляді рядків з назвами папок.
   - Функція classes_to_numbers() перетворює рядки класів на цілі числа, що відповідають їхньому порядковому номеру в заданому списку класів.
   - Функція numbers_to_classes() виконує зворотню операцію, перетворюючи цілі числа назад на рядки класів.
3. Нормалізація зображень:
   - Зображення нормалізуються шляхом поділу на 255.0, щоб значення пікселів були у діапазоні від 0 до 1. Це допомагає стабілізувати навчання моделі та поліпшити її продуктивність.
4. Розбиття даних на тренувальний та валідаційний набори:
   - Зображення та відповідні класи розділяються на тренувальний та валідаційний набори за допомогою функції train_test_split() з модулю sklearn.model_selection. В даному випадку, 80% даних використовуються для тренування, а 20% – для валідації.

Попередня обробка даних включає кілька важливих етапів. Спочатку, зображення (рис. 3.2) конвертуються в відтінки сірого, що спрощує подальшу обробку та аналіз. Потім зображення змінюють розмір на задані значення висоти та ширини, щоб забезпечити однаковий розмір вхідних даних для моделі. Після цього дані нормалізуються, поділивши значення пікселів на 255.0, що перетворює їх на діапазон від 0 до 1. Це допомагає стабілізувати процес навчання та покращити продуктивність моделі.

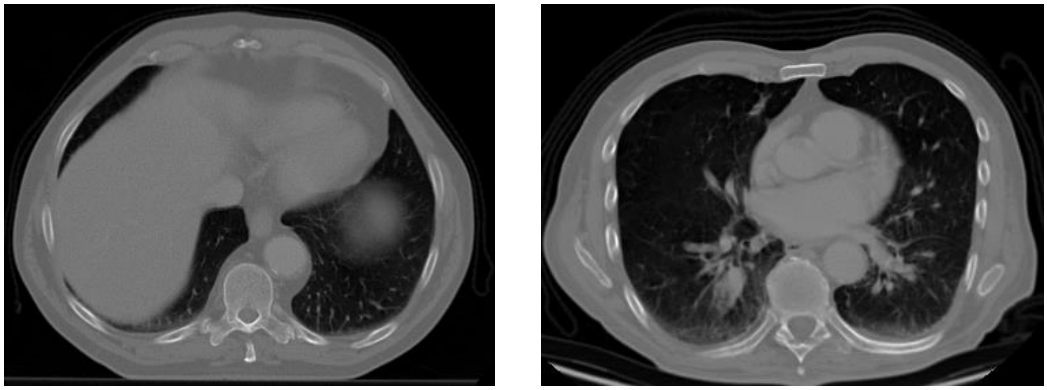


Рисунок 3.2 – Приклад зображень для тренування

Наступний крок – розбиття даних на тренувальний та валідаційний набори. Зображення та відповідні класи розділяються за допомогою функції train_test_split(). В результаті отримуємо тренувальні дані, які будуть використовуватися для навчання моделі, та валідаційні дані, які допоможуть оцінити продуктивність моделі під час навчання.

Також у коді присутні функції, які конвертують текстові класи в числові значення та навпаки. Це допомагає відображати класи в числовому форматі, який використовується для навчання моделі та визначення прогнозів.

Усі ці кроки аналізу та попередньої обробки даних допомагають забезпечити належну підготовку даних перед навчанням моделі, що покращує її точність та ефективність у виконанні завдання сегментації КТ-зображень легенів та аналізу наявності хвороби.

## 3.5 Архітектура системи

Архітектура даної системи, що реалізує програму штучного інтелекту для сегментації КТ-зображень легенів та аналізу виявлення хвороби, базується на згорткових нейронних мережах (Convolutional Neural Networks).

Головною складовою архітектури є згортковий шар (Convolutional Layer), який відповідає за виявлення локальних особливостей у зображеннях. У програмі присутні

три згорткових шари з різними параметрами, такими як кількість фільтрів і розмір ядра. Кожен шар використовує функцію активації ReLU для нелінійної активації.

Після кожного згорткового шару використовується шар пулінгу (Pooling Layer), який зменшує розмір зображення та кількість параметрів, сприяючи виявленню більш загальних ознак.

Далі використовується шар розгортання (Flatten Layer), який перетворює вихід згорткових шарів в одновимірний вектор для подальшої обробки шаром повнозв'язних шарів (Dense Layers). У програмі є два таких шари з різною кількістю нейронів та функцією активації ReLU.

Заключним шаром є повнозв'язний шар з кількістю нейронів, що відповідає кількості класів. Для задачі класифікації використовується функція активації softmax, яка дозволяє отримати ймовірності належності зображення до кожного класу.

Усі ці шари складаються разом у модель, яка створюється за допомогою фреймворку TensorFlow та бібліотеки Keras. Модель компілюється з використанням оптимізатора Adam і функції втрати sparse_categorical_crossentropy, яка підходить для задачі багатокласової класифікації. Така архітектура згорткової нейронної мережі дозволяє автоматично виявляти важливі ознаки у зображеннях, що допомагає ефективно сегментувати КТ-зображення легенів та визначати, чи присутня хвороба.

## 3.6 Інтерфейс програми

Інтерфейс програми реалізований з використанням фреймворку JupyterLab. За допомогою JupyterLab користувач може працювати з програмою штучного інтелекту, що сегментує КТ-зображення легенів. Він може відкривати та редагувати програмний код в Jupyter ноутбуках, які містять усі використані функції та методи для обробки та аналізу даних (рис. 3.3).

[2]:
```
# set the path to the dataset folders
data_folder = './dataset'

# image specs
image_height = 256
image_width = 256
```

[3]:
```
# create the CNN model
def create_model(input_shape, num_classes):
    model = tf.keras.Sequential()

    model.add(tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=input_shape))
    model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))

    model.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu'))
    model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))

    model.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu'))
    model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))

    model.add(tf.keras.layers.Flatten())

    model.add(tf.keras.layers.Dense(64, activation='relu'))
    model.add(tf.keras.layers.Dense(num_classes, activation='softmax'))

    return model
```

Рисунок 3.3 – Приклад програмного коду

Користувач може виконувати окремі комірки коду, переглядати результати, включаючи візуалізацію зображень, графіків та метрик моделі. Він також може змінювати параметри моделі, налаштовувати гіперпараметри та повторно навчати модель для поліпшення результатів.

JupyterLab надає зручну робочу область з багаторядковим текстом для пояснень та документації, а також можливість додавання роздільників для відділення різних частин програми. Користувач може легко навігувати по коду, вставляти нові комірки для додавання нового функціоналу або аналізу даних, а також зберігати результати роботи виконання коду.

В цьому інтерфейсі програми користувач може відобразити прогнози моделі для тестових зображень, порівняти їх зі справжніми класами та загальною характеристикою (доброякісна або злоякісна) класифікованої легені. Він може також

проводити додатковий аналіз результатів та вносити зміни у код для експериментів та поліпшення моделі.

## 3.7 Практичні результати

Результати роботи даної програми полягають у сегментації КТ-зображень легенів та класифікації їх як хворих або здорових. За допомогою згорткової нейронної мережі, навченої на попередньо оброблених даних, програма може давати прогнози щодо стану легенів на основі вхідних зображень.

Результати представлені у вигляді прогнозованого класу для кожного тестового зображення, а також узагальненого характеру, що вказує, чи присутня злоякісна патологія у легенях. Користувач може оцінити якість роботи програми, порівняючи прогнози зі справжніми класами. Це дозволяє визначити точність, чутливість та специфічність моделі. Також можна провести аналіз неправильно класифікованих зображень для подальшого вдосконалення моделі.

Результати візуалізовані та представлені у зручному форматі, що дозволяє користувачеві швидко оцінити ефективність програми та зробити висновки про стан легенів на основі оброблених КТ-зображень (рис. 3.4).

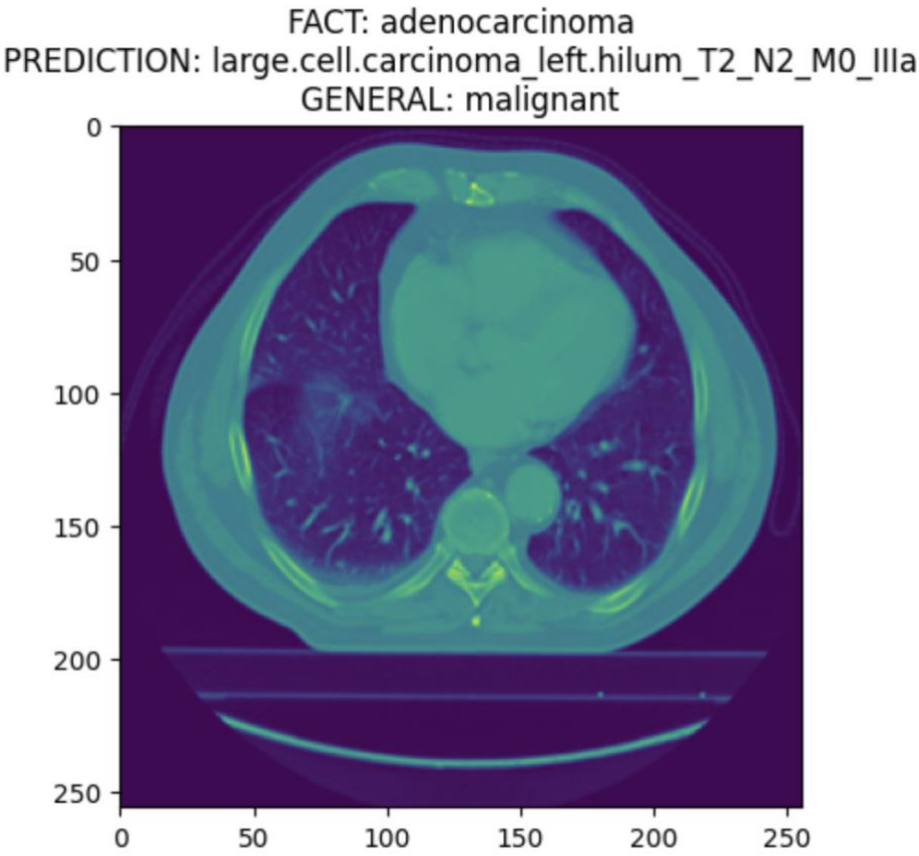



Рисунок 3.4 – Графічний результат роботи програми

Виконання програми показало, що у даному КТ-зображені присутня злоякісна патологія легенів, а саме Аденокарцинома легенів. Програма класифікувала дане зображення як злоякісне, що є вірним. Проте помилка виникла при класифікації захворювання. Точність – 0.9449 (рис. 3.5).

```
accuracy: 0.3184

accuracy: 0.5857

accuracy: 0.7939

accuracy: 0.8837

accuracy: 0.9449
```

Рисунок 3.5 – Точність результатів

Також дана програма реалізує текстові результати (рис. 3.6). На рисунку видно 5 рандомних прикладів. Кожний індекс це є окреме КТ-зображення. Як бачимо, програма майже правильно розрізняє скани легенів. Проте, як і в попередньому випадку, помилка виникла у класифікуванні злоякісної пухлини (індекс: 232).

```
INDEX: 137
FACT: large.cell.carcinoma
PREDICTION: large.cell.carcinoma_left.hilum_T2_N2_M0_IIIa
GENERAL: malignant

INDEX: 314
FACT: normal
PREDICTION: normal
GENERAL: benign

INDEX: 232
FACT: adenocarcinoma
PREDICTION: large.cell.carcinoma_left.hilum_T2_N2_M0_IIIa
GENERAL: malignant

INDEX: 138
FACT: large.cell.carcinoma
PREDICTION: large.cell.carcinoma_left.hilum_T2_N2_M0_IIIa
GENERAL: malignant

INDEX: 105
FACT: large.cell.carcinoma
PREDICTION: large.cell.carcinoma_left.hilum_T2_N2_M0_IIIa
GENERAL: malignant
```

Рисунок 3.6 – Текстовий результат роботи програми

Загалом, результати роботи даної програми можуть бути використані для автоматизації процесу аналізу КТ-зображень легенів та допомогти лікарям у виявленні патологій та прийнятті важливих медичних рішень.

## 3.8 Висновки до розділу 3

У цьому розділі було проведено аналіз архітектури та інтерфейсу даного програмного продукту. Було детально розглянута архітектура системи, включаючи використовувані бібліотеки та модулі, процес аналізу та попередньої обробки даних, архітектуру системи, інтерфейс програми та були проаналізовані результати роботи програми.

Описана архітектура системи базується на використанні фреймворка TensorFlow та бібліотеки Keras для розробки та навчання моделі глибокого навчання. Була створена згорткова нейромережа з кількома шари, яка здатна класифікувати зображення на основі попередньо навчених даних. Для цього було використано різні шари, такі як згорткові, пулінгу, повністю з'єднані шари та функції активації.

Також було описано інтерфейс програми, що використовує фреймворк JupyterLab. За допомогою JupyterLab можна виконувати код, відображати результати та взаємодіяти з програмою. Використання JupyterLab дозволяє зручно аналізувати дані, навчати модель, виконувати прогнози та візуалізувати результати.

Таким чином, дана програма має добре організовану архітектуру, яка дозволяє ефективно обробляти та аналізувати дані про рак легенів. Інтерфейс програми забезпечує зручну роботу з програмою та взаємодію з результатами. Ця програма може бути корисною для аналізу та класифікації зображень, що допоможе в ранньому виявленні та лікуванні раку легенів.

У майбутньому можна розглянути наступні можливості для вдосконалення даної програми:

- розширення датасету: можна зібрати більше даних про рак легенів, включаючи різні типи раку та їх різноманітні клінічні характеристики. Це дозволить покращити точність класифікації та розпізнавання патологій;
- вдосконалення моделі: можна експериментувати з різними архітектурами нейромереж та параметрами моделі для поліпшення її точності та швидкодії. Також можна розглянути використання передових алгоритмів машинного навчання та глибокого навчання;
- додаткові функції аналізу: розглянути включення додаткових функцій аналізу зображень, таких як сегментація, виявлення аномалій та відстеження змін патологій на зображеннях з часом. Це може допомогти виявити та відстежувати рак легенів на ранніх стадіях;

- вдосконалення інтерфейсу: поліпшити інтерфейс програми для зручної взаємодії користувача з програмою, включаючи можливість завантаження власних зображень для класифікації та візуалізацію результатів;
- валідація та перевірка: провести додаткову валідацію та перевірку результатів програми, порівняти їх зі стандартними клінічними протоколами та експертними думками, щоб забезпечити надійність та точність програми;

Вдосконалення програми в цих напрямках може покращити її ефективність та корисність для клінічних досліджень, медичних спеціалістів та пацієнтів.

# РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

В заданому розділі буде проведено оцінювання основних характеристик для майбутнього програмного продукту, що спеціалізується на дослідженні демографічного стану.

Дана реалізація буде сприяти проведенню усіх необхідних досліджень, що дасть змогу якісно дослідити питання не лише в Україні, проте у всьому світі.

Також в даному дослідженні показано різні варіанти реалізації для забезпечення найбільш коректної та оптимальної стратегії вибору, що має вплив на економічні фактори та сумісність з майбутнім програмним продуктом. Для цього застосовувався апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) передбачає собою технологію, що дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. ФВА проводиться з метою виявлення резервів зниження витрат за рахунок ефективніших варіантів виробництва, кращого співвідношення між споживчою вартістю виробу та витратами на його виготовлення. Для проведення аналізу використовується економічна, технічна та конструкторська інформація.

Алгоритм функціонально-вартісного аналізу включає в себе визначення послідовності етапів розробки продукту, визначення повних витрат (річних) та кількості робочих часів, визначення джерел витрат та кінцевий розрахунок вартості програмного продукту.

## 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 програмного продукту, який вирішує проблему сегментації цифрових КТ-зображень органів грудної клітини і розпізнає захворювання, таке як рак легенів. Беручи за основу цю функцію, можна виділити наступні:

$F_1$ – вибір самої програми;

$F_2$ – якісний аналіз даних;

$F_3$ – вибір середовища розробки.

Кожна з цих функцій має декілька варіантів реалізації:

Функція $F_1$:

а) Python;

б) C++.

Функція $F_2$.

а) Застосування вбудованих функцій;

б) Створення своїх обчислень значень.

Функція $F_3$:

а) JupyterLab;

б) Visual Studio.

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

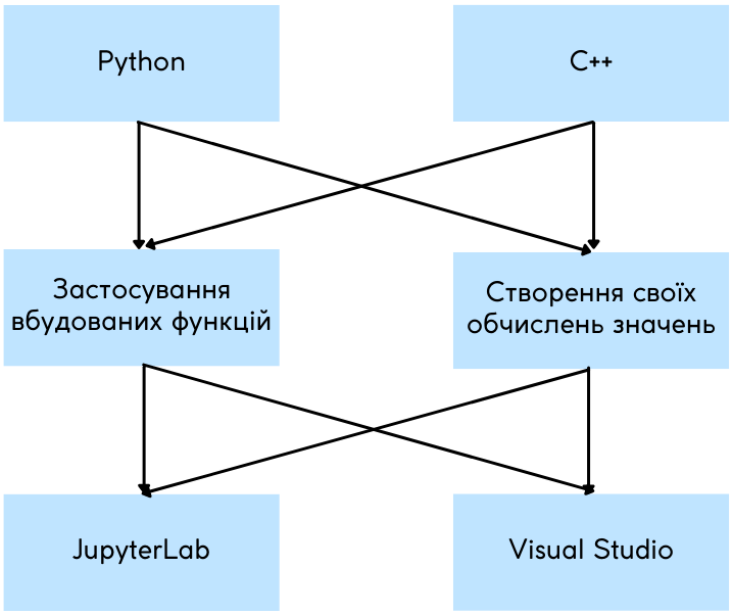



Рисунок 4.1 – Морфологічна карта

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

Таблиця 4.1 – Позитивно-негативна матриця

| Функції | Варіанти реалізації | Переваги | Недоліки |
|---|---|---|---|
| $F_1$ | $A$ | Доступність багатьох бібліотек, швидка розробка, кросплатформеність | Повільна продуктивність, особливо в умовах, коли необхідно обробляти значні обсяги даних |

| | | | |
|---|---|---|---|
| | *Б* | Швидке виконання коду | Розробка програми вимагає значних затрат часу |
| $F_2$ | *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* | Має підтримку для різних мов програмування, без проблем запускається на будь-якому сервері | Вимагає постійного з'єднання з Інтернетом і не може працювати в автономному режимі |
| | *Б* | Широкий набір інструментів і забезпечення високого рівня безпеки | Може працювати з однією мовою програмування одночасно |

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 швидкості навчання, простоті використання та наявності стандартних бібліотек для обчислень. У зв'язку з цим, варіант Б має бути відхилений для спрощення процесу написання коду.

Функція $F_2$:

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

Функція $F_3$:

Якщо вибрана мова програмування – Python, ми віддаємо перевагу варіанту А.

Таким чином, будемо розглядати такий варіанти реалізації ПП:

$F_1$а – $F_2$а – $F_3$а

$F_1$а – $F_2$б – $F_3$а

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 | 10000 | 14600 | 19800 |

| | | | | | |
|---|---|---|---|---|---|
| Об’єм пам’яті | X2 | Мб | 398 | 370 | 332 |
| Час попередньої обробки даних | X3 | мс | 5 | 4 | 3 |
| Потенційний об’єм програмного коду | X4 | кількість рядків коду | 420 | 350 | 208 |

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

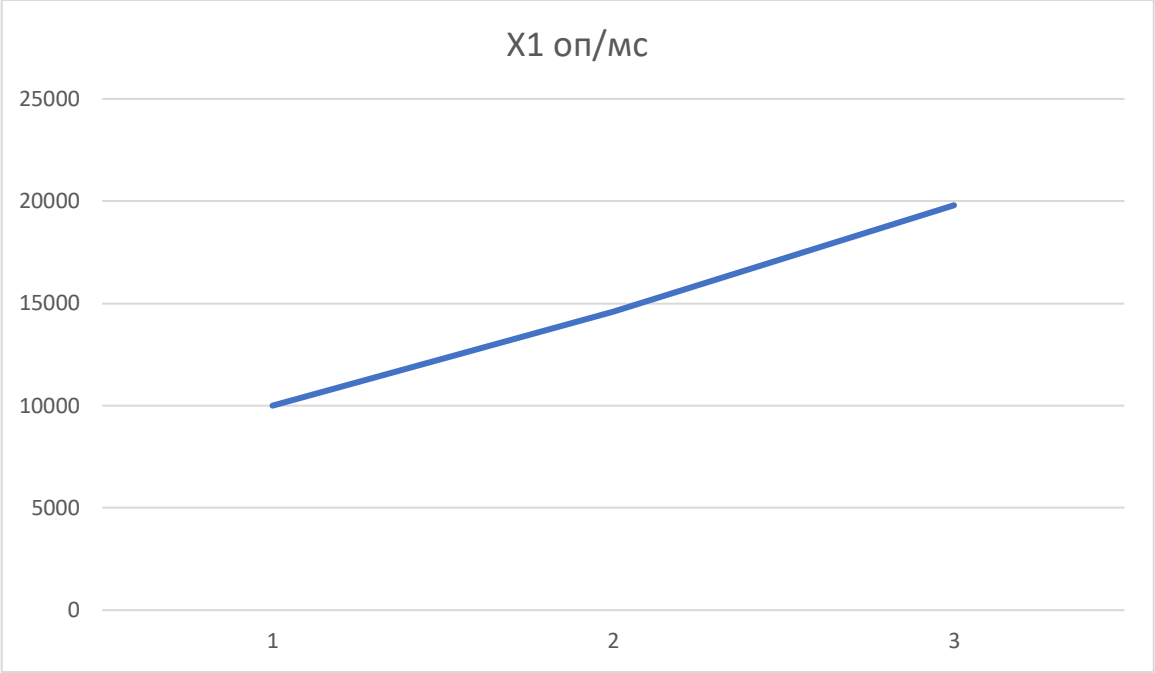



Рисунок 4.2 – X1, швидкодія мови програмування

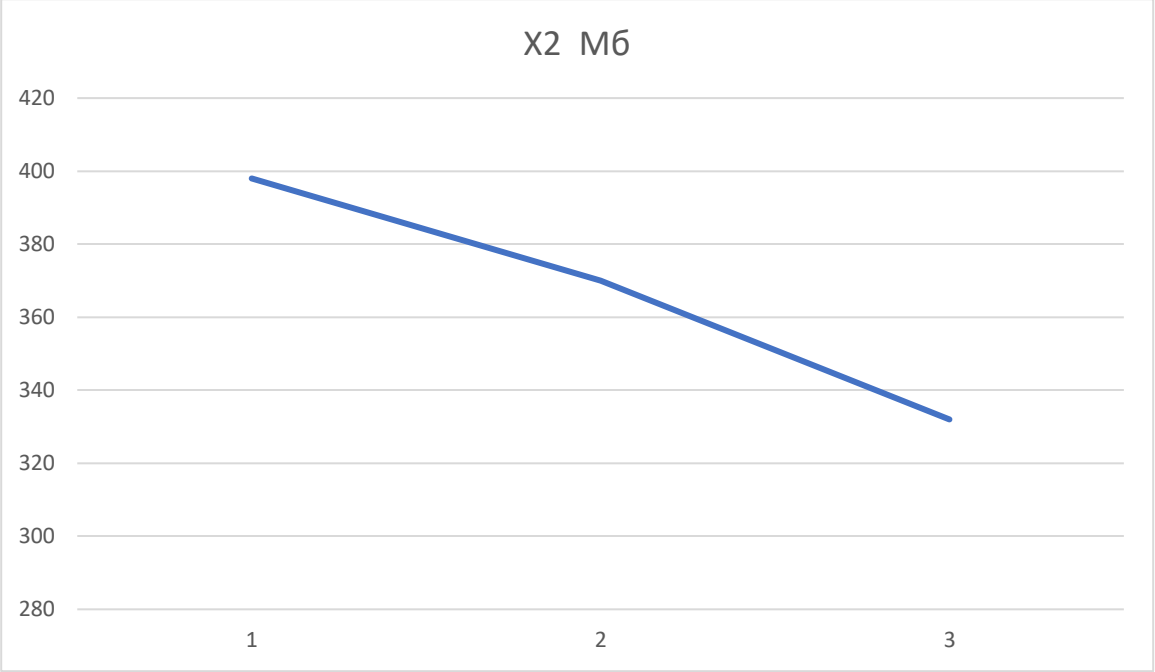



Рисунок 4.3 – X2, об’єм пам’яті

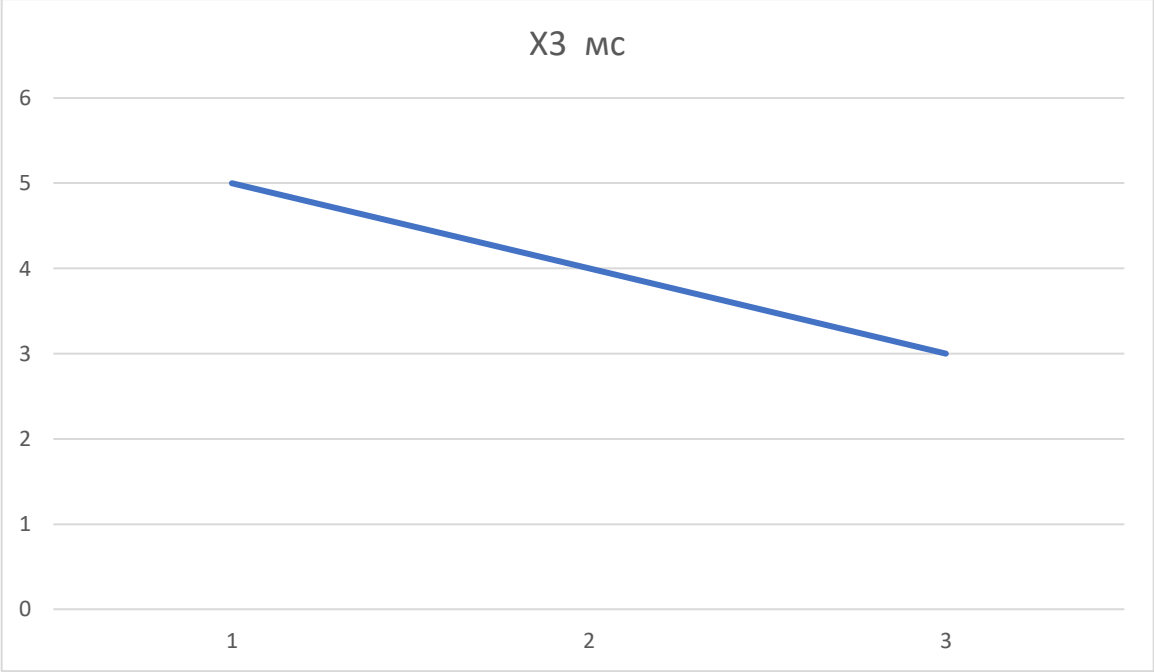



Рисунок 4.4 – X3, час попередньої обробки даних


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 $\Delta_i$ | $\Delta_i^2$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | | |
| *X1* | Швидкодія мови програмування | Оп/мс | 1 | 2 | 2 | 1 | 1 | 2 | 1 | 10 | -7,5 | 56,25 |
| *X2* | Об’єм пам’яті | Mb | 3 | 3 | 3 | 3 | 4 | 4 | 4 | 24 | 6,5 | 42,25 |
| *X3* | Час попередньої обробки даних | мс | 2 | 2 | 1 | 2 | 1 | 2 | 1 | 11 | -6,5 | 42,25 |
| *X4* | Потенційний об’єм програмного коду | Кількість рядків коду | 3 | 3 | 4 | 4 | 4 | 4 | 3 | 25 | 7,5 | 56,25 |
| | Разом | | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 70 | 0 | 197 |

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

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

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

де $N$ – число експертів,

$n$ – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 17{,}5 \qquad (4.2)$$

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

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

Сума відхилень по всім параметрам повинна дорівнювати 0;

г) загальна сума квадратів відхилення:

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

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 197}{7^2(4^3 - 4)} = 0{,}754 > W_k = 0{,}67. \quad (4.5)$$

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 | > | > | > | < | < | < | > | > | 1,5 |
| X1 і X4 | < | < | < | < | < | < | < | < | 0,5 |
| X2 і X3 | > | > | > | > | > | > | > | > | 1,5 |
| X2 і X4 | < | > | < | > | > | > | < | > | 1,5 |
| X3 і X4 | < | < | < | < | < | < | < | < | 0,5 |

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

$$a_{ij} = \begin{cases} 1.5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases}. \quad (4.6)$$

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

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

$$K_{\text{вi}} = \frac{b_i}{\sum_{i=1}^{n} b_i} \tag{4.7}$$

$$b_i = \sum_{i=1}^{N} a_{ij} \tag{4.8}$$

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:

$$K_{\text{вi}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}, \tag{4.9}$$

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

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 | $b_i$ | $K_{\text{вi}}$ | $b_i^1$ | $K_{\text{вi}}^1$ | $b_i^2$ | $K_{\text{вi}}^2$ |
| X1 | 1 | 0,5 | 0,5 | 0,5 | 2,5 | 0,16 | 9,25 | 0,16 | 34,125 | 0,16 |
| X2 | 1,5 | 1 | 1,5 | 1,5 | 5,5 | 0,38 | 17,25 | 0,38 | 60,125 | 0,38 |
| X3 | 1,5 | 0,5 | 1 | 1,5 | 4,5 | 0,32 | 13,25 | 0,31 | 51,875 | 0,3 |
| X4 | 1,5 | 1,5 | 1,5 | 1 | 5,5 | 0,34 | 21,25 | 0,35 | 77,875 | 0,36 |
| Всього: | | | | | 18 | 1.2 | 61 | 1.2 | 224 | 1.2 |

## 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):

$$K_K(j) = \sum_{i=1}^{n} K_{вi,j} B_{i,j}\,, \qquad (4.11)$$

де $n$ – кількість параметрів;

$K_{вi}$ – 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 | 101 | 27 | 0,19 | 4,88 |
| F3 | А | X2 | 90 | 32 | 0,27 | 7,26 |
| | Б | X3 | 28 | 19 | 0,26 | 2,93 |
| F4 | А | X4 | 23 | 21 | 0,33 | 9,42 |

За даними з таблиці 4.6 за формулою:

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

Визначаємо рівень якості кожного з варіантів:

$$K_{K1} = 4{,}88 + 7{,}26 + 9{,}42 = 21{,}56;$$

$$K_{K2} = 4{,}88 + 2{,}93 + 9{,}42 = 17{,}23.$$

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:

$$\text{Т}_\text{О} = \text{Т}_\text{Р} \cdot \text{К}_\text{П} \cdot \text{К}_\text{СК} \cdot \text{К}_\text{М} \cdot \text{К}_\text{СТ} \cdot \text{К}_\text{СТ.М}, \quad (4.13)$$

де $T_Р$ – трудомісткість розробки ПП;

$К_П$ – поправочний коефіцієнт;

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

$К_М$ – 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: $T_Р$ = 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:

$$T_1 = 37 \cdot 1.8 \cdot 0.9 = 59{,}94 \text{ людино-днів.}$$

Проведемо аналогічні розрахунки для подальших завдань.

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

$$T_2 = 29 \cdot 0.9 \cdot 0.8 = 20.88 \text{ людино-днів.}$$

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

$$T_I = (59{,}94 + 20.88 + 4.8 + 20.88) \cdot 8 = 852 \text{ людино-годин.}$$

$$T_{II} = (59{,}94 + 20.88 + 6.91 + 20.88) \cdot 8 = 868{,}88 \text{ людино-годин.}$$

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь два програмісти з окладом 25000 грн., один аналітик в області даних з окладом 21000. Визначимо середню зарплату за годину за формулою:

$$С_Ч = \frac{М}{T_m \cdot t} \text{ грн.}, \quad (4.14)$$

де М – місячний оклад працівників;

$T_m$ – кількість робочих днів тиждень;

$t$ – кількість робочих годин в день.

$$С_Ч = \frac{25000 + 25000 + 21000}{3 \cdot 21 \cdot 8} = 140{,}87 \text{ грн.} \quad (4.15)$$

Тоді, розрахуємо заробітну плату за формулою:

$$С_{ЗП} = С_Ч \cdot Т_i \cdot К_Д, \quad (4.16)$$

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. $С_{ЗП} = 140.87 \cdot 852 \cdot 1.2 = 144025{,}48$ грн.

II. $С_{ЗП} = 140.87 \cdot 868.88 \cdot 1.2 = 146878{,}95$ грн.

Відрахування на єдиний соціальний внесок становить 22%:

I. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 144025{,}48 \cdot 0.22 = 31685{,}6$ грн.

II. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 146878{,}95 \cdot 0.22 = 32313{,}36$ грн.

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

Так як одна ЕОМ обслуговує одного програміста з окладом 25000 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:

$$С_{ЗП} = С_Г \cdot (1 + К_З) = 60000 \cdot (1 + 0.2) = 72000 \text{ грн.}$$

Відрахування на соціальний внесок:

$$С_{ВІД} = С_{ЗП} \cdot 0.22 = 72000 \cdot 0,22 = 15840 \text{ грн.}$$

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

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.4 \cdot 0.25 \cdot 10000 = 3500 \text{ грн.,}$$

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:

$$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.4 \cdot 10000 \cdot 0.08 = 1120 \text{ грн.,}$$

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 12 – 16) \cdot 8 \cdot 0.35 = 627,2 \text{ години,}$$

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:

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 627,2 \cdot 0,2 \cdot 0,3 \cdot 4,79 = 180,25 \text{ грн.},$$

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:

$$С_Н = Ц_{ПР} \cdot 0.67 = 10000 \cdot 0,67 = 6700 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н, \qquad (4.17)$$

$$С_{ЕКС} = 72000 + 15840 + 3500 + 1120 + 180,25 + 6700 = 99340,25 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 69376,90 / 627,2 = 110,60 \text{ грн/год.}$$

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:

$$С_М = С_{М-Г} \cdot Т, \quad (4.18)$$

I. $С_М = 110,6 \cdot 852 = 94231,2$ грн.

II. $С_М = 110,6 \cdot 868,88 = 96098,12$ грн.

Накладні витрати складають 67% від заробітної плати:

$$С_Н = С_{ЗП} \cdot 0,67, \quad (4.19)$$

I. $С_Н = 144025,48 \cdot 0,67 = 96497,07$ грн.

II. $С_Н = 146878,95 \cdot 0,67 = 98408,89$ грн.

Отже, вартість розробки ПП за варіантами становить:

$$С_{ПП} = С_{ЗП} + С_{ВІД} + С_М + С_Н, \quad (4.20)$$

I. $С_{ПП} = 144025,48 + 31685,6 + 94231,2 + 96497,07 = 366439,35$ грн.

II. $С_{ПП} = 146878,95 + 32313,36 + 96098,12 + 98408,89 = 373699,32$ грн.

## 4.7 Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕР j} = К_{Кj} / С_{Фj}, \quad (4.21)$$

$$К_{ТЕР1} = 21,56 / 366439,35 = 5,884 \cdot 10^{-5},$$

$$К_{ТЕР2} = 17,23 / 373699,32 = 4,611 \cdot 10^{-5}.$$

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1} = 5,884 \cdot 10^{-5}$.

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 $К_{ТЕР} = 5,884 \cdot 10^{-5}$.

Цей варіант реалізації програмного продукту має такі параметри:

– Вибір програмного продукту – Python;

– Реалізація важливої постановки з допомогою вбудованих функцій;

– Вибір середовища розробки – JupyterLab.

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

## 4.8 Висновки до розділу 4

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 другий варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості:

Цей варіант реалізації програмного продукту має такі параметри:

− завантаження зображення за допомогою КТ-апарату;

- спектральний аналіз для сегментації зображення;
- використання нейронної мережі для обробки даних;
- визначення групи хвороби, якщо вона присутня.

Даний варіант виконання програмного комплексу дає користувачу зручний інтерфейс, хороший функціонал і швидкодію.

# ВИСНОВКИ ДО РОБОТИ

У ході виконання даної дипломної роботи, була проведена детальна аналітична та практична робота з сегментації КТ-зображень легенів на основі згорткових нейронних мереж.

Застосування згорткових нейронних мереж для сегментації КТ-зображень легенів є ефективним підходом. Вони дозволяють автоматично виділити та окреслити області легенів на зображеннях з високою точністю. Для досягнення найкращих результатів сегментації легенів, важливо вибрати оптимальну архітектуру згорткової нейронної мережі, відповідні гіперпараметри та оптимізаційний алгоритм. Ретельний підбір цих параметрів може покращити точність та швидкодію сегментації. Для тренування моделі був використаний відповідний датасет КТ-зображень легенів з анотаціями. Цей датасет був важливим етапом у підготовці моделі до сегментації та забезпечив достатню кількість прикладів для навчання та тестування.

Результати експериментів показали, що розроблена модель згорткової нейронної мережі має високу точність та здатність до точної сегментації легенів на КТ-зображеннях. Вона здатна виділити навіть малі деталі та структури легенів, що може бути важливим для діагностики та лікування різних захворювань. Дослідження також показало, що швидкість обробки зображень за допомогою згорткових нейронних мереж є задовільною, що дозволяє використовувати цей підхід у реальному часі або на великому обсязку даних.

З даного дослідження, можна зробити висновок, що сегментація КТ-зображень легенів на основі згорткових нейронних мереж є потужним інструментом для аналізу та діагностики захворювань легенів. Впровадження такої технології може сприяти покращенню точності та швидкості діагностики, забезпечуючи більш ефективне лікування та довгостроковий догляд за пацієнтами з різними патологіями легенів.

У результаті дослідження був розроблений програмний продукт, що дозволяє автоматично виділяти та окреслювати області легенів на зображеннях з високою точністю. Він використовує потужні згорткові нейронні мережі, навчені на великому обсязку даних, що дозволяє отримати найкращі результати сегментації. Програмний продукт надає зручний інтерфейс для візуалізації результатів сегментації. Він може відображати оригінальне КТ-зображення разом з виділеними областями легенів. Продукт має інтуїтивно зрозумілий та зручний інтерфейс користувача, що дозволяє легко взаємодіяти з програмою та виконувати різні операції без особливих труднощів. Програма побудована з урахуванням принципів розширюваності та модульності. Вона може бути легко розширена та модифікована для виконання додаткових завдань та врахування специфічних вимог користувача.

.

# ПЕРЕЛІК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

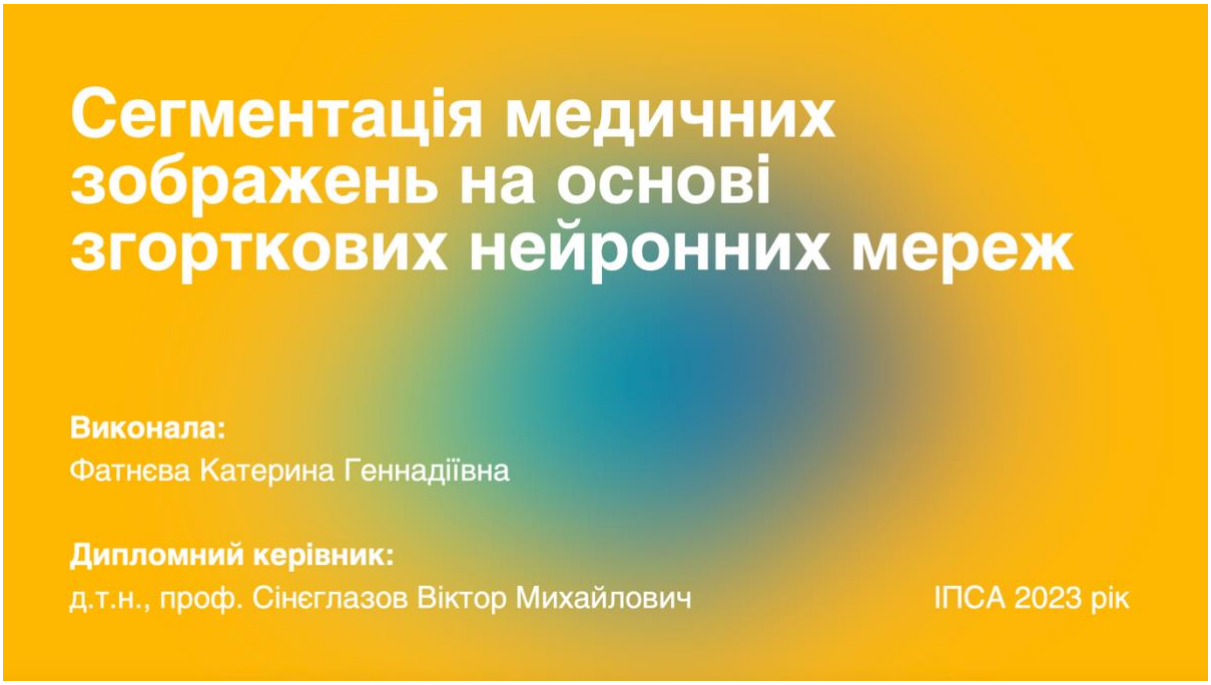

1. M. Thoma, “A Survey of Semantic Segmentation,” arXiv.org, 2016. [Електронний ресурс]. Режим доступу до ресурсу: https://arxiv.org/abs/1602.06541. [Дата звернення: 07.05.2023].
2. E. Shelhamer, J. Long, and T. Darrell, “Fully Convolutional Networks for Semantic Segmentation,” IEEE transactions on pattern analysis and machine intelligence, vol. 39, no. 4, pp. 640–651, 2017. [Електронний ресурс]. Режим доступу до ресурсу: https://www.ncbi.nlm.nih.gov/pubmed/27244717. [Дата звернення: 09.05.2023].
3. O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation,” 2015 [Електронний ресурс]. Режим доступу до ресурсу: https://arxiv.org/pdf/1505.04597.pdf. [Дата звернення: 09.05.2023].
4. L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 40, no. 4, pp. 834–848, Apr. 2018. [Електронний ресурс]. Режим доступу до ресурсу: https://arxiv.org/pdf/1606.00915.pdf. [Дата звернення: 09.05.2023].
5. K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Jun. 2016. [Електронний ресурс]. Режим доступу до ресурсу: https://ieeexplore.ieee.org/document/7780459. [Дата звернення: 09.05.2023].
6. О. С. Сєгрієнко, В. І. Сєгрієнко, В. О. Яременко, "Системи комп'ютерного зору: Навч. посібник", ВПЦ "Київський університет", Київ, 2018.
7. Nationaldefensemagazine.org, 2023. [Електронний ресурс]. Режим доступу до ресурсу: https://www.nationaldefensemagazine.org/articles/2023/3/24/ukraine-a-living-lab-for-ai-

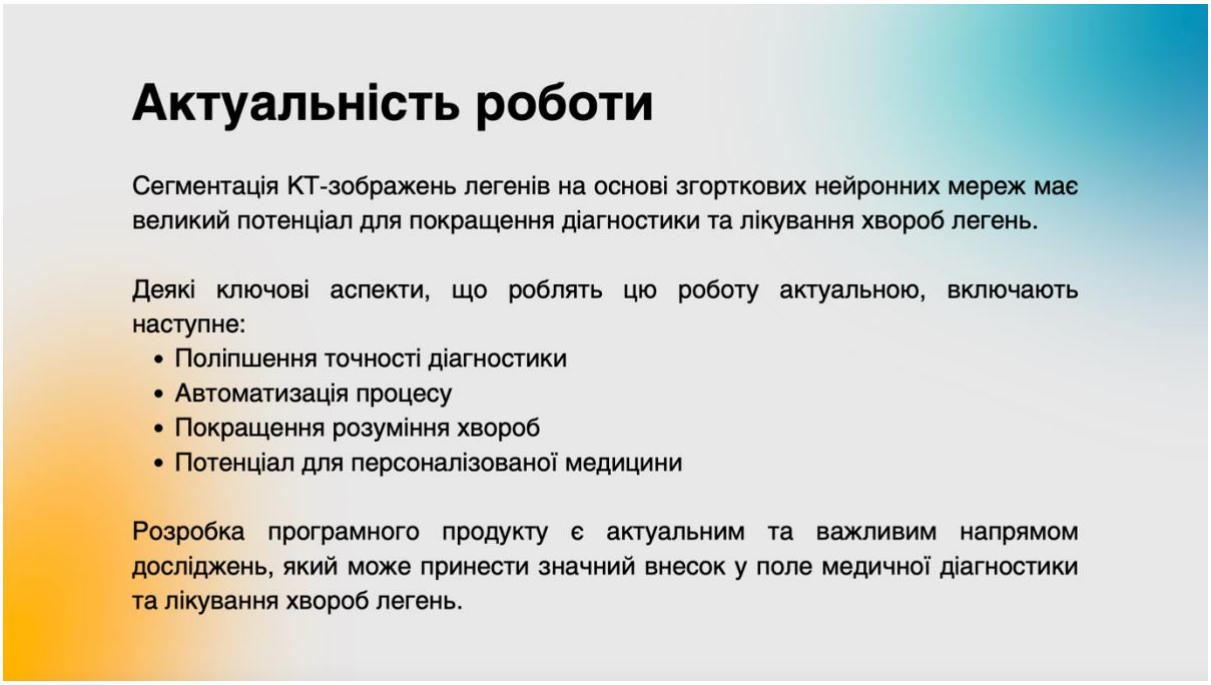
warfare#:~:text=In%20March%202022%2C%20Ukraine's%20defense,in%20electronic%20warfare%20and%20encryption.. [Дата звернення: 09.05.2023].

8. S. Russell, “AI weapons: Russia’s war in Ukraine shows why the world must enact a ban,” vol. 614, no. 7949, pp. 620–623, Feb. 2023, doi: https://doi.org/10.1038/d41586-023-00511-5. [Електронний ресурс]. Режим доступу до ресурсу: https://www.nature.com/articles/d41586-023-00511-5. [Дата звернення: 23.05.2023].
9. M. J. A. Khan, J. Ren, and H. A. Gabbar, “Applying Deep Learning for Image Segmentation: A Survey,” World Congress on Electrical Engineering and Computer Systems and Science, Jul. 2022, doi: https://doi.org/10.11159/mvml22.101. [Електронний ресурс]. Режим доступу до ресурсу: https://avestia.com/EECSS2022_Proceedings/files/paper/MVML/MVML_101.pdf. [Дата звернення: 23.05.2023].
10. I. Goodfellow, Y. Bengio, A. Courville. Deep Learning. MIT Press, 2016.
11. Termin.in.ua, “Нейромережа — що це таке, як працює та навіщо потрібна.,” Termin.in.ua, Apr. 28, 2023. [Електронний ресурс]. Режим доступу до ресурсу: https://termin.in.ua/neyromerezha/#lwptoc14. [Дата звернення: 23.05.2023].
12. Медична інформатика в модулях: практикум / за ред. І.Є. Булах.Київ: «Медицина», 2009. – 208 с.
13. Медична інформатика: підручник для студентів медичних ВНЗ / за ред. В.Г. Кнігавко. – Харків: ХНМУ, 2015. – 288 с.
14. Візуалізація медико-біологічних даних. Обробка й аналіз медичних зображень «Медична інформатика» / упор. Рисована Л.М., Радзішевська Є.Б. – Харків: ХНМУ, 2016. – 23 с.
15. McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 5(4), 115-133.
16. Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review, 65(6), 386-408.

17. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323(6088), 533-536.
18. “Convolutional Neural Network: Benefits, Types, and Applications,” Datagen, May 22, 2023. [Електронний ресурс]. Режим доступу до ресурсу: https://datagen.tech/guides/computer-vision/cnn-convolutional-neural-network/. [Дата звернення: 07.06.2023].
19. M. Hany, “Chest CT-Scan images Dataset,” Kaggle.com, 2020. [Електронний ресурс]. Режим доступу до ресурсу: https://www.kaggle.com/datasets/mohamedhanyyy/chest-ctscan-images?resource=download. [Дата звернення: 03.06.2023].

# ДОДАТОК А ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ



## Об’єкт дослідження

Об'єктом дослідження є КТ-зображення легенів, які потребують точної сегментації для визначення меж об'єктів та структур у легенях. Ці зображення отримуються за допомогою комп'ютерної томографії, яка надає детальну інформацію про структуру та характеристики легенів.

## Предмет дослідження

Предметом дослідження є розробка та застосування згорткових нейронних мереж для автоматичної сегментації КТ-зображень легенів. Головна мета полягає в досягненні високої точності та ефективності визначення границь легенів та розпізнавання різних структур, таких як повітряні простори, судини, пухлини або патологічні утворення.

## Мета роботи

Метою даної роботи є розробка програмного продукту для сегментації КТ-зображень легенів на основі згорткових нейронних мереж.

Основними завданнями є розробка та тренування моделей глибокого навчання для автоматичної сегментації легенів на зображеннях, інтеграція цих моделей у програмний продукт, а також оцінка продукту на реальних медичних даних.

Робота спрямована на вирішення проблем, пов'язаних з ручним або напівавтоматичним процесом сегментації, шляхом розробки алгоритмів, які використовують згорткові нейронні мережі для автоматичного виділення та класифікації областей на медичних зображеннях.

# Постановка завдання

**1. Збір та підготовка даних:** здійснюється збір КТ-зображень легенів, які включають різні типи патологій (аденокарцинома, великоклітинна карцинома, плоскоклітинна карцинома та нормальні зображення).

**2. Розробка архітектури мережі:** буде розроблена згорткова нейронна мережа для сегментації КТ-зображень легенів.

**3. Попередня обробка даних:** ресайз зображень до необхідного розміру, нормалізацію значень пікселів та можливу аугментацію даних для збільшення розмаїття тренувального набору.

**4. Тренування моделі:** треба навчити згорткову нейронну мережу розпізнавати та сегментувати патологічні області на зображеннях легенів.

# Постановка завдання

**5. Оцінка та валідація моделі:** після завершення тренування модель потрібно оцінити та перевірити її ефективність на невикористовуваних раніше тестових даних.

**6. Вдосконалення та оптимізація:** може включати зміну параметрів мережі, додавання шарів або використання більш складних архітектур для покращення точності та швидкості сегментації.

**7. Аналіз результатів та висновки:** на основі оцінки результатів та порівняння з іншими методами будуть зроблені висновки про ефективність розробленого програмного продукту для сегментації КТ-зображень легенів.

## Проблеми сегментації зображень

При роботі з медичними даними, зокрема КТ-зображеннями легенів, можуть виникати деякі потенційні проблеми, які варто враховувати при застосуванні алгоритмів сегментації:

- Низька якість зображення
- Варіабельність форми та розміру
- Великий обсяг даних
- Помилкові діагнози та анатомічні відхилення

Урахування цих потенційних проблем та розробка алгоритмів, які можуть працювати ефективно з вищезгаданими викликами, є важливим аспектом розвитку сегментації КТ-зображень легенів.

## Архітектура CNN

Архітектура CNN складається з трьох основних шарів: шару згортки, шару пулінгу та повнозв'язаного шару. Може бути кілька шарів згортки та пулінгу. Чим більше шарів у мережі, тим більша складність і точність моделі машинного навчання.

**Шар згортки** – основний будівельний блок мережі, де відбувається більшість обчислень.

**Шар пулінгу** – дозволяє зменшити розмір вхідних даних і підвищити ефективність мережі.

**Повнозв'язний шар** – виконує завдання класифікації, використовуючи особливості, які були виявлені попередніми шарами.

## Огляд програмного продукту

Програмний продукт є потужним інструментом для сегментації КТ-зображень легенів та аналізу їх стану, що сприяє прогнозуванню раку легенів – хвороби, що вражає легені, а також виявленню типу хвороби.

**Мова програмування:** Python
**Фреймворк:** JupyterLab

Використання Python та JupyterLab у розробці даної програми є вигідним вибором, оскільки разом вони надають потужні інструменти, легкість використання, а також доступ до великої кількості ресурсів.

Використовуються різні бібліотеки та модулі:
os, numpy, matplotlib.pyplot, tensorflow, tensorflow.keras.layers, PIL.Image.

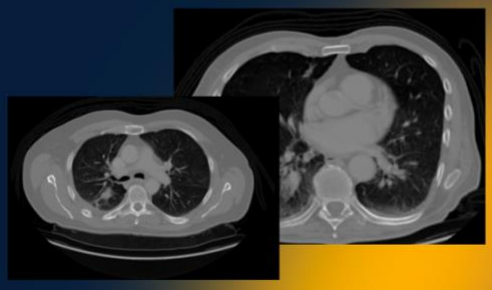
## Вхідні дані

**Формат зображень:** JPG або PNG

Дані містять 3 типи раку легень: аденокарцинома (adenocarcinoma), великоклітинний карцинома (large cell carcinoma) та плоскоклітинний карцинома (squamous cell carcinoma), а також 1 папку для нормальних клітин (normal).

Аналіз та попередня обробка даних:

1. Зчитування та обробка зображень
2. Кодування класів
3. Нормалізація зображень
4. Розбиття даних на тренувальний та валідаційний набори



# Архітектура системи

Архітектура даної системи, що реалізує програму штучного інтелекту для сегментації КТ-зображень легенів та аналізу виявлення хвороби, основана на згорткових нейронних мережах.

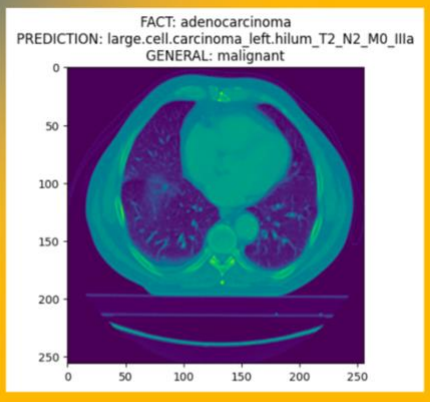
**Convolutional Layer – Pooling Layer – Flatten Layer – Dense Layers**

Усі ці шари складаються разом у модель, яка створюється за допомогою фреймворку TensorFlow та бібліотеки Keras.

```python
# create the CNN model
def create_model(input_shape, num_classes):
    model = tf.keras.Sequential()

    model.add(tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=input_shape))
    model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))

    model.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu'))
    model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))

    model.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu'))
    model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))

    model.add(tf.keras.layers.Flatten())

    model.add(tf.keras.layers.Dense(64, activation='relu'))
    model.add(tf.keras.layers.Dense(num_classes, activation='softmax'))

    return model
```

# Аналіз результатів

Результати представлені у вигляді прогнозованого класу для кожного тестового зображення, а також узагальненого характеру, що вказує, чи присутня злоякісна патологія у легенях. Користувач може оцінити якість роботи програми, порівняючи прогнози зі справжніми класами.




Виконання програми показало, що у даному КТ-зображені присутня злоякісна патологія легенів, а саме Аденокарцинома легенів. Програма класифікувала дане зображення як злоякісне, що є вірним. Проте помилка виникла при класифікації захворювання. Точність – 0.9449.

# Аналіз результатів

Також дана програма реалізує текстові результати. На рисунку видно 5 рандомних прикладів. Кожний індекс це є окреме КТ-зображення. Програма майже правильно розрізняє скани легенів. Проте, як і в попередньому випадку, помилка виникла у класифікуванні злоякісної пухлини.

```
INDEX: 137
FACT: large.cell.carcinoma
PREDICTION: large.cell.carcinoma_left.hilum_T2_N2_M0_IIIa
GENERAL: malignant

INDEX: 314
FACT: normal
PREDICTION: normal
GENERAL: benign

INDEX: 232
FACT: adenocarcinoma
PREDICTION: large.cell.carcinoma_left.hilum_T2_N2_M0_IIIa
GENERAL: malignant

INDEX: 138
FACT: large.cell.carcinoma
PREDICTION: large.cell.carcinoma_left.hilum_T2_N2_M0_IIIa
GENERAL: malignant

INDEX: 105
FACT: large.cell.carcinoma
PREDICTION: large.cell.carcinoma_left.hilum_T2_N2_M0_IIIa
GENERAL: malignant
```

Загалом, результати роботи даної програми можуть бути використані для автоматизації процесу аналізу КТ-зображень легенів та допомогти лікарям у виявленні патологій та прийнятті важливих медичних рішень.

# Висновки

- Розглянуто загальну проблему сегментації зображень, яка виникає при розділенні зображення на окремі об'єкти або регіони.
- Досліджено та порівняно різні підходи до сегментації.
- Проаналізовано потенційні проблеми в даних для алгоритмів сегментації.
- Розроблено модель згорткової нейронної мережі, яка має доволі високу точність та здатність до точної сегментації легенів на КТ-зображеннях.
- Розроблено програмний продукт, який дозволяє автоматично виділяти та окреслювати області легенів на зображеннях з високою точністю, використовуючи нейронні мережі, навчені на великому обсягу даних.

## Подальші дослідження

У майбутньому можна розглянути наступні можливості для вдосконалення даної програми:

- Розширення датасету
- Вдосконалення моделі
- Додаткові функції аналізу
- Вдосконалення інтерфейсу
- Валідація та перевірка

Вдосконалення програми в цих напрямках може покращити її ефективність та корисність для клінічних досліджень, медичних спеціалістів та пацієнтів.

## Дякую за увагу!

# ДОДАТОК Б ЛІСТИНГ ПРОГРАМИ

```
# import basic libs
import os
import numpy as np
import matplotlib.pyplot as plt

# import ML libs
import tensorflow as tf
from tensorflow.keras import layers
from sklearn.model_selection import train_test_split
from PIL import Image

# some more magic hehe
import warnings
warnings.filterwarnings("ignore")



# set the path to the dataset folders
data_folder = './dataset'

# image specs
image_height = 256
image_width = 256

# create the CNN model
def create_model(input_shape, num_classes):
    model = tf.keras.Sequential()
```

```
    model.add(tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=input_shape))
    model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))

    model.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu'))
    model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))

    model.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu'))
    model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))

    model.add(tf.keras.layers.Flatten())

    model.add(tf.keras.layers.Dense(64, activation='relu'))
    model.add(tf.keras.layers.Dense(num_classes, activation='softmax'))

    return model

# convert strings to ints
def classes_to_numbers(_masks):
    temp_masks = []

    classes_str = ['squamous.cell.carcinoma_left.hilum_T1_N2_M0_IIIa','adenocarcinoma_left.lower.lobe_T2_N0_M0_Ib','normal','large.cell.carcinoma_left.hilum_T2_N2_M0_IIIa']
    classes_int = [i[0] for i in enumerate(classes_str)]

    d = {classes_str[i] : classes_int[i] for i in range(len(classes_str))}
```

```
    for i in range(len(_masks)):

        temp_masks.append(d[_masks[i]])

    return temp_masks

# convert ints to strings
def numbers_to_classes(_masks):
    temp_masks = []

    classes_str =
['squamous.cell.carcinoma_left.hilum_T1_N2_M0_IIIa','adenocarcinoma_left.lower.lobe_
T2_N0_M0_Ib','normal','large.cell.carcinoma_left.hilum_T2_N2_M0_IIIa']
    classes_int = [i[0] for i in enumerate(classes_str)]

    d = {classes_int[i] : classes_str[i] for i in range(len(classes_str))}

    for i in range(len(_masks)):

        temp_masks.append(d[_masks[i]])

    return temp_masks

# normalize images
def normalize_images(images):
    normalized_images = images / 255.0
    return normalized_images
```

```
# load and preprocess train and validation data
def load_data():
    images = []
    masks = []

    train_data_path = os.path.join(data_folder, 'train')

    # iterate over the files in the dataset folder
    for subdir in os.listdir(train_data_path):

        # avoid .DS_store
        if subdir[0]!='.':
            # print(subdir)

            # iterate through images in folder
            for img_name in os.listdir(os.path.join(train_data_path, subdir)):

                # get image path
                img_path = os.path.join(train_data_path, subdir, img_name)

                # Convert to grayscale
                img = Image.open(img_path).convert('L')

                # resize image
                img = img.resize((image_width, image_height))

                # convert img to array and add to existing data
```

```
            image_array = np.array(img)
            images.append(image_array)

            # append class of this image
            masks.append(subdir)

    # convert the lists to NumPy arrays
    images = np.array(images)
    masks = np.array(classes_to_numbers(masks))

    # normalize the images
    images = normalize_images(images)

    # split the data into training and validation sets
    train_images, val_images, train_masks, val_masks = train_test_split(images, masks, test_size=0.2, random_state=42)

    return train_images, val_images, train_masks, val_masks

# load test data
def load_test_data():
    test_images = []
    test_masks = []

    # set test images path
    test_data_path = os.path.join(data_folder, 'test')

    # iterate through test images
```

```
for subdir in os.listdir(test_data_path):

    # avoid .DS_store
    if subdir[0]!='.':
        # print(subdir)

        # same stuff as with loading train data
        for img_name in os.listdir(os.path.join(test_data_path, subdir)):

            img_path = os.path.join(test_data_path, subdir, img_name)

            img = Image.open(img_path).convert('L')  # Convert to grayscale

            img = img.resize((image_width, image_height))

            image_array = np.array(img)
            test_images.append(image_array)

            test_masks.append(subdir)



# convert the list to a NumPy array
test_images = np.array(test_images)
test_masks = np.array(test_masks)

# normalize the test images
test_images = normalize_images(test_images)
```

```
    return test_images, test_masks

# post-processing results
def postprocess_predictions(predictions):

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

    processed_predictions = numbers_to_classes(processed_predictions)

    return processed_predictions

# main part
if __name__ == '__main__':

    # load train and validation data
    train_images, val_images, train_masks, val_masks = load_data()

    # load test data
    test_images, test_masks = load_test_data()

    # creating model
    model = create_model((image_height, image_width, 1), 4)

    # adding optimizer and loss function. Setting the target metric
    model.compile(
        optimizer='adam',
        loss='sparse_categorical_crossentropy',
        metrics=['accuracy']
```

```
    )

    # training model
    model.fit(train_images, train_masks, validation_data=(val_images, val_masks), epochs=5, batch_size=32)

    # get predictions on test images
    predictions = model.predict(test_images)

    # postprocessing predictions
    processed_predictions = postprocess_predictions(predictions)

    # same stuff but in text
    print('\n\nSHOWING PREDICTIONS AND REAL CLASSES:\n\n')

    # get random images and their class
    for i in range(10):

        idx = np.random.randint(0, len(test_images))

        print(f"INDEX: {idx}\nFACT: {test_masks[idx]}\nPREDICTION: {processed_predictions[idx]}\nGENERAL: {'benign' if processed_predictions[idx]=='normal' else 'malignant'}\n")
```