КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра технологій управління

Спеціальність 122 - Комп'ютерні науки,
програма "Інформаційна аналітика та впливи"

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему:

"Аналіз та прогнозування виявлення аномалій мозку методами Data Science"

Студента 2-го курсу групи ІАВ-21

Дмитро ЖОВТУХІН

(ім'я, прізвище)

(підпис студента)

Науковий керівник:

д.т.н.,професор

(науковий ступінь, вчене звання)

Юлія ХЛЕВНА

(ім'я, прізвище)

(дата) *(підпис)*

Попередній захист:

(Висновок: "До захисту в Державній екзаменаційній комісії")

Завідувач кафедри
технологій управління,
проф

Віктор МОРОЗОВ

(підпис) *(прізвище, ініціали)* *(дата)*

Київ – 2022

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій

Кафедра технологій управління
Освітньо-кваліфікаційний рівень Магістр
Спеціальність 122 - Комп'ютерні науки
Освітня програма Інформаційна аналітика та впливи

ЗАТВЕРДЖУЮ
Завідувач кафедри
професор Віктор МОРОЗОВ
«____» _____20___року

З А В Д А Н Н Я
НА ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Студент: **Дмитро ЖОВТУХІН**

Група: ІАВ-21

1. Тема кваліфікаційної роботи Аналіз та прогнозування виявлення аномалій мозку методами Data Science

Затверджена протоколом від «17» листопада 2021 р. № 4.

2. Строк подання студентом готової роботи – "18" травня 2022 р.

3. Цільова установка та вихідні дані до роботи система аналізу та прогнозування виявлення аномалій мозку побудована за допомогою мови програмування Python і реалізована із використанням навченої нейромережі на даних, що складається з 2556 зображень із них 1373 мають пухлини.

4. Зміст роботи провів аналіз задачі аналізу аномалій на медичних зображеннях; визначив математичні інструменти й методи, які використовуються для вирішення задач сегментації зображень; визначив дані, що будуть використовуватися при побудові системи; вивчив методи глибокого навчання для сегментації зображень та обрав підходящу модель та специфікації для її навчання; реалізував програмну імплементацію моделі пошуку аномалій мозку на мові програмування Python; створив програмну імплементацію побудованої технології виявлення пухлин та визначив спосіб для надання користувачам можливості доступу до її використання у майбутньому;

5. Перелік графічного матеріалу (слайдів) 25 рисунків, 34 формул, 4-ох додатків, 23 слайдів презентації доповіді.

6. Календарний план виконання роботи:

№ п/п	Назва частин роботи	%	Виконання роботи	
			За планом	Фактично
1.	Вибір теми дипломної роботи	3	01.20.21	01.20.21

2.	Протокол кафедри ТУ про затвердження тем дипломних робіт та призначення наукових керівників	2	17.11.21	17.11.21
3.	Формування переліку нормативних матеріалів, літератури з проблематики дипломної роботи	10	07.01.22	07.01.22
4.	Складання розгорнутого плану кваліфікаційної роботи	5	18.01.22	18.01.22
5.	Ознайомлення наукового керівника з розгорнутим планом кваліфікаційної роботи. Внесення змін.	5	19.01.22 - 20.01.22	20.01.22
6.	Підготовка розділу 1 «Огляд задачі виявлення аномалій мозку»	10	14.02.22	15.02.22
7.	Підготовка розділу 2 «Метод на основі нейронних мереж для сегментації медичних зображень»	14	08.03.22	08.03.22
8.	Підготовка розділу 3 «Реалізація методу пошуку пухлин за допомогою нейромережі U-Net»	14	01.04.22	01.04.22
9.	Підготовка розділу 4 «Практичне застосування отриманої моделі сегментації аномалій мозку»	13	20.04.22	20.04.22
10	Оформлення кваліфікаційної роботи. Підготовка висновків і пропозицій	15	03.05.22	03.05.22
11	Передача кваліфікаційної роботи науковому керівникові	2	04.05.22	04.05.22
12	Передача кваліфікаційної роботи рецензенту для рецензування	2	11.05.22	11.05.22
13	Попередній захист кваліфікаційної роботи	5	19.05.22	19.05.22

Дата видачі завдання «17» листопада 2022 р.

Керівник роботи доктор технічних наук, професор Юлія ХЛЕВНА

_____
(підпис)

Завдання прийняв до виконання:
Здобувач освіти групи ІАВ-21 Дмитро ЖОВТУХІН

_____
(підпис)

ЗМІСТ

АНОТАЦІЯ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра технологій управління
Спеціальність 122 - Комп'ютерні науки,
освітня програма "Інформаційна аналітика та впливи"

Дипломна робота магістра Жовтухіна Дмитра Геннадійовича.

Тема роботи – «Аналіз та прогнозування виявлення аномалій мозку методами Data Science».

Мета дипломної роботи магістра – розробка технології виявлення аномалій мозку, яка здатна надавати користувачам допомогу при постановці діагнозу.

Об'єкт дослідження – процес пошуку новоутворень головного мозку, що слід своєчасно виявляти на медичних знімках.

Предмет дослідження – моделі, методи Data Science для виявлення аномалій мозку.

Наукова новизна одержаних результатів у тому, що розроблено концептуальну модель сегментації аномалій мозку, в основі якої методи Data Science. Відмінність пропонованої моделі від існуючих є здатність обробляти зображення на основі FLAIR технології для отримання МРТ знімків із точністю 95% на тестовому наборі даних, за метрикою чутливості.

Практична значимість у тому, що реалізовано програмну імплементацію технології виявлення аномалій мозку на медичних зображеннях, що базується на мові програмування Python та в подальшому може використовуватися як медичними спеціалістами так і пацієнтами для постановки діагнозу.

У роботі досліджуються існуючі підходи до сегментації пухлин на зображеннях, використання підходу глибокого навчання у задачах сегментації зображень, зокрема медичної візуалізації. Розробляється нейронна модель для пошуку аномалій мозку, створюється застосунок для її використання, а також описується концепція впровадження запропонованого методу аби користуватися

ним могли зацікавлені сторони. Описується результати роботи моделі та її обмеження.

Дипломна робота складається зі вступу, основної частини, яка включає чотири розділи, висновків та списку використаних джерел. Всього основного тексту налічує 74 сторінок та перелік посилань з 62 джерел на 6 сторінках.

Ключові слова: пухлина, сегментація, Data Science, глибоке навчання, штучна нейронна мережа, веб-застосунок.

ВСТУП

Актуальність дослідження.

Насьогодні область комп'ютерного зору стрімко розвивається та приносить користь у багатьох сферах. Обробка зображень може слугувати цінним ресурсом у вирішенні ряду завдань, що допомагають покращити життя простим громадянам, наприклад, пошук об'єктів може зменшити забруднення навколишнього середовища і допомогти у сортуванні сміття [1]. Медична сфера також може отримати зиск від технологій автоматичної обробки зображень.

Пухлини головного мозку включають найбільш небезпечні типи пухлин у всьому світі. Гліома, найпоширеніша первинна пухлина мозку, виникає внаслідок канцерогенезу гліальних клітин у спинному та головному мозку. Гліома характеризується кількома гістологічними та злоякісними ступенями, а середній час виживання пацієнтів з гліобластомою становить менше 14 місяців після встановлення діагнозу [2]. Вона може поширюватися на будь-яку частину мозку, що ускладнює виявлення [3]. За статистикою, щорічна захворюваність гліомою становить близько 5 випадків на 100 000 осіб [4, 5]. Хоча гліоми зустрічаються рідше, ніж інші смертельні захворювання, вони погано піддаються лікуванню і мають вищий рівень смертності [6]. Всесвітня організація охорони здоров'я поділяє гліоми на 4 ступеня, 1 і 2 класи – це гліоми низького рівня, або LGG, а 3 і 4 – високоякісні гліоми, HGG, час виживання пацієнтів з гліомою 4 ступеня менше ніж один рік. Тому рання діагностика відіграє важливу роль. Дане дослідження базується на гліомах низького рівня.

Як швидкий і неінвазивний метод візуалізації, МРТ дозволяє досліджувати анатомічні структури як здорового, так і хворого мозку, а також надає значну інформацію для діагностики гліоми. Однак процес ручного виявлення гліом – це дуже виснажлива і трудомістка робота для рентгенологів. Таким чином, розробка автоматичного та ефективного інструменту діагностики полегшила б роботу фахівцям і життя їх пацієнтам у процесі лікування.

В даний час запропоновано безліч методів автоматичної діагностики. У ранніх роботах такі характеристики і функції, як гістограма, середнє значення,

дисперсія, площа, периметр, нерівномірність, контраст, ключові точки, градієнти, зазвичай витягується для класифікації біометричних зображень [7]. Після виділення ознак, використовується моделі класифікації, наприклад, K-найближчі сусіди, машина опорних векторів, випадковий ліс.

У цій роботі мета полягає в розробці іншої системи класів гліоми. Пропонується автоматичне виявлення гліом на знімках, отриманих за допомогою FLAIR магнітно-резонансної томографії. Запропонований метод складається з одного етапу, сегментації з використанням технології глибоко навчання.

Метою роботи є розробка технології виявлення аномалій мозку, яка здатна надавати користувачам допомогу при постановці діагнозу. Для реалізації поставленої мети необхідним є вирішення таких завдань:

- провести аналіз задачі пошуку аномалій на медичних зображеннях;
- визначити математичні інструменти й методи, які використовуються для вирішення задач сегментації зображень;
- визначити дані, що будуть використовуватися при побудові системи;
- вивчити методи глибокого навчання для сегментації зображень та обрати підходящу модель та специфікації для її навчання;
- реалізувати програмну імплементацію моделі пошуку аномалій мозку на мові програмування Python;
- створити програмну імплементацію побудованої технології виявлення пухлин;
- визначити спосіб для надання користувачам можливості доступу до її використання у майбутньому;
- формалізувати виявлення аномалій мозку методами Data Science у вигляді концептуальної моделі.

Об’єктом дослідження виступає процес пошуку новоутворень головного мозку, що слід своєчасно виявляти на медичних знімках.

Предметом дослідження є моделі, методи Data Science для виявлення аномалій мозку.

Методами дослідження є опис та узагальнення щодо існуючих підходів створення моделей сегментації, елементи штучного інтелекту, нейронних мереж, методи вимірювання для оцінювання якості розробленої моделі та методи графічного аналізу.

Наукова новизна одержаних результатів

Розроблено концептуальну модель сегментації аномалій мозку, в основі якої методи Data Science. Відмінність пропонованої моделі від існуючих є здатність обробляти зображення на основі FLAIR технології для отримання МРТ знімків із точністю 95% на тестовому наборі даних, за метрикою чутливості.

Практична значимість роботи полягає в тому, що реалізовано програмну імплементацію технології виявлення аномалій мозку на медичних зображеннях, що базується на мові програмування Python та в подальшому може використовуватися як медичними спеціалістами так і пацієнтами для постановки діагнозу.

Особистий внесок здобувача. Частина наукових результатів, які відображено у кваліфікаційній роботі, що отримані автором дало змогу підприємству ТОВ “Ай Ес Ді Дизайн” поліпшити існуючи в них алгоритми у сфері обробки інформації та аналізу даних для практичних цілей бізнесу. В Додатку А документ від підприємства, що засвідчує цю інформацію.

Публікації. Основні наукові положення, висновки і результати магістерської кваліфікаційної роботи знайшли відображення у 2 друкованих працях, з них: 1 тези доповіді на конференції та 1 стаття у журналі, включеному до наукометричної бази даних Scopus. Обидві праці написано англійською мовою.

Структура та обсяг роботи. Кваліфікаційна робота складається зі вступу, 4 розділів, висновків, списку літератури з 62 пунктів та 4 додатків. Загальний обсяг кваліфікаційної роботи становить 94 сторінок, із них 74 сторінок основного тексту, який містить 25 рисунки, 34 формули.

РОЗДІЛ 1. ОГЛЯД ЗАДАЧІ ВИЯВЛЕННЯ АНОМАЛІЙ МОЗКУ

1.1 Опис задачі пошуку аномалій мозку

Медична візуалізація, себто картинки, відіграє центральну роль у діагностиці пухлин головного мозку. Черепно-мозкова травма та пошук пухлини – це два діагнози, які використовують зображення сканування мозку для діагностики патології, моніторингу прогресування чи відновлення захворювання або ж локалізації пошкодженої тканини при хірургічному втручанні. Ранні методи візуалізації – інвазивні та іноді небезпечні – такі як пневмоенцефалографія та церебральна ангіографія були залишені на користь неінвазивних методів високої роздільної здатності. Серед основних методів для візуалізації м’яких тканин мозку можна виділити такі:

- Позитронно-емісійна томографія (ПЕТ) [8] використовує особливий тип радіоактивних індикаторів. Метаболічні особливості пухлини мозку, такі як кровотік, метаболізм глюкози, синтез ліпідів, споживання кисню та метаболізм амінокислот, аналізуються за допомогою ПЕТ. Він досі вважається одним з найпотужніших метаболічних методів і використовує фтордезоксиглюкозу (ФДГ). ФДГ є широко використовуваним ПЕТ-індикатором у зображеннях мозку. Тим не менш, такі зображення мають обмеження, наприклад, нездатність диференціювати випромінювання некрозу та рецидивуючу пухлину високого рівня [9]. Більше того, під час ПЕТ-сканування радіоактивні індикатори можуть завдати шкідливого впливу на організм людини, викликаючи після сканування алергічну реакцію. Деякі пацієнти мають алергію на аспартам і йод. Крім того, ПЕТ-трекери не забезпечують точної локалізації анатомічної структури, оскільки мають відносно низьку просторову роздільну здатність.
- Зображення комп’ютерної томографії (КТ) дають більш глибоку інформацію, ніж зображення, отримані при звичайному рентгенівському знімку. Комп’ютерна томографія отримала широкі рекомендації та

поширення з моменту її створення. Дослідження [10] визначило, що тільки в США щорічна частота комп'ютерної томографії становить 62 мільйони, з них 4 мільйони для дітей. КТ показує м'які тканини, кровоносні судини та кістки різних частин людського тіла. Воно використовує більше випромінювання, ніж звичайний рентген. Це випромінювання може підвищити ризик розвитку раку при виконанні кількох КТ.

- МРТ-сканування використовується для повного аналізу різних частин тіла, а також допомагає виявити аномалії в мозку на більш ранніх стадіях, ніж інші методи візуалізації [11].
- Послідовності МРТ використовуються для аналізу уражень інсульту на основі кількох параметрів, таких як вік, локалізація та протяжність областей [12]. У контексті лікування комп'ютеризований метод може бути використаний для точної діагностики швидкості прогресування захворювання [13]. Послідовність МРТ, така як дифузійне зважене зображення (DWI) і ослаблене рідиною інверсійне відновлення (FLAIR), використовуються для виявлення уражень інсульту, проте може використовуватися і для інших захворювань. Так було показано, що DWI є корисними для класифікації пухлин, що раніше було неможливим за допомогою лише структурної візуалізації [14].

Пухлини мозку часто проявляються у вигляді темніших або яскравіших за тканину мозку, масових уражень на зображеннях. Проте ручна сегментація та аналіз структурних зображень пухлин головного мозку є важким і трудомістким завданням, яке поки що може бути виконане лише професійними нейрорадіологами [15]. Широко підготовлені лікарі залишаються золотим стандартом у вилученні інформації та постановки діагнозів з радіологічних зображень. Рентгенологи визначають як дискретні, так і дифузні ознаки на зображеннях м'яких тканин і диференціюють патологію від помилок зображення або інших доброякісних структурних аномалій, таких як

відкладення кальцію. Виявлення аномалій у зображеннях м’яких тканин головного мозку, тим не менш, є складним завданням навіть для кваліфікованого фахівця, тому для сприяння процесу діагностики розроблені спеціальні методи візуалізації та методи обробки зображень. Рентгенологи знаходять потрібну інформацію та перехресно посилають кілька наборів зображень для кожного пацієнта, щоб створити цілісну картину проблеми, отриманої з великого обсягу необроблених даних. Остаточний діагноз потім встановлюється на основі збору результатів візуалізації.

Незважаючи на те, що радіологи мають високу кваліфікацію та мають доступ до передових інструментів для діагностики, дослідження показали, що два радіологи не погоджуються щодо діагнозу від 5 до 20 відсотків часу [16]. Можливість інструментів прийняття рішень на основі машинного навчання полягає в тому, аби зменшити діагностичну помилку. Таким чином, автоматична та надійна сегментація пухлини мозку матиме значний вплив на діагностику та лікування пухлин головного мозку. Крім того, це також може призвести до своєчасної діагностики та лікування неврологічних розладів, таких як новоутворення, хвороба Альцгеймера, шизофренія та деменція. Автоматична методика сегментації ураження може допомогти радіологам надати ключову інформацію про об’єм, локалізацію та форму пухлин, включаючи розширення серцевинних ділянок пухлини та їх ділянок, щоб зробити прогрес терапії більш ефективним та значущим. Існує кілька відмінностей між пухлиною та її нормальною сусідньою тканиною, які перешкоджають ефективності сегментації в аналізі медичних зображень, наприклад, розмір, артефакт через неправильне отримання зображення, розташування та форма.

У літературі реалізовано декілька моделей, які намагаються знайти точні та ефективні границі пухлин головного мозку на медичних зображеннях. Ці моделі можна розділити на три основні категорії:

- Підходи машинного навчання вирішують ці проблеми, використовуючи різні класифікатори, такі як випадковий ліс, машина опорних векторів, нечітка кластеризація [17]. Як початковий крок у цьому виді, ключова

інформація витягується з вхідного зображення за допомогою деякого алгоритму виділення ознак, сегментації, а потім дискримінаційну модель навчають розпізнавати пухлину з нормальних тканин. Методи та алгоритми вилучення ознак мають знаходити деталі, пов'язані з краями. Коли межі між здоровими тканинами та пухлинами нечіткі, ці методи демонструють низькі результати.

- Алгоритми Multi-atlas registration (MAS) засновані на реєстрації та об'єднанні міток кількох нормальних атласів мозку до нової модальності зображення [18]. Через труднощі з реєстрацією нормальних атласів мозку та потребу у великій кількості атласів ці алгоритми MAS не успішно працюють із додатками, які потребують швидкості.
- Методи глибокого навчання автоматично витягують важливі функції. Ці підходи дали видатні результати в різних областях застосування, наприклад, виявлення пішоходів розпізнавання та розуміння мовлення та сегментація пухлин головного мозку [19].

Як видно, два з трьох методів використовують сегментацію задля пошуку анамілій мозку. Тож для виконання задачі даного дослідження було обрано саме сегметувати область пухлини. Ці два методи представляють собою машине та глибоке навчання, де перший шукає пухлини в два етапи, першим з яких і є сегментація, а другий одразу повертає необхідні регіони зображення. Перед зануренням у методи сегментації мозку, опишемо власне задачу сегментації, задля вибору одного з них, для практичної реалізації.

1.2 Опис задачі сегментації зображень

У цифровій обробці зображень і комп'ютерному зорі сегментація зображення — це процес поділу цифрового зображення на кілька сегментів, також відомих як області або об'єкти зображення, набори пікселів. Мета сегментації полягає в тому, щоб спростити та/або змінити представлення зображення на щось, що є більш значущим і легшим для аналізу [20, 21]. На

вхід подаються пікселі зображення, а на виході очікується зображення того ж розміру, але замість пікселів, що показують саме зображення повертаються мітки об’єктів. Кожна мітка, тобто унікальне число, позначає певний об’єкт, що треба було знайти. Для бінарної сегментації вихідна маска має чорні пікселі для фону і білі для знайденого об’єкту.

Сегментація зображень зазвичай використовується для визначення місця розташування об’єктів і меж, наприклад ліній, кривих, на зображеннях. Це процес присвоєння мітки кожному пікселю зображення таким чином, що пікселі з однаковою міткою мають певні характеристики.

Завдання сегментації зображень можна розділити на три групи залежно від кількості та типу інформації, яку вони передають.

- Семантична сегментація відноситься до класифікації пікселів у зображенні на семантичні класи. Пікселі, що належать до певного класу, просто класифікуються до цього класу без урахування іншої інформації чи контексту. Не слід використовувати, коли в зображенні є тісно згруповані кілька екземплярів одного класу.
- Сегментація екземплярів. Моделі сегментації екземплярів класифікують пікселі за категоріями на основі «екземплярів», а не класів. Алгоритм сегментації екземплярів не має уявлення про клас, до якого належить класифікована область, але може відокремлювати об’єкти, що перекриваються або дуже схожі області об’єктів на основі їх меж.
- Паноптична сегментація може бути виражена як комбінація семантичної сегментації та сегментації екземплярів, де кожен екземпляр об’єкта на зображенні відокремлюється, а ідентичність об’єкта прогнозується. Цей тип знаходять широкомасштабне застосування в популярних завданнях, таких як самокеровані автомобілі, де величезна кількість інформації про безпосереднє оточення має бути захоплена за допомогою відеопотоку. Паноптична сегментація дає нам сегментні карти всіх об’єктів будь-якого конкретного класу, присутніх на зображенні.

Сегментація зображень застосовується у таких видатних галузях, як робототехніка, медична візуалізація, автономні транспортні засоби та інтелектуальна аналітика відео. Окрім цих додатків також використовується супутниками на аерофотознімках для сегментації доріг, будівель та дерев. Ось кілька найпопулярніших реальних випадків використання сегментації зображень:

- Робототехніка. Сегментація зображень допомагає машинам сприймати навколишнє середовище та переміщуватись у ньому, вказуючи на об’єкти на їхньому шляху руху, що дозволяє їм ефективно змінювати шляхи та розуміти контекст свого середовища. Окрім пересування, сегментація зображень допомагає машинам відокремлювати об’єкти, з якими вони працюють, що дозволяє їм взаємодіяти з реальними об’єктами, використовуючи лише картинку як орієнтир. Це дозволяє машині бути корисною практично будь-де без особливих обмежень.
- Самокеровані автомобілі. Автомобілі з автопілотом — одна з найбільших сфер для користування сегментації зображень. Семантична сегментація та сегментація екземплярів допомагає цим транспортним засобам ідентифікувати шаблони доріг та інші транспортні засоби, тим самим забезпечуючи безпечну та комфортну поїздку.
- Пошук на основі зображень. Пошукові системи, які пропонують засоби пошуку на основі зображень, використовують методи сегментації зображень, щоб ідентифікувати об’єкти, присутні на ньому, і порівнюють їхні результати з відповідними зображеннями, які вони знаходять, щоб видати оптимальні результати пошуку.
- Ідентифікація номерних знаків. Багато світлофорів і камер використовують ідентифікацію номерних знаків для нарахування штрафів чи задля допомоги слідству при кримінальних справах. Технологія ідентифікації номерних знаків дозволяє системі дорожнього руху розпізнавати автомобіль та отримувати інформацію про його власника. Він використовує сегментацію зображення, щоб відокремити номерну

табличку та її інформацію від решти об’єктів, присутніх у полі зору. Ця технологія значно спростила процес накладення штрафів для урядів.

- Сільське господарство. Супутникові зображення обробляються для ідентифікації різних моделей, об’єктів, географічних контурів, інформації про ґрунт тощо, які пізніше можуть бути використані для сільського господарства, видобутку корисних копалин, геодозондування.
- У виробництві. Наприклад, деякі виробники почали використовувати методи сегментації зображень, щоб знайти неякісні продукти. Тут алгоритм фіксує лише необхідні компоненти з зображення об’єкта та класифікує їх як несправні чи оптимальні. Ця система знижує ризик людських помилок і робить процес тестування більш ефективним для організації.
- Медична візуалізація є важливою областю комп’ютерного зору, яка зосереджується на діагностиці захворювань на основі візуальних даних, як у формі простих візуальних даних, так і біомедичних сканів. Допомагає лікарям швидко та точно визначити можливі злоякісні ознаки на зображеннях. Використовуючи сегментацію зображень, діагностику захворювань можна не тільки пришвидшити, але й здешевити, що принесе користь тисячам людей по всьому світу.

При застосуванні до зображень, типових для медичної візуалізації, тобто набору зрізів, наприклад, після МРТ, отримані контури після сегментації можна використовувати для створення тривимірних реконструкцій за допомогою алгоритмів інтерполяції, таких як маршируючі куби [22].

Процес сегментації можна розділити на два підходи на основі властивостей зображення. Перший - виявлення подібностей, що д ґрунтується на виявленні подібних пікселів у зображенні – на основі порогу, збільшення області, розширення області та злиття регіонів. Алгоритми машинного навчання, такі як кластеризація, спираються на цей підхід виявлення подібностей за невідомим набором ознак, так само як і класифікація, яка

виявляє схожість на основі попередньо визначеного відомого набору ознак. Виявлення розривів - це повна протилежність підходу виявлення подібностей, коли алгоритм скоріше шукає границі. Алгоритми сегментації зображень, такі як виявлення країв, виявлення точок, виявлення лінії, дотримуються цього підходу, коли краї визначаються на основі різних показників розриву, як-от інтенсивність. На основі цих двох підходів існують різні прийоми, які застосовуються при розробці алгоритмів сегментації зображень. Ці методи використовуються на основі типу зображення, яке необхідно обробити та проаналізувати, і їх можна класифікувати на три більш широкі категорії [23]:

- Методи структурної сегментації. Ці набори алгоритмів вимагають, по-перше, знати структурну інформацію про зображення. Це можуть бути пікселі, щільність пікселів, гістограми, розподіл кольорів. По-друге, потрібна структурна інформація про об’єкт, який ми збираємося отримати із зображення. Такий підхід потрібен, якщо треба визначити цільову область, яка є дуже специфічною для певної задачі, які треба вирішити і є критичним фактором. У цих наборах алгоритмів буде використовуватися підхід, заснований на подібності.
- Методи стохастичної сегментації. У цій групі алгоритмів основна інформація, яка потрібна — це знати значення дискретних пікселів повного зображення, а не вказувати на структуру необхідної частини зображення. Це виявляється вигідним у випадку більшої групи зображень, де існує високий ступінь невизначеності щодо необхідного об’єкта. Цей підхід використовують алгоритми з нейронними мережами та машинного навчання, які використовують, наприклад, k-середніх.
- Гібридні методи. Як випливає з назви, ці алгоритми для сегментації зображення використовують комбінацію структурного методу та стохастичних методів, тобто використовують як структурну інформацію регіону, так і інформацію про дискретні пікселі зображення.

1.3 Аналіз існуючих способів сегментації зображень

Спершу слід зазначити, що задача виключно сегментації зображень існує окремо від задачі пошуку пухлин. Втім для визначення місцезнаходження аномалій зазвичай сегментують зображення, тож спершу зробимо огляд класичних підходів до сегментації зображень.

Сегментація зображень спочатку почалася з цифрової обробки зображень у поєднанні з алгоритмами оптимізації. Ці примітивні алгоритми використовували такі методи, як вирощування регіонів і алгоритм змій, де вони встановлювали початкові області, а алгоритм порівнював значення пікселів, щоб отримати уявлення про сегменти. Ці методи брали локальне уявлення про особливості зображення та зосереджувалися на локальних відмінностях та градієнтах у пікселях.

Алгоритми, які брали глобальне інформацію про вхідне зображення, з’явилися набагато пізніше, коли серед класичних методів обробки зображень були запропоновані такі методи, як адаптивне порогове значення, метод Оцу та алгоритми кластеризації. Тож зараз існують наступні прийоми сегментації зображень:

- Порогові методи. Зосереджені на пошуку пікових значень на основі гістограми зображення, щоб знайти подібні пікселі. Не вимагають складної попередньої обробки, є доволі простими і зрозумілими. Багато деталей може бути пропущено.
- Методи на основі пошуку країв. Працюють на основі виявлення розриву на відміну від виявлення подібності. Добре для зображень із кращим контрастом між об’єктами. Не підходять для зображень із шумом.
- Методи на основі регіонів. Працюють на основі поділу зображення на однорідні області. Дуже добре підходять для зображень із значною кількістю шуму, може використовувати маркери користувача для швидкої оцінки. Вимагають багато часу і пам'яті.
- Методи на основі кластеризації. Розбиває зображення на певну кількість однорідних, взаємовиключних кластерів. Перевірені методи, підкріплені

нечіткою логікою та більш корисні для застосування в реальному часі. Визначення функції для мінімізації різниці між кластерами може бути складним.

- Методи на основі водорозділу. Працюють на основі типологічної інтерпретації меж зображення. Отримані сегменти більш стабільні, виявлені межі чіткі. Розрахунок градієнта в цих випадках складний.
- Алгоритми глибокого навчання – згорткові нейронні мережі. Мають готові бібліотеки з реалізацією, що доступні на Python, для створення більш практичних додатків. Але навчання моделі для займає багато часу та ресурсів.

Дослідимо детальніше кожен із них.

Порогові методи.

Застосування порогового значення є одним із найпростіших методів сегментації зображення, де встановлюється поріг для поділу пікселів на два класи. На основі інтенсивності пікселі зображення діляться шляхом порівняння інтенсивності пікселя з пороговим значенням. Пікселі зі значеннями, більшими за порогове, отримують значення 1, а пікселі зі значеннями, меншими за порогове — 0. Таким чином, зображення перетворюється на двійкову карту, в результаті чого відбувається процес, який часто називають бінаризацією. Це, мабуть, найпростіший і водночас потужний метод визначення потрібних об’єктів на зображенні. Пороговий метод виявляється вигідним, коли передбачається, що об’єкти на зображенні мають більшу інтенсивність, ніж фон або інші небажані компоненти зображення. Тобто, якщо різниця у значеннях пікселів між двома цільовими класами дуже велика, і як порогове значення легко вибрати середнє значення. Але такий підхід може виявитися хибним, для випадків з великою кількістю шуму на зображенні. Таким чином, можна або підтримувати поріг постійним, або динамічно змінювати його на основі властивостей зображення і таким чином отримати кращі результати.

В найпростішому випадку ця техніка замінює пікселі на зображенні на чорні або білі. Якщо інтенсивність пікселя $I_{i,j}$ у позиції (i, j), де і та j координати, менша за поріг T, то він замінюється чорним, а якщо більше, то на білий. Це бінарний підхід до визначення порогу. Порогове значення часто використовується для бінаризації зображень, щоб можна було використовувати додаткові алгоритми, такі як виявлення контурів та інші, що працюють лише на двійкових зображеннях.

Хоча в деяких випадках порогове значення T може або потрібно вибирати користувачем вручну, є багато випадків, коли користувач хоче, аби поріг автоматично встановлювався алгоритмом. У цих випадках порогове значення має бути найкращим порогом у тому сенсі, що він має розділяти об'єкти, які вважаються частиною переднього плану, і об'єкти, які вважаються частиною фону. Існує багато типів автоматичних методів визначення порогу, найвідомішим і широко використовуваним є метод Оцу [24]. Візьмемо зображення, гістограма якого має два піки, так зване бімодальне зображення, один для фону і один для переднього плану. Відповідно до бінаризації Оцу, для цього зображення значення в середині цих піків можна обрати як порогове. Простіше кажучи, він автоматично обчислює порогове значення з гістограми зображення. Метод погано працює на зображеннях, які не є бімодальними, гістограма зображення яких має кілька піків або один із наявних класів має високу дисперсію. Однак бінаризація Оцу широко використовується при скануванні документів, видаленні небажаних кольорів з документа, розпізнаванні візерунків.

Стале значення порогу може бути не оптимальним варіантом у тих умовах, коли зображення має різне освітлення для фону і переднього плану в різних ділянках. Тут потрібен адаптивний підхід, який може змінювати поріг для різних компонентів зображення. При цьому алгоритм ділить зображення на різні менші частини і обчислює поріг для цих частин зображення окремо. Отже, отримується різні пороги для різних областей одного зображення. Це, у свою чергу, дає кращі результати для зображень із різним освітленням. Алгоритм

може автоматично обчислити порогове значення. Порогове значення може бути середнім з усіх яскравостей пікселів для певної ділянки або це може бути їх зваженою сумою, де ваги є гаусовим вікном.

Методи на основі пошуку країв

Виявлення країв — це процес визначення країв зображення, що є дуже важливим кроком до розуміння особливостей зображення. З точки зору сегментації, можна сказати, що виявлення країв відповідає класифікації того, які пікселі зображення є крайовими пікселями, і виділення цих пікселів краю відповідно до окремого класу. Вважається, що границі складаються із значущих ознак і містять суттєву інформацію. Це значно зменшує розмір зображення, яке буде оброблено в подальшому, і відфільтровує інформацію, яка може вважатися менш актуальною, зберігаючи та зосереджуючи виключно на важливих структурних властивостях зображення. Алгоритми виявлення країв в основному діляться на дві категорії – методи на основі градієнта та гістограм.

Алгоритми сегментації на основі країв працюють для виявлення країв зображення на основі різних розривів рівня сірого, кольору, текстури, яскравості, насиченості, контрасту. Для подальшого покращення результатів необхідно виконати додаткові кроки обробки, щоб об’єднати всі краї в ланцюжки, які краще відповідають межам на зображенні.

Зазвичай виконується за допомогою спеціальних фільтрів, які дають нам краї зображення при згортці. Ці фільтри розраховуються спеціальними алгоритмами, які працюють над оцінкою градієнтів зображення в координатах x і y окремо. У цих алгоритмах використовуються базові оператори визначення меж, такі як оператор Собеля [25], Канні [26] так само як і Лапласіан. Ці оператори допомагають виявляти розриви країв і, отже, позначають межі краю. Алгоритм Канні є одним з найпопулярніших. Кінцева мета полягає в досягненні принаймні часткової сегментації за допомогою цього процесу, коли групуються всі локальні границі в нове двійкове зображення, де присутні лише ланцюжки країв, які відповідають необхідним існуючим об’єктам або частинам зображення.

При застосуванні до набору зображень, типового для медичної візуалізації, отримані контури можна використовувати для створення тривимірних реконструкцій за допомогою алгоритмів інтерполяції.

Методи на основі регіонів

Алгоритми сегментації на основі регіонів працюють, шукаючи подібності між сусідніми пікселями та групуючи їх у загальний клас, де пікселі мають подібні характеристики. Зазвичай процедура сегментації починається з деяких пікселів або менших частин зображення, встановлених як початкові елементи, і алгоритм працює, виявляючи безпосередні межі початкових пікселів і класифікуючи їх як схожі чи відмінні. Далі використовуються певні підходи, щоб додати більше пікселів до початкових точок або ще більше зменшити початкову точку до менших сегментів і злитися з іншими меншими початковими точками. Після цього найближчі сусіди розглядаються як початкові, кроки повторюються, доки все зображення не буде сегментовано. На основі цього методу існують дві основні методики.

Перший - це метод зростання регіонів [27]. Він працює знизу вгору, коли сегментація починається з меншого набору пікселів і поступово накопичує або ітеративно об’єднує їх на основі певних заздалегідь визначених обмежень подібності. Цей алгоритм починається з вибору довільного початкового пікселя на зображенні та порівняння його з сусідніми пікселями. Якщо є збіг або подібність у сусідніх пікселях, то вони додаються до початкового пікселя, таким чином збільшуючи розмір області. Коли зростання цієї області не може продовжуватися далі, алгоритм вибирає інший початковий піксель, який обов’язково не належить до жодного з попередньо визначених регіонів, і починає процес знову.

Методи зростання регіонів часто досягає ефективної сегментації, яка добре відповідає краям. Але іноді, коли алгоритм дозволяє регіону рости, перш ніж спробувати інші вихідні дані, це зазвичай зміщує сегментацію на користь регіонів, які сегментовані першими. Щоб протидіяти цьому ефекту, більшість алгоритмів спочатку вимагають вводити користувачем інформацію про

подібність пікселів, жодному регіону не дозволяється домінувати та повністю рости, а кільком регіонам дозволяється розвиватися одночасно.

Зростання регіонів, також є алгоритмом на основі порогового значення, але основна відмінність полягає в тому, що порогове значення виділяє велику область на основі подібних пікселів з будь-якої точки зображення, тоді як зростання регіонів витягує лише сусідні пікселі. Технології зростання регіонів є кращими для зображень із шумом, де дуже важко виявити краї.

Методи сегментації, засновані на розділенні та об'єднанні, використовують дві основні методики, які виконуються разом – розділення регіонів та об'єднання регіонів для сегментації зображення. Розбиття включає ітераційне поділ зображення на області, що мають подібні характеристики, а об'єднання використовує об'єднання сусідніх областей, які дещо схожі один на одного. Розділ на регіон, на відміну від зростання регіону, розглядає все вхідне зображення. Потім він намагається знайти відповідність відомому набору параметрів або попередньо визначеним обмеженням подібності та обирає всі області пікселів, які відповідають критеріям. Це метод «розділяй і володарюй» на відміну від алгоритму зростання регіону.

Після виконання процесу поділу є багато однаково позначених областей, розкиданих по всіх пікселях зображення, тобто остаточна сегментація міститиме розсіяні кластери сусідніх областей, які мають ідентичні або подібні властивості. Щоб завершити процес, необхідно виконати злиття, яке після кожного розбиття порівнює сусідні регіони, і, якщо потрібно, на основі ступенів подібності, об'єднує їх. Такі алгоритми ще називаються split-merge.

Прикладом подібного підходу є популярний **алгоритм водорозділу** [28], який працює, виходячи з локальних максимумів евклідової карти відстаней, і зростає за умови, що жодні дві початкові точки не можуть бути класифіковані як такі, що належать до одного регіону або сегменту.

Водорозділ — це метод на основі регіонів, який слідує концепції топологічної інтерпретації. Розглядається аналогія географічного ландшафту з хребтами та долинами для різних компонентів зображення. Нахил і висота

згаданої топографії чітко визначаються сірими значеннями відповідних пікселів, які називаються величиною градієнта. На основі цього тривимірного зображення, яке зазвичай використовується для ландшафтів Землі, перетворення водорозділу розкладає зображення на області, які називаються басейнами водозбору. Для кожного локального мінімуму водозбірний басейн містить усі пікселі, чий шлях найкрутішого спуску сірих значень закінчується на цьому мінімумі.

Простими словами, алгоритм розглядає пікселі як локальну топографію , або висоту, часто ініціалізуючи себе з визначених користувачем маркерів. Потім алгоритм визначає те, що називається басейни, які є точками мінімуму, і, отже, басейни “заливаються” з маркерів до тих пір, поки басейни не зустрінуться на лініях водорозділу. Водорозділи, які тут так утворені, відокремлюють басейни один від одного. Таким чином, зображення сегментується, тому що ми маємо пікселі, призначені кожному такому регіону або водорозділу.

Сегментація на основі кластеризації.

Алгоритми кластеризації є неконтрольованими алгоритмами, на відміну від алгоритмів класифікації, де користувач не має попередньо визначеного набору функцій, класів або груп. Алгоритми кластеризації допомагають отримати основну, приховану інформацію з таких даних, як структури, кластери та групи, які зазвичай невідомі з евристичної точки зору.

Методи, засновані на кластеризації, сегментують зображення на кластери або групи пікселів, що не перетинаються, за схожими характеристиками. Завдяки базовим властивостям кластеризації даних елементи даних розбиваються на кластери таким чином, що елементи в одному кластері більш схожі один на одного, ніж в інших кластерах.

Сучасні процедури сегментації, які залежать від методів обробки зображень, зазвичай використовують алгоритми кластеризації для сегментації. Вони працюють краще, ніж їхні аналоги, і можуть забезпечити досить хороший поділ за невеликий проміжок часу. Популярні алгоритми, такі як алгоритми кластеризації K-середніх, MeanShift, CamShift [29], є неконтрольованими

алгоритмами, які працюють шляхом групування пікселів із загальними атрибутами які належних до певного сегмента. На відміну від методів вирощування регіонів, методи на основі кластеризації не потребують початкової точки для початку сегментації, до того ж вони використовують кілька статистичних параметрів, якими можна налаштовувати якість фінального результату.

K-середніх [30] — це один із найпростіших алгоритмів навчання без вчителя, який може вирішити проблеми кластеризації загалом. Процес відбувається за простим і легким способом класифікації даного зображення за допомогою певної кількості кластерів, які фіксуються попередньо. Алгоритм фактично починається з цієї точки, де простір зображення поділено на k пікселів, що представляють k центроїдів групи. Потім кожен з об'єктів призначається до групи на основі його відстані від кластера. Коли всі пікселі призначені для всіх кластерів, центроїди переміщуються і перепризначаються. Ці кроки повторюються до тих пір, поки центроїди не перестануть зміщуватися. Наприкінці цього алгоритму ми маємо області всередині зображення, сегментовані на групи, де складові пікселі демонструють деякі рівні подібності.

K-середніх дозволяє розділити та групувати разом пікселі зображення, які мають певний ступінь подібності. Однією з вражаючих особливостей K-середніх є те, що групи та їх члени повністю взаємовиключні. Метод нечіткої кластеризації, Fuzzy C Means [31], дозволяє об'єднати пікселі в більш ніж один кластер. Іншими словами, група пікселів може належати більш ніж одному кластеру або групі, але вони можуть мати рівну подібність з кожною із груп.

Нейронні мережі.

Окремим видом стоїть сегментація на основі нейронних мереж. Підхід до використання сегментації зображень із використанням нейронних мереж часто називають розпізнаванням зображень. Він використовує штучний інтилект для автоматичної обробки та ідентифікації таких компонентів зображення, як об'єкти, обличчя, текст, рукописний текст. Згорткові нейронні мережі

спеціально використовуються для цього процесу через їхню конструкцію для ідентифікації та обробки даних зображення високої чіткості.

Зображення, виходячи з використаного підходу, розглядається або як таблиця пікселів із числовими значеннями кольорів. Таблиця перетворюється на простіші компоненти, які представляють складові фізичних об’єктів та особливості зображення. Системи комп’ютерного зору можуть логічно аналізувати ці конструкції, виділяючи найважливіші, а потім упорядковуючи дані за допомогою алгоритмів вилучення ознак і алгоритмів класифікації.

Хоча нейронні мережі забирають багато часу, коли справа доходить до навчання даних, кінцеві результати краще за попередні розглянуті методи, і їх застосування дуже успішне.

1.4 Формалізація вхідних даних виявлення аномалій мозку

Процес збору даних має вирішальне значення для розробки ефективної моделі для будь-якої системи. Якість і кількість набору даних безпосередньо впливають на процес прийняття рішень моделями. І ці два фактори визначають надійність, точність і продуктивність алгоритмів на основі штучного інтелекту. Як наслідок, збір і структурування даних часто займає більше часу, ніж власне навчання моделі на цих даних.

За збором даних слідує анотація зображень, процес ручного надання інформації про істину в даних. Простими словами, анотація зображень — це процес вказівки на ті описи картинок, які модель повинна навчитися виявляти, наприклад, розташування, типу об’єктів, його клас чи характеристика.

Власні навчальні набори даних для машинного навчання можна створити шляхом збору за допомогою програмних інструментів для веб-скрепінгу, камер та інших пристроїв із датчиком, як от мобільний телефон, відеокамера, веб-камера. Незалежно від джерела збору даних, важливо узгоджувати дані з конкретними цілями та характеристиками задач машинного навчання або комп’ютерного зору. Крім того, необхідно власноруч анотувати дані та

позначити їх належним чином, щоб вони добре відповідали поставленій для моделі задачі. Для цього також існують інструменти ц вільному доступі.

Загальнодоступні набори даних містять колекції даних для машинного навчання, деякі з яких містять мільйони даних і величезну кількість анотацій, які можна повторно використовувати для навчання або тонкого налаштування моделей. Порівняно зі створенням власного спеціального набору даних шляхом збору відеоданих або зображень, використовувати загальнодоступний набір даних набагато швидше та дешевше. Використання повністю підготовленого набору даних є сприятливим, якщо завдання виявлення включає загальні об’єкти або ситуації і не є дуже специфічними.

Для даного дослідження було обрано датасет, що містить МРТ-зображення мозку разом із масками сегментації аномалій FLAIR. Ослаблене рідиною інверсійне відновлення (FLAIR) — це спеціальна послідовність відновлення інверсії з тривалим часом інверсії [32]. Була винайдена доктором Гремом Байддером на початку 1990-х. FLAIR можна використовувати як із тривимірним зображенням (3D FLAIR), так і з двовимірним зображенням (2D FLAIR). Тканина мозку на зображеннях FLAIR зображає сіру речовину яскравішу за білу, а спинномозкова рідина темна. Незважаючи на дуже довгий час візуалізації, техніка FLAIR неодноразово зарекомендувала себе, виявляючи широкий спектр уражень, включаючи кортикальні, перивентрикулярні та менінгеальні захворювання, які важко було побачити на звичайних зображеннях. До кінця 1990-х років час обробки зображень скоротився, і FLAIR став стандартним протоколом для рутинної обробки зображень.

Збір даних атласу генома раку низької гліоми (TCGA-LGG) [33] є частиною більшої праці зосередженої на зв’язуванні фенотипів раку з генотипами шляхом надання клінічних зображень, поєднаних із суб’єктами з Атласу геному раку (TCGA). Клінічні, генетичні та патологічні дані зберігаються на порталі даних Genomic Data Commons (GDC), тоді як радіологічні дані зберігаються в The Cancer Imaging Archive (TCIA).

Узгоджені ідентифікатори пацієнтів TCGA дозволяють досліджувати бази даних TCGA/TCIA щодо кореляції між генотипом тканини, радіологічним фенотипом та результатами пацієнтів. Тканини для TCGA були зібрані з багатьох сайтів по всьому світу, щоб досягти своїх цільових показників, зазвичай близько 500 зразків для кожного типу раку. З цієї причини набори даних зображення також надзвичайно неоднорідні з точки зору способів сканування, виробників і протоколів отримання. У більшості випадків зображення були отримані в рамках звичайного догляду, а не в рамках контрольованого дослідження або клінічного випробування.

Для дослідження було використано частину загального датасету. Дані мають інформацію по 110 пацієнтам, включеним до колекції гліом нижчого класу The Cancer Genome Atlas (TCGA), з наявними принаймні послідовністю ослабленого рідиною інверсії (FLAIR) та наявними даними геномного кластера. Оригінальний датасет містить інформацію по 199 пацієнтам з 241183 зображеннями. Його можна знайти на платформі Kaggle [34], що відома серед спеціалістів по даним наявнісю великої кількості датасетів у відкритому доступі для досліджень.

Використані дані мають зображення як уражених так і здорових ділянок мозку. Незважаючи на те, що всі пацієнти мали пухлину, не на всіх МРТ сканах їх видно, оскільки знімки зображають мозок шар за шаром. Так більшість зрізів не мають гліом. В загалом 2556 зображень без пухлинам (рис. 1.2) і 1373 мають аномалії (рис. 1.1). Приклад на наступних картинках.

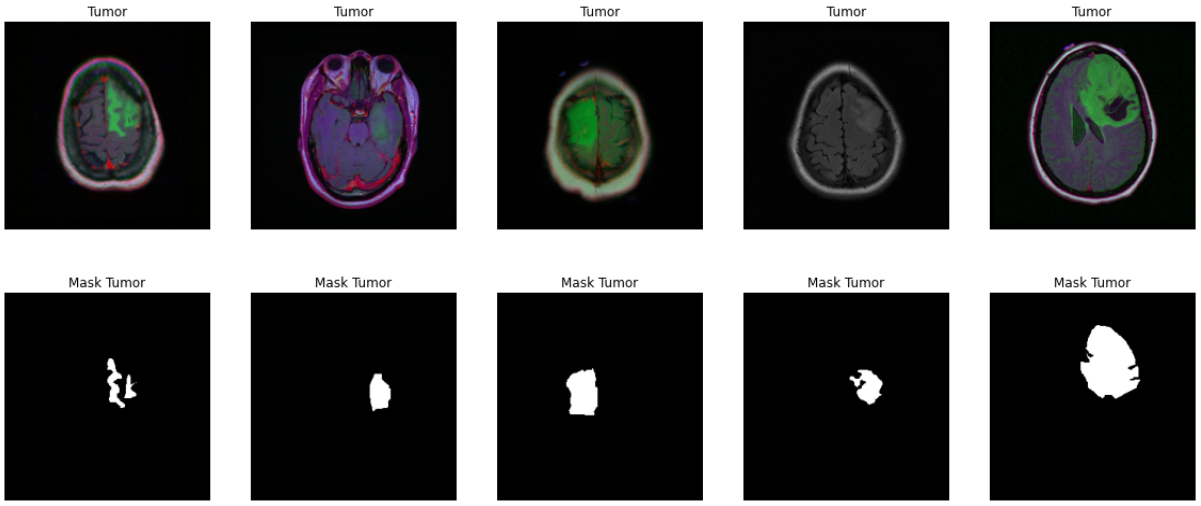


Рисунок 1.1 – Приклад зображень з датасету з пухлинами і їх маски.

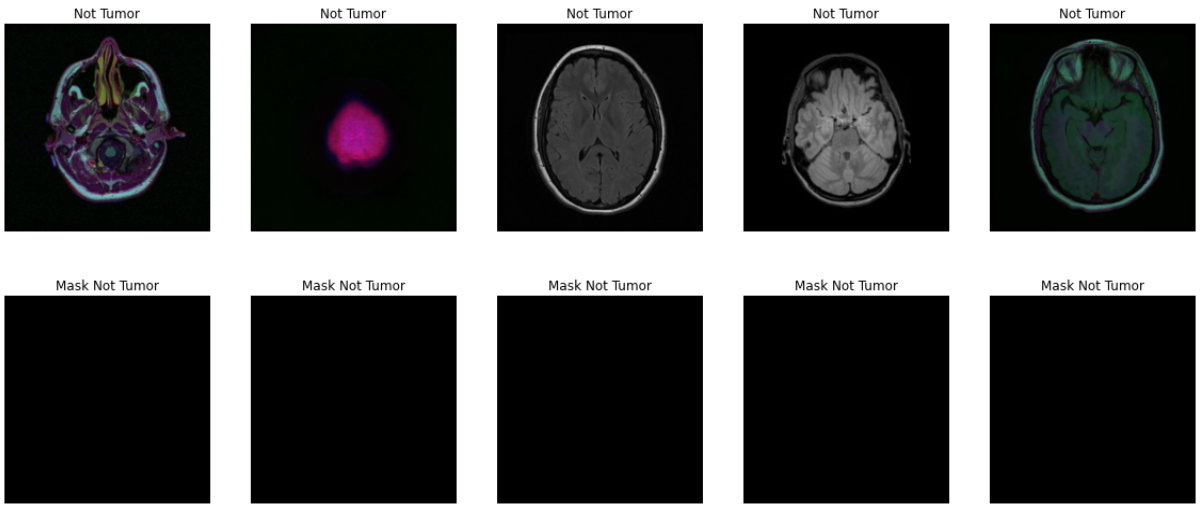


Рисунок 1.2 – Приклад зображень з датасету з без пухлин.

Слід зауважити, що оригінальні зображення не є чорно-білими, як зазвичай МРТ знімки, це обумовлено тим, що було використано FLAIR. Так кожне зображення містить 3 канали - червоний, синій та зелений. Інформації про спосіб конвертування у звичний для розуміння формат не знайдено. Тож потенційно це накладає обмеження на майбутнє користування системою, проте концепції, що були досліджені у ході роботи можливо застосувати і для звичайних МРТ/КТ знімків. Рішення обрати саме таке представлення було

зроблено через те, що даний вид медичної візуалізації краще себе показав у задачі сегментації, що було описано вище.

1.5 Постановка задачі та висновки

Отже, за результатами першого розділу можна стверджувати, що завдання сегментації, а зокрема у медичній сфері є актуальним і активно досліджується науковцями. Оскільки об’єктом дослідження виступає пошук утворень головного мозку, за допомогою сегментації, то в наступних розділах буде висвітлено саме практичне реалізації питання.

Необхідно вивчити методи глибокого навчання для сегментації зображень та вибрати підходящу модель та специфікації для її навчання. Використання штучних нейронних мереж для виявлення аномалій мозку на зображеннях МРТ мозку за FLAIR процедурою, а також побудова програмного продукту для їх зручного використання необхідно описати. Такий вибір методу сегментації обрано з урахуванням того факту, що він дає найточніший результат, до того ж не вимагає побудови двох систем, першу для отримання ознак, а другу для виокремлення з них аномалій.

Також треба визначити, якими методами глибинного навчання слід скористатися та реалізувати програмну імплементацію моделі пошуку аномалій мозку. Додатково слід визначити спосіб для надання користувачам можливості доступу до її використання у майбутньому і формалізувати виявлення аномалій мозку методами Data Science у вигляді концептуальної моделі.

РОЗДІЛ 2. МЕТОД НА ОСНОВІ НЕЙРОННИХ МЕРЕЖ ДЛЯ СЕГМЕНТАЦІЇ МЕДИЧНИХ ЗОБРАЖЕНЬ

Для вирішення поставленого завдання пошуку аномалій мозку, на основі огляду з розділу 1, було обрано метод глибокого навчання, тобто з використанням нейронних мереж. Тож в цьому розділі опишемо що це таке та основні теоретичні відомості щодо обраного алгоритму.

2.1 Deep learning підхід для сегментації медичних зображень

Глибоке навчання — це підзадача машинного навчання, яке по суті є нейронною мережею з трьома або більше шарами. Ці нейронні мережі намагаються імітувати поведінку людського мозку — хоча й далеко не відповідати його здібностям — дозволяючи йому «вчитися» на великих обсягах даних. Хоча нейронна мережа з одним шаром все ще може робити приблизні прогнози, додаткові приховані шари можуть допомогти оптимізувати та уточнити для точності.

Якщо глибоке навчання є також і машинним навчанням, то чим вони відрізняються? Глибоке навчання відрізняється від класичного машинного навчання типом даних, з якими воно працює, і методами, за допомогою яких воно навчається.

Алгоритми машинного навчання використовують структуровані, зазвичай розмічені, дані для прогнозування, тобто конкретні функції визначаються з вхідних даних для моделі та організовуються в таблиці. Це не обов’язково означає, що використовуються неструктуровані дані, але якщо це так, то вони зазвичай проходить певну попередню обробку, щоб упорядкувати дані у структурований формат. Глибоке навчання усуває частину попередньої обробки даних, яка зазвичай пов’язана з машинним навчанням. Ці алгоритми можуть приймати й обробляти неструктуровані дані, як-от текст і зображення, а також автоматизують вилучення ознак, усуваючи деяку залежність від експертів чи проміжних алгоритмів. Методами глибокого навчання можна визначити, які особливості, наприклад, границі на зображеннях, є найважливішими,

щобвирішувати поставленні задачі. У машинному навчанні ця ієрархія функцій встановлюється вручну інженером. В процесі навчання, за допомогою градієнтного спуску та зворотного поширення, алгоритми глибокого навчання шукають оптимальні характеристики зважаючи на точність прогнозу.

Ідея персептронів, попередників штучних нейронів, полягає в тому, що можна імітувати певні частини нейронів головного мозку, такі як дендрити, тіла клітин і аксони, використовуючи спрощені математичні моделі того, які обмежені знання ми маємо про їхню внутрішню роботу: сигнали можна отримувати від дендритів і відправив вниз аксон, як тільки було отримано достатню кількість сигналів. Цей вихідний сигнал потім може бути використаний як інший вхід для інших нейронів, повторюючи процес. Деякі сигнали важливіші за інші і можуть викликати легше збудження деяких нейронів. Зв’язки можуть ставати міцнішими або слабшими, нові зв’язки можуть з’являтися, а інші можуть переставати існувати. Науковці можуть імітувати більшу частину цих процесів, придумавши функцію, яка отримує список зважених вхідних сигналів і виводить якийсь сигнал, якщо сума цих зважених вхідних даних досягає певного значення (рис. 2.1 [35]). Зауважте, що ця спрощена модель не імітує ні створення, ні руйнування зв’язків між нейронами та ігнорує час сигналу. Однак сама ця обмежена модель достатньо потужна, щоб працювати з простими завданнями такими як класифікація.

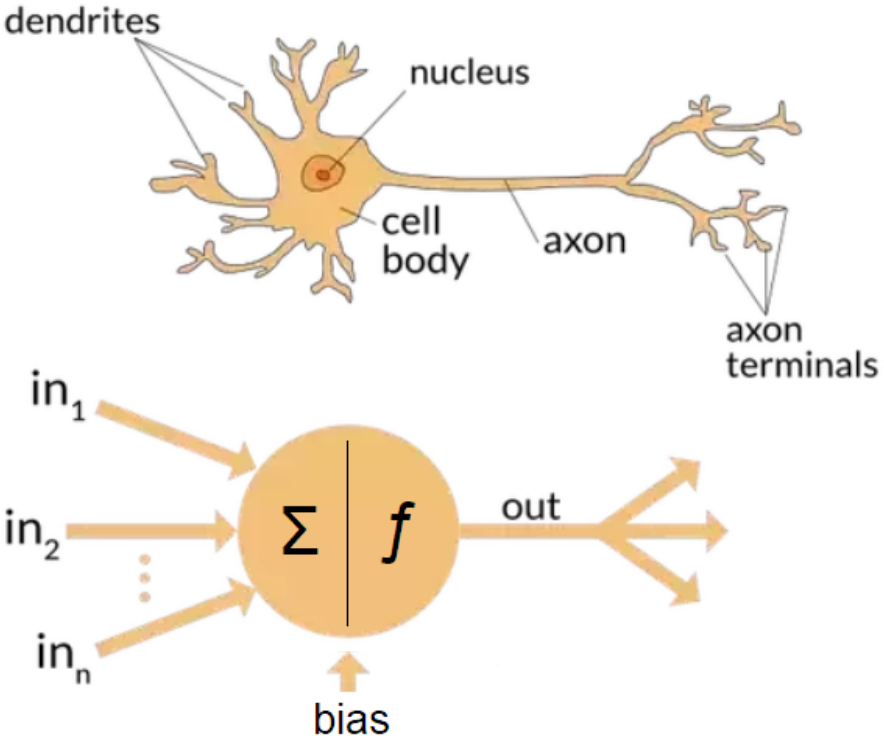


Рисунок 2.1 – Біологічний і штучний нейрон

Однак недоліки цього підходу були швидко усвідомлені, оскільки один шар персептронів сам по собі не в змозі вирішити проблеми нелінійної класифікації, наприклад, вивчення простої функції XOR. Цю проблему можна подолати лише за допомогою кількох прихованих шарів. Однак не існує простого і дешевого способу навчання кількох шарів персептронів, крім випадкового підштовхування всіх їх ваги, тому що немає способу визначити, який невеликий набір змін в кінцевому підсумку значною мірою вплине на вихід інших нейронів далі. Цей недолік призвів до того, що дослідження штучної нейронної мережі роками зупинялися. Тоді новому виду штучних нейронів вдалося вирішити цю проблему, трохи змінивши певні аспекти в своїй моделі, що дозволило з’єднати кілька шарів без втрати можливості їх тренувати. Замість того, щоб працювати як перемикач, який міг приймати та виводити лише двійкові сигнали, тобто отримувати або 0, або 1 залежно від відсутності чи присутності сигналу, штучні нейрони замість цього будуть використовувати значення з безперервними функціями активації.

Отже, глибокі нейронні мережі складаються з кількох шарів взаємопов’язаних вузлів, кожен з яких будується на попередньому шарі для уточнення й оптимізації передбачення або категоризації. Таке просування обчислень по мережі називається прямим поширенням. Вхідні та вихідні шари глибокої нейронної мережі називаються видимими шарами. Вхідний рівень — це місце, де модель глибокого навчання приймає дані для обробки, а вихідний рівень — це місце, де робиться остаточне передбачення або класифікація. Приклад мережі на рисунку 2.2 [36].

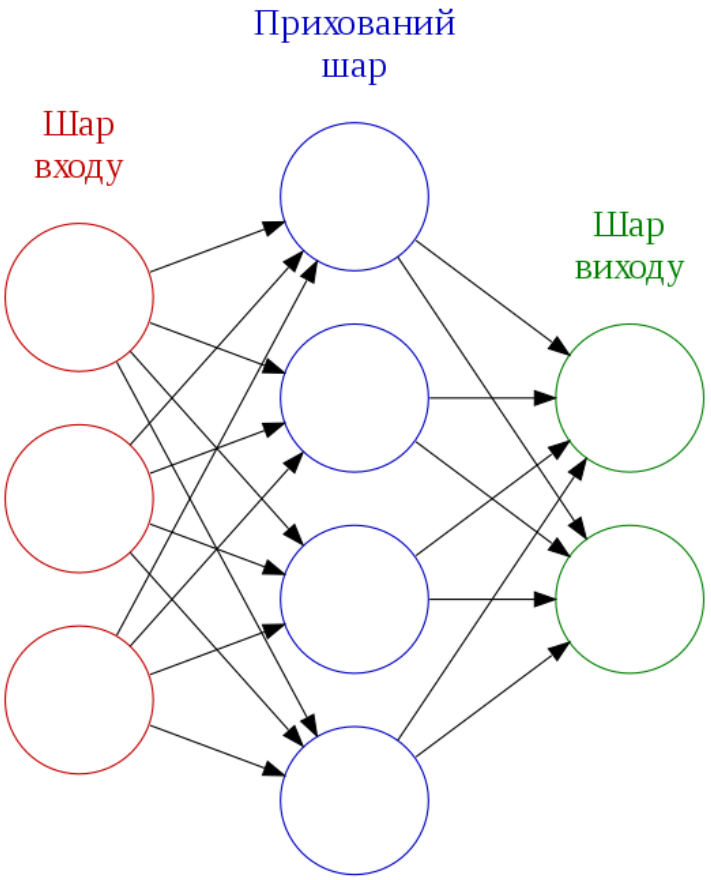


Рисунок 2.2 – Біологічний і штучний нейрон

Інший процес, який називається зворотним поширенням, використовує алгоритми, такі як градієнтний спуск, для обчислення помилок у передбаченнях, а потім коригує ваги та зміщення функції шляхом переміщення

назад через шари, щоб навчити модель. Разом пряме та зворотне поширення дозволяють нейронній мережі робити прогнози та відповідно виправляти будь-які помилки в них. З часом алгоритм стає точнішим.

Технічно в прямому поширенні виконується два основних кроки: сумування добутку і проходження його через функцію активації. Перший крок означає множення вектора ваги на заданий вхідний вектора це триває до останнього шару, де приймається рішення. Далі сума добутку ваги та вхідного вектора пропускається через функцію активації, щоб отримати вихідний шар. І тоді вихід одного шару стає входом наступного шару, який потрібно помножити на вектор ваги в цьому шарі. І цей процес триває до моменту активації вихідного шару. Формально кожен шар можна описати так:

$$u = activation(\sum_j w_{ij} x_j + b_i), \quad (2.1)$$

де activation - функція активації;

i - номер шару мережі;

j - номер елементу шару;

w_{ij} - ваги елементу j шару i;

b_i - вектор зміщення.

Зворотне поширення обчислює градієнт функції втрат по відношенню до ваги мережі. Включає в себе 3 кроки: обчислення помилки, знаходження різниці, сумування добутку. Спершу рахується різниця між очікуваним результатом і прогнозованим результатом, отриманим у прямому проході. Потім помилку множать на похідну, щоб отримати різницю, множать її на вхідний вектор і сумують. Головним, найдовшим і обчислювально найскалднішим тут є процес отримання похідної і він базується на привилі диференціації складної функції або chain rule [37] і має вигляд:

$$\frac{dz}{dx} = prod(\frac{dz}{dy} * \frac{dy}{dx}) \quad (2.2)$$

Проте навчання цих мереж було настільки дорогим у обчислювальному відношенні, що люди рідко використовували їх для завдань машинного навчання, бо лише віднедавна стало легше отримати великі обсяги прикладних даних, а комп'ютери стали набагато швидше. Оскільки штучні нейронні мережі важко навчати і вони не є точними моделями того, що насправді відбувається в нашій голові, більшість вчених все ще вважали їх глухими кутами машинного навчання. Ажіотаж повернувся, коли в 2012 році архітектурі глибокої нейронної мережі AlexNet вдалося вирішити проблему класифікації великого набіру даних із понад 14 мільйонами анотованих зображень. Відтоді нейронні мережі знову стали актуальними.

Машинне навчання, зокрема і глибоке, може відображати вхідні функції та вихідні дані ефективніше, ніж люди, особливо в областях, де дані доступні лише у незрозумілій для нас формі: ми не бачимо зображення як купу чисел, що представляють значення кольорів і країв, але моделі не мають проблем з навчанням із такого представлення. Моделі машинного навчання, включаючи моделі глибокого навчання, вивчають зв'язки у представленні даних. Це означає, що якщо представлення є неоднозначним і залежить від контексту, навіть найточніші моделі не вдасться навчитися відповідати на питання. Однак, якби люди коли-небудь побудували б таку розумну, як ми, машину, вона автоматично була б кращою за нас, завдяки перевагам у швидкості. Вже зараз глибоке навчання керує багатьма програмами та службами штучного інтелекту, які покращують автоматизацію, виконуючи аналітичні та фізичні завдання без участі людини. Технологія глибокого навчання лежить в основі повсякденних продуктів і послуг таких як: цифрові помічники, голосові пульти від телевізора, виявлення шахрайства з кредитними картками, а також новітніх технологій самокерованих автомобілів чи управління голосом.

2.2 Типи глибокого навчання

Оскільки існує багато нейронних мереж та їх специфікацій, то слід дослідити деякі з них аби обрати підходящі для подальшої прикладної реалізації з урахуванням задачі сегментації медичних зображень.

2.2.1 За підходом до навчання

Машинне навчання та моделі глибокого навчання здатні до різних типів навчання, які відрізняються за характеристиками і наявністю вхідних даних. Виділяють наступні типи [38]:

- Навчання з вчителем означає наявність повного набору розмічених даних під час навчання алгоритму. Кожен приклад у наборі навчальних даних позначений правильною відповіддю, яку алгоритм повинен передбачити. Дві основні області, де навчання з вчителем корисно: проблеми класифікації та проблеми регресії.
- Навчання без вчителя. Чисті, ідеально позначені набори даних знайти непросто. А іноді дослідники задають питання алгоритму, на які вони не знають відповіді. Ось тут і приходить навчання без вчителя. Набір навчальних даних – це набір прикладів без конкретного бажаного результату або правильної відповіді. Потім нейронна мережа намагається автоматично знайти структуру в даних, витягуючи корисні функції та аналізуючи їх структуру. Популярні задачі з цієї категорії: кластеризація, виявлення аномалій, асоціацій, автокодери.
- Напівконтрольоване навчання — коли навчальний набір даних із розміченими, так і не розміченими даними. Цей метод особливо корисний, коли вилучення відповідних ознак із даних є складним, а маркування прикладів є трудомістким завданням для експертів. Популярне у медичному домені, зокрема на медичних зображеннях, такі як КТ або МРТ. Підготовлений рентгенолог може пройти і позначити невелику підгрупу сканувань на наявність пухлин або захворювань, а решту зробить алгоритм.

- Навчання з підкріпленням. У цьому типі машинного навчання моделі намагаються знайти оптимальний шлях для досягнення певної мети або покращення ефективності виконання конкретного завдання. Коли агент виконує дії, спрямовані до мети, він отримує винагороду. Загальна мета: передбачити найкращий наступний крок, щоб отримати найбільшу остаточну винагороду. Ця техніка особливо корисна для навчальних роботів, які приймають ряд рішень у таких завданнях, як керування автономним транспортним засобом або управління запасами на складі.

2.2.2 За видами архітектур

На початку розділу описано найпростіший тип глибокої нейронної мережі в найпростіших термінах. Проте реальні мережі значно складніші та їх велика кількість, кожна відповідає своєму завданню. Можна виділити такі типи:

- Штучна нейронна мережа (ANN)
- Згорткові нейронні мережі (CNN) [39]
- Рекурентні нейронні мережі (RNN) [40]
- Генеративні змагальні мережі (GAN) [41]
- Графові нейронні мережі (GNN) [42]
- Трансформери (Transformers) [43]

Штучна нейронна мережа — це група з кількох персептронів на кожному шарі. Доволі проста але зрозуміла архітектура. Рисунок 2.2 демонструє одношарову ANN. В таких мережах кожен вихід подається на вхід кожному входу наступного шару. Може застосовуватися до даних представлених у табличному вигляді. Також зображення можна перетворити на простий вектор чисел, позбавившись частини інформації і пропустити через цю мережу, але результат буде не найкращим.

Згорткові нейронні мережі використовуються в основному в комп’ютерному баченні. Можуть виявляти особливості та закономірності в зображенні. У 2015 році CNN вперше переміг людину в задачі розпізнавання

об’єктів. Центральним у згортковій нейронній мережі є згортковий шар, який дає мережу назву. Цей шар виконує операцію під назвою згортка.

У контексті згорткової нейронної мережі згортка — це лінійна операція, яка включає множення набору вагових коефіцієнтів на вхідні дані, подібно до традиційної нейронної мережі. Враховуючи, що методика була розроблена для двовимірного входу, множення виконується між масивом вхідних даних і двовимірним масивом ваг, який називається фільтром або ядром. Формула фільтру має наступний вигляд [44]:

$$G(x,\ y,\ \sigma) = \frac{1}{2\pi\sigma^2} e^{\frac{-\left(x^2+y^2\right)}{2\sigma^2}},\ x,\ y\ = \overline{-\frac{k}{2}, \frac{k}{2}},\quad (2.3)$$

де k - розмір ядра, що завжди непарне число,

σ - стандартне відхилення розподілу Гауса.

Фільтр менший за вхідні дані, а тип множення, який застосовується між вхідним фрагментом розміром з фільтр, і фільтром є точковим добутком. Точковий добуток — це поелементне множення між фрагментом розміру фільтра вхідного і фільтра, яке потім підсумовується, що завжди призводить до єдиного значення. Оскільки це призводить до єдиного значення, операцію часто називають скалярним добутком.

Використовують фільтр, менший за вхід, оскільки це дозволяє помножити один і той же фільтр, набір ваг, на вхідний масив кілька разів у різних точках входу. Зокрема, фільтр застосовується систематично до кожної частини, що перекривається, або фрагмента розміру фільтра вхідних даних, зліва направо, зверху вниз.

Якщо фільтр призначений для виявлення певного типу об’єкта у вхідних даних, то систематичне застосування цього фільтра по всьому вхідному зображенню дає фільтру можливість виявити цю функцію в будь-якому місці зображення. Результатом одноразового множення фільтра на вхідний масив є

одне значення. Оскільки фільтр застосовується кілька разів до вхідного масиву, результатом є двовимірний масив вихідних значень, які представляють фільтрацію вхідних даних. Таким чином, двовимірний вихідний масив цієї операції називається картою ознак.

Рекурентні нейронні мережі зазвичай використовуються в програмах розпізнавання природної мови та мовлення, оскільки вони використовують послідовні дані або дані часового ряду.

Подібно до прямих і згорткових нейронних мереж, рекурентні нейронні мережі використовують дані для навчання. Вони відрізняються своєю пам’яттю, оскільки вони беруть інформацію з попередніх входів, щоб впливати на поточний вхід і вихід. У той час як традиційні глибокі нейронні мережі припускають, що входи та виходи незалежні один від одного, вихід рекурентних нейронних мереж залежить від попередніх елементів у послідовності.

Проста RNN має петлю зворотного зв’язку, яка повертає вихід більш пізніх шарів до більш раннього. Розгорнута мережа дуже схожа на нейронну мережу з прямим зв’язком. Іншою відмінною характеристикою рекурентних мереж є те, що вони мають спільні параметри на кожному рівні мережі. У той час як мережі прямого зв’язку мають різні ваги в кожному вузлі, рекурентні нейронні мережі мають однаковий параметр ваги в кожному шарі мережі. Тим не менш, ці ваги все ще коригуються в процесі зворотного поширення та градієнтного спуску, щоб полегшити навчання.

Рекурентні нейронні мережі використовують алгоритм зворотного поширення через час (BPTT [45]) для визначення градієнтів, який дещо відрізняється від традиційного зворотного поширення, оскільки він специфічний для даних послідовності. Принципи в BPTT такі ж, як і у традиційного зворотного поширення, коли модель тренується, обчислюючи помилки від вихідного рівня до вхідного. Ці розрахунки дозволяють належним чином налаштувати та підігнати параметри моделі. BPTT відрізняється від традиційного підходу тим, що BPTT підсумовує помилки на кожному кроці

часу, тоді як мережі з прямим зв’язком не повинні підсумовувати помилки, оскільки вони не поділяють параметри на кожному шарі.

Завдяки цьому процесу RNN, як правило, стикаються з двома проблемами, відомими як градієнти, що вибухають, і градієнти, що зникають. Коли градієнт занадто малий, він продовжує зменшуватися, оновлюючи параметри ваги, доки вони не стануть незначними, тобто. 0. Коли це відбувається, алгоритм більше не навчається. Вибухові градієнти виникають, коли градієнт занадто великий, створюючи нестабільну модель. У цьому випадку ваги моделі виростуть занадто великими, і в кінцевому підсумку вони будуть представлені як NaN. Одним із рішень цих проблем є зменшення кількості прихованих шарів у нейронній мережі, усуваючи частину складності моделі RNN.

Генеративні змагальні мережі — це клас алгоритмів штучного інтелекту, що використовуються в навчанні без учителя, реалізовані системою двох штучних нейронних мереж, які змагаються одна з одною в рамках гри з нульовою сумою. Ця методика дозволяє створювати фотографії, які для для людини виглядають як справжні та мають багато реалістичних елементів. GAN захоплюють своєю здатністю генерувати реалістичні приклади в ряді проблемних областей, таких як переклад фотографій літа на зиму або з дня до ночі, а також при створенні фотореалістичних фотографій об’єктів, сцен і людей, які навіть люди не можуть розпізнати як підробки.

Генеративні змагальні мережі є підходом до генеративного моделювання з використанням методів глибокого навчання, таких як згорткові нейронні мережі. Генеративне моделювання — це неконтрольована задача в машинному навчанні, яка включає автоматичне виявлення та вивчення закономірностей у вхідних даних таким чином, що модель можна використовувати для генерування або виведення нових прикладів, які правдоподібно можна було б взяти з вихідного набору даних. GAN – це розумний спосіб навчання генеративної моделі, формуючи проблему як проблему навчання під наглядом за допомогою двох підмоделей: моделі генератора, яку навчають для створення нових прикладів, і

моделі дискримінатора, яка намагається класифікувати приклади як реальні або підробка. Дві моделі навчаються разом у змагальній грі з нульовою сумою, поки модель дискримінатора не буде обдурена приблизно в половині часу, що означає, що модель генератора генерує правдоподібні приклади.

Графові нейронні мережі використовують для обробки даних, які можуть бути представлені як графи. Вони були популяризовані завдяки використанню задля пошуку властивостей різних молекул [42]. GNN можна розуміти як узагальнення згорткових нейронних мереж на дані, структуровані в графі, бо картинки можна представити у вигляді простого графу.

Дані графів настільки складні, що створюють багато проблем для існуючих алгоритмів машинного навчання. Причина в тому, що звичайні інструменти машинного навчання та глибокого навчання спеціалізуються на простих типах даних. Як зображення з однаковою структурою та розміром, які ми можемо розглядати як сітки фіксованого розміру. Текст і мова — це послідовності, тому ми можемо розглядати їх як лінійні графи. Але є більш складні графи, без фіксованої форми, зі змінним розміром невпорядкованих вузлів, де вузли можуть мати різну кількість сусідів.

У теорії графів реалізується концепція представлення вузлів у вигляді векторів. Це означає відображення вузлів у d-вимірний простір, простір меншої розмірності аніж сам граф, так що подібні вузли в графі залишаються близько один до одного. Функцією подібності може бути евклідова відстань. Саме для побудови цього представлення і потрібні графові мережі.

Існують так звані графові згорткові мережі (GCN). Вони були вперше представлені [46] як метод застосування нейронних мереж до даних, структурованих у графі. Найпростіший GCN має лише три різні оператори: згортка графу, лінійний шар, нелінійна активація. Операції зазвичай виконуються в такому порядку і разом утворюють один мережевий шар. Можна об’єднати один або кілька шарів, щоб утворити повний GCN. Але проблема цієме мережі в розрізі сегментації пухлин полягає в тому, що вона не повертає маску, а лише вивчає представлення зображення.

Трансформери — це моделі які імітують механізм уваги, роздільно зважуючи важливість кожної частини даних входу. Її використовують переважно в області обробки природної мови та комп'ютерному баченні.

Як і LSTM [47], підвид рекурентної мережі, трансформери є архітектурою для перетворення однієї послідовності в іншу за допомогою двох частин: кодера і декодера, але вона відрізняється від раніше існуючих моделей тим, що не передбачає жодних повторень мережі. Досі повторювані мережі були одним із найкращих способів фіксувати своєчасні залежності у послідовностях. Однак команда, яка перша презентувала цю мережу, довела, що архітектура лише з механізмами уваги без будь-яких RNN може покращити результати в задачі перекладу[43].

Механізми уваги дозволяють моделі витягнути інформацію з будь-якої попередньої точки послідовності. Яскравим прикладом цінності уваги є мовний переклад, де контекст має важливе значення для визначення значення слова в реченні. У системі перекладу з англійської на французьку перше слово французького виводу, швидше за все, сильно залежить від перших кількох слів англійського введення. Однак у класичній моделі LSTM, щоб отримати перше слово французького результату, моделі дається лише вектор стану останнього англійського слова. Теоретично цей вектор може кодувати інформацію про все англійське речення, даючи моделі всі необхідні знання. На практиці ця інформація часто погано зберігається LSTM. Для вирішення цієї проблеми можна додати механізм уваги: декодер отримує доступ до векторів стану кожного англійського вхідного слова, а не тільки останнього, і може дізнатися вагові коефіцієнти уваги, які диктують, скільки потрібно звертати на кожен вхідний англійський вектор стану.

При додаванні до RNN механізми уваги підвищують продуктивність. Розвиток архітектури трансформерів показав, що механізми уваги є потужними сам по собі і що послідовна повторна обробка даних не потрібна для досягнення якості як у RNN з увагою. Трансформери використовують механізм уваги без RNN, обробляючи всі входи одночасно і обчислюючи ваги уваги між

ними в послідовних шарах. Оскільки механізм уваги використовує інформацію лише про інші входи з нижчих рівнів, його можна обчислити паралельно, що призводить до покращення швидкості навчання.

Зараз трансформери – це тенденція в комп'ютерному зору. UNETR – це перша успішна архітектура-трансформер для сегментації 3D медичних зображень [48]. Вона використовує трансформатор як кодер для вивчення представлень послідовності вхідного обсягу та ефективного захоплення глобальної багатомасштабної інформації, а також слідуючи успішній «U-подібній» мережі. Проте залишається велика проблема з цими моделями - вони вимагають надто потужних оточень для навчання та багато часу. До того ж виконання триває довше за більш традиційні нероні мережі.

2.2.3 Функції активації

Також важливим елементом є функції активації, що мають на меті додати нелінійності в процес навчання. Таким чином, на відміну від біологічних нейронів, штучні нейрони не просто спрацьовують, а посилають безперервні значення замість бінарних сигналів. Залежно від функцій активації сила цих сигналів може бути різна. Існують такі типи:

- Порогова функція активації. Ця функція активації дуже проста. В основному це класифікатор з порогом, що допомагає вирішити, який нейрон повинен бути активований. Її вихідний сигнал дорівнює 1, тобто активовано, коли значення > 0 (поріг), а в іншому випадку виводиться 0, не активовано. Похідна для випадків коли вхід є нулем не визначено. Формули функції і її похідної мають вигляд:

  $$f(x) = 0\ for\ x < 0; 1\ for\ x \geq 0 \quad (2.4)$$

  $$f'(x) = 0\ for\ x \neq 0 \quad (2.5)$$

- Функція лінійної активації також називається "ідентичність" або "відсутність активації". Вона жодним чином не змінює зважену суму

вхідних даних, а натомість безпосередньо повертає значення. Формули функції і її похідної мають вигляд:

$$f(x) = x \quad (2.6)$$

$$f'(x) = 0 \quad (2.7)$$

- Rectified Linear unit або ReLU є найбільш широко використовуваною функцією активації, яка коливається від 0 до нескінченності. Вона повертає x, якщо x додатне, а 0 - інакше. Формули функції і її похідної мають вигляд:

$$f(x) = 0 \; for \; x < 0; x \; for \; x \geq 0 \quad (2.8)$$

$$f'(x) = 0 \; for \; x < 0; 1 \; for \; x \geq 0 \quad (2.9)$$

- Leaky ReLU. Функція Leaky ReLU потрібна, аби вирішити проблему «вмирання ReLU», коли всі негативні вхідні значення дуже швидко перетворюються на нуль, а у випадку з Leaky ReLU всі негативні входи зводяться до значення, близького до нуля, що вирішує основну проблему функції активації ReLU. Формули функції і її похідної мають вигляд:

$$f(x) = \alpha x \; for \; x < 0; x \; for \; x \geq 0 \quad (2.10)$$

$$f'(x) = \alpha \; for \; x < 0; 1 \; for \; x \geq 0 \quad (2.11)$$

- Параметрична функція ReLU — це ще один варіант ReLU, який має на меті вирішити проблему зникаючого градієнта для лівої половини осі. Ця функція надає нахил від'ємної частини функції як аргумент a. Виконуючи зворотне поширення, вивчається найбільш відповідне значення a.

$$f(x) = max(ax, x) \quad (2.12)$$

$$f^{'}(x) = 1\ for\ ax < x; a\ for\ ax \geq x \text{ (2.13)}$$

- Експоненціальна лінійна одиниця, або скорочено ELU, також є варіантом ReLU, який змінює нахил від'ємної частини функції. ELU використовує криву логарифмів для визначення від'ємних значень. ELU стає плавним повільно та вирішує проблему зникаючого градієнта.

$$f(x) = \alpha(e^{x} - 1)\ for\ x < 0; x\ for\ x \geq 0 \text{ (2.14)}$$

$$f^{'}(x) = f(x) + \alpha\ for\ x < 0; 1\ for\ x \geq 0 \text{ (2.15)}$$

- Функція сигмоїдної активації використовується здебільшого, оскільки вона застосовує імовірнісний підхід до прийняття рішень і коливається в межах від 0 до 1, тому, коли потрібно прийняти рішення або передбачити результат, слід використати цю функцію. Формули функції і її похідної мають вигляд:

$$f(x) = \frac{1}{1 + e^{-x}} \text{ (2.16)}$$

$$f^{'}(x) = f(x)(1 - f(x)) \text{ (2.17)}$$

- Гіперболічна функція активації (Tanh). Ця функція активації дещо краща, ніж сигмоїдна функція. Вона також використовується для прогнозування або розмежування між двома класами, але вона відображає негативний вхід лише в негативну величину і коливається в діапазоні від -1 до 1. Формули функції і її похідної мають вигляд:

$$f(x) = \frac{2}{1 + e^{-2x}} - 1 \text{ (2.18)}$$

$$f^{'}(x) = 1 - f(x)^{2} \text{ (2.19)}$$

- Softmax використовується головним чином на останньому шарі, тобто вихідному для прийняття рішень. Ця функція генерує вихід, який коливається в межах від 0 до 1 і із сумою ймовірностей, що дорівнює 1. Формула має вигляд:

$$P(u = j \mid \theta^{(i)}) = e^{\theta^{(i)}} / \sum_{j=0}^{k} e^{\theta_k^{(i)}}, \quad (2.20)$$

де θ - матриця з рядками, що позначають правильний клас, де 0 неправильний, а 1 правильний;

j - клас для якого обчислюється функція;

k - кількість класів.

Отже можна визначити, що перспективи для успішного навчання матимуть моделі з згортовими шарами. Таке рішення обумовлено тим, що ці нейромережі створювалися саме для обробки картинок і добре вписуються в концепцію навчання з вчителем, адже є розмічені дані для цього. Як функції активації слід обирати з сімейства ReLU чи сігмоїдну чи параболічну. На останній шар передбачення можна використовувати софтмакс функцію, бо вона повертає ймовірність від 0 до 1.

2.3 Опис архітектури Unet

Для реалізації задачі пошуку аномалій було обрано одну з відомих архітектур згорткових нейронних мереж Unet. U-Net — це згорткова нейронна мережа, розроблена для сегментації біомедичних зображень на факультеті комп'ютерних наук Фрайбурзького університету [50]. Мета U-Net полягає в тому, щоб охопити як особливості контексту, так і локалізацію. Основна ідея реалізації полягає у використанні послідовних стискаючих шарів, за якими відразу йдуть оператори підвищення дискретизації для досягнення більш високої роздільної здатності на вхідних зображеннях.

Мережа заснована на повністю згортковій мережі, а її архітектура була модифікована та розширена для роботи з меншою кількістю навчальних зображень і отримання більш точної сегментації. Сегментація зображення розмірністю 512 на 512 пікселів за допомогою цієї мережі займає менше секунди на сучасному графічному процесорі. Реалізація за допомогою мови програмування Python та фреймворку TensorFlow у Додатку Б.

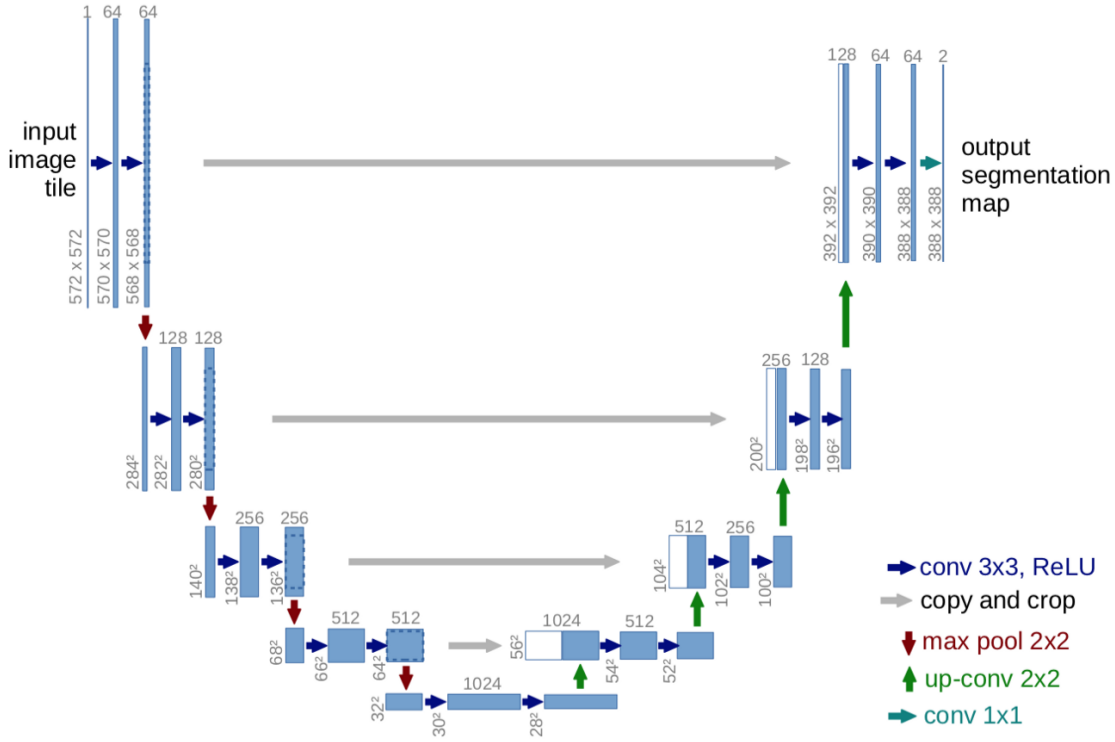


Рисунок 2.3 – Архітектура U-Net [49]

Легко помітити (рис. 2.3), що мережа має U-подібну форму, звідси і назва. Архітектура є симетричною і складається з двох основних частин — ліва частина називається стискаючою частиною, яка складається із загального згорткового процесу, права частина — це шлях розширення, який складається з обернених згорткових шарів.

Опишемо стискаючий шлях. Кожен етап складається з двох шарів згортки з ядром 3 на 3 і активацією ReLU, і кількість каналів змінюється від 1 → 64, оскільки процес

згортки збільшує глибину зображення. Після кожного кроку відбувається max pooling, тобто об'єднання сусідніх пікселів з урахуванням найбільшого. У результаті розмір зображення зменшується вдвічі . Процедуру повторюють ще 3 рази. Наприкінці є 2 згорткових шари, але без об'єднання. Розмір зображення в цей момент становить 28x28x1024.

На розширюючому шляху зображення збільшується до початкового розміру. Транспонована згортка — це техніка, яка розширює розмір зображень. По суті, вона виконує деяке доповнення до зображення з наступною операцією згортки 3 на 3 та ReLU активацією. Після транспонованої згортки зображення з'єднується з відповідним зображенням зі шляху стискання і разом створює зображення з вдвічі більшою кількістю каналів. Причина полягає в тому, щоб об'єднати інформацію з попередніх шарів, аби отримати більш точний прогноз. Так само цей процес повторюється ще 3 рази:

Останній крок архітектури — змінити форму зображення, щоб задовольнити наші вимоги щодо передбачення. Останній шар являє собою шар згортки з 1 фільтром розміром 1 на 1. У результаті отримуємо зображення такого самого розміру, як і вхідне.

Загальною формулою звичайної операції згортки є:

$$g(x,\ y) = w * f(x,y) = \sum_{dx=-a}^{a} \sum_{dy=-b}^{b} w(dx,\ dy) f(x - dx, y - dy), \quad (2.21)$$

де g(x,y) – вихідне зображення,

f(x,y) – вхідне зображення,

w — ядро згортки, є квадратною матрицею.

Опишемо детальніше механізм роботи транспонованої згортки (рис 2.4). Розглянемо матрицю чисел розміром 2 на 2, яку потрібно збільшити до розміру

3 на 3. Беремо ядро розміром 2 на 2 з одиничним кроком і нульовим відступом. Беремо кожен елемент вхідної матриці і множимо його на кожен елемент ядра. Деякі елементи отриманих матриць перекриваються. Щоб вирішити цю проблему, просто додаємо числа між собою, на прикладі це середній елемент, де 0 + 2 + 2 + 0 = 4.

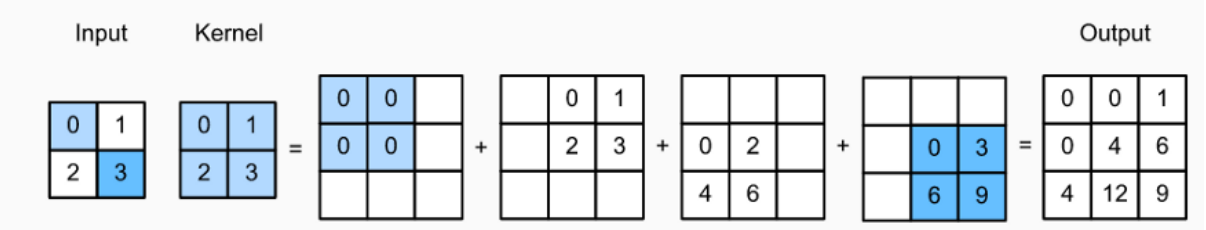


Рисунок 2.4 – Приклад роботи транспонованої згортки [51]

Транспонована згортка також відома як деконволюція, що не правильне трактування, оскільки деконволюція передбачає видалення ефекту згортки, якого ми не прагнемо досягти. Вона також відома як згортка з підвищеною дискретизацією, або дробова згортка, згортка з стрибками назад.

Цей шар має певну проблему - нерівномірне перекриття деяких частин зображення, що спричиняє артефакти. Це можна виправити або зменшити, використовуючи розмір ядра, що ділиться на крок, наприклад, взяти розмір ядра 2 або 4 при кроці 2.

Отже у даному розділі було розглянуто методи глибинного навчання, архітектури та деякі способи їх навчання. Було обрано архітектуру згортової мережі, оскільки, вона найбільш підходить під задачу сегментації зображень мозку. В наступному розділі буде описано процес навчання обраної нейромережі на мові програмування Python від обробки даних до візуалізації результатів процесу навчання.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ МЕТОДУ ПОШУКУ ПУХЛИН ЗА ДОПОМОГОЮ НЕЙРОМЕРЕЖІ U-NET

Основним інструментом реалізації методу сегментації було обрано мову програмування Python, а зокрема фреймворк Tensorflow та його розширення Keras.

Python — це високорівнева, інтерпретована мова програмування загального призначення. Вона постійно займає місце як одна з найпопулярніших мов програмування як у світі так і в Україні. Python часто використовується в проектах зі штучним інтелектом і в проектах машинного навчання.

TensorFlow — це безкоштовна бібліотека програмного забезпечення з відкритим вихідним кодом для машинного навчання та штучного інтелекту. Здебільшого він зосереджений на навчанні та застосуванні глибоких нейронних мереж [52]. Був розроблений командою Google Brain для внутрішнього використання Google у дослідженнях і виробництві. В даному досліджені було обрано саме Python. Початкова версія була випущена під ліцензією Apache License 2.0 у 2015 році [53]. Назва TensorFlow походить від операцій, які такі нейронні мережі виконують над багатовимірними масивами даних, які називаються тензорами.

TensorFlow можна використовувати в широкому спектрі мов програмування, особливо в Python, а також у Javascript, C++ і Java. Він доступний на 64-разрядных платформах Linux, macOS, Windows і мобільних обчислювальних платформах, включаючи Android та iOS. Його гнучка архітектура дозволяє легко розгортати обчислення на різних платформах (CPU, GPU, TPU) і від настільних комп’ютерів до кластерів серверів до мобільних і граничних пристроїв.

Keras — це бібліотека програмного забезпечення з відкритим вихідним кодом, яка надає інтерфейс для бібліотеки TensorFlow для комфортної роботи зі штучними нейронними мережами. Він містить численні реалізації типових будівельних блоків нейронної мережі, таких як шари, функції активації, оптимізатори та безліч інструментів, які полегшують роботу з графічними та

текстовими даними, щоб спростити кодування, необхідне для написання глибокого коду нейронної мережі. На додаток до стандартних нейронних мереж, Keras підтримує згорткові та рекурентні нейронні мережі [54].

Keras дозволяє користувачам створювати глибокі моделі на смартфонах iOS і Android, в Інтернеті або на віртуальній машині Java. Це може знадобитися в майбутньому, якщо постане потреба розробки мобільного застосунку. Необхідні дії з перетворення моделі будуть мінімальні адже початкова бібліотека підтримує виконання моделей на мобільних пристроях.

Для обробки датасету боло використано відкриті бібліотеки scikit-image [55] та OpenCV [56]. Вони надають доступ до читання та редагування зображень. Scikit-image — бібліотека обробки зображень з відкритим вихідним кодом для мови програмування Python. Включає в себе алгоритми для сегментації, геометричних перетворень, маніпуляцій з колірним простором, аналізу, фільтрації, морфології. В основному написаний на Python, а деякі основні алгоритми написані на Cython, версія що написана на C - подібних мовах що дає змогу пришвидшити обчислення, для досягнення продуктивності. OpenCV, Open Source Computer Vision Library, — це бібліотека в основному спрямованих на комп'ютерний зір у реальному часі. Починаючи з 2011 року, OpenCV має функцію прискорення GPU для операцій у реальному часі.

3.1 Підготовка даних

Застосовуючи згорткові нейронні мережі для класифікації зображень, важливим кроком є саме підготовка зображень для моделювання, наприклад, масштабування або нормалізація значень пікселів. Крім того, збільшення кількості даних може використовуватися для покращення продуктивності моделі та зменшення помилки при навчанні, а збільшення часу тестування може використовуватися для покращення прогнозної продуктивності моделі.

Оскільки використовується датасет з відкритих джерел, що вже анотовано і підготовано у необхідному форматі, то підготовка даних включає в себе лише програмну зміну зображень до одного виду для того аби модель змогла

обробити їх відповідно до трьох етапів: навчання, валідації та тестування. На першому буде власне модель навчатися, тобто оновлюватиме ваги шарів. На валідаційному буде робитися перевірка точності моделі відповідно до обраних метрик під час навчання. Останній набір даних знадобиться вже після навчання аби перевірити як модель працюватиме на реальних задачах, даних, яких вона раніше не зустрічала а також для візуалізації результатів навчання.

Датасет було поділено на три вибірки: для навчання, валідації під час навчання та тестування. У результаті отримали по 2838, 501, 590 зображення відповідно.

Одним з важливих обмежень, які існують у деяких алгоритмах машинного навчання, таких як CNN, є необхідність змінити розмір зображень у датасету до єдиного виміру. Оскільки модель не може приймати на вхід різні розміри картинок, то спершу кожне зображення було приведено до одного розміру, а саме 256 на 256 пікселів.

Часто в комп'ютерному зорі виконують перетворення кольорових зображень у відтінки сірого. Це пояснюється тим, що в багатьох об’єктах колір не потрібен для розпізнавання та інтерпретації зображення. Відтінки сірого можуть бути достатньо хорошими для розпізнавання певних об’єктів. Оскільки кольорові зображення містять більше інформації, ніж чорно-білі зображення, вони можуть додавати непотрібну складність і займати більше місця в пам’яті. У випадку використання FLAIR знімків мозку є корисним залишити інформацію про колір, адже саме цим і славиться такий вид МРТ - наявністю кольорової підсвітки певних областей, що допомагає у постановці діагнозу. Втративши цю інформацію, буде знівельовано обгрунтування застосування саме такої технології медичної візуалізації і модель не відрізнятиметься від більшості аналогів, що працюють зі звичайними чорно-білими МРТ знімками.

Для збільшення кількості тренувальних прикладів була застосована аугментація. При кожному звертанні до датасету, з певною ймовірністю до картинки застосовуються наступні види перетворень: горизонтальне віддзеркалення, поворот до 20 градусів, зсув вправо чи вліво, нахил,

збільшення. Для заповнення прогалин після трансформацій застосовується сценарій найближчого сусіда. Ці перетворення актуальні лише для тренувальних даних. Для тестових чи валідаційних зображень немає необхідності збільшувати кількість, а просто робляться перетворення, аби модель могла обробити зображення. До них належать зміна розміру картинки та перетворення її у відповідний тип даних для входу моделі.

Аугментація виконується при звертанні алгоритму навчання до картинки і не зберігається в пам'яті, тобто фактично вибірка не збільшується просто модель кожного разу приймає трохи змінені дані для більшої репрезентативності.

Також кожне зображення було нормалізоване діленням. Оскільки представлення пікселя від 0 до 255, то для базової мінімізації слід поділити кожне на 255. Таким чином модель швидше навчиться, а також це запобігає надмірному споживанню пам'яті.

Задля запобіганню виділення зайвої пам’яті використано функцію ImageDataGenerator. Об’єкти що повертає ця функція не зберігаються в пам'яті напряму, кожного разу, як процес тренування потребує нових даних, нові зображення завантажуються, а після цього пам’ять знов звільняється для наступного батчу. Батч це невелика вибірка з основної, на якій навчається модель. У даній роботі використовувалося значення 32.

Також слід зазначити, що дані для масок, себто істинних значень локалізації пухлин, та самих зображень завантажуються різним способом. Маски також є зображеннями, але мають лише один канал, на відміну від картинок мозку. Тому, для запобігання помилок, вони завантажуються окремими об’єктами з однаковим сценарієм попередньої обробки.

3.2 Специфікація навчання моделі

В цьому підрозділі опишемо головні елементи, які необхідні для навчання та були застосовані при реалізації даної роботи.

Незбалансованість класів є частою проблемою, яка виникає при спробі навчити мережі сегментації зображень. Пікселі, які потрібно сегментувати, становлять дуже невеликий відсоток від загальної кількості. Все, що потрібно було зробити моделі, — це передбачити повністю чорне зображення, тобто без сегментованих частин, і це було винагороджено точністю понад 90%. Це поширена проблема, яка зустрічається в більшості завдань сегментації зображень, де зона фону набагато більша, аніж інші класи. Тож слід використовувати метрики та функції втрат, які можуть працювати стабільно за незбалансованих даних. У Додатку В код для програмної реалізація описаних функцій на мові програмування Python, що використовувались задля навчання моделі.

3.2.1 Метрики

Оцінка алгоритму машинного навчання є важливою частиною будь-якого проекту. Вибір правильної метрики має вирішальне значення під час оцінки моделей машинного навчання. Модель може дати задовільні результати при оцінці за допомогою однієї метрики, але може погані результати при оцінці за іншими показниками.

Для відстеження за ходом навчання моделі було обрано такі метрики: dice coefficient, sensitivity, specificity, tversky index. Але перед детальним їх описом, введемо ще кілька додаткових понять:

True Positive (TP): об’єкт виявлено правильно

False Positive (FP): об’єкт виявлено, але хибно

False Negative (FN): об’єкт не виявлено за його наявності

True Negative (TN): об’єкт не виявлено, об’єкта і не було

Dice coefficient. Цей коефіцієнт добре відомий, як основний показник оцінки для сегментації зображення, але він також може служити функцією втрат. Хоча він не так широко використовується, як інші метрики, проте гарно показує себе, коли справа доходить до дисбалансу класів. Він враховує лише клас сегментації, а не фоновий клас, тому він не може домінувати над меншим

класом при розрахунках. Пікселі класифікуються як істинно позитивні (TP), хибнонегативні (FN) і хибнопозитивні (FP). Тоді формула має вигляд:

$$BSC = \frac{2TP}{2TP + FN + FP} \quad (3.1)$$

Коефіцієнт кубика є мірою перекриття передбаченої маски та істинної маскою. Він повертає оцінку в діапазоні від 0 до 1, де 1 є ідеальним перекриттям істинної маски, тобто ідеальним передбаченням.

Однак однією з підводних каменів цієї метрики, як функції втрат є можливість вибуху градієнтів. На початку навчання коефіцієнт близький до 0, що може спричинити нестабільність у навчанні через вибухові градієнти в результаті того, що мережа робить великі зміни ваг і внаслідок цього повільніше тренується. Однак цим легко керувати за допомогою пакетної нормалізації та ReLU.

Tversky index — це асиметрична міра подібності, яка є узагальненням коефіцієнта кубика та індексу Жаккарда [57]:

$$TI = \frac{TP}{TP + \alpha FN + \beta FP}, \quad (3.2)$$

де α і $\boldsymbol{\beta}$ такі дійсні числа, що $\alpha + \boldsymbol{\beta} = 1$

Індекс Тверського додає два параметри, α і $\boldsymbol{\beta}$. У випадку, коли $\alpha = \boldsymbol{\beta} = 0{,}5$, він спрощується до коефіцієнта кубика. Він спрощується до індексу Жаккарда, якщо $\alpha = \boldsymbol{\beta} = 1$. Встановивши значення $\alpha > \boldsymbol{\beta}$, можна більше штрафувати модель за хибнопозитивні передбачення. У випадку застосування цієї метрики, як функції втрат при дуже незбалансованих наборах даних, модель дає кращу дрібномасштабну сегментацію, адже має додатковий рівень контролю, ніж звичайний коефіцієнт кубку.

Чутливість і специфічність (sensitivity and specificity) математично описують точність передбачення, який повідомляє про наявність або відсутність

певного стану. Особи, для яких умова виконана, вважаються «позитивними», а ті, для яких вона не є виконаною – «негативними». Широко застосовуються у медичному домені при задачі бінарної класифікації.

Чутливість (sensitivity, TPR) описує ймовірність позитивного передбачення за умови дійсно позитивного результату:

$$TPR = \frac{TP}{TP + FN} \quad (3.3)$$

Специфічність (specificity, TNR) описує ймовірність негативного передбачення за умови, що він дійсно негативний:

$$TNR = \frac{TN}{TN + FP} \quad (3.4)$$

3.2.2 Функція втрат

Варто зазначити, що метрики відрізняється від функції втрат. Функції втрат — це функції, які показують міру продуктивності моделі і використовуються для навчання моделі машинного навчання з використанням певної оптимізації, і зазвичай диференційовані за параметрами моделі. З іншого боку, метрики використовуються для моніторингу та вимірювання ефективності моделі під час навчання та тестування, і не мають бути диференційованими. Однак, якщо для деяких завдань показник продуктивності є диференційованим, його можна використовувати і як функцію втрат, іноді з додаванням деяких регуляризацій. Саме такий випадок з Tversky index. Як функцію втрат вирішено використовувати його видозмінену версію.

Focal Tversky Loss (FTL, Фокальна Тверська функція втрат) [58] є узагальненням функції втрат оберненої до Тверського індексу. Нелінійний характер втрат дає контроль над тим, як поводиться функція при різних значеннях отриманого індексу Тверського. Має формулу:

$$FTL = (1 - TI)^{\gamma}, (3.5)$$

де $\gamma > 0$, є параметром, який контролює нелінійність втрат.

Коли γ прямує до плюс нескінченності, то і градієнт прямує до плюс нескінченності, коли індекс Тверського близький до 1. Коли γ прямує до 0, градієнт прямує до 0, коли TI близько до 1. По суті, зі значенням $\gamma < 1$ градієнт втрат вищий для прикладів, де TI > 0,5, що змушує модель зосередитися на таких прикладах. Така поведінка може бути корисною наприкінці навчання, оскільки модель все ще має стимул до навчання, навіть якщо TI наближається до стабілізації. Однак, водночас, на ранніх етапах навчання буде зосереджуватись на інших приклади, що може призвести до поганого навчання в цілому.

У разі дисбалансу класів, FTL стає корисним, коли $\gamma > 1$. Це призводить до більшого градієнта втрат для прикладів, де TI < 0,5. Це змушує модель зосередитися на більш складних прикладах, особливо на дрібномасштабних ділянках для сегментації, які зазвичай отримують низькі значення TI.

Table 1. Performance on BUS 2017 Dataset B with 40 test images

Model	Parameters	DSC	Precision	Recall
Attn U-Net + Multi-Input + DL	$\alpha = 0.5, \beta = 0.5$	0.716 ± 0.041	0.759 ± 0.092	0.751 ± 0.046
Attn U-Net + Multi-Input + TL	$\alpha = 0.7, \beta = 0.3$	0.751 ± 0.042	0.802 ± 0.073	0.768 ± 0.056
Attn U-Net + Multi-Input + FTL	$\alpha = 0.7, \beta = 0.3, \gamma = 4/3$	$\mathbf{0.804 \pm 0.024}$	$\mathbf{0.829 \pm 0.027}$	$\mathbf{0.817 \pm 0.022}$

Table 2. Performance on ISIC 2018 with 649 test images

Model	Parameters	DSC	Precision	Recall
Attn U-Net + Multi-Input + DL	$\alpha = 0.5, \beta = 0.5$	0.827 ± 0.055	$\mathbf{0.896 \pm 0.019}$	0.829 ± 0.076
Attn U-Net + Multi-Input + TL	$\alpha = 0.7, \beta = 0.3$	0.841 ± 0.012	0.823 ± 0.038	0.912 ± 0.026
Attn U-Net + Multi-Input + FTL	$\alpha = 0.7, \beta = 0.3, \gamma = 4/3$	$\mathbf{0.856 \pm 0.007}$	0.858 ± 0.020	0.897 ± 0.014

Рисунок 3.1 – Таблиця результатів навчання U-Net з Focal Tversky Loss [59].

У наборі даних BUS Фокальна Тверська функція втрат показала значні покращення в усіх категоріях. У наборі даних ISIC, хоча загальний коефіцієнт кубиків був найвищим, модель не мала найвищої точності чи узагальнення даних. Це видно з рисунку 3.1, де зображено таблицю зі статті “Нова фокальна

функція втрат Тверського з покращеною мережею Attention U-Net для сегментації уражень [59].

Хоча моделі, які тренуються за допомогою FTL, можуть не мати абсолютно найвищої точності, важливо зазначити, що баланс між точністю і узагальнення був найкращим під час навчання за допомогою FTL, що показує, що мета подолання дисбалансу класів досягнута.

3.3.3 Оптимизатор

Основна мета глибокого навчання полягає в тому, щоб створити модель, яка добре працює і дає точні прогнози. Для цього потрібна оптимізація.

Оптимізація в глибокому навчанні – це процес налаштування гіперпараметрів з метою мінімізації функції втрат за допомогою одного з методів оптимізації. Важливо мінімізувати функцію втрат, оскільки вона описує невідповідність між істинним значенням оцінюваного параметра і тим, що передбачила модель.

Яка різниця між параметрами та гіперпараметрами моделі. Перед початком навчання моделі встановлюють її гіперпараметри. Вони включають кількість, швидкість навчання, кількість епох та інші. Гіперпараметри описують структуру моделі. З іншого боку, параметри моделі отримуються під час навчання. Немає можливості отримати їх заздалегідь. Прикладами є ваги нейронних мереж. Ці дані є внутрішніми для моделі та змінюються на основі навчання на даних.

Існує багато різних типів алгоритмів оптимізації, які можна використовувати для задач оптимізації безперервних функцій при навчанні нейронних мереж. Далі буде описано лише принцип роботи одного з них, що було обрано для практичної реалізації моделі.

Для навчання обраної мережі було вирішено використовувати оптимізатор Adam. Оптимізатор Адам — це розширена версія стохастичного градієнтного спуску, яка застосовується в різних завданнях глибокого навчання, таких як комп’ютерний зір та обробка природної мови. Вперше він був представлений на

відомій конференції для дослідників глибокого навчання під назвою ICLR 2015 [60]. Його називали Адамом, оскільки використовує оцінки першого та другого моментів, градієнтів першого порядку, щоб адаптувати швидкість навчання ваг нейронної мережі.

Має перевагу в тих випадках, де потреба в пам'яті занадто мала. Також його гіперпараметри мають інтуїтивно зрозумілі інтерпретації і, отже, їх підбір полегшується [61].

Adam є комбінацією двох методів градієнтного спуску, Momentum і RMSP, які пояснюються нижче. Але ці два оптимізатори мають деякі проблеми, такі як узагальнення продуктивності. Зокрема, у одній із наукових статей говориться, що Адам бере на себе атрибути двох вищевказаних оптимізаторів і використовує їх, щоб забезпечити більш оптимізований градієнтний спуск [62].

Momentum - це алгоритм оптимізації, який враховує «експоненціально зважене середнє» та прискорює градієнтний спуск. Це розширення алгоритму оптимізації градієнтного спуску. Алгоритм гальмує зміну градієнта що, у свою чергу, призводить до зменшення розміру кроку з кожною новою точкою в просторі пошуку оптимума функції втрат. Складається з двох частин. Перша – розрахунок зміну положення (формула 3.6), а друга – оновлення старих ваг (формула 3.7).

$$update = \alpha * m_t, \quad (3.6)$$

$$w_{t+1} = w_t - update, \quad (3.7)$$

$$m_t = \beta * w_t + (1 - \beta) * (\frac{\delta L}{\delta w_t}), \quad (3.8)$$

де α є параметром, який керує рухом у просторі пошуку, який також називається швидкістю навчання,

m_t – це градієнти у момент часу t,

m_{t+1} – це градієнти у момент часу після t,

w_t - ваги в момент часу t,

w_{t+1}- ваги в момент часу після t,

∂L – похідна функції втрат,

∂w_t - похідна ваг при t,

β - середній параметр,

e - константа.

RMSP – це адаптивний алгоритм оптимізації, який є покращеною версією AdaGrad. RMSP вирішує проблеми імпульсу, моменту. Якщо в AdaGrad використовується кумулятивне підсумовування квадратів градієнтів, то в RMSP буде застосовуватися «експоненціальне середнє». Формула має вигляд:

$$w_{t+1} = w_t - (\frac{\alpha_t}{\sqrt{v_t}} + e) * (\frac{\delta L}{\delta w_t}), (3.9)$$

$$v_t = \beta * v_t + (1 - \beta) * (\frac{\delta L}{\delta w_t})^2 (3.10)$$

Власне оптимізатор Адам починається з формул 3.8 та 3.10. Спочатку і m_t , і v_t встановлені на 0. Обидва мають тенденцію бути майже рівним до 0, оскільки β дорівнює 1. Ця проблема виправляється за допомогою таких обчислень:

$$\hat{m}_t = m_t \div (1 - \beta_1^t), (3.11)$$

$$\hat{v}_t = v_t \div (1 - \beta_2^t) (3.12)$$

Отже остаточна формула для обчислення ваг для наступного кроку за оптимізатором Адам має вигляд:

$$w_t = w(t-1) - \alpha * \left(\frac{\hat{m}_t}{\sqrt{\hat{v}_t}} + e\right) \quad (3.13)$$

3.3 Аналіз результату навчання

Отже, після самого навчання можна отримати списки метрик та функцій втрат, що використовувалися, на кожному кроці аби порівняти результати, провести аналіз та мати уявлення про якість роботи моделі. Наступні графіки демонструють функцію втрат, метрики кубика, чутливості, специфічності, індексу тверського з плином епох відповідно. Вони були використані як один з показників для оцінки фінальної моделі.

Навчання зупинилося на 141 епосі внаслідок того, що валідаційна функція втрат не зменшувалася впродовж 20 епох. Кінцеве значення її становить 0.15838.

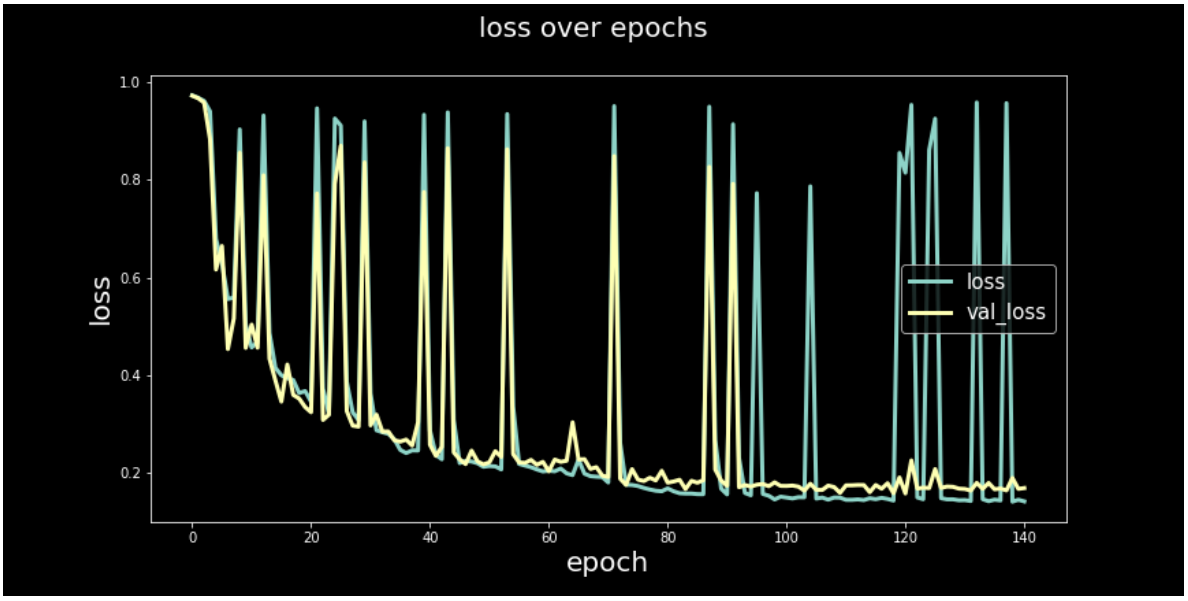


Рисунок 3.2 – Функція втрат протягом навчання

По графіку функції втрат видно (рис 3.2), що для валідаційної вибірки функція вийшла на плато, а отже знайшла мінімум. Може здатися, що цей мінімум не глобальний, а локальный, але варто зауважити, що валідаційна

вибірка ніяк не оброблялася додатковими аугментаціями, а отже представляє дані в тому вигляді в якому вони будуть застосовуватися в реальних задачах. Тож можна сказати, що модель навчилася достатньо аби передбачати локалізацію пухлин.

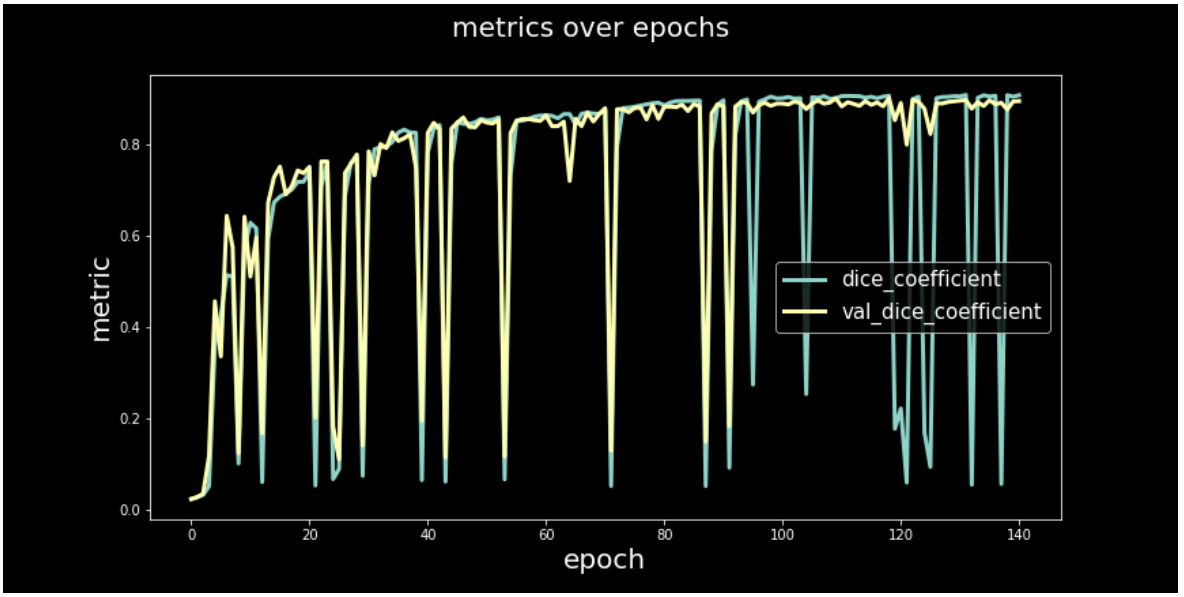


Рисунок 3.3 – Метрика dice протягом навчання

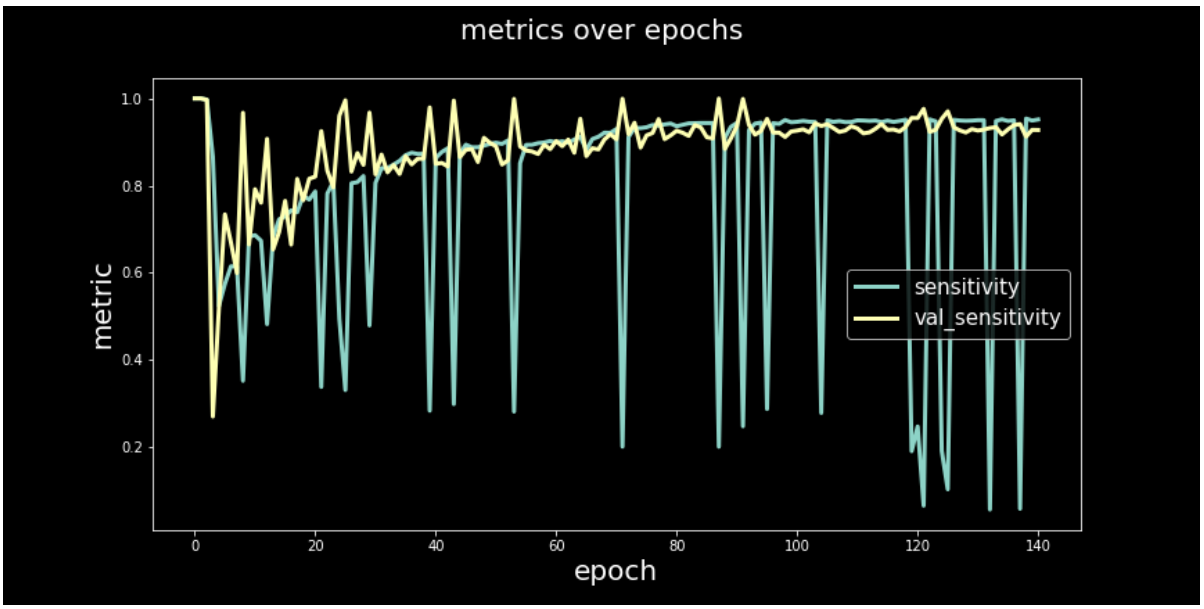


Рисунок 3.4 – Метрика sensitivity протягом навчання

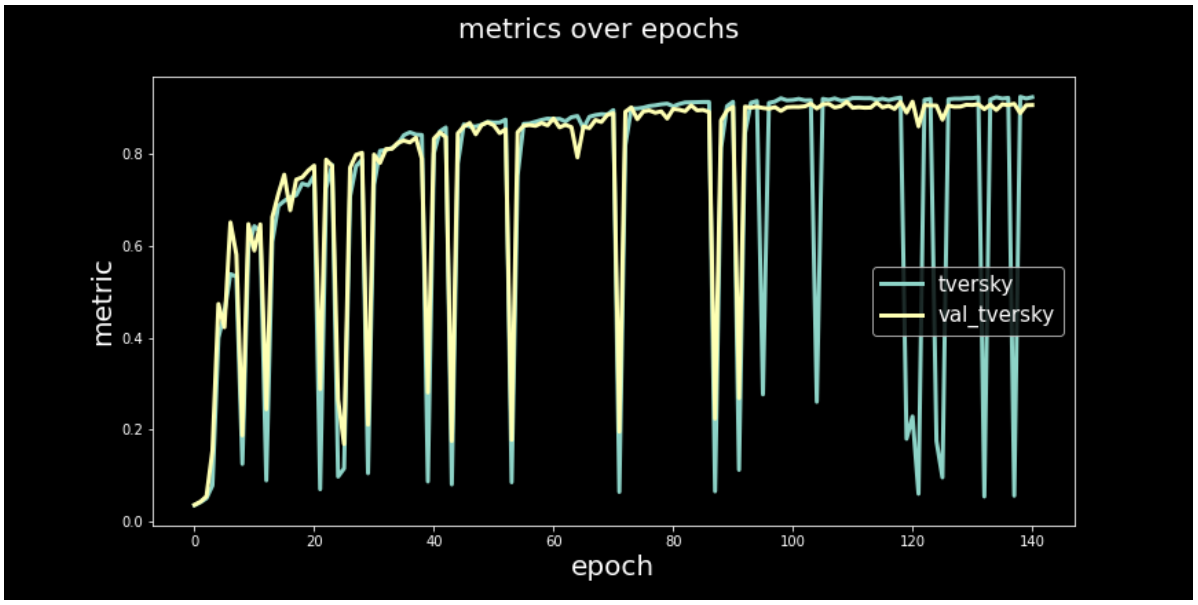


Рисунок 3.5 – Метрика tversky протягом навчання

Для метрик (рис 3.3 - 3.5), крім специфічності (рис 3.6), ситуація подібна до функції втрат. А от графіки для тренувального та валідаційного набору даних по специфічності майже співпадають. Це обумовлено тим, що ця метрика бере до уваги ситуації, коли клас фону, тобто здорові ділянки мозку, визначаються моделлю, як не здорові. З того, що і для тренувальної вибірки метрика вийшла на плато, можна говорити, що зміни датасету для ділянок пухлин не відповідають клінічній картині звичайних пацієнтів і спотворюють настільки, що модель не може їх вивчити. Про це також свідчить не постійний характер поганих оцінок, тобто тільки на тих прикладах, де відбувається аугментація, а вона застосовуються випадковим чином.

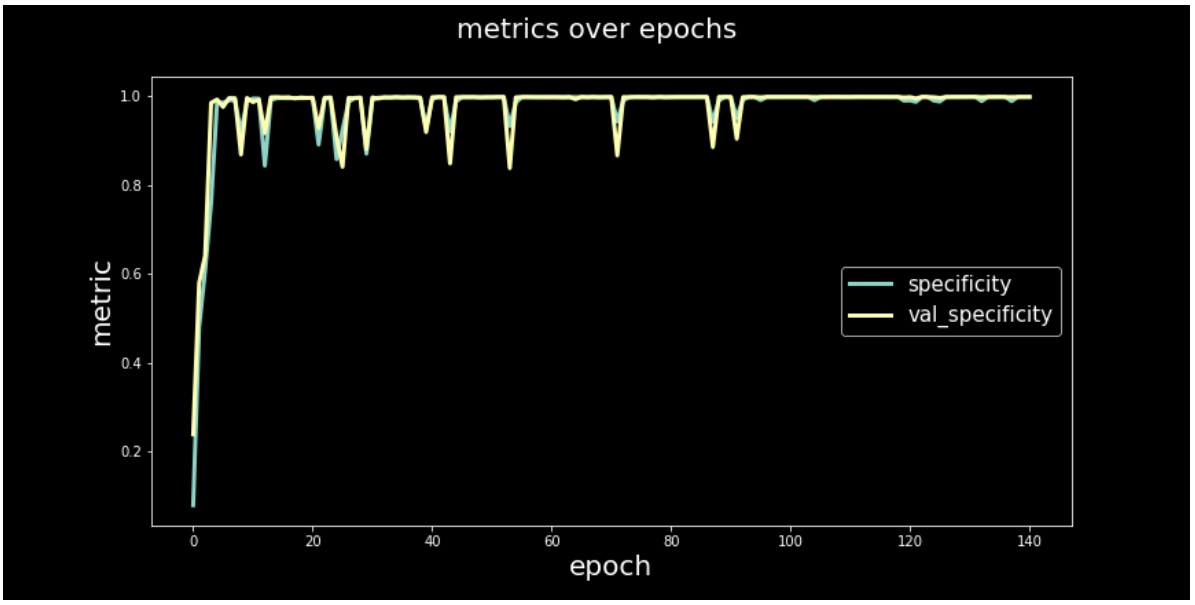


Рисунок 3.6 – Метрика specificity протягом навчання

Кінцеві метрики для валідаційного набору даних такі: індекс тверського - 0.9140, коефіцієнт кубика - 0.8893, чутливість - 0.9549 - специфічність - 0.9981 Гарною практикою було б запустити навчання наново без проблемної аугментації, і може бути одним із кроків при продовженні даного дослідження за межами цієї роботи.

Отже у результаті дослідження у даному розділу було отримано модель сегментації пухлин головного мозку, що показує гарні результати на вілідаційній вибірці. Також було описано спосіб навчання, що включає попередню обробку даних, визначення метрик, функції втрат та оптимізатора. У ході аналізу моделі було виявлено, що її результати на обрани метриках задовольняють задачі дослідження поставлені у вступі і її можна використовувати для подальшого впровадження. Безумовно є що покращювати, проте на даному етапу можна зупинитися на досягнутому. У наступному розділі буде показано роботу моделі, описно як її можна використовувати на практиці, буде запропоновано застосунок і подальші кроки для його розгортання.

РОЗДІЛ 4. РЕАЛІЗАЦІЯ МЕТОДУ ПОШУКУ ПУХЛИН ЗА ДОПОМОГОЮ НЕЙРОМЕРЕЖІ U-NET

Для того аби розроблена в минулому розділі модель знайшла практичне застосування у медичних установах чи веб сторінках зацікавлених у самодіагностиці пацієнтів, необхідно розробити середовище, де модель зможе бути запущена на зображення та видаватиме результати своєї роботи користувачеві. Далі буде описано як цього досягти, але спершу давайте поглянемо на результати роботи моделі, це допоможе визначитися з типом застосунку для розробки.

4.1 Приклади роботи моделі

Після навчання доцільно подивитися наскільки модель виконує поставлену задачу. У випадку комп'ютерного зору, можливо наочно переконатися у роботі моделі. Отже на наступних зображеннях продемонстровано роботу моделі, де перше зображення є вхідним, середнє зображає маску реальної локалізації пухлини, яка створена експертом і наостанок зображення передбачення моделі.

Ці візуалізації створені засобами на мові програмування Python. Для тесту використовувалися зображення з вибірки даних саме для тесту, потім було обрано частину з них для включення в роботи аби продемонструвати окремі випадки роботи моделі.

Кожний рисунок міститиме 2 знімки, по 2 приклади для кожної ситуації і будуть містити варіанти як хорошої роботи моделі так і хибного сегментування.

На двох зображеннях (рис 4.1) видно, що модель впоралася з ситуацією, коли пухлини немає. Вона правильно визначає маску, тобто відсутність аномалій.

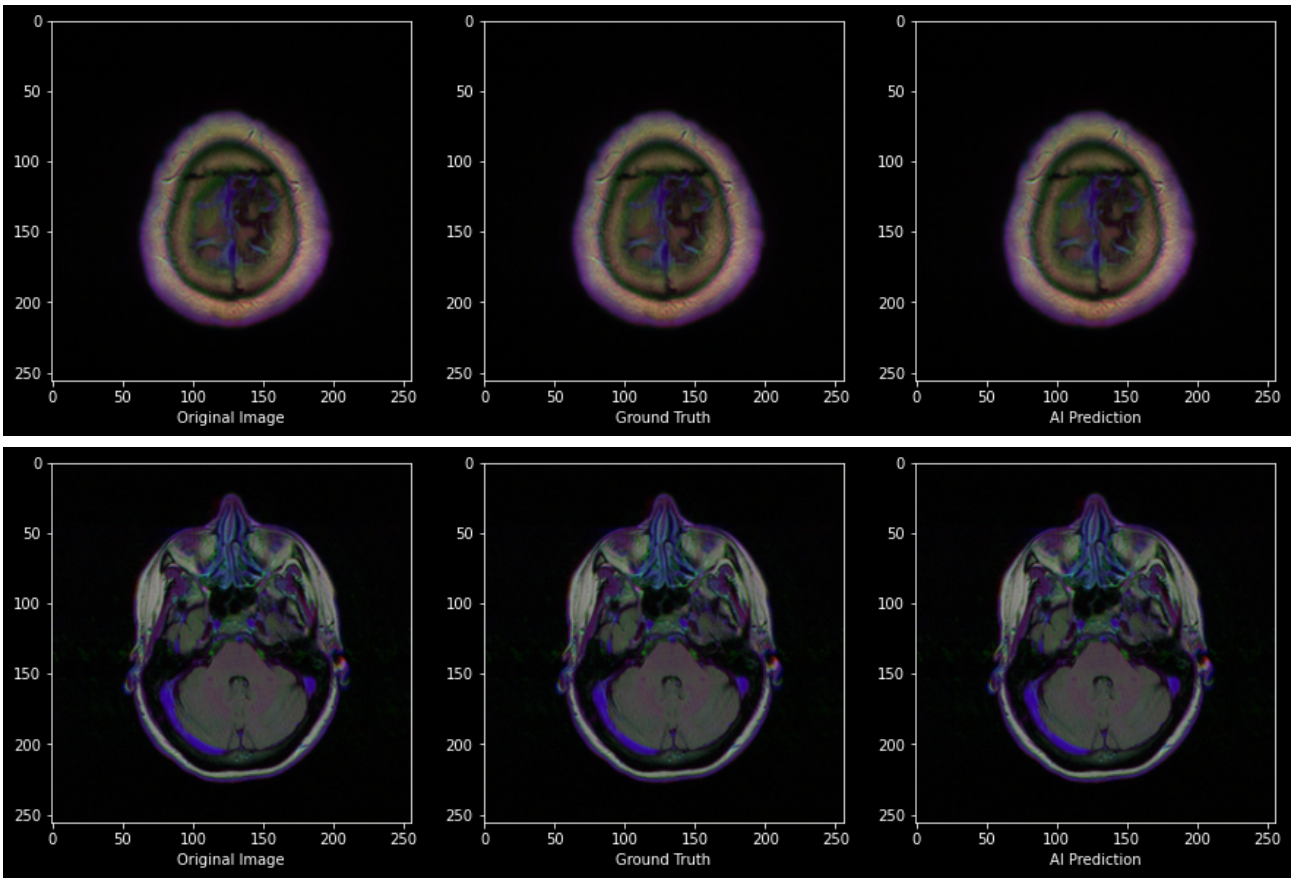


Рисунок 4.1 – Приклади роботи на зображеннях без пухлин

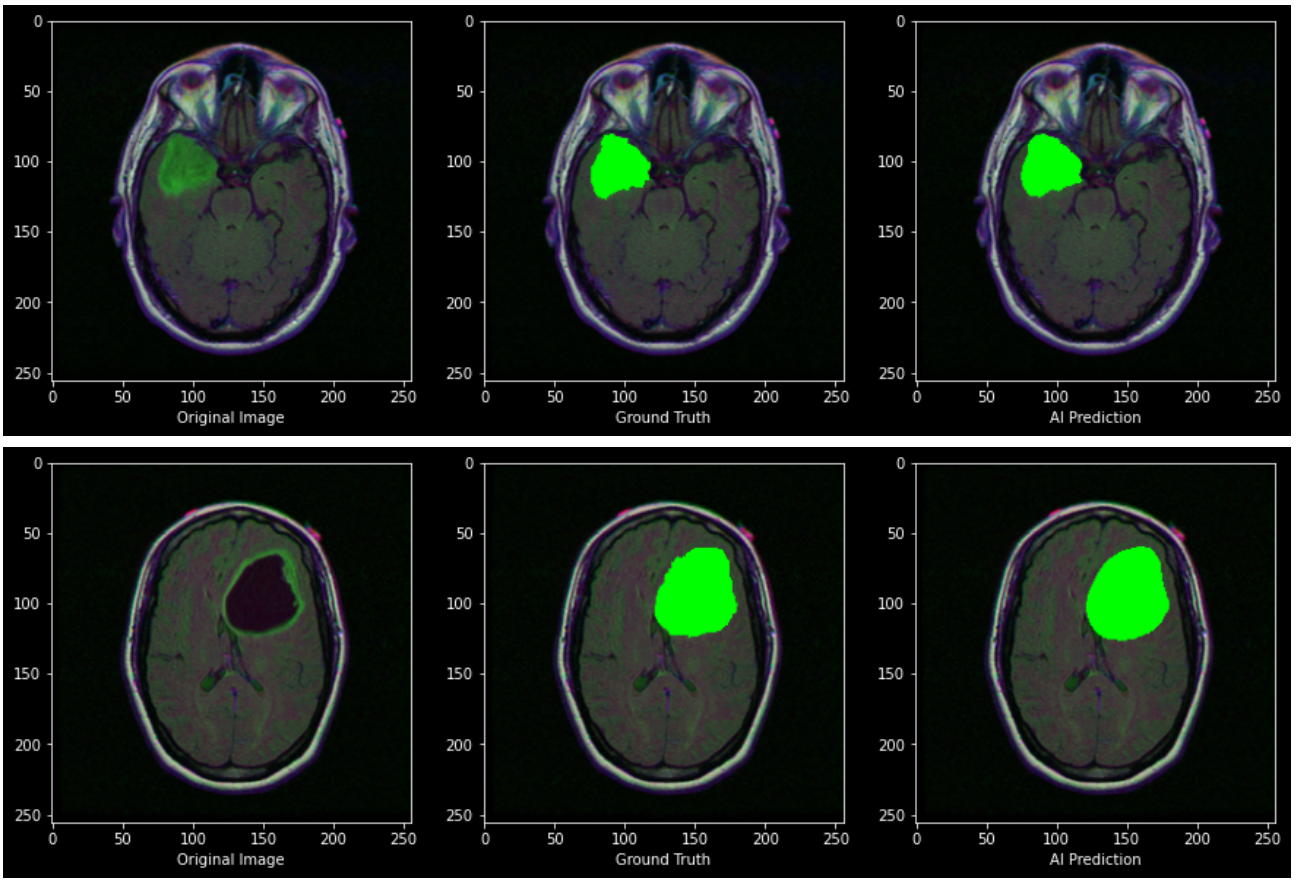


Рисунок 4.2 – Приклади роботи на зображеннях з чіткими пухлинами

На двох знімках (рис 4.2) видно пухлину навіть для людини, що не розуміється на медицині. Тут також модель спрацювала чудово, локалізація майже ідеальна. Проте злегка згладжена, в порівнянні з істиною.

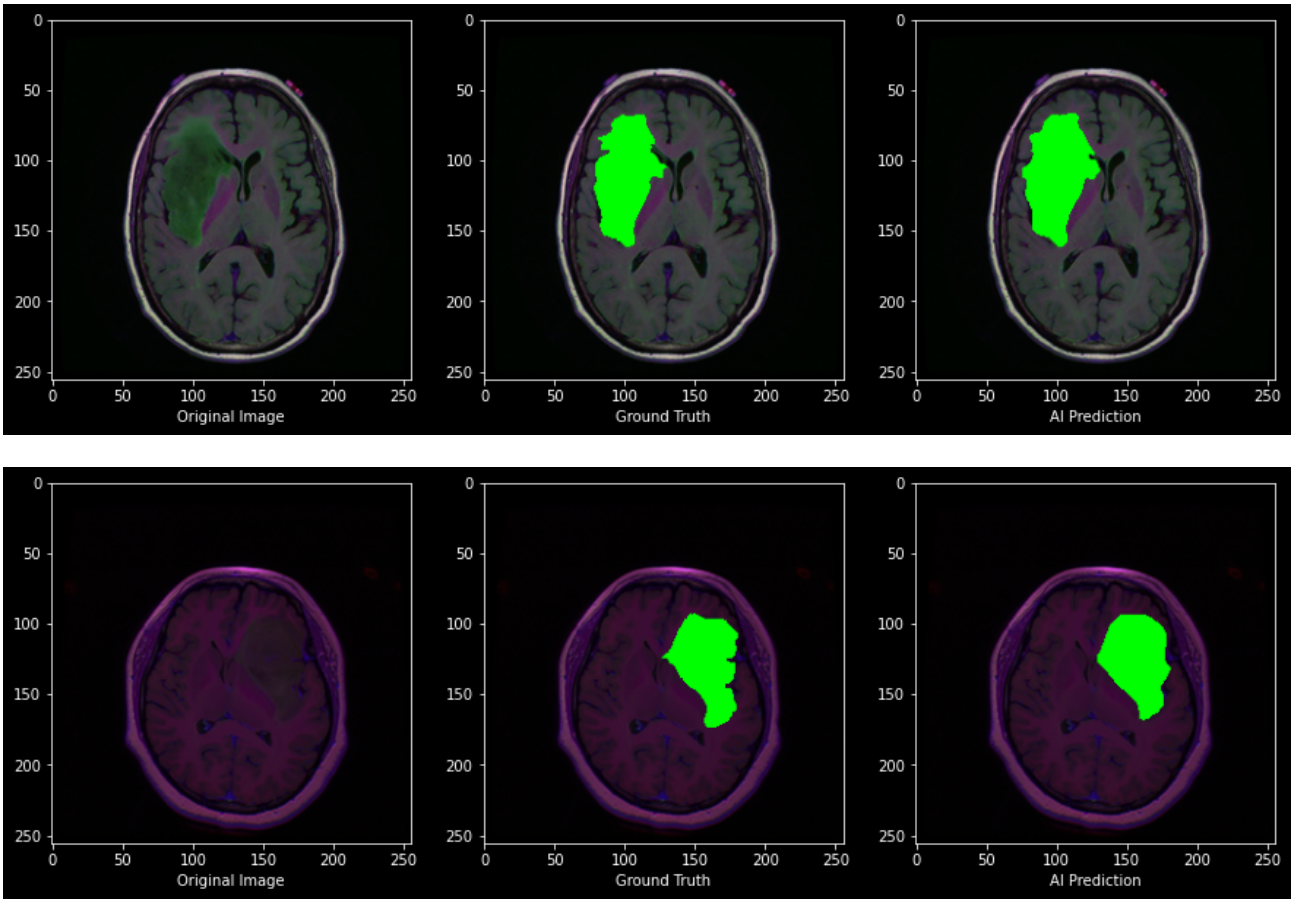


Рисунок 4.3 – Приклади роботи на зображеннях з не чіткими пухлинами

Наступні 2 приклади (рис 4.3) вже не є такими очевидними. Модель визначає маску здебільшого правильно.

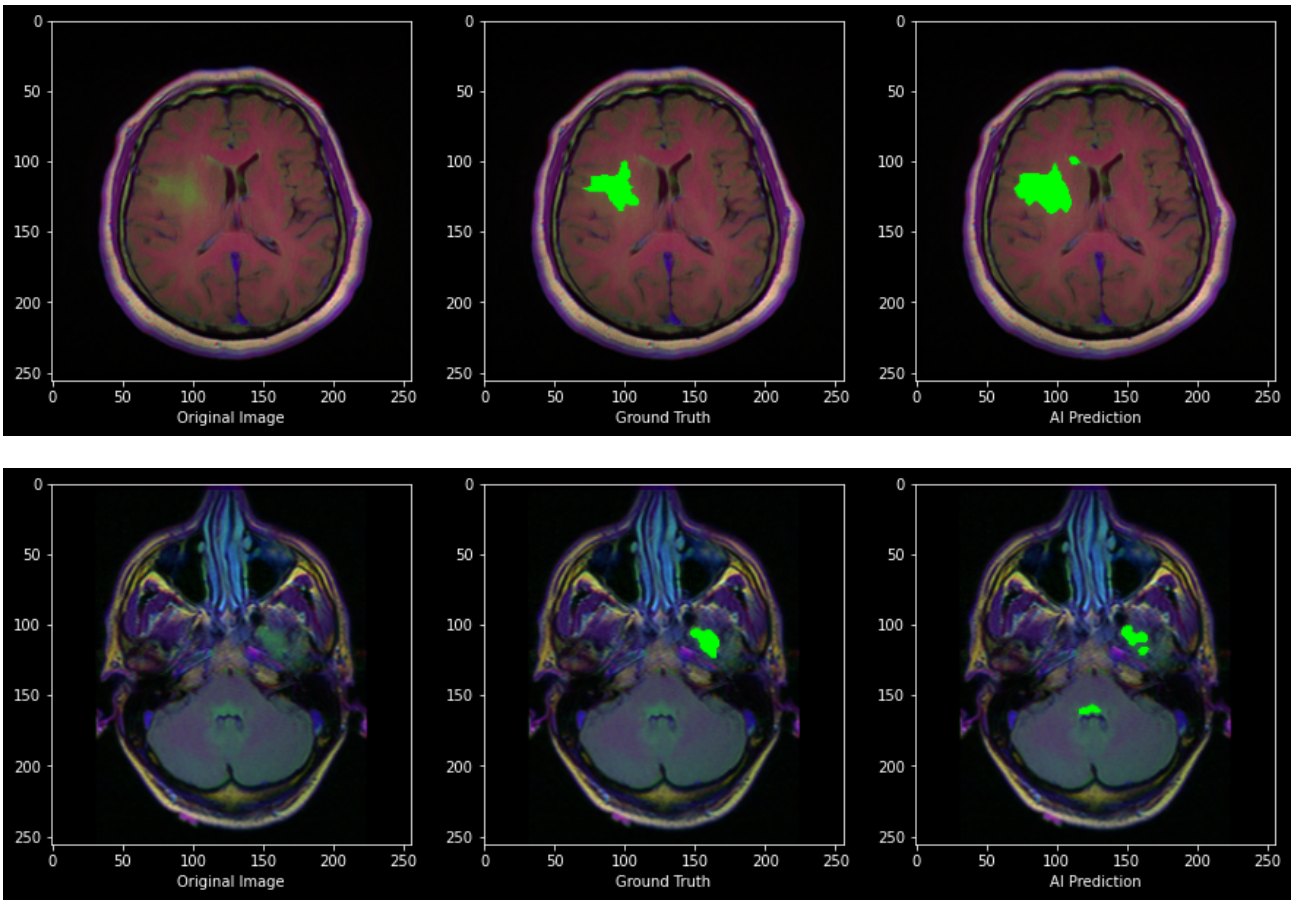


Рисунок 4.4 – Приклади неточної роботи моделі

А от на прикладах з рисунку 4.4 все не так чудово. Модель то перебільшує регіон ураження, то навпаки визначає надто малу область. До того ж на обох знімках присутнє хибне спрацювання. Тобто модель назвала здорові області пухлинами.

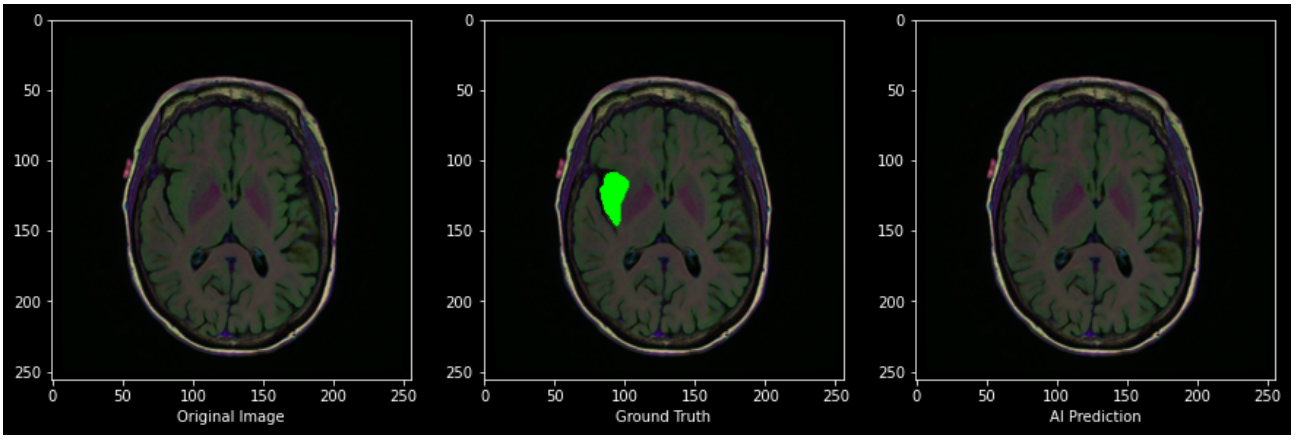


Рисунок 4.5 – Приклади хибної роботи моделі

Рисунок 4.5 показує, що іноді алгоритм не визначає пухлину. Це може свідчити про недоліки у самій моделі і необхідне перенавчання, але також і дані можуть слугувати причиною, адже на вхідній картинці непомітно змін, що можна прийняти за проблему. Для таких випадків слід у подальшому проконсультуватися з людиною-експертом у області гліом мозку.

З огляду на вищеописане, можна зазначити, що модель в теперішньому її стані не може бути самостійною одиницею для визначення локалізації аномалій мозку. Наразі вона слугує дише підказкою для лікаря, проте подальші дослідження можуть виправити ситуацію. Слід подумати у бік зміни архітектури, гіперпараметрів навчання а також датасету, збільшити чи взагалі замінити.

Також варто зазначити, що подібна візуалізація добре підходить з користувацького боку - одразу видно область на яку слід звернути увагу і досліджувати біль детально. Тому для практичного застосування моделі схожа візуалізація результатів буде використана.

4.2 Концепція пошуку пухлин методами Data Science

Спочатку формалізуємо виявлення аномалій мозку методами Data Sciencey вигляді концептуальної моделі. Була створена концептуальна модель виявлення аномалій мозку методами Data Science.

Основними елементами для досягнення стабільного застосунку для пошуку аномалій мозку є: створення відповідної колекції даних чи з відкритих джерел, чи зібраних власноруч; попередня обробка зображень, що може включати розширення кількості даних та зміну їх розміру; створення моделі, що відповідає вхідним даним та на виході передбачає маску з локалізацією пухлин; визначення метрик, функції втрат та оптимізатора для навчання; ініціалізація тренувальних гіперпараметрів; тренування моделі; її тестування; за незадовільних результатів зміна тренувальних параметрів чи даних та нова спроба отримати якісну модель під час тренування; за успішного навчання відбувається розгортання моделі у застосунку; розгортання застосунку з

використанням хмарних технологій, для надання доступу користувачам. Наприкінці циклу розробки моделі ведеться моніторинг за роботою системи і можливе покращення моделі у випадку отримання нових даних яи скарг від користувачів. Схема зображена на рисунку 4.6.

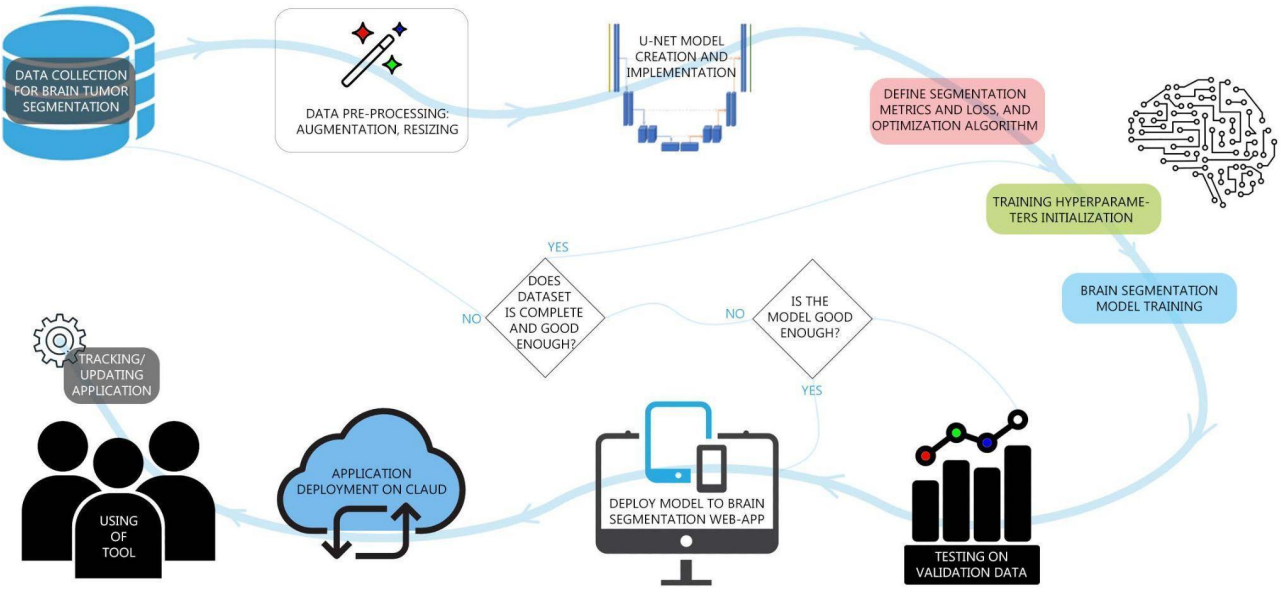


Рисунок 4.6 – Концептуальна модель виявлення мозку методами Data Science

Далі буде описано більш предметно етапи, що стосуються розробки та роботи самого застосунку для використання побудованої моделі.

Оскільки мета даного дослідження у тому аби надати змогу працівникам медичної сфери біль якісно виконувати обстеження та лікування гліом мозку, було вирішене реалізувати застосунок для цього. Основною ціллю якого є надання користувачам змогу використовувати модель пошуку аномалій на практиці. Так лікарі зможуть швидко підвантажити зображення, що було отримано в ході обстеження і робити висновки спираючись, в тому числі і на продукт розроблений в ході цієї роботи.

Спочатку опишемо концептуально як має виконуватися код крок за кроком. Для цього було розроблено схему роботи алгоритму, що дасть змогу завантажувати зображення та отримувати результат сегментації. На рисунку 4.7 можна подивитися на головні етапи, що зображені у схемі.

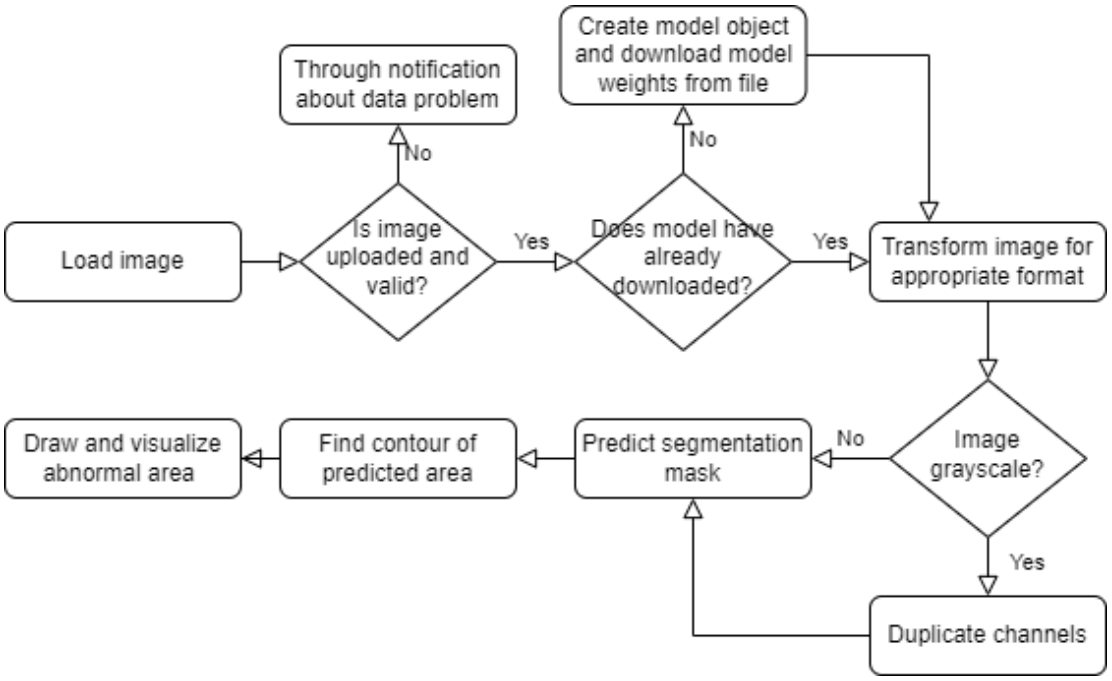


Рисунок 4.7 – Схема роботи застосунку

Як наведено у схемі програма має такі елементи: завантаження даних, їх валідація, завантаження моделі, трансформація даних, застосування моделі до даних, пошук контурів пухлини і візуалізація.

Програма дозволяє завантажувати файли з розширеннями ві png, jpg, jpeg, tif. Вони відповідають картинках, та може статися, що файл зіпсовано, тому додаткова перевірка виключає надходження далі некоректних даних. Далі виконується обробка зображення для входу у модель, серед них: нормалізація, приведення до одного розміру та формату тензору. Також є можливість завантажити зображення в градаціях сірого, тоді алгоритм додасть ще 2 канали, це буде теж зображення, просто замість одного каналу, будуть три однакові, адже модель передбачає на вхід саме 3 канали.

Сама модель ініціалізується не на початку програми, а при необхідності першого передбачення. Тоді ж і завантажуються ваги з окремого файлу, що було отримано у ході навчання у попередньому розділі. Після передбачення знаходиться область з найбільшою ймовірністю знаходження там пухлини і візуалізується результат.

Для користування отриманою моделі у описаний вище спосіб було розроблено веб-інтерфейс. Код програми написано на мові програмування Python з використанням бібліотеки Streamlit. Дана бібліотека надає інструментарій для написання інтерфейсів у зручний для програміста спосіб.

Як зазначено у документації [62], Streamlit є найпотужнішим способом створення програм із даними, включаючи можливість відображати та стилізувати дані, малювати діаграми та карти, додавати інтерактивні віджети, налаштовувати макети додатків, обчислювати кеш і визначати теми. Він працює на Windows, macOS та Linux. Все це робить його привабливим для створення кросплатформенних застосунків, що і необхідно для розробки способу використання створеної моделі сегментації, адже передбачається, що і звичайні пацієнти зможуть спробувати опрацювати ним свої МРТ знімки.

Отже, приклад інтерфейсу можна побачити на рисунку 4.8. В майбутньому можлива переробка даного сайту аби підлаштувати під інші середовища виконання та потреби користувачів.

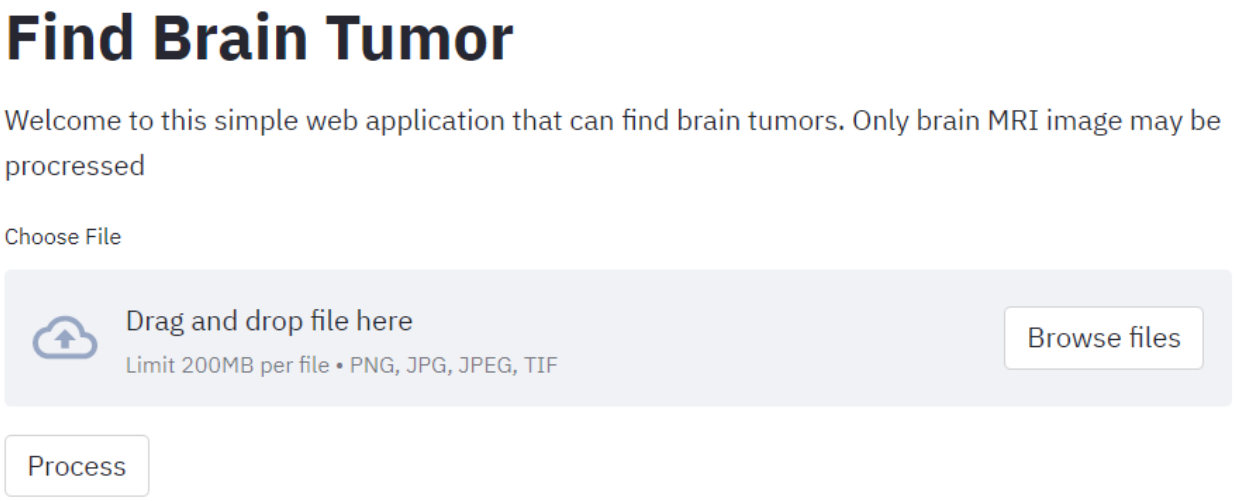


Рисунок 4.8 – Інтерфейс застосунку.

Як можна бачити всього присутньо 3 головні елементи: опис застосунку, форма для завантаження зображення та кнопка, що запускає виконання моделі. Повний приклад сторінки при виконання аналізу можна побачити у Додатку Г.

Після підвантаження картинки, виконується пошук пухлини за допомогою моделі і виводиться на екран 2 зображення. Перше показує передбачену маску, друге оригінал картинки з накладеною на нього маскою для наочного представлення результату.

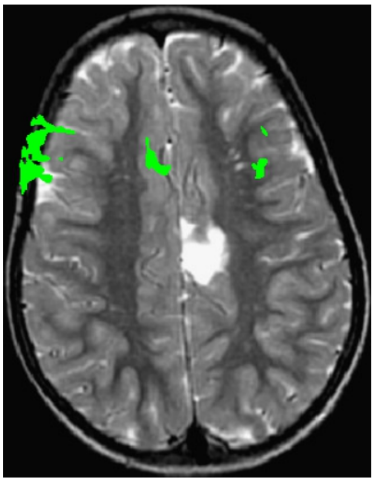


Рисунок 4.9 – Приклад застосування моделі до звичайного МРТ знімку.

Завантажувати можна як чорно білі так і кольорові картинки, проте результат роботи моделі на виключно чорно-білих знімках буде хибним. Наприклад, результат на рисунку 4.9. Було використано звичайне МРТ зображення і передбачена маска підкреслила область пухлини неправильно. Видно, що всередині існує аномальна ділянка, але її було проігноровано. Такий результат прогнозовано, адже для навчання моделі було використано знімки з FLAIR технологією, отже і застосовувати її слід до таких самих зображень.

В майбутньому можливо розширити застосунок аби всі види сканування мозку підтримувались. Один з варіантів навчити модель на картинках з різних методів діагностики, або натренувати ще одну модель і перемикатися між ними при обробці зображення на сайті.

4.3 Концепція по розгортанню моделі для користувачів

В цьому підрозділі буде описано концепцію майбутнього розгортання застосунку аби кожен міг мати до нього доступ.

Наразі увесь код на Python працює на машині, де був написаний. Якщо інший науковець чи програміст запустить його у себе, він може не отримати ті ж результати. Середовище Python може відрізняється від початкового. Середовище Python включає всі бібліотеки та залежності з точними версіями, за допомогою яких було створено та протестовано програму. Тому необхідно привести систему до ідентичного стану як і оригінальна, а це не завжди можливо.

До того ж задача отримати робочу програму для не технічних спеціалістів, які не володіють інструментами програмування по типу мови Python. Необхідно створити середовище, яке можна буде передати на інші машини, то буде можливість відтворити результати будь-де.

Одним із рішень є контейнер — це тип програмного забезпечення, яке упаковує програму та всі її залежності, щоб програма надійно працювала від одного обчислювального середовища до іншого. Іншою альтернативою для створення ізольованого середовища є віртуальні машини. Контейнери тут є кращими, оскільки вони вимагають менше ресурсів, дуже портативні та швидше розгортаються.

Docker — це компанія, яка надає програмне забезпечення, яке дозволяє користувачам створювати, запускати та керувати контейнерами. Хоча контейнер Docker є найпоширенішим, існують інші менш відомі альтернативи, такі як LXD і LXC, які надають інструменти для створення контейнерів.

Після того, як застосунок буде запаковано в контейнер Docker, його можна відправити на іншу машину або хмару, і він продовжить працювати, як і оригінальна версія.

Навіть після розробки та розгортання моделі життєвий цикл машинного навчання далекий від завершення. Щоб отримати максимальну віддачу від моделі, потрібно стежити за нею протягом усього часу використання.

Після створення контейнеру його вже можуть використовувати інші програмісти чи фахівці з науки по даним. Для того аби застосунок могли побачити звичайні користувачі можна застосувати хмарні технології.

Завдяки розгортання контейнера на хмарному сервері, наприклад, AWS Google Cloud, можна створити сайт доступний для всіх в мережі інтернет.

Таким чином веб-застосунок, що було розроблено, зможе використовувати будь-хто зацікавлений у аналізу мозку на предмет гліом лише зайшовши на сайт. А також залишиться можливість розгортання застосунку на інших машинах за потреби у персоналізованих змін чи для продовження досліджень.

ВИСНОВКИ

Отже, а ході виконання магістерської роботи було проведено аналіз задачі пошуку аномалій на медичних зображеннях. Виявлено, що дане завдання відноситься до проблеми сегментації в комп'ютерному баченні. Робота в напрямку сегментації пухлин проводилася ще в минулому сторіччі. Переломним моментом стало винайдення та початок використання глибоко навчання для такої задачі як сегментації. Методи сегментації використовуються для пошуку гліом і раніше, але з деякими відмінностями і не мали вражаючих результатів.

Визначено математичні інструменти й методи, які використовуються для вирішення задач сегментації зображень. І виявлено, що за останні десятиліття здобувають популярність методи з використанням нейронних мереж, адже раніше не було достатньо потужних машин для їх навчання. Основним принципом є те, що ці алгоритми власноруч шукають тисячі чисел аби отримати розуміння про вхідні дані та на їх основі видати результат. Це одноступеневий метод, тоді як його попередники спочатку сегментували зображення по певним критеріям, а вже потім інший алгоритм навчався з цих даних передбачати локалізацію пухлин. Ці методи мають більш зрозуміле пояснення, проте результати сегментації не вражають, адже вони обробляють обмежену кількість даних з зображення.

Було визначено дані, що будуть використовуватися при побудові системи. У результаті вивчення особливостей медичних даних для візуалізації було обрано МРТ технологію. Вивчено FLAIR метод для створення магнітно резонансної томографії і для подальшого дослідження використовувалися саме дані отримані за допомогою цього типу візуалізації. Дані для навчання було отримано з відкритих джерел. Воин були перевірені професіоналами в рентгенології та анатовані. Датасет складається з 2556 зображень із них 1373 мають пухлини. Додатково описано процеси підготовки зображень для навчання моделі.

При вивченні методів глибокого навчання для сегментації зображень та підборі підходящої модель та специфікації для її навчання було обрано архітектуру мережі U-Net. Це мережа була створена здебільшого задля сегментації у медичному домені і показує гарні результати на багатьох відомих задачах. Класична архітектура мережі включає в собі два рівні: стиснення та розгортання. На першому етапі завдяки згортам виконується пошук характеристи на зображеннях завдяки різним фільтрам. На другому стиснене зображення повертається до початкових розмірів і передбачає де знаходиться пухлина завдяки вивчених фільтрів і інформації, що була збережена про зображення на першому етапі. Також було описано процес її навчання. Зокрема визначено критерії оцінки якості сегментацій мозку такі як метрики, функції активації та оптимізатор. Dice coefficient, sensitivity, specificity, tversky index використовувалися як метрики, tversky loss як функція втрат а для оптимізації під час навчання було обрано Adam алгоритм.

Було реалізовано програмну імплементацію моделі пошуку аномалій мозку на мові програмування Python. І у ході дослідження отримано модель, що може знаходити гліоми на зображеннях МРТ мозку, що зроблені за FLAIR технологією за допомогою інструментів, представлених бібліотекою машинного навчання для мови програмування Python TensorFlow. Модель має 95% точності за метрикою чутливості та може передбачати аномалії досить гарно.

Створено програмну імплементацію побудованої технології виявлення пухлин. Як результат було розроблено веб-застосунок для виявлення аномалій.

Разом з тим було визначено спосіб для надання користувачам можливості доступу до її використання у майбутньому. Запропоновано концепцію розгортання застосунку на хмарних сервісах з використанням технології контейнерів. Це дасть змогу потенційним користувачам оцінити застосунок власноруч.

У результаті формалізації виявлення аномалій мозку методами Data Science у вигляді концептуальної моделі отримано 12 кроків, що зображають життєвий цикл розробки застосунку пошуку аномалій. На першому етапі

відбувається пошук даних, що закінчується наданням користувачам доступу до застосунку і моніторингом фінальної моделі та її своєчасним поновленням у разі необхідності.

Не виключено, що поточна реалізація має недоліки, і деякі варіанти покращення системи у майбутньому описано у роботі, наприклад, зміну гіперпараметрів для більш точної моделі чи розширення її можливості задля сегментації на звичайних МРТ знімках. Але вже зараз результати цього дослідження можуть бути використаними для подальших досліджень у сфері сегментації зображень методами Data Science в цілому та для медичної сфери зокрема. Крім цього ефективність розробленої моделі дозволяє використовувати її вже зараз для пошуку аномалій. Поки що лише пацієнтами, а у майбутньому, після перевірки фахівцями та, можливим удосконаленням, навіть і фахівцями у онкології для постановки більш точних діагнозів, що може врятувати багато життів у довгій перспективі.

ПЕРЕЛІК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Zhovtukhin D., Yehorchenkov O. Classification of Bottles Images Using Convolutional Neural Networks. VII International conference Information Technology and Interactions (Satellite), 2020, 186-191 pp.
2. Van Meir, E. G. et al. Exciting new advances in neuro-oncology: the avenue to a cure for malignant glioma. CA. Cancer J. Clin. 60, 2010, 166–193.
3. Hussain S, Anwar SM, Majid M. Segmentation of Glioma Tumors in Brain Using Deep Convolutional Neural Network. Neurocomputing, 2017, 248–261.
4. Reardon DA, Wen PY. Therapeutic Advances in the Treatment of Glioblastoma: Rationale and Potential Role of Targeted Agents. Oncologist, 2006, 112-152
5. Brandes AA. State-of-the-Art Treatment of High-Grade Brain Tumors. Semin Oncol, 2003, 30
6. Automated Brain Tumor Segmentation Using Spatial Accuracy-Weighted Hidden Markov Random Field. Nie J, Xue Z, Liu T, Young GS, Setayesh K, Guo L. Computerized Med Imaging Graphics, 2009, 431–441.
7. An automatic glioma grading method based on multi-feature extraction and fusion. Tianming Zhana, Piaopiao Fengd, Xunning Hongd, Zhenyu Luc, Liang Xiaob Yudong Zhange.
8. Scott AM PET imaging in oncology. In: Bailey DL, Townsend DW, Valk PE, Maisey MN Positron emission tomography. Springer, London, 2005, 311–325
9. Wong TZ, van der Westhuizen GJ, Coleman RE Positron emission tomography imaging of brain tumors. Neuroimaging Clin, 2002, 615–626
10. Brenner DJ, Hall EJ Computed tomography—an increasing source of radiation exposure. N Engl J Med 2007, 2277–2284
11. Saad NM, Bakar SARSA, Muda AS, Mokji MM Review of brain lesion detection and classification using neuroimaging analysis techniques. J Teknol, 2015, 1–13

12. Joo L, Jung SC, Lee H, Park SY, Kim M, Park JE et al Stability of MRI radiomic features according to various imaging parameters in fast scanned T2-FLAIR for acute ischemic stroke patients. Sci Re, 2021, 1–11
13. Chen H, Zou Q, Wang Q Clinical manifestations of ultrasonic virtual reality in the diagnosis and treatment of cardiovascular diseases. J Healthc Eng, 2021, 1–12
14. Margiewicz S, Cordova C, Chi AS, Jain R. "State of the Art Treatment and Surveillance Imaging of Glioblastomas". Seminars in Roentgenology, 2018, 23–36
15. Tang, Z., Ahmad, S., Yap, P. T. & Shen, D. Multi-atlas segmentation of MR tumor brain images using low-rank based image recovery. IEEE Trans. Med. Imaging, 2224–2235
16. Am. J. Roentgenol. Defending the “missed” radiographic diagnosis. Berlin, L, 2001, 317–322
17. Khosravanian, A., Rahmanimanesh, M., Keshavarzi, P. & Mozaffari, S. Fast level set method for glioma brain tumor segmentation based on superpixel fuzzy clustering and lattice boltzmann method. Comput. Methods Programs Biomed
18. Tang, Z., Ahmad, S., Yap, P. T. & Shen, D. Multi-atlas segmentation of MR tumor brain images using low-rank based image recovery. IEEE Trans. Med. Imaging, 2018, 2224–2235
19. Iqbal, S. et al. Deep learning model integrating features and novel classifiers fusion for brain tumor segmentation. Microsc. Res. Tech. 2019, 1302–1315
20. Linda G. Shapiro and George C. Stockman: “Computer Vision”, 2001, 279–325
21. Barghout, Lauren, and Lawrence W. Lee. "Perceptual information processing system." Paravue Inc. U.S. Patent Application, 2003
22. Zachow, Stefan, Michael Zilske, Hans-Christian Hege. "3D reconstruction of individual anatomy from medical image data: Segmentation and geometry processing.", 2007

23. Веб-сайт. URL: https://www.analytixlabs.co.in/blog/what-is-image-segmentation/
24. Nobuyuki Otsu. A threshold selection method from gray-level histograms. IEEE Trans. Sys. Man. Cyber. 1979, 62–66
25. Irwin Sobel. History and Definition of the Sobel Operator, 2014.
26. R. Deriche, Using Canny's criteria to derive a recursively implemented optimal edge detector, Int. J. Computer Vision, 1987, 167–187.
27. Canny, J. A Computational Approach To Edge Detection, IEEE Transactions on PatternAnalyzing breast tumor heterogeneity to predict the response to chemotherapy using 3D MR images registration. ACM. pp Analysis and Machine Intelligence /Adoui, Mohammed El; Drisis, Stylianos; Benjelloun, Mohammed/, 1986, 679–698
28. Luc Vincent and Pierre Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. In IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 13, Num. 6, 1991, 583–598
29. Веб-сайт. URL: https://docs.opencv.org/4.x/d7/d00/tutorial_meanshift.html
30. Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties /Fix, Evelyn; Hodges, Joseph L./, 1951
31. Dunn, J. C. A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. Journal of Cybernetics. 1973, 32–57
32. Веб-сайт. URL: https://radiopaedia.org/articles/fluid-attenuated-inversion-recovery
33. Веб-сайт. URL: https://wiki.cancerimagingarchive.net/display/Public/TCGA-LGG
34. Веб-сайт. URL: https://www.kaggle.com/datasets/mateuszbuda/lgg-mri-segmentation
35. Веб-сайт. URL: https://www.quora.com/What-is-the-differences-between-artificial-neural-network-computer-science-and-biological-neural-network

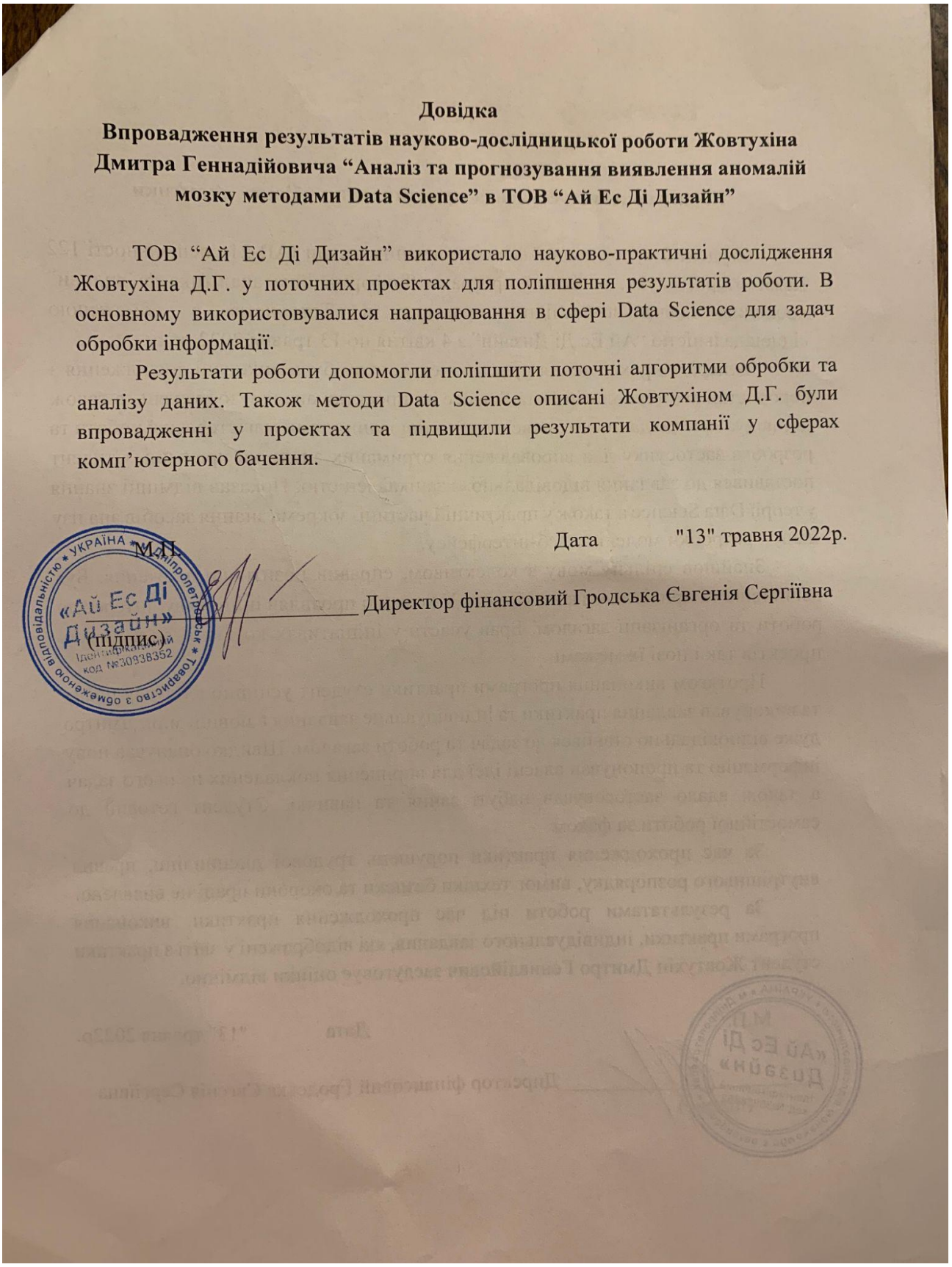
36. Веб-сайт. URL: https://www.wikiwand.com/uk/%D0%A8%D1%82%D1%83%D1%87%D0%BD%D0%B0_%D0%BD%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D0%B0_%D0%BC%D0%B5%D1%80%D0%B5%D0%B6%D0%B0
37. George F. Simmons, Calculus with Analytic Geometry, 1985, 93.
38. Веб-сайт. URL: https://blogs.nvidia.com/blog/2018/08/02/supervised-unsupervised-learning/
39. Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. /Valueva, M.V.; Nagornov, N.N.; Lyakhov, P.A.; Valuev, G.V.; Chervyakov, N.I. / Mathematics and Computers in Simulation. Elsevier BV. 2020, 232–243.
40. Hochreiter, Sepp; Schmidhuber, Jürgen. Long Short-Term Memory. Neural Computation. 1997, 1735–1780.
41. Generative Adversarial Networks / Goodfellow, Ian; Pouget-Abadie, Jean; Mirza, Mehdi; Xu, Bing; Warde-Farley, David; Ozair, Sherjil; Courville, Aaron; Bengio, Joshua. /, 2014.
42. Neural Message Passing for Quantum Chemistry / Gilmer, Justin; Schoenholz, Samuel S.; Riley, Patrick F.; Vinyals, Oriol; Dahl, George E. / International Conference on Machine Learning, 2017, 1263–1272.
43. Attention Is All You Need / Polosukhin, Illia; Kaiser, Lukasz; Gomez, Aidan N.; Jones, Llion; Uszkoreit, Jakob; Parmar, Niki; Shazeer, Noam; Vaswani, Ashish. / , 2017.
44. Iulia Khlevna, Dmytro Zhovtukhin. Using image segmentation neural network model for motion representation in sport analytics. Mathematical Modeling and Simulation of Systems (Scopus), 2022
45. Robinson, A. J. & Fallside, F. The utility driven dynamic error propagation network. Cambridge University, Engineering Department, 1987
46. Spectral Networks and Deep Locally Connected Networks on Graphs, Bruna, 2014

47. Sepp Hochreiter; Jürgen Schmidhuber. Long short-term memory. Neural Computation, 1997, 1735—1780
48. UNETR: Transformers for 3D Medical Image Segmentation / Hatamizadeh, Ali; Tang, Yucheng; Nath, Vishwesh; Yang, Dong; Myronenko, Andriy; Landman, Bennett; Roth, Holger; Xu, Daguang/, 2021
49. Ronneberger O, Fischer P, Brox T. "U-Net: Convolutional Networks for Biomedical Image Segmentation", 2015.
50. Веб-сайт. URL: https://towardsdatascience.com/unet-line-by-line-explanation-9b191c76baf5
51. Веб-сайт. URL: https://towardsdatascience.com/transposed-convolution-demystified-84ca81b4baba
52. TensorFlow: A system for large-scale machine learning / Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, Xiaoqiang Zheng /, 2016
53. Веб-сайт. URL: https://www.apache.org/licenses
54. Веб-сайт. URL: keras.io.
55. Веб-сайт. URL: https://scikit-image.org/
56. Веб-сайт. URL: https://opencv.org/
57. Jaccard P. Distribution de la flore alpine dans le Bassin des Dranses et dans quelques regions voisines // Bull. Soc. Vaudoise sci. Natur, 1901, 241—272.
58. Nabila Abraham, Naimul Mefraz Khan, A Novel Focal Tversky loss function with improved Attention U-Net for lesion segmentation, 2018
59. Веб-сайт. URL: https://www.iclr.cc/archive/www/doku.php%3Fid=iclr2015:main.html

60. A. Agnes Lydia and , F. Sagayaraj Francis, Adagrad - An Optimizer for Stochastic Gradient Descent, Department of Computer Science and Engineering, Pondicherry Engineering College, 2019.
61. Wendyam Eric Lionel Ilboudo, Taisuke Kobayashi, Kenji Sugimoto, TAdam: A Robust Stochastic Gradient Optimizer, 2020.
62. Веб-сайт. URL: https://docs.streamlit.io/

ДОДАТКИ

Додаток А. Довідка від бази практики



Довідка
Впровадження результатів науково-дослідницької роботи Жовтухіна Дмитра Геннадійовича “Аналіз та прогнозування виявлення аномалій мозку методами Data Science” в ТОВ “Ай Ес Ді Дизайн”

ТОВ “Ай Ес Ді Дизайн” використало науково-практичні дослідження Жовтухіна Д.Г. у поточних проектах для поліпшення результатів роботи. В основному використовувалися напрацювання в сфері Data Science для задач обробки інформації.

Результати роботи допомогли поліпшити поточні алгоритми обробки та аналізу даних. Також методи Data Science описані Жовтухіном Д.Г. були впровадженні у проектах та підвищили результати компанії у сферах комп’ютерного бачення.

Дата "13" травня 2022р.

М.П.

(підпис) Директор фінансовий Гродська Євгенія Сергіївна

```
from tensorflow.keras import Input
from tensorflow.keras.models import Model
from tensorflow.keras.layers import *

def naive_inception_module(layer_in, filters):
    f1 = f2 = f3 = filters
    # 1x1 conv
    conv1 = Conv2D(f1, (1,1), padding='same', activation='relu')(layer_in)
    # 3x3 conv
    conv3 = Conv2D(f2, (3,3), padding='same', activation='relu')(layer_in)
    # 5x5 conv
    conv5 = Conv2D(f3, (5,5), padding='same', activation='relu')(layer_in)
    # 7x7 conv
    #   conv7 = Conv2D(f3, (7,7), padding='same', activation='relu')(layer_in)
    # 3x3 max pooling
    pool = MaxPooling2D((3,3), strides=(1,1), padding='same')(layer_in)
    # concatenate filters, assumes filters/channels last
    layer_out = concatenate([conv1, conv3, conv5, pool], axis=-1)
    return layer_out

def unet(input_size = (256,256,3)):
    #Input Layer
    inputs = Input(input_size)

    #Encoder network
    conv1 = naive_inception_module(inputs,16)
    pool1 = MaxPooling2D(pool_size= (2,2))(conv1)

    conv2 = naive_inception_module(pool1,32)
    pool2 = MaxPooling2D(pool_size= (2,2))(conv2)

    conv3 = naive_inception_module(pool2,64)
    pool3 = MaxPooling2D(pool_size= (2,2))(conv3)

    conv4 = naive_inception_module(pool3,128)
    pool4 = MaxPooling2D(pool_size= (2,2))(conv4)
```

```
conv5 = naive_inception_module(pool4,256)

#decoder network
up6 = Conv2DTranspose(128, (2,2), strides = (2,2), padding = "same")(conv5)
up6 = concatenate([up6, conv4], axis = 3)
conv6 = naive_inception_module(up6, 128)

up7 = Conv2DTranspose(64, (2,2), strides = (2,2), padding = "same")(conv6)
up7 = concatenate([up7, conv3], axis = 3)
conv7 = naive_inception_module(up7, 64)

up8 = Conv2DTranspose(32, (2,2), strides = (2,2), padding = "same")(conv7)
up8 = concatenate([up8, conv2], axis = 3)
conv8 = naive_inception_module(up8, 32)

up9 = Conv2DTranspose(16, (2,2), strides = (2,2), padding = "same")(conv8)
up9 = concatenate([up9, conv1], axis = 3)
conv9 = naive_inception_module(up9, 16)

#output layer
conv10 = Conv2D(1, (1,1), activation= "sigmoid")(conv9)

model = Model(inputs = [inputs], outputs = [conv10])

return model
```

```
from keras.losses import binary_crossentropy

beta = 0.25
alpha = 0.25
gamma = 2
epsilon = 1e-5
smooth = 1

class Semantic_loss_functions(object):
    def __init__(self):
        print ("semantic loss functions initialized")

    def dice_coef(self, y_true, y_pred):
        y_true_f = K.flatten(y_true)
        y_pred_f = K.flatten(y_pred)
        intersection = K.sum(y_true_f * y_pred_f)
        return (2. * intersection + K.epsilon()) / (
                K.sum(y_true_f) + K.sum(y_pred_f) + K.epsilon())

    def sensitivity(self, y_true, y_pred):
        true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
        possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
        return true_positives / (possible_positives + K.epsilon())

    def specificity(self, y_true, y_pred):
        true_negatives = K.sum(
            K.round(K.clip((1 - y_true) * (1 - y_pred), 0, 1)))
        possible_negatives = K.sum(K.round(K.clip(1 - y_true, 0, 1)))
        return true_negatives / (possible_negatives + K.epsilon())

    def convert_to_logits(self, y_pred):
        y_pred = tf.clip_by_value(y_pred, tf.keras.backend.epsilon(),
                                  1 - tf.keras.backend.epsilon())
        return tf.math.log(y_pred / (1 - y_pred))

    def weighted_cross_entropyloss(self, y_true, y_pred):
```

```
    y_pred = self.convert_to_logits(y_pred)
    pos_weight = beta / (1 - beta)
    loss = tf.nn.weighted_cross_entropy_with_logits(y_true,y_pred,pos_weight)
    return tf.reduce_mean(loss)

def focal_loss_with_logits(self, logits, targets, alpha, gamma, y_pred):
    weight_a = alpha * (1 - y_pred) ** gamma * targets
    weight_b = (1 - alpha) * y_pred ** gamma * (1 - targets)

    return (tf.math.log1p(tf.exp(-tf.abs(logits))) + tf.nn.relu(
        -logits)) * (weight_a + weight_b) + logits * weight_b

def focal_loss(self, y_true, y_pred):
    y_pred = tf.clip_by_value(y_pred, tf.keras.backend.epsilon(),
                              1 - tf.keras.backend.epsilon())
    logits = tf.math.log(y_pred / (1 - y_pred))

    loss = self.focal_loss_with_logits(logits=logits, targets=y_true,
                               alpha=alpha, gamma=gamma, y_pred=y_pred)

    return tf.reduce_mean(loss)

def depth_softmax(self, matrix):
    sigmoid = lambda x: 1 / (1 + K.exp(-x))
    sigmoided_matrix = sigmoid(matrix)
    softmax_matrix = sigmoided_matrix / K.sum(sigmoided_matrix, axis=0)
    return softmax_matrix

def dice_coefficient(self, y_true, y_pred):
    smooth = 1.
    y_true_f = K.flatten(y_true)
    y_pred_f = K.flatten(y_pred)
    intersection = K.sum(y_true_f * y_pred_f)
    score = (2. * intersection + smooth) / (
              K.sum(y_true_f) + K.sum(y_pred_f) + smooth)
    return score

def dice_loss(self, y_true, y_pred):
    loss = 1 - self.dice_coefficient(y_true, y_pred)
```

```
    return loss

def bce_dice_loss(self, y_true, y_pred):
    loss = binary_crossentropy(y_true, y_pred) + \
           self.dice_loss(y_true, y_pred)
    return loss / 2.0

def confusion(self, y_true, y_pred):
    smooth = 1
    y_pred_pos = K.clip(y_pred, 0, 1)
    y_pred_neg = 1 - y_pred_pos
    y_pos = K.clip(y_true, 0, 1)
    y_neg = 1 - y_pos
    tp = K.sum(y_pos * y_pred_pos)
    fp = K.sum(y_neg * y_pred_pos)
    fn = K.sum(y_pos * y_pred_neg)
    prec = (tp + smooth) / (tp + fp + smooth)
    recall = (tp + smooth) / (tp + fn + smooth)
    return prec, recall

def true_positive(self, y_true, y_pred):
    smooth = 1
    y_pred_pos = K.round(K.clip(y_pred, 0, 1))
    y_pos = K.round(K.clip(y_true, 0, 1))
    tp = (K.sum(y_pos * y_pred_pos) + smooth) / (K.sum(y_pos) + smooth)
    return tp

def true_negative(self, y_true, y_pred):
    smooth = 1
    y_pred_pos = K.round(K.clip(y_pred, 0, 1))
    y_pred_neg = 1 - y_pred_pos
    y_pos = K.round(K.clip(y_true, 0, 1))
    y_neg = 1 - y_pos
    tn = (K.sum(y_neg * y_pred_neg) + smooth) / (K.sum(y_neg) + smooth)
    return tn

def tversky(self, y_true, y_pred):
    y_true_pos = K.flatten(y_true)
    y_pred_pos = K.flatten(y_pred)
```

```
        true_pos = K.sum(y_true_pos * y_pred_pos)
        false_neg = K.sum(y_true_pos * (1 - y_pred_pos))
        false_pos = K.sum((1 - y_true_pos) * y_pred_pos)
        alpha = 0.7
        return (true_pos + smooth) / (true_pos + alpha * false_neg + (
                1 - alpha) * false_pos + smooth)

    def tversky_loss(self, y_true, y_pred):
        return 1 - self.tversky(y_true, y_pred)

    def focal_tversky(self, y_true, y_pred):
        pt_1 = self.tversky(y_true, y_pred)
        gamma = 0.75
        return K.pow((1 - pt_1), gamma)

    def log_cosh_dice_loss(self, y_true, y_pred):
        x = self.dice_loss(y_true, y_pred)
        return tf.math.log((tf.exp(x) + tf.exp(-x)) / 2.0)

    def loss(self, y_true, y_pred):
        return 0.5*s.focal_tversky(y_true, y_pred) +
0.2*self.weighted_cross_entropyloss(y_true, y_pred) + 0.3*surface_loss(y_true,
y_pred)
```


Find Brain Tumor
Welcome to this simple web application that can find brain tumors. Only brain MRI image may be procressed
Choose File
Drag and drop file here
Limit 200MB per file • PNG, JPG, JPEG, TIF
Browse files
TCGA_CS_4941_19960909_13.tif 198.3KB
Uploaded Image
Process
Processed
Prediction
Predicted mask
Predicted area
Prediction takes 0.938317 sec