КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА
ФАКУЛЬТЕТ РАДІОФІЗИКИ, ЕЛЕКТРОНІКИ ТА КОМП'ЮТЕРНИХ СИСТЕМ
Кафедра медичної радіофізики

«На правах рукопису»

Робота допущена до захисту в ЕК
рішенням кафедри медичної радіофізики
від___ ____2024 року, протокол № _____.
Завідувач кафедри кандидат фіз.-мат. наук, доцент
_____Сергій РАДЧЕНКО

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА
на тему:
«ВИДІЛЕННЯ ОКРЕМИХ ТИПІВ БІОЛОГІЧНИХ ТКАНИН МЕТОДАМИ НЕЙРОННИХ МЕРЕЖ ДЛЯ ТОМОГРАФІЧНИХ ДАНИХ»

Виконав:
студент 4-го курсу денної форми навчання
спеціальності 105 Прикладна фізика та наноматеріфали
ОПП «Електроніка та інформаційні технології в медицині»
Морозов Володимир Миколайович_____

Науковий керівник:
Кандидат фіз.-мат. наук,
доцент кафедри математики та теоретичної радіофізики
Нетреба Андрій В'ячеславович_____

Рецензент:
канд.фіз-мат.наук,
асистент кафедри квантової радіофізики та наноелектроніки
Лень Юрій Анатолійович_____

Засвідчую, що у цій бакалаврській роботі
немає запозичень з праць інших авторів без
відповідних посилань
Студент: _____Володимир МОРОЗОВ

Київ-2024

СПИСОК СКОРОЧЕНЬ

МРТ — магніторезонансна томографія

СКВ — середньоквадратичне відхилення

КСКВ — корінь середньоквадратичного відхилення

САВ — середнє абсолютне відхилення

DL — глибоке навчання

AF — функція активації

SGD — стохастичний градієнтний спуск

Реферат

Дипломна робота бакалавра – 38с., 16 зображень, 17 джерел.

Проведено аналіз літератури, щодо пошуку побудови, навчання, тестування, аналізу точності нейронних мереж та баз даних за допомогою яких нейронна мережа навчається та тестується.

Обрано модель на основі архітектури U-Net, яка належить до згорткових нейронних мереж, для сегментації трьох мозкових тканин на зображеннях отриманих шляхом магнітно-резонансної томографії, та проведено пошук оптимальних гіперпараметрів таких як розмір бетчу, вид оптимізатора та функції активації.

Ключові слова: нейронні мережі, гіперпараметри, функція активації, бетч, оптимізатор, томографічні зображення.

ЗМІСТ

ВСТУП

Сегментація областей для медичних зображень тканин є важливою технікою обробки зображень, що дозволяє точно виділяти різні структури для подальшого аналізу та діагностики. Цю можливість було досягнуто завдяки розвитку нейронних мереж та глибокого навчання, що революціонізували обробку медичних зображень.

Дані технології не лише автоматизують процеси, які раніше вимагали значних зусиль та часу медичних фахівців, але й прискорюють діагностику захворювань, дозволяють виявляти відмінності у структурах частин тіла, органах тканинах та допомагають у моніторингу та плануванні лікування пацієнтів з різними захворюваннями.

Крім того, такі технології є важливим інструментом у дослідженнях мозкової активності та функцій, що розширює наше розуміння роботи цього складного органу. Це відкриває широкі можливості для наукових досліджень та медичної практики, що дозволяє швидше та точніше аналізувати мозкові структури та виявляти патології.

Актуальність цього підходу зростає з кожним роком, відповідаючи зростаючим вимогам до швидкої та точної діагностики, а також глибокого розуміння функцій та структур мозку.

Метою цієї роботи є виділення трьох основних типів мозкових тканин - сірої речовини, білої речовини та цереброспінальної рідини - на зображеннях, отриманих за допомогою магнітно-резонансної томографії, що дозволить розвивати нові технології медичної діагностики.

1 Особливості побудови нейронних мереж. Гіперпараметри

У нейронних мережах гіперпараметри - це параметри, які визначають структуру та поведінку моделі під час навчання. На відміну від параметрів мережі, які навчаються на основі даних, гіперпараметри налаштовуються вручну перед початком навчання. Гіперпараметри представляють із себе:

- Функція активації: Функція, яка використовується для перетворення вхідних даних нейрона на його вихід.
- Алгоритм оптимізації(градієнтний спуск): Алгоритм, який використовується для оновлення ваг мережі під час навчання.
- Швидкість навчання: Контролює, наскільки сильно ваги мережі оновлюються під час кожного оновлення.
- Кількість шарів: Кількість шарів нейронів у мережі.
- Кількість нейронів у кожному шарі: Кількість нейронів у кожному шарі мережі.
- Розмір бетчу: Кількість зразків даних, які обробляються одночасно під час одного оновлення ваг.
- Кількість епох: Кількість разів, коли вся навчальна вибірка використовується для навчання моделі.

1.1 Лінійна регресія

Регресія – це тип навчання з учителем у машинному навчанні, який використовується для прогнозування безперервного вихідного значення на основі однієї або кількох вхідних ознак. До поширених моделей регресії належать лінійна регресія, поліноміальна регресія та регресійні дерева. Лінійна регресія передбачає лінійний зв'язок між незалежними та залежними змінними і описується рівнянням

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n \qquad (1.1.1)$$

де ŷ – прогнозоване значення, β_0 – зміщення(bias), β_1, β_2, ..., β_n – коефіцієнти або ваги, що відповідають вхідним ознакам $x_1, x_2, \ldots, x_n$, а n – кількість вхідних ознак [1]. Завдання полягає у знаходженні значень зміщення та коефіцієнтів, які мінімізують різницю між прогнозованими та фактичними значеннями, зазвичай використовуючи функцію втрат, таку як середнє квадратичне відхилення (MSE) або середня абсолютна помилка (MAE).

Поліноміальна регресія моделює зв'язок між незалежною змінною x та залежною змінною y як поліном n-го степеня, що дозволяє захоплювати складні, нелінійні зв'язки між вхідними та вихідними змінними. Загальна форма рівняння поліноміальної регресії виглядає як

$$\hat{y} = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \cdots + \beta_n x^n \quad (1.1.2)$$

де ŷ – прогнозоване значення, β_0 – перетин або зміщення, β_1, β_2, ..., β_n – коефіцієнти, які відповідають степеням x, а n – ступінь полінома. Як і у випадку лінійної регресії, метою є знаходження зміщення та коефіцієнтів, що мінімізують різницю між прогнозованими та фактичними значеннями. Однак поліноми високого ступеня можуть призводити до перенавчання, коли модель стає чутливою до шуму в тренувальних даних, що призводить до низької точності на нових даних, які модель бачить вперше.

1.2 Перцептрон

На початку 20 століття, коли наука лише починала розкривати таємниці людського мозку, двоє вчених, Уоррен Маккаллок та Вальтер Піттс, заклали фундамент для розуміння нейронних мереж. Їхня праця, опублікована в 1943 році, ґрунтувалася на тогочасних знаннях про анатомію та фізіологію нервової системи, описуючи нейрони, їх зв'язки та динаміку сигналів [2].

Основні принципи нейронних мереж:

- Нейрони: Ці фундаментальні одиниці мозку, отримують, обробляють та передають інформацію. Вони з'єднані між собою мережею синапсів, що дозволяє їм об'єднувати зусилля, створюючи складні сигнали.
- Сигнали: Схожі на електричні імпульси, які пробігають по дротах, нервові сигнали подорожують по нейронах, несучи інформацію з одного нейрона на інший.
- Поріг: Щоб активувати нейрон, сигнал повинен перевищити певний рівень, відомий як поріг. Це гарантує, що лише важливі та значущі сигнали пробиваються крізь шум, роблячи роботу нервової системи більш ефективною.
- Швидкість проведення: Нервові сигнали не подорожують миттєво. Швидкість їх поширення залежить від діаметра нейронів, подібно до того, як товстий шланг може пропускати більше води, ніж тонкий.
- Синаптична затримка: Незважаючи на те, що нервові сигнали рухаються швидко, передача інформації через синапси потребує додаткового часу. Це пов'язано з тим, що хімічні речовини, які використовуються для передачі сигналу, потребують певного часу, щоб пройти через синаптичний простір.
- Вогнетривкість: Після генерації імпульсу нейрон стає "вогнетривким" на короткий проміжок часу, роблячи його нездатним до генерування нового імпульсу. Це запобігає перевантаженню нейрона та гарантує, що сигнали не будуть занадто розмитими.
- Гальмування: Не всі сигнали збуджують нейрони. Деякі з них мають протилежний ефект, гальмуючи активність нейрона. Цей процес дозволяє нервовій системі регулювати потік інформації та формувати складні відповіді на зовнішні стимули.

На основі цих тверджень Маккаллок та Піттс розробили штучний нейрон(перцептрон)(рис.1.1.1.), який став будівельним блоком нейронної мережі.

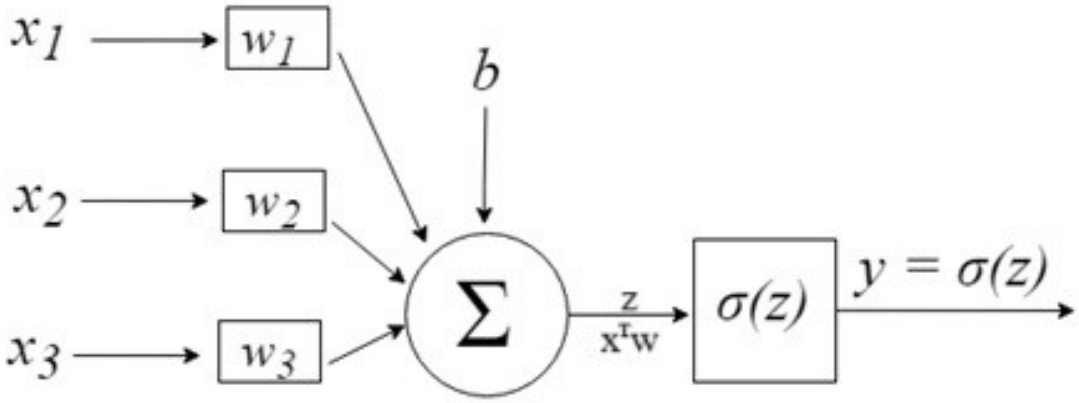

Рисунок 1.2.1. Структурна схема перцептрона.

Перцептрон складається з:

- $x_1, x_2, x_3 ..... x_N$ - вхідних сигналів - інформація, яку отримує нейрон з інших нейронів або з зовнішніх джерел. Ці сигнали можуть бути представлені у вигляді числових значень, які описують різні дані, наприклад, інтенсивність пікселів зображення або значення датчиків.
- b - зміщення(bias).
- $w_1, w_2, w_3 ..... w_N$ - вагів - коефіцієнти, які помножуються на вхідні сигнали. Ваги відображають силу зв'язку між перцептронами та визначають, наскільки кожен вхідний сигнал впливає на вихід нейрона. Ваги можуть бути як позитивними, так і негативними, що дозволяє нейрону навчатися та формувати складні зв'язки між даними.
- $\sum$ - суматор, у ньому відбувається підсумовування зважених вхідних сигналів. Суматор обчислює загальний збуджуючий або гальмуючий вплив, який отримує нейрон від своїх входів. Отримуємо скалярний добуток вектора вхідних сигналів на вектор вагів.

$$z = \left(x^T w \right) \qquad (1.2.1)$$

- σ(z) – функція активації — нелінійна функція, результатом якої є вихідний сигнал, який поступає в інший штучний нейрон, або є вихідним сигналом всієї нейронної мережі. Ця функція вводить нелінійність у модель, дозволяючи нейрону реагувати на складні

комбінації вхідних даних немонотонним способом. Завдяки активаційній функції штучний нейрон може моделювати складні нелінійні залежності, подібно до того, як це робить біологічний нейрон.

1.3 Архітектура нейронної мережі

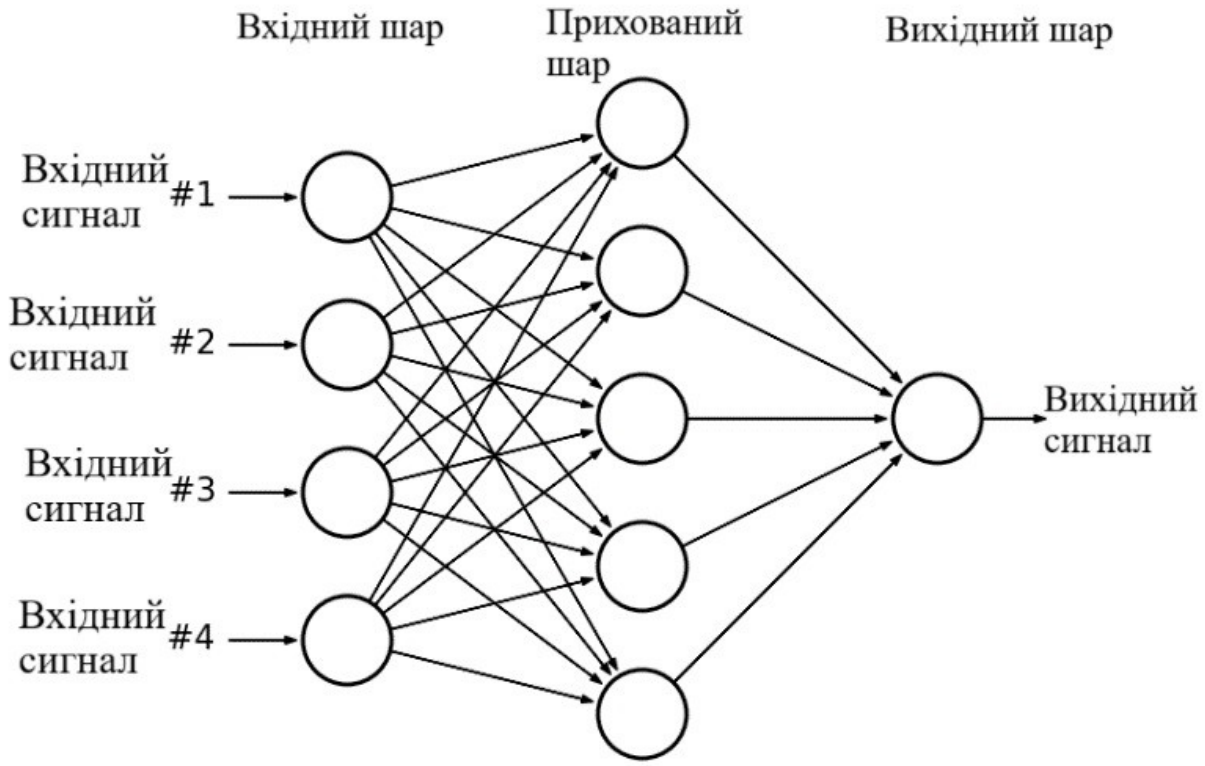

Рисунок 1.3.1. Загальний вигляд архітектури нейронної мережі.

Нейронні мережі, розроблені за аналогією до спрощеного представлення роботи людського мозку, складаються з численних взаємопов'язаних штучних нейронів(рис.1.3.1.)(кожен круг на рисунку відповідає окремому персептрону). Ці нейрони організовані в шари, кожен з яких виконує конкретну роль у процесі обробки інформації [3].

Перший шар у нейронній мережі, який приймає зовнішні дані, називається вхідним шаром. Кількість нейронів у цьому шарі відповідає розмірності вхідних даних. Наприклад, для задачі розпізнавання рукописних цифр вхідний шар може містити 784 нейрони, що відповідає 28x28 пікселям

зображення. Основна функція вхідного шару - передавати отримані дані до наступного шару без будь-якої обробки.

Приховані шари розташовані між вхідним та вихідним шарами. Мережа може містити один або кілька прихованих шарів. Кількість нейронів у прихованому шарі може бути різною і залежить від складності завдання та архітектури мережі. Приховані шари виконують складну обробку інформації, розпізнаючи приховані закономірності та зв'язки в даних. Використання нелінійних активаційних функцій у прихованих шарах дозволяє мережі моделювати складні нелінійні залежності, що є ключовим для успішного вирішення багатьох завдань.

Останній шар у нейронній мережі, який генерує кінцевий результат, називається вихідним шаром. Кількість нейронів у вихідному шарі визначається конкретним завданням. Наприклад, для задачі розпізнавання рукописних цифр вихідний шар може містити 10 нейронів, по одному для кожної цифри від 0 до 9. Нейрони у вихідному шарі генерують ймовірності для кожної категорії, що дозволяє мережі класифікувати вхідні дані.

Кожен нейрон в шарі з'єднаний з усіма нейронами наступного шару, утворюючи повнозв'язну структуру. З'єднання між нейронами мають ваги, що визначають силу взаємодії одного нейрона з іншим. В процесі навчання мережі ваги з'єднань налаштовуються, що дозволяє мережі покращувати свої прогнози або виконувати певні завдання більш точно. Архітектура нейронної мережі, зокрема кількість нейронів у кожному шарі, типи їх функцій активації та кількість шарів, суттєво впливає на її навчальні можливості та ефективність у виконанні завдань.

Існує кілька типів нейронних мереж, кожен з яких має свою унікальну архітектуру та призначення. Прямі нейронні мережі містять один або кілька прихованих шарів між вхідним та вихідним шарами. Інформація в них рухається тільки в одному напрямку - від вхідного шару до вихідного. Зворотні нейронні мережі використовують зворотній зв'язок для корекції помилок, що дозволяє їм покращувати свої прогнози через процес навчання. Згорткові нейронні мережі застосовуються для обробки даних зі збереженням їхньої

просторової структури, що є особливо корисним для аналізу зображень. Рекурентні нейронні мережі призначені для роботи з послідовностями даних, таких як текст або мовлення, де порядок елементів є важливим.

1.4 Градієнтний спуск

Градієнтний спуск – це фундаментальний алгоритм оптимізації, широко використовуваний у машинному навчанні для пошуку мінімуму функції. Це ітеративний процес, який повторно оновлює параметри моделі в напрямку, що зменшує функцію втрат [4]. Цей процес триває доти, доки функція втрат не зійдеться до мінімуму(рис.1.4.1.) .

Опис роботи алгоритму:

- Ініціалізація параметрів: Вибір початкових значень параметрів.
- Вибір випадкового зразка: Вибір випадкової точки, початку ініціалізації.
- Обчислення градієнта: Обчислення градієнта функції.
- Оновлення параметрів: Оновлення позиції у напрямку протилежному до градієнта: $\begin{bmatrix} x_k \\ y_k \end{bmatrix} = \begin{bmatrix} x_{(k-1)} \\ y_{(k-1)} \end{bmatrix} - \alpha \nabla f\left(x_{(k-1)}, y_{(k-1)}\right)$ (1.4.1)
- Повторення: Процес повторюється до досягнення збіжності.

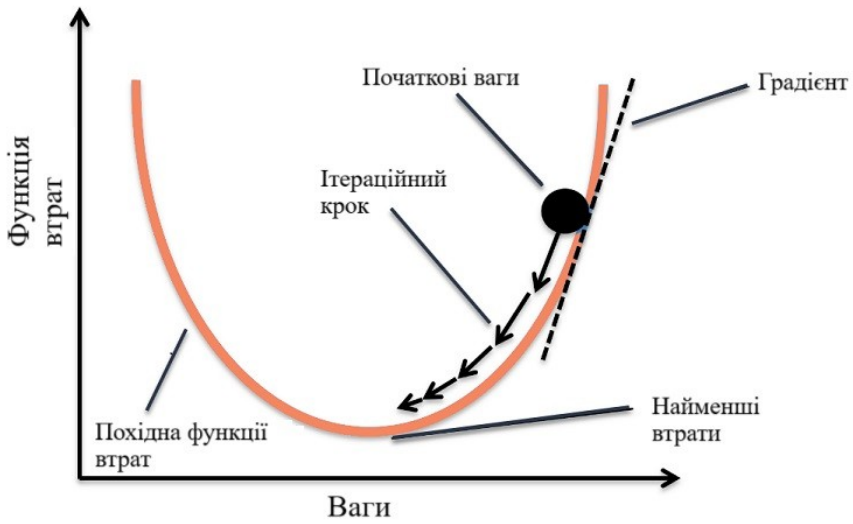

Рисунок 1.4.1. Принцип роботи алгоритму градієнтного спуску.

Важливі моменти:

- Локальні мінімуми: Градієнтний спуск може застрягти в локальних мінімумах, точках, де градієнт дорівнює нулю, але функція втрат не знаходиться в глобальному мінімумі.
- Швидкість навчання: Вибір правильної швидкості навчання є важливим. Занадто мала швидкість навчання може призвести до повільного наближення до мінімуму, а занадто велика - до перескакування мінімуму.

1.5 Функція активації та глибоке навчання

Нейронні мережі успішно використовуються в різних нових галузях для вирішення реальних складних проблем, і до сьогодні розроблено безліч архітектур глибинного навчання (DL). Щоб досягти цих найсучасніших показників, архітектури DL використовують функції активації (AF), щоб виконувати різноманітні обчислення між прихованими шарами та вихідними шарами будь-якої даної архітектури DL [5].

Алгоритми глибинного навчання є багаторівневими техніками навчання представлень, які дозволяють простим нелінійним модулям перетворювати представлення від сирого вхідного сигналу до вищих рівнів абстрактних представлень, причому багато з цих перетворень створюють вивчені складні функції. Дослідження в галузі глибинного навчання були натхнені обмеженнями традиційних алгоритмів навчання, особливо їхньою здатністю обробляти дані в сирому вигляді, [6] та техніками людського навчання шляхом зміни ваг симульованих нейронних з'єднань на основі досвіду, отриманого з минулих даних.

Використання навчання представлень, яке є технікою, що дозволяє машинам виявляти зв'язки з сирих даних, необхідне для виконання певних завдань, таких як класифікація та виявлення.

Протягом останніх шести десятиліть поле машинного навчання, гілка штучного інтелекту, почало швидко розширюватися, і дослідження в цій галузі набрали обертів, щоб диверсифікуватися в різні аспекти людського існування. Машинне навчання є областю вивчення, яка використовує принципи статистики та комп'ютерних наук для створення статистичних моделей, які використовуються для виконання основних завдань, таких як прогнозування. Ці моделі є наборами математичних зв'язків між вхідними та вихідними даними даної системи. Процес навчання — це процес оцінки параметрів моделей таким чином, щоб модель могла виконувати визначене завдання [7]. Для машин процес навчання намагається надати їм здатність навчатися без явного програмування.

2 Принципи навчання та оцінки точності нейронних мереж для обробки медичних зображень

2.1 Функції втрат

Середньоквадратичне відхилення (СКВ) - це поширена метрика помилки, яка використовується в машинному навчанні для оцінки точності моделі регресії. СКВ вимірює середню квадратичну різницю між фактичними та прогнозованими значеннями [8].

СКВ має міцну теоретичну основу в статистиці та теорії інформації. Вона ґрунтується на розподілі Гаусса (нормальному розподілі), який є поширеним розподілом ймовірностей у багатьох реальних задачах. Це робить MSE зручною для аналізу та інтерпретації. MSE тісно пов'язана з поняттям максимальної правдоподібності (maximum likelihood). Модель регресії, яка мінімізує СКВ, також максимізує ймовірність того, що дані були отримані з цієї моделі. Це робить СКВ обґрунтованим вибором для навчання моделей машинного навчання.

$$CKB = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2 \qquad (2.1.1)$$

де y_i - фактичні значення, $\hat{y}_i$ - прогнозовані значення.

СКВ є диференційованою функцією, що робить її зручною для оптимізації за допомогою алгоритмів градієнтного спуску. Це одні з найпоширеніших алгоритмів навчання в машинному навчанні. Градієнтний спуск використовується для ітеративного покращення параметрів моделі, щоб мінімізувати СКВ. Ефективність оптимізації СКВ робить її популярним вибором для навчання моделей машинного навчання. Її можна використовувати з різними типами моделей, включаючи лінійні регресії, нейронні мережі та дерева рішень.

Середньоквадратичне відхилення має просту інтерпретацію. Одиниця СКВ відповідає квадрату одиниці вимірювання цільової змінної. Це робить СКВ легко порівнянною з іншими метриками помилок, такими як середнє абсолютне відхилення (САВ) або корінь середньоквадратичного відхилення (КСКВ). Інтерпретація СКВ робить її зручною для оцінки точності моделі регресії. Ви можете легко порівняти СКВ різних моделей, щоб визначити, яка з них має найкращу продуктивність.

Незважаючи на свої переваги, СКВ має деякі обмеження, які слід враховувати. Середньоквадратичне відхилення чутливе до великих помилок, оскільки квадратична операція посилює їх вплив. Це може бути недоліком у деяких випадках, коли важливо уникати великих помилок, наприклад, у задачах медичного діагностування. СКВ залежить від масштабу цільової змінної, що означає, що моделі, які прогнозують значення в різних масштабах, не можуть бути безпосередньо порівняні за їх СКВ. Щоб СКВ працювала правильно, дані повинні бути нормалізовані, тобто значення цільової змінної повинні мати однаковий середній та стандартний відхилення.

2.2 Коефіцієнт кореляції Пірсона

Для оцінювання ефективності роботи нейронної мережі використано коефіцієнт кореляції Пірсона, позначений як r, R або r Пірсона, є статистичною мірою, що використовується для оцінки лінійного зв'язку між двома змінними [9]. Він показує, наскільки зміни значень однієї змінної співвідносяться зі змінами значень іншої змінної. Розрахунок коефіцієнта кореляції Пірсона ґрунтується на коваріації та стандартних відхиленнях двох змінних. Формула для його обчислення виглядає так:

$$r = \frac{\sum \left(x_i - \bar{x}\right)\left(y_i - \bar{y}\right)}{\sqrt{\sum \left(x_i - \bar{x}\right)^2 \sum \left(y_i - \bar{y}\right)^2}} \qquad (2.2.1)$$

де:

- r - коефіцієнт кореляції Пірсона
- Σ - сума по всіх спостереженнях
- x_i - значення і-го спостереження для змінної X
- $\bar{x}$ - середнє значення змінної X
- y_i - значення і-го спостереження для змінної Y
- $\bar{y}$ - середнє значення змінної Y

Значення коефіцієнта кореляції Пірсона може варіюватися від -1 до +1:

- -1: свідчить про абсолютно негативну кореляцію, тобто коли значення однієї змінної збільшуються, значення іншої змінної зменшуються.
- 0: свідчить про відсутність кореляції, тобто між значеннями змінних немає лінійного зв'язку.
- +1: свідчить про абсолютно позитивну кореляцію, тобто коли значення однієї змінної збільшуються, значення іншої змінної також збільшуються.

2.3 Пряме та зворотнє поширення

Пряме поширення: вхідні дані подаються на вхідний шар нейронної мережі, активуються значення для кожного нейрона від вхідного шару до вихідного, а вихідні значення мережі порівнюються з бажаними (цільовими) значеннями [4]. Далі розраховується втрат для кожного нейрона шляхом вимірювання різниці між його вихідним значенням та бажаним значенням.

Втрати поширюються назад через мережу шар за шаром, використовуючи правило ланцюгового диференціювання.

$$\omega = \omega - \alpha\, dL \qquad (2.3.1)$$

Ваги з'єднань між нейронами оновлюються в напрямку, що зменшує загальні втрати, за допомогою методу градієнтного спуску, який регулює ваги пропорційно негативному градієнту втрат. Цей процес повторюється для набору навчальних даних знову і знову. З кожною ітерацією мережа краще відображає зв'язок між вхідними та вихідними даними.

Алгоритм використовує метод ланцюгового правила для ефективного розрахунку градієнтів втрат, а ваги оновлюються невеликими кроками, щоб уникнути коливань і гарантувати стійке навчання. Навчання може тривати протягом багатьох ітерацій, поки не буде досягнута бажана точність.

Серед переваг алгоритму зворотного поширення: він є ефективним для навчання нейронних мереж з довільною структурою, може використовуватися для навчання на великих наборах даних та є основою для багатьох сучасних методів глибокого навчання. Проте, є і недоліки: алгоритм може застрягати в локальних мінімумах, не знаходячи оптимальне рішення, є чутливим до шуму в даних та може потребувати великої кількості обчислювальних ресурсів системи для складних мереж.

2.4 U-net

Нейронна мережа U-Net (скорочення від англ. "U-shaped Convolutional Neural Network") – це архітектура нейронної мережі, розроблена для сегментації медичних зображень. Її вперше представили у 2015 році Олаф Роннебергер та Філіп Фішер у своїй праці "U-Net: Convolutional Networks for Biomedical Image Segmentation". Архітектура U-Net складається з шляху звуження та шляху розширення [10].

Шлях звуження складається з блоків згортки, що зменшують просторову розмірність зображення (зменшують його висоту та ширину), але збільшують кількість каналів. Ця частина мережі відповідає за витягування складних просторових залежностей в даних. Використовуються різні типи згорткових шарів, наприклад, згортки з подвійним розміром ядра та згортки з пулінгом. Після кожного згорткового шару використовується нелінійна функція активації, наприклад, ReLU або функція активації з витіканням.

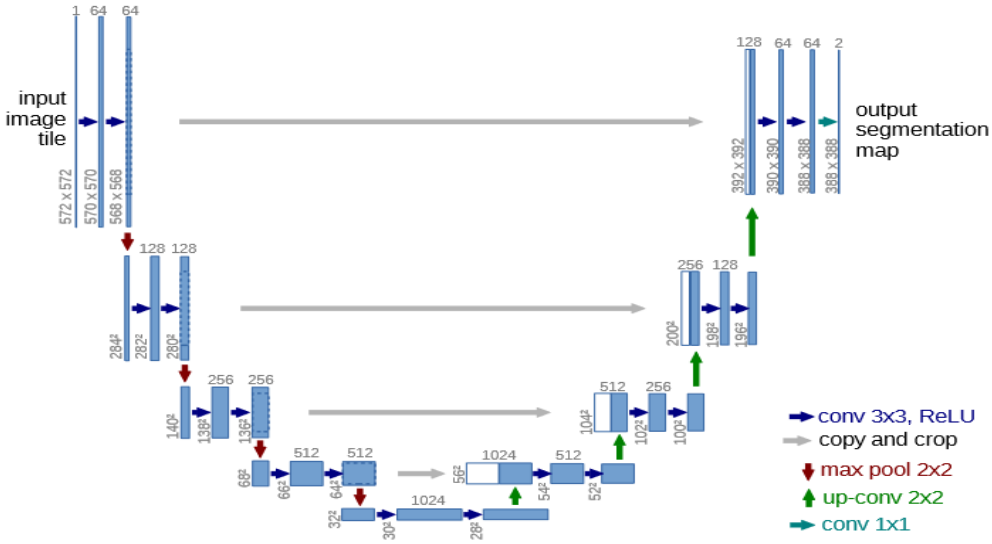

Рисунок 2.4.1. Архітектура U-Net [10].

Шлях розширення складається з блоків деконволюції, що збільшують просторову розмірність зображення (збільшують його висоту та ширину), але зменшують кількість каналів. Ця частина мережі відповідає за локалізацію рис та генерування точного сегментування. Використовуються деконволюційні шари для відновлення просторової розмірності зображення. Між відповідними блоками шляху звуження та шляху розширення використовуються з'єднання злиття, які дозволяють мережі поєднувати інформацію з різних рівнів просторової ієрархії, що істотно покращує точність сегментації.

U-Net ефективно виконує сегментацію зображень і чудово зарекомендувала себе в сегментації медичних зображень, таких як МРТ, КТ та УЗД, завдяки своїй здатності точно визначати межі між об'єктами. Завдяки своїй архітектурі U-Net може з високою точністю локалізувати краї на зображеннях,

що робить її цінним інструментом для медичних досліджень та діагностики. Крім того, U-Net має відносно просту структуру, що робить її легкою для навчання та впровадження. Вона також ефективно використовує пам'ять, що робить її придатнішою для використання на пристроях з обмеженими ресурсами.

U-Net широко використовується для сегментації різних типів медичних зображень, наприклад, сегментації пухлин, органів та тканин. U-Net також використовується в біомедичних дослідженнях для аналізу та візуалізації даних зображень. Крім медичних додатків, U-Net знаходить застосування в інших сферах, таких як комп'ютерне бачення, робототехніка та автономне керування. Її можна використовувати для сегментації об'єктів, виявлення об'єктів та картування глибини.

2.5 Різновидності оптимізаторів

Стохастичний градієнтний спуск (SGD) - це ітеративний алгоритм оптимізації, який широко використовується в машинному навчанні для тренування моделей [11]. Він використовується для пошуку мінімуму функції втрат, яка вимірює, наскільки добре модель прогнозує дані. SGD є ефективним методом для оптимізації великих та складних моделей, оскільки він може оновлювати параметри моделі за допомогою лише невеликої частини даних за раз.

SGD ґрунтується на ідеї оновлення параметрів моделі в напрямку, який призводить до зменшення функції втрат. Це робиться шляхом обчислення градієнта функції втрат щодо параметрів моделі. Градієнт вказує напрямок, в якому функція втрат змінюється найшвидше. Оновивши параметри моделі в протилежному напрямку градієнта, ми можемо зменшити значення функції втрат.

У SGD замість обчислення градієнта всієї функції втрат використовується лише градієнт невеликої частини даних, що називається пактеом(бетчем). Це робить алгоритм більш ефективним, особливо для великих наборів даних.

Оновлення параметрів моделі в SGD здійснюється за наступним правилом:

$$\theta = \theta - \alpha * \nabla L(\theta, B) \quad (2.5.1)$$

де:

- θ - вектор параметрів моделі
- α - крок навчання, який визначає, наскільки далеко ми рухаємося в напрямку градієнта
- ∇L(θ, B) - градієнт функції втрат L щодо параметрів θ, обчислений на бетчі B

Крок навчання α є важливим гіперпараметром SGD, який потрібно налаштувати вручну. Занадто малий крок навчання може призвести до повільної конвергенції, тоді як занадто великий крок навчання може призвести до нестабільності та коливань.

SGD має ряд переваг, які роблять його популярним вибором для оптимізації моделей машинного навчання:

- Ефективність: SGD може оновлювати параметри моделі за допомогою лише невеликої частини даних за раз, що робить його ефективним для великих наборів даних.
- Масштабованість: SGD
- можна легко масштабувати на розподілені системи, оскільки обчислення градієнта на бетчі можна виконувати паралельно.
- Простота: SGD є простим алгоритмом, який легко реалізувати та зрозуміти.

SGD недоліки:

- Шумність: Оскільки SGD використовує лише градієнт невеликої частини даних, оновлення параметрів можуть бути шумними. Це може призвести до коливань під час оптимізації та ускладнити визначення того, чи сходиться алгоритм до мінімуму.
- Підбір гіперпараметрів: Крок навчання α є важливим гіперпараметром SGD, який потрібно налаштувати вручну. Занадто малий крок навчання може призвести до повільної конвергенції, тоді як занадто великий крок навчання може призвести до нестабільності та коливань.

Adam (Adaptive Moment Estimation) - це алгоритм оптимізації першого порядку, що широко використовується в машинному навчанні для тренування нейронних мереж [11]. Він ґрунтується на алгоритмі стохастичного градієнтного спуску (SGD) та поєднує в собі два адаптивні моменти, щоб покращити стійкість та швидкість збіжності.

Adam використовує два адаптивні моменти: момент середнього значення (m) та момент квадрату середнього значення (v), щоб динамічно коригувати крок навчання. Ці моменти розраховуються за попередніми градієнтами, тим самим враховуючи історію оновлень параметрів.

Оновлення параметрів Adam описуються наступними формулами:

$$m_t = \beta_1 * m_{t-1} + \left(1 - \beta_1\right) * g_t \quad (2.5.2)$$

$$v_t = \beta_2 * v_{t-1} + \left(1 - \beta_2\right) * g_t^2 \quad (2.5.3)$$

$$\widehat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad (2.5.4) \qquad \widehat{v}_t = \frac{v_t}{1 - \beta_2^t} \quad (2.5.5)$$

$$\theta_t = \theta_{t-1} - \frac{\alpha}{\sqrt{\left(\widehat{v}_t + \varepsilon\right)}} \widehat{m}_t \quad (2.5.6)$$

де:

- θ_t - вектор параметрів моделі на t-му кроці
- g_t - градієнт функції втрат на t-му кроці
- α - крок навчання
- β_1, β_2 - гіперпараметри, що контролюють швидкість експоненціального забування
- ε - невелике значення, що запобігає діленню на нуль

Adam має низку переваг, які роблять його популярним вибором для оптимізації нейронних мереж:

- Швидкість збіжності: Adam часто сходиться швидше, ніж інші алгоритми оптимізації, такі як SGD або RMSProp.
- Стабільність: Adam менш схильний до коливань, ніж SGD, що робить його більш стійким до шуму в даних.
- Простота використання: Adam має лише кілька гіперпараметрів, які зазвичай добре працюють за замовчуванням.

Adam також має деякі недоліки:

- Може не працювати добре з деякими функціями втрат: Adam може бути не таким ефективним з деякими функціями втрат, як, наприклад, квадратична помилка.
- Може потребувати більше пам'яті: Adam зберігає два адаптивні моменти, що може потребувати більше пам'яті, ніж інші алгоритми оптимізації.

Adam часто порівнюють з SGD, оскільки обидва алгоритми широко використовуються для тренування нейронних мереж. Adam, як правило, має перевагу над SGD у швидкості збіжності та стійкості, особливо для задач з великими наборами даних. Проте, SGD може бути кращим вибором для задач з невеликими наборами даних або для функцій втрат, де Adam не працює добре.

2.6 Застосування функцій активації

Функції активації є невід'ємною частиною нейронних мереж, додаючи нелінійність у модель та дозволяючи їй навчатися складним зв'язкам між даними. Серед поширених активаційних функцій виділяються Tanh та ReLU, які мають свої унікальні характеристики та переваги.

Tanh(гіперболічний тангенс)(рис.2.6.1) вводить нелінійність, стискаючи вхідні значення до діапазону від -1 до 1. Її математичне представлення:

$$\sigma(x) = \tanh(x) = \frac{\left(e^{x} - e^{(-x)}\right)}{\left(e^{x} + e^{(-x)}\right)} \qquad (2.6.1)$$

При реалізації оцінювання областей, які відповідають різним біологічним тканинам для медичних зображень у роботі використано функцію активації Tanh.

Переваги Tanh:

- Зберігає негативні значення: на відміну від ReLU [12], Tanh не обнуляє негативні входи, що може бути корисним для деяких завдань, де важливі як позитивні, так і негативні сигнали.
- Центровані виходи: Tanh центрує свої виходи навколо 0, що може полегшити навчання нейронної мережі.
- Плавна диференційованість: Tanh є гладкою диференційованою функцією, що робить її придатною для методів оптимізації, заснованих на градієнтах.

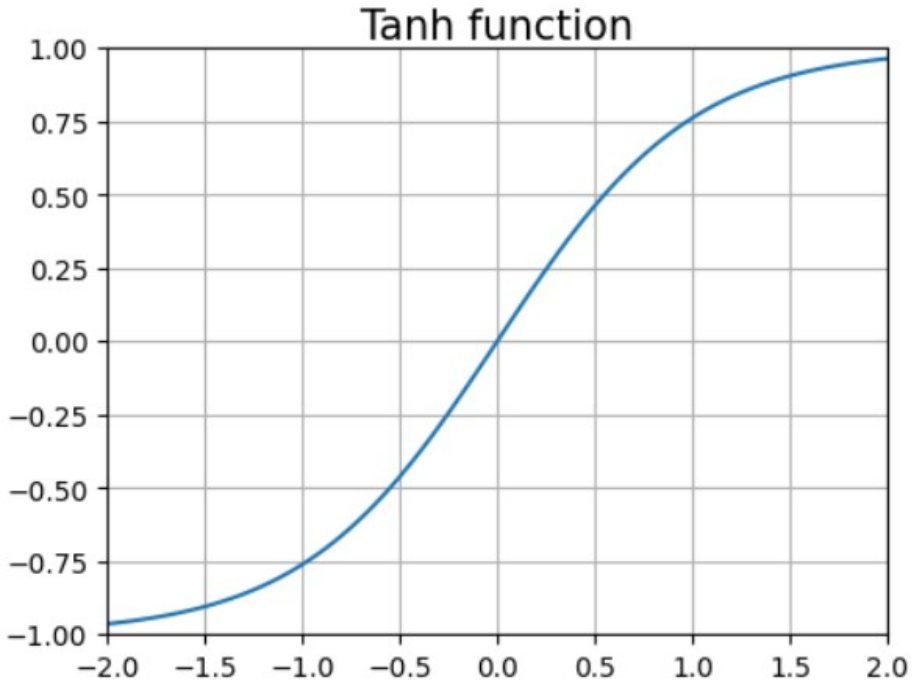

Рисунок 2.6.1. Функція активації Tanh.

Недоліки Tanh:

- Зникаючий градієнт: Tanh може страждати від проблеми зникаючого градієнта, коли похідні наближаються до 0, ускладнюючи навчання глибоких нейронних мереж.
- Насиченість: Tanh насичується в крайніх значеннях (-1, 1), що може призвести до втрати інформації при великих активаціях.

ReLU(випрямлена лінійна одиниця)(рис.2.6.2) встановлює від'ємні вхідні значення на 0, а позитивні залишає без змін:

$$\sigma(x) = ReLU(x) = max(0, x) \qquad (2.6.2)$$

Переваги ReLU:

- Простота: ReLU має просту та обчислювально ефективну реалізацію, що робить її швидшою за інші активаційні функції.

- Незникаючий градієнт: ReLU не страждає від проблеми зникаючого градієнта, оскільки похідна завжди 0 або 1.
- Розрідженість: ReLU призводить до розрідженості активацій, де більшість нейронів мають вихід 0, що може бути корисним для деяких алгоритмів регуляризації.

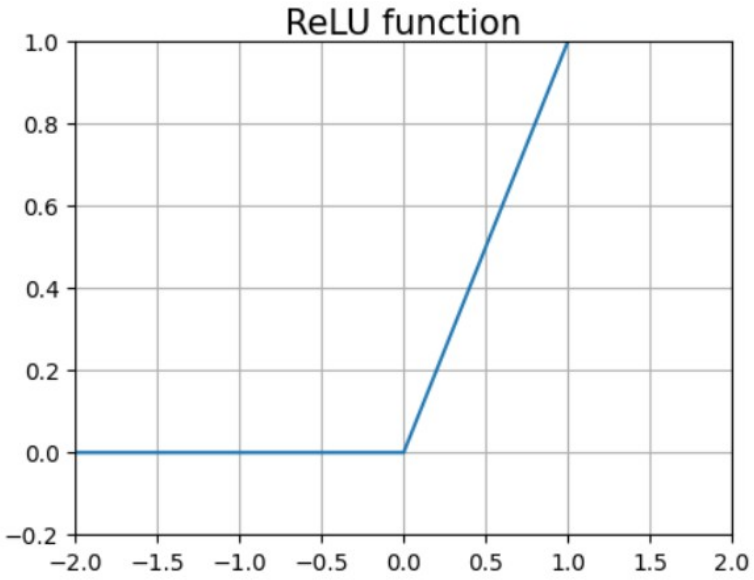

Рисунок 2.6.2. Функція активації ReLU.

Недоліки ReLU:

- "Вмираючі" нейрони: ReLU може призвести до "вмираючих" нейронів [13], які завжди виводять 0, що може ускладнити їх навчання.
- Чутливість до шуму: ReLU чутливий до шуму в даних, що може призвести до нестійкості під час навчання.
- Відсутність градієнта при 0: ReLU не має градієнта в точці 0, що може ускладнити навчання деяким алгоритмам оптимізації.

3 Параметричний експеримент

Було проведено підбір гіперпараметрів для задачі сегментації трьох типів мозкових тканин - сірої речовини, білої речовини та цереброспінальної рідини, на базі даних трьохвимірних знімків МРТ [14] та масками для кожної з тканин(рис.3.1.) , бібліотеки для роботи з нейронними мережами [15] на мові програмування Python, використовуючи архітектуру нейронної мережі U-Net.

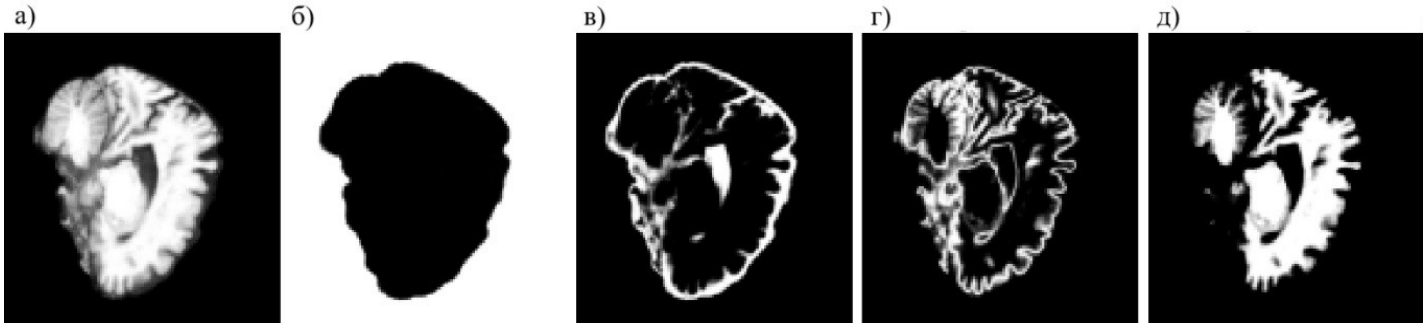

Рисунок 3.1. Зображення мозку отримане шляхом МРТ [14]
а) не оброблене зображення мозку б) сегментований задній фон в) сегментована цереброспінальна рідина г)сегментована сіра речовина д) сегментована біла речовина.

3.1 Розмір бетчу(batch size)

Бетч (batch) - це група зразків, які одночасно обробляються під час оновлення моделі машинного навчання [16]. Розмір бетча - це кількість зразків у бетчі. Бетчі використовуються для покращення ефективності навчання моделі. Обробка зразків бетчами дозволяє:

- Зменшити обчислювальні витрати: Замість того, щоб оновлювати модель для кожного зразка окремо, бетчі дозволяють оновлювати модель лише один раз для групи зразків. Це економить час та обчислювальні ресурси.

- Збільшити пропускну здатність: Обробка зразків бетчами може призвести до кращого використання пам'яті та пропускної здатності GPU або CPU, що може призвести до більш швидкого навчання.
- Зменшити шум: Оновлення моделі на основі більшої кількості зразків (бетча) може призвести до більш стійких та надійних оновлень. Це може допомогти зменшити шум градієнта та покращити загальну продуктивність моделі.

Розмір бетча може мати значний вплив на продуктивність навчання моделі [17]:

- Більші бетчі:
    - Зазвичай призводять до більш швидкого навчання.
    - Можуть призвести до кращого використання обчислювальних ресурсів.
    - Можуть призвести до більш стійких та надійних оновлень.
    - Можуть призвести до гіршої точності.
    - Можуть призвести до проблем з збіжністю.
- Менші бетчі:
    - Зазвичай призводять до більш повільного навчання.
    - Можуть призвести до гіршого використання обчислювальних ресурсів.
    - Можуть призвести до більш шумних та нестійких оновлень.
    - Можуть призвести до кращої точності.
    - Можуть допомогти з проблемами збіжності.

Оптимальний розмір бетча залежить від багатьох факторів, таких як:

- Тип моделі машинного навчання
- Розмір набору даних
- Доступні обчислювальні ресурси
- Цілі навчання

Немає універсального оптимального розміру бетча. Зазвичай рекомендується експериментувати з різними розмірами бетча, щоб знайти той, який найкраще підходить для конкретного завдання.

3.2 Оцінка розміру бетчу та кількості епох

Оскільки ключовими характеристиками є швидкість і точність навчання нейронної мережі, то для зменшення швидкості та збільшення точності базу даних розбивають на бетчі. Оскільки МРТ мозку людини є типовими, то при меншому розміру бетчу мережа отримує більше інформації з кожного зразка про що свідчать результати експерименту(рис. 3.1.1), зі збільшенням розміру бетчу, падає точність. Проте чим менше бетч, тим менше обчислювальних ресурсів використовується. А завеликі бетчі можуть призвести до проблем з пам'яттю. Нейронні мережі з великими бетчами потребують більше пам'яті для зберігання даних і оновлення ваг(рис.3.1.2). Оптимальним розміром бетчу було обрано 6 зразків.

Рисунок 3.2.1. Залежність втрат(loss) від розміру бетчу(batch size).

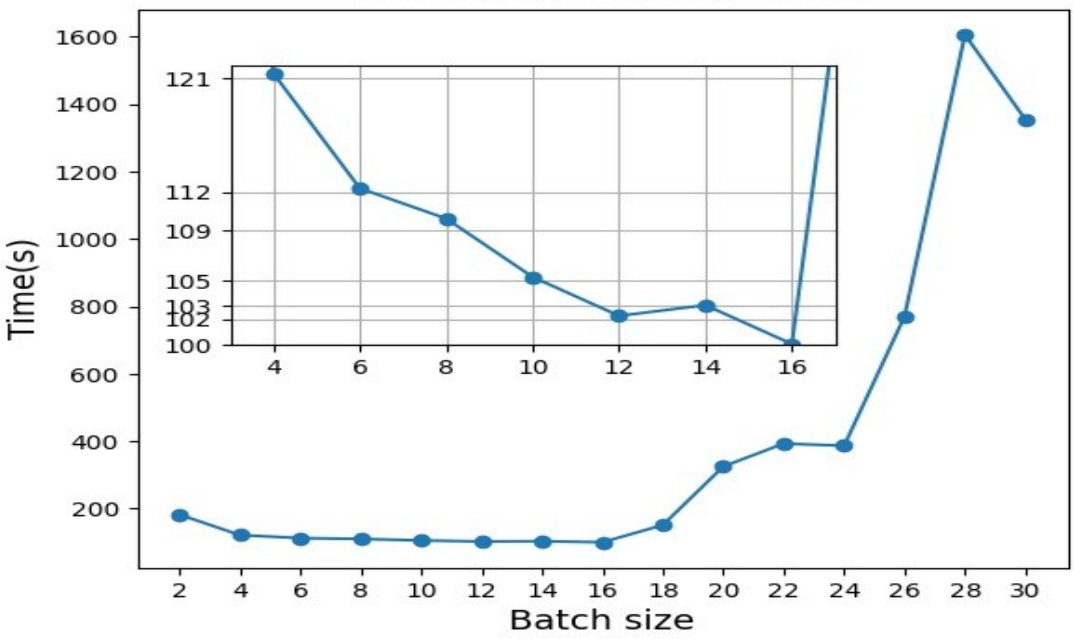

Рисунок 3.2.2. Залежність часу обчислення(time) від розміру бетчу(batch size).

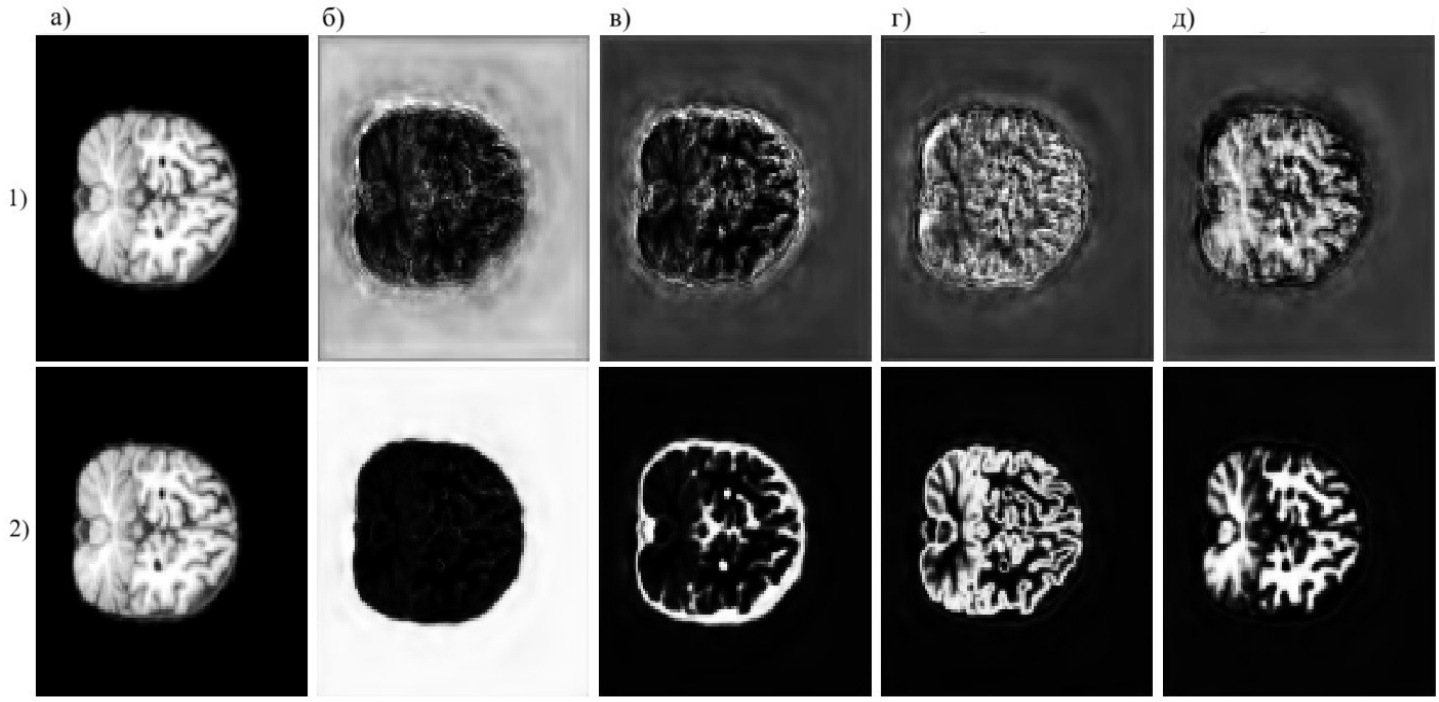

Рисунок 3.2.3. Зображення мозку отримане шляхом МРТ, сегментоване нейронною мережею

1) 2 епохи 2) 30 епох

а) не оброблене зображення мозку б) сегментований задній фон в) сегментована цереброспінальна рідина г) сегментована сіра речовина д) сегментована біла речовина.

Нейронна мережа проводить навчання епохами, що є одним з важливих аспектів процесу навчання моделей машинного навчання. Епоха означає один повний прохід через весь навчальний набір даних. Під час тренування відбувається оновлення параметрів навчання, під час валідації — оцінка оновлених параметрів, з використанням даних на яких мережа не тренувалася. За збільшення кількості епох зростає збіжність нейронної мережі(рис3.2.3.)

3.3 Оцінка ефективності оптимізаторів

На основі поставленого параметричного експерименту:

SGD та Adam мають подібну швидкість конвергенції: SGD досягає схожого рівня точності на валідації, що й Adam, приблизно за 20 епох(рис.3.3.2.). Adam трохи випереджає SGD на ранніх епохах(рис.3.2.1), але різниця не є значною. Також за рахунок простоти алгоритму SGD швидше обчислюється(рис.3.3.3.), проте вона є суттєво відмінною від алгоритму Adam. Adam дає кращу узагальнювану здатність: Adam трохи випереджає SGD за точністю на валідації в кінці навчання. Хоча рівномірне зростання SGD може здатися кращим, це не завжди гарантує кращу узагальнювальну здатність. Adam, завдяки своїй стійкості до локальних мінімумів, врешті-решт приходить до кращої точності на валідації, навіть якщо його крива зростання не така гладка.

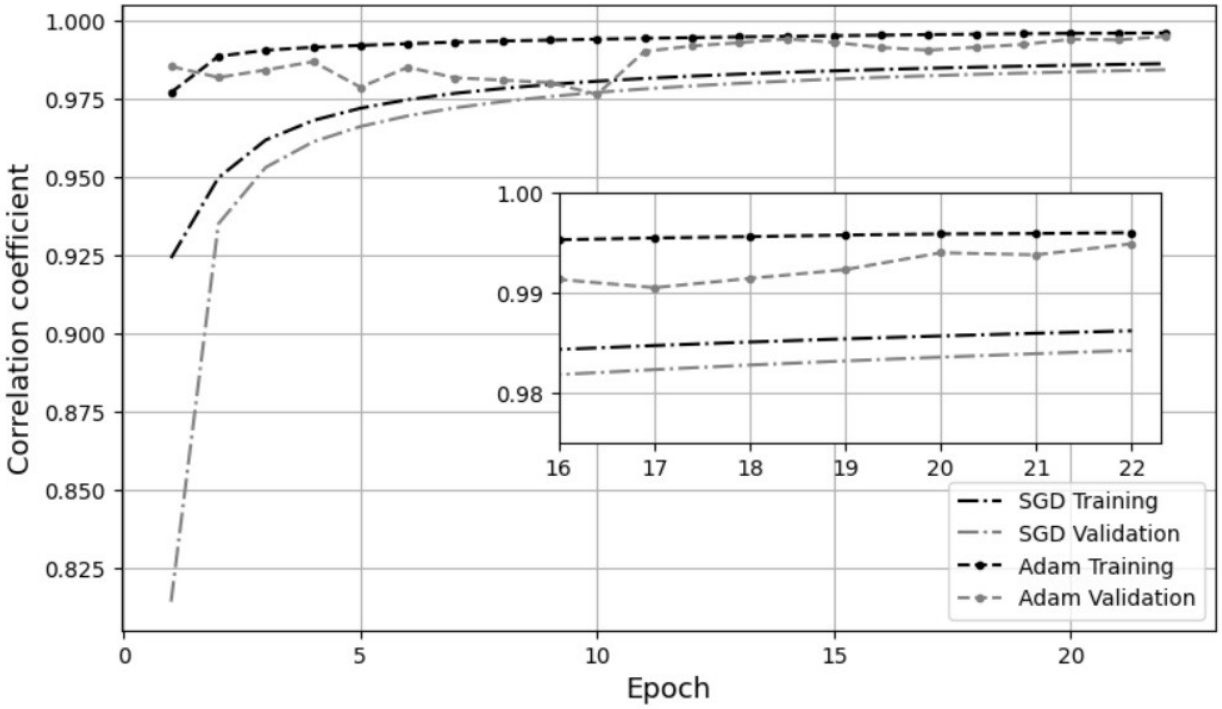

Рисунок 3.3.1. Графік оцінки вагових коефіцієнтів коефіцієнтом кореляції Пірсона зі збільшенням епох навчання, з використанням алгоритмів SGD та Adam.

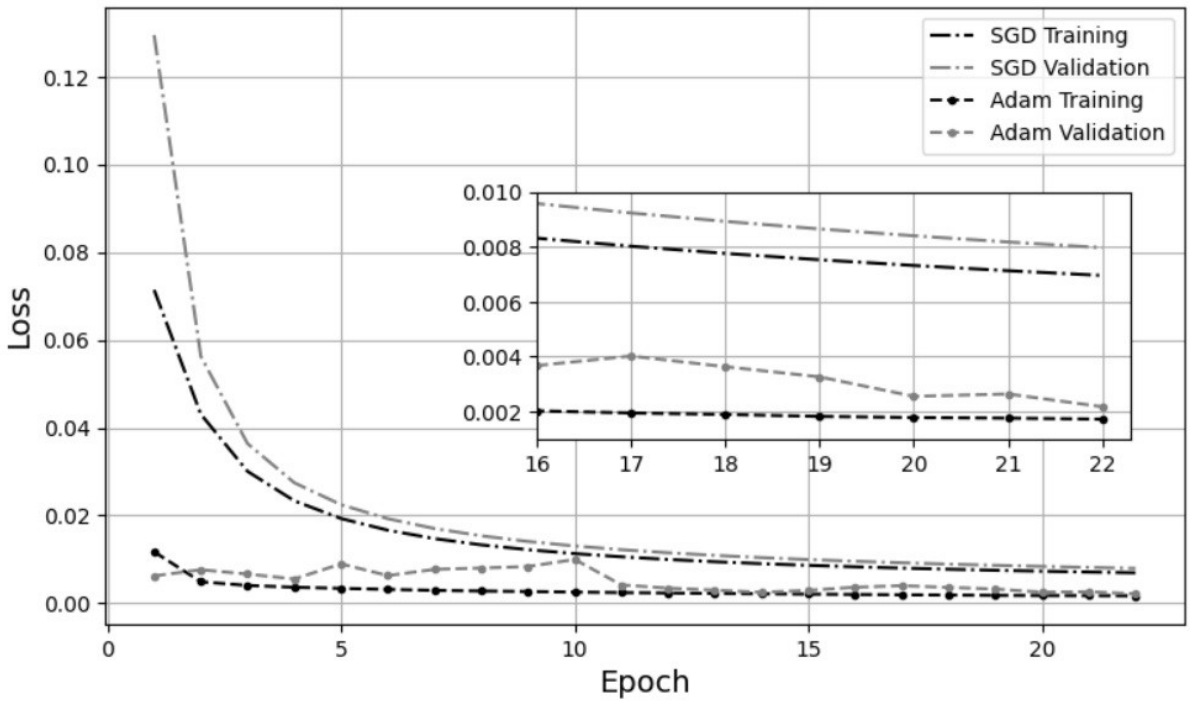

Рисунок 3.3.2. Графік оцінки точності вагових коефіцієнтів методом СКВ зі збільшенням епох навчання(Epoch), з використанням алгоритмів SGD та Adam.

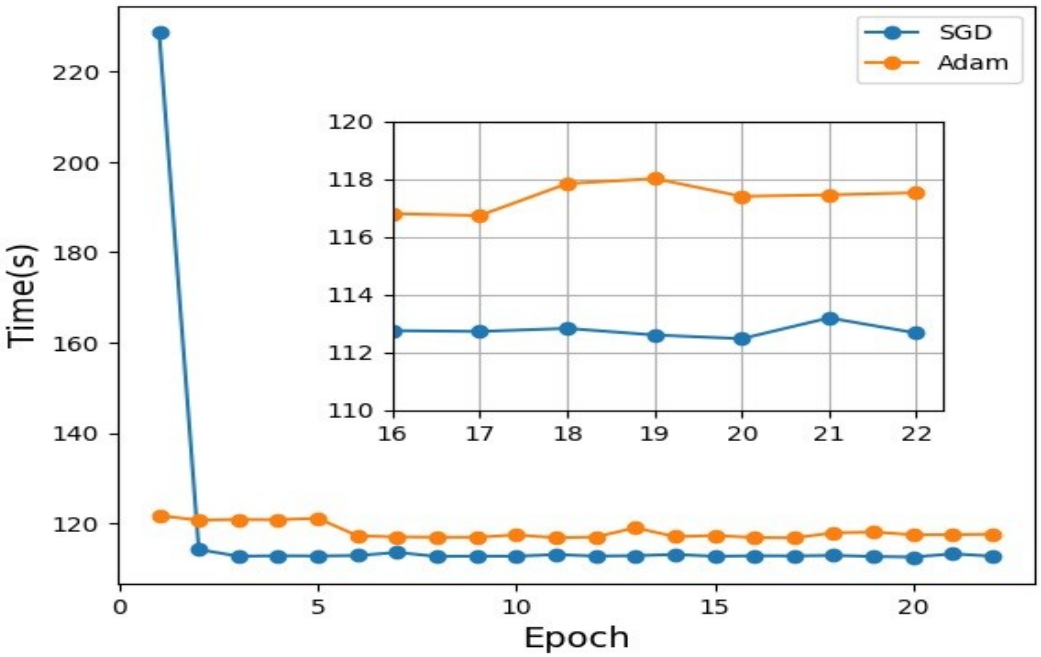

Рисунок 3.3.3. Залежність часу обчислення(time) від оптимізатора по епохах(Epoch).

3.4 Оцінка ефективності функцій активації

На основі поставленого параметричного експерименту:

ReLU та Tanh мають подібну швидкість конвергенції: обидві функції активації приводять до схожих рівнів точності на валідації протягом 20 епох. Tanh трохи випереджає ReLU на ранніх епохах, але різниця не є значною(рис.3.4.1.). Проте в подальшому ReLU випереджає Tanh в точності і дає кращу протидію зникаючим градієнтам. Також за рахунок меншої складності час обчислення функції активації ReLU менший за час обчислення Tanh(рис.3.4.3.).

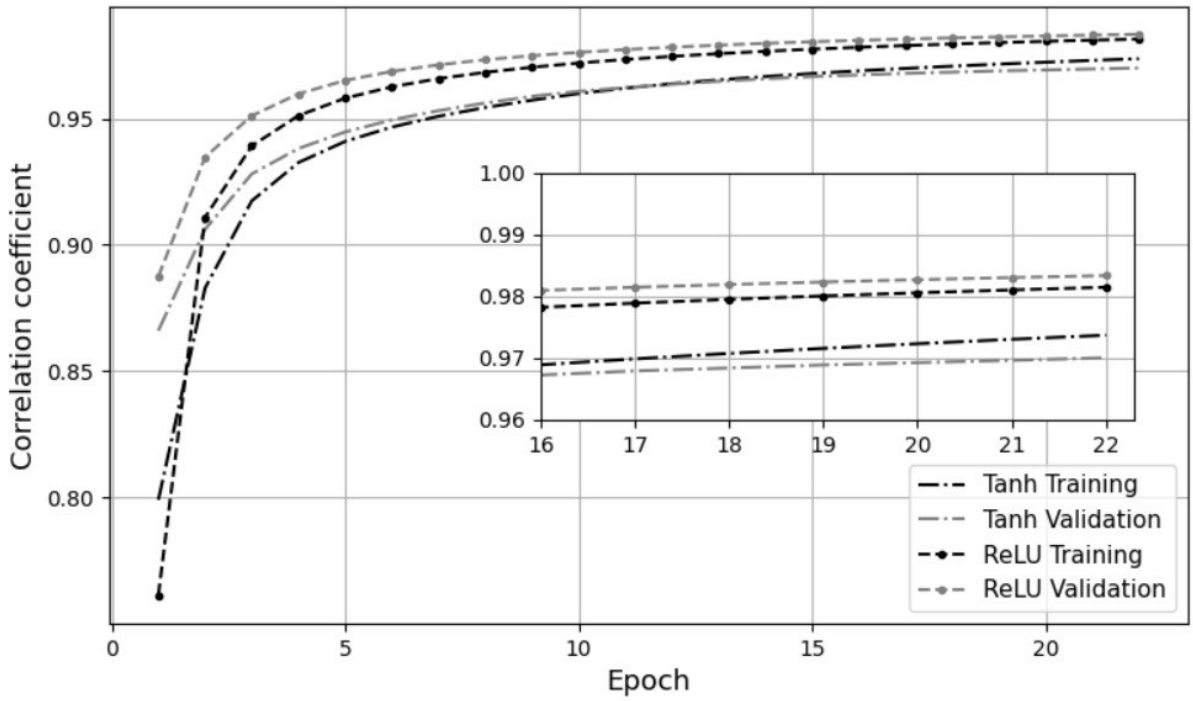

Рисунок 3.4.1. Графік оцінки вагових коефіцієнтів коефіцієнтом кореляції Пірсона зі збільшенням епох навчання, з використанням функцій активації Tanh та ReLU.

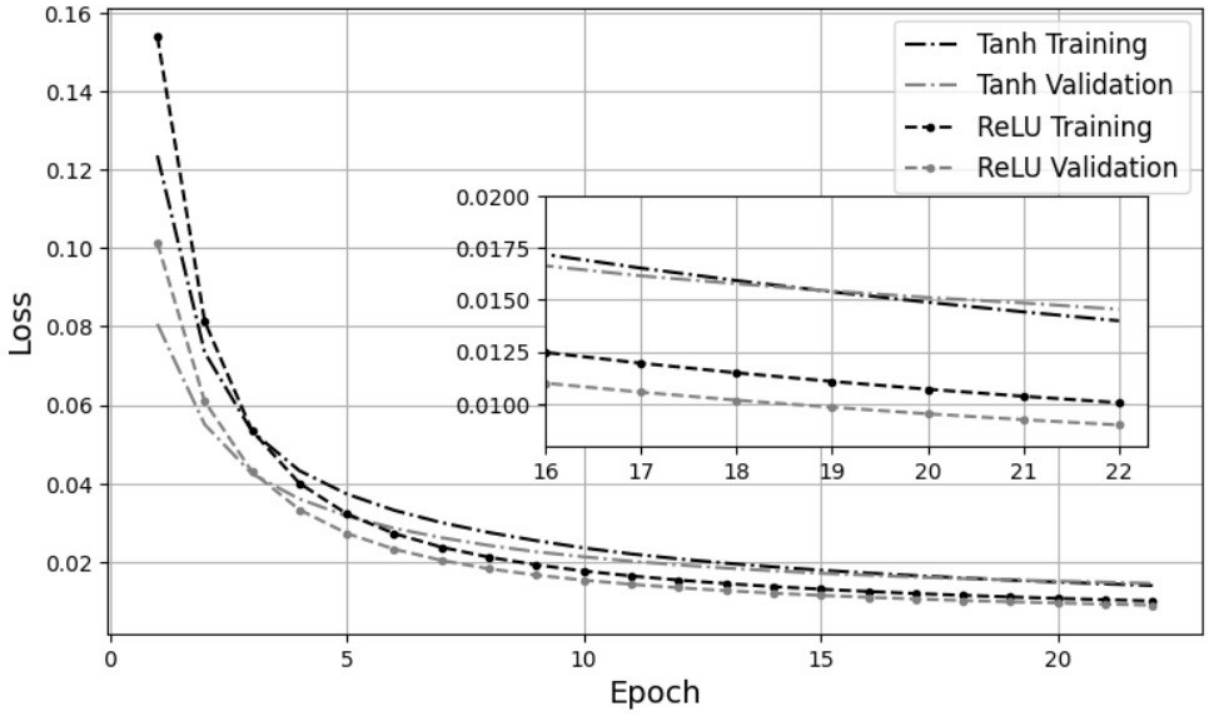

Рисунок 3.4.2. Графік оцінки точності вагових коефіцієнтів методом СКВ зі збільшенням епох навчання(Epoch), з використанням функцій активації Tanh та ReLU.

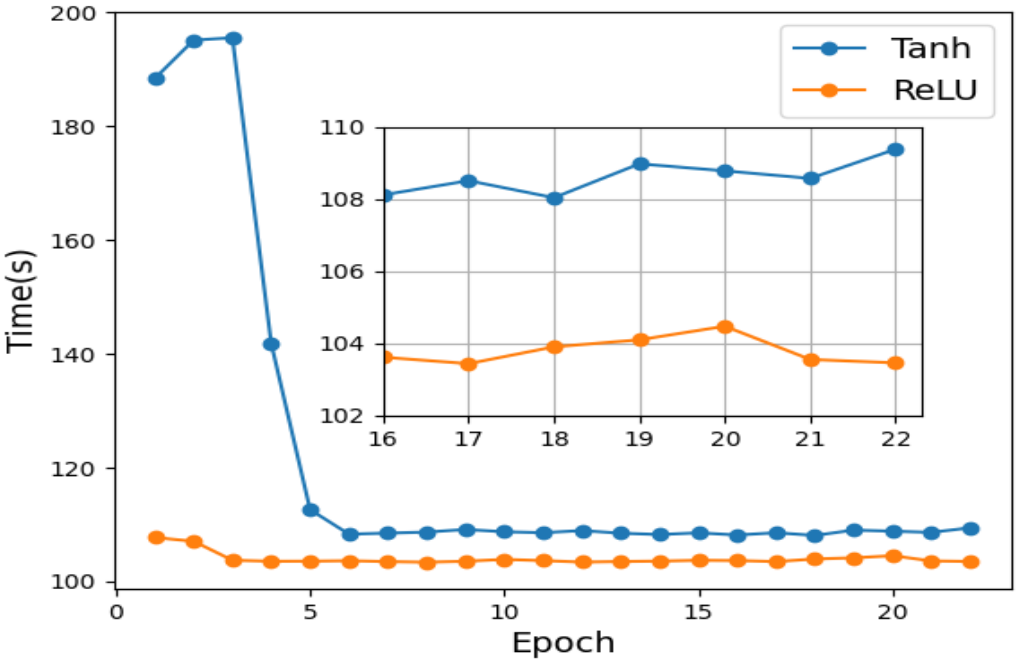

Рисунок 3.4.3. Залежність часу обчислення(time) від функції активації по епохах(Epoch).

ВИСНОВКИ

За допомогою нейронної мережі на архітектурі U-net, що базується на згорткових нейронних мережах, було проведено сегментацію трьох основних типів тканин мозку (сірої речовини, білої речовини та цереброспінальної рідини) на зображеннях, отриманих магнітно-резонансною томографією. На основі проведеного аналізу даних та поставлених експериментів було обрано оптимальний набір гіперпараметрів:

- Бетч розмір: Розмір бетчу було встановлено на 6 зразків. Це значення забезпечило баланс між швидкістю обчислень та стабільністю процесу навчання, дозволяючи ефективно використовувати обчислювальні ресурси.
- Функція активації: Була обрана функція активації ReLU (Rectified Linear Unit) для шарів мережі. ReLU допомагає уникнути проблеми зникання градієнтів, сприяючи швидшому і стабільнішому навчанню мережі. Використовуючи ReLU вдалося досягти 0.009 втрат, в той час як з Tanh 0.014.
- Оптимізатор: Оптимізатор Adam (Adaptive Moment Estimation) був обраний для процесу навчання. Adam поєднує переваги методів стохастичного градієнтного спуску з моментом, забезпечуючи швидке досягнення збіжності та хорошу узагальнювальну здатність. Adam досяг 0.002 втрат, в той час як SGD 0.008.

Проведені дослідження та експерименти підтвердили ефективність використання архітектури U-net для задач сегментації тканин мозку на МРТ-зображеннях. Завдяки своїй здатності об'єднувати інформацію як з низьким, так і з високим рівнем абстракції, U-net змогла точно виділити межі між різними типами тканин, причому зробивши великий стрибок у точності за п’ять епох(рис.3.3.1-3.3.2 та рис.3.4.1-3.4.2.) та досягла високої точності при втратах у

0.002, всього за 22 епохи. Це особливо важливо для медичних зображень, де точність сегментації може впливати на діагностику та лікування.

Використання нейронних мереж для сегментації тканин на томографічних зображеннях має значний потенціал для практичного застосування в медичній діагностиці. Автоматизована сегментація може допомогти лікарям у більш швидкій та точній оцінці стану пацієнтів, зменшуючи людський фактор і підвищуючи ефективність процесу діагностики.

ЛІТЕРАТУРА

1. James, G., Witten, D., Hastie, T., Tibshirani, R., & Ge, Y. (2013). Introduction to statistical learning with applications in Python (2nd ed.).

https://www.statlearning.com/

2. McCulloch, W. S., & Pitts, W. H. (1943). A logical calculus of ideas immanent in nervous activity.

3. Mike X Cohen (2024).A deep understanding of deep learning (with Python intro)

https://www.udemy.com/course/deeplearning_x/

4. Richard S. Sutton(1986). Two problems with backpropagation and other steepest-descent learning procedures for networks.

5. Nwankpa, C. E., Ijomah, W., Gachagan, A., & Marshall, S. (2013). Activation Functions: Comparison of Trends in Practice and Research for Deep Learning. https://doi.org/10.48550/arXiv.1811.03378

6. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. http://dx.doi.org/10.1038/nature14539

7. Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Hasan, M., Esesn, B. V., & Asari, V. K. (2018). The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches. https://doi.org/10.48550/arXiv.1803.01164

8. Fesl, B., Koller, M., & Utschick, W. (2023). On the Mean Square Error Optimal Estimator in One-Bit Quantized Systems. https://doi.org/10.1109/TSP.2023.3282063

9. Harary, M. (2024). Efficient algorithms for the sensitivities of the Pearson correlation coefficient and its statistical significance to online data. https://doi.org/10.48550/arXiv.2405.14686

10. Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. https://doi.org/10.48550/arXiv.1505.04597

11. Ruder, S. (2017). An overview of gradient descent optimization algorithms.

12. Nwankpa, C., Ijomah, W., Gachagan, A., & Marshall, S. (2018). Activation Functions: Comparison of Trends in Practice and Research for Deep Learning. https://doi.org/10.48550/arXiv.1609.04747

13. Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Searching for Activation Functions. https://doi.org/10.48550/arXiv.1710.05941

14. База даних МРТ зображень мозку та масками для сегментації трьох типів тканин:

https://www.kaggle.com/datasets/soroush361/3dbraintissuesegmentation/data

15. Бібліотека для роботи з нейронними мережами на мові програмування Python:

https://pytorch.org/

16. Yoon, J., Gupta, A., & Anumanchipalli, G. (2024). Is Bigger Edit Batch Size Always Better? -- An Empirical Study on Model Editing with Llama-3. https://doi.org/10.48550/arXiv.2405.00664

17. Li, S., Zhao, P., Zhang, H., Sun, X., Wu, H., Jiao, D., Wang, W., Liu, C., Fang, Z., Xue, J., Tao, Y., Cui, B., & Wang, D. (2024). Surge Phenomenon in Optimal Learning Rate and Batch Size Scaling. https://doi.org/10.48550/arXiv.2405.14578