НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
(КПІ ім. Ігоря Сікорського)

ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ
кафедра БІОМЕДИЧНОЇ КІБЕРНЕТИКИ

«До захисту допущено»

В.о. завідувач кафедри БМК

_____ Євген НАСТЕНКО

“___”_____2023р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою
«Комп'ютерні технології в біології та медицині»
спеціальності *122 «Комп'ютерні науки»*

на тему: **Система сегментації COVID-19 за допомогою моделі на основі трансформеру**

Виконав: студент **IV** курсу, групи _БС-93_

ОЛЕКСЕНКО ІЛЛЯ ОЛЕГОВИЧ _____
(прізвище, ім'я, по батькові) (підпис)

Керівник: *в.о.зав.каф. БМК, професор, д.б.н., к.т.н.*
Настенко Євген Арнольдович _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Консультант з розділів дипломної роботи:
інженер ДУ "ІЯМПД" **Бабенко Віталій Олегович** _____
(посада, вчене звання, науковий ступінь, прізвище, ім'я, по батькові) (підпис)

Рецензент: *зав. каф. біомедичної інженерії, д.т.н., доцент*
Шликов Владислав Валентинович _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

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

Студент _____
(підпис)

Київ – 2023 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет біомедичної інженерії
Кафедра біомедичної кібернетики

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні технології в біології та медицині»

ЗАТВЕРДЖУЮ
В.о. завідувач кафедри БМК
_____ Євген НАСТЕНКО
«_*30*_»__*травня*___2023 р.

ЗАВДАННЯ
на дипломну роботу студенту

ОЛЕКСЕНКО ІЛЛЯ ОЛЕГОВИЧ
(прізвище, ім'я, по батькові)

1. Тема роботи **Система сегментації COVID-19 за допомогою моделі на основі трансформеру**

Керівник роботи
Настенко Євген Арнольдович, проф., д.б.н., к.т.н.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «31_»_*травня*_ 2023 р. №***2106-с***

2. Термін подання студентом роботи ***06-08 червня 2023р.***
3. Вихідні дані до роботи: архів знімків КТ, мова програмування Python, фреймворки PyTorch, Tkinter, ONNX.
4. Зміст роботи: Анотації (на двох мовах), вступ, огляд літературних джерел, теоретична частина (основні поняття, методи підготовки і обробки даних, алгоритми), аналітична частина (аналіз і порівняння з існуючими алгоритмами, аналіз функціоналу, переваги і недоліки), практична реалізація, розрахунок економічного ефекту, безпека життєдіяльності та охорона здоров'я, висновки, список використаних джерел.
5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): 16 слайдів презентації, 57 рисунків, 28 таблиць.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Дипломної роботи	Бабенко Віталій Олегович, інженер ДУ "ІЯМПД"	10.04.2023	05.06.2023

7. Дата видачі завдання ***30 травня 2023 року.***

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роб оти	Примітка
1	Отримати завдання за темою ДР на практику	До 15.02.2023р.	***виконано***
2	Переддипломна практика	За графіком	***виконано***
3	Виконання розділів ДР (Вступ, аналітичний огляд літературних джерел, теоретична частина)	До кінця практики	***виконано***
4	Виконання розділів ДР (практична частина, загальні висновки, список джерел)	Не пізніше 1 тижня до засідання каф-ри	***виконано***
5	Перевірка ДР науковим керівником	Не пізніше 1 тижня до засідання каф-ри	***виконано***
6	Подання в електронному вигляді ДР та анотації до неї на перевірку нормоконтролера та плагіат (UNICHECK).	---- « ------	***виконано***
7	Надання документів на засідання кафедри	За день до засідання	***виконано***
8	Предзахист ДР та допуск до захисту дисертації	Згідно плану каф.	***виконано***
9	Подання ДР рецензенту. Отримання рецензії.	До подання пакету документів до ЕК	***виконано***
10	Подання пакету документів по ДР та супровідних до неї документів до захисту в ЕК[1]	За 5 днів до дати захисту ДР за графіком	***виконано***
11	Захист ДР в ЕК		

Студент _____ (підпис) **Ілля ОЛЕКСЕНКО** (ім'я, ПРІЗВИЩЕ)

Керівник ДР _____ (підпис) **Євген НАСТЕНКО** (ім'я, ПРІЗВИЩЕ)

Нормоконтролер _____ (підпис) **Галина КОРНІЄНКО** (ім'я, ПРІЗВИЩЕ)

[1] не пізніше ніж за 5 днів до затвердженої дати захисту ДР в ЕК

АНОТАЦІЯ

Дипломна робота за темою «Система сегментації COVID-19 за допомогою моделі на основі трансформеру» виконана студентом кафедри біомедичної кібернетики ФБМІ *Олексенком Іллею Олеговичем зі спеціальності 122 «Комп'ютерні науки» за освітньо-професійною програмою «Комп'ютерні технології в біології та медицині» та* складається зі: вступу; 3 розділів (аналітичний огляд літературних джерел за темою ДР, теоретична частина, практична реалізація задачі за темою ДР), висновків до кожного з цих розділів; загальних висновків; списку використаних джерел, який налічує 39 джерел. Загальний обсяг роботи 69 сторінки.

Актуальність теми. В останні декілька років поширеність моделі штучного інтелекту трансформер набула надзвичайних масштабів. Ефективність цієї моделі та вражаючі результати систем побудованих на її основі набули широкого розголосу та привернули увагу не тільки наукової спільноти, а й усього світу.

Не зважаючи на те, що ця архітектура була створена для галузі обробки природної мови, вона швидко розповсюдилась і на інші області штучного інтелекту. Особливо цікавою є перспектива застосування цієї технології у області сегментації медичних зображень, де вже багато років одноосібно домінує модель U-Net та її модифікації. Архітектура трансформер базується навколо механізму уваги, який славиться своїм умінням розраховувати глибокі просторові ознаки, які недоступні повністю згортковим моделям, до яких належить U-Net. Особливо перспективним є використання трансформеру в задачі сегментації COVID-19 в КТ зображеннях, де оперування глибокими просторовими ознаками може значно підвищити якість вихідних сегментаційних масок.

Мета і завдання роботи. Метою роботи є застосування нейронних мереж на базі архітектури трансформер до задачі сегментування COVID-19 в зображеннях комп'ютерної томографії та порівняння результатів точності таких моделей з традиційними, повністю згортковими мережами.

Її досягнення передбачає вирішення наступних завдань:

1. Аналіз вітчизняних та зарубіжних джерел.
2. Проведення попередньої обробки зображень
3. Тренування моделей обох типів
4. Аналіз результатів точності моделей обох типів
5. Розробка графічного інтерфейсу користувача

Використані методи. В процесі проведеного аналізу було визначено та обґрунтовано використання: мови програмування Python, безкоштовні бібліотеки для аналізу даних та оптимізації коду NumPy, ONNX, фреймворк PyTorch.

Отримані результати. В результаті роботи було виявлено, що нейронні мережі на основі архітектури трансформер мають більшу точність (близько 15%) в даній задачі ніж повністю згорткові мережі. Найбільш ефективною виявилась модель SwinUNETR, для якої було побудовано інтерфейс користувача.

Ключові слова. COVID-19, комп'ютерна томографія, сегментація, згорткова мережа, трансформер, рецептивне поле, механізм уваги.

Бібліографічний опис ДР

Олексенко І. О. Система сегментації COVID-19 за допомогою моделі на основі трансформеру : дипломна роб. бакалавра : 122 Комп'ютері науки / Олексенко Ілля Олегович. – Київ, 2023. – 69 с.

ABSTRACT

The thesis on the topic "The system of segmentation of COVID-19 using a transformer-based model" was completed by the student of the department of biomedical cybernetics of the FBMI *Oleksenko Illia Olehovich from the specialty 122 "Computer sciences" under the educational and professional program "Computer technologies in biology and medicine"* and consists of: introduction; 3 sections (analytical review of literary sources on the topic of DR, theoretical part, practical implementation of the task on the topic of DR), conclusions to each of these sections; general conclusions; of the list of used sources, which includes 39 sources. The total volume of work is 69 pages.

Actuality of theme. In the last few years, the spread of the transformer artificial intelligence model has gained extraordinary proportions. The effectiveness of this model and the impressive results of the systems built on its basis gained wide publicity and attracted the attention of not only the scientific community, but also the whole world.

Despite the fact that this architecture was created for the field of natural language processing, it quickly spread to other areas of artificial intelligence. The prospect of using this technology in the field of medical image segmentation, where the U-Net model and its modifications have been single-handedly dominant for many years, is particularly interesting. The transformer architecture is based around the attention mechanism, which is famous for its ability to compute deep spatial features that are inaccessible to fully convolutional models, to which U-Net belongs. The use of the transformer in the task of segmentation of COVID-19 in CT images is especially promising, where operating with deep spatial features can significantly improve the quality of the initial segmentation masks.

The purpose and tasks of the work. The purpose of the work is to apply neural networks based on the transformer architecture to the task of segmenting COVID-19 in computed tomography images and comparing the accuracy results of such models with traditional, fully convolutional networks.

Its achievement involves solving the following tasks:

1. Analysis of domestic and foreign sources.
2. Pre-processing of images

3. Training of models of both types
4. Analysis of the accuracy results of both types of models
5. Development of a graphical user interface

Used methods. In the course of the analysis, the use of: Python programming language, free libraries for data analysis and code optimization NumPy, ONNX, PyTorch framework was defined and justified.

The results obtained. As a result of the work, it was found that neural networks based on the transformer architecture have higher accuracy (about 15%) in this problem than uniformly convolutional networks. The SwinUNETR model, for which the user interface was built, turned out to be the most effective.

Keywords. COVID-19, computed tomography, segmentation, convolutional network, transformer, receptive field, attention mechanism.

ЗМІСТ

ВСТУП

В останні декілька років поширеність моделі штучного інтелекту трансформер набула надзвичайних масштабів. Ефективність цієї моделі та вражаючі результати систем побудованих на її основі набули широкого розголосу та привернули увагу не тільки наукової спільноти, а й усього світу.

Не зважаючи на те, що ця архітектура була створена для галузі обробки природної мови, вона швидко розповсюдилась і на інші області штучного інтелекту. Особливо цікавою є перспектива застосування цієї технології у області сегментації медичних зображень, де вже багато років одноосібно домінує модель U-Net та її модифікації. Архітектура трансформер базується навколо механізму уваги, який славиться своїм умінням розраховувати глибокі просторові ознаки, які недоступні повністю згортковим моделям, до яких належить U-Net. Особливо перспективним є використання трансформеру в задачі сегментації COVID-19 в КТ зображеннях, де оперування глибокими просторовими ознаками може значно підвищити якість вихідних сегментаційних масок.

Мета і завдання роботи. Метою роботи є застосування нейронних мереж на базі архітектури трансформер до задачі сегментування COVID-19 в зображеннях комп'ютерної томографії та порівняння результатів точності таких моделей з традиційними, повністю згортковими мережами.

Її досягнення передбачає вирішення наступних завдань:

1. Аналіз вітчизняних та зарубіжних джерел.
2. Проведення попередньої обробки зображень
3. Тренування моделей обох типів
4. Аналіз результатів точності моделей обох типів
5. Розробка графічного інтерфейсу користувача

Використані методи. В процесі проведеного аналізу було визначено та обґрунтовано використання: мови програмування Python, безкоштовні

бібліотеки для аналізу даних та оптимізації коду NumPy, ONNX, фреймворк PyTorch.

Отримані результати. В результаті роботи було виявлено, що нейронні мережі на основі архітектури трансформер мають більшу точність (близько 15%) в даній задачі ніж поівністю згорткові мережі. Найбільш ефективною виявилась модель SwinUNETR, для якої було побудовано інтерфейс користувача.

Структура роботи

Дипломна робота за темою «Система сегментації COVID-19 за допомогою моделі на основі трансформеру» виконана студентом *Олексенком Іллею Олеговичем зі спеціальності 122 «Комп'ютерні науки» за освітньо-професійною програмою «Комп'ютерні технології в біології та медицині»,* побудована за класичним типом та викладена на 69 сторінках машинописного тексту. Вона складається з: вступу; 3 розділів (аналітичний огляд літературних джерел за темою ДР, теоретична частина, практична реалізація задачі за темою ДР), висновків до кожного з цих розділів; загальних висновків; списку використаних джерел, який налічує 39 джерел та додатків (3 – на кирилиці, 36 – на латиниці). В роботі представлено 57 рисунків і 28 таблиць.

Робота була виконана в рамках ініціативної наукової роботи кафедри БМК НДР д/р № 0123U100866 «Методи та моделі ідентифікації станів об'єктів в задачах прийняття медичних рішень». Зареєстовано 06-02-2023.

РОЗДІЛ 1
ОГЛЯД ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1.1 Діагностика COVID-19 та перспективи застосування штучного інтелекту

Вперше виявлений у 2019 році COVID-19, збудником якого є вірус SARS-CoV-2, швидко поширився по всьому світу, ставши глобальною пандемією [1][2]. На сьогоднішній день це захворювання стало причиною смерті мільйонів людей по всьому світу. Виявлення і діагностика цього захворювання є ключовими факторами у боротьбі з його поширенням. На сьогоднішній день існує кілька методів діагностики COVID-19, серед найбільш поширених можна назвати полімеразна ланцюгова реакція (ПЛР), імунологічні тести, тести на антигени та багато інших [3].

Однак, окрім виявлення COVID-19 часто також необхідно локалізувати та визначити ступінь поширеності інфекції. Для вирішення цієї задачі використовують комп’ютерну томографію (КТ).

Принцип роботи КТ полягає в застосуванні рентгенівського випромінювання та комп'ютерної обробки отриманих даних. Під час КТ обстеження пацієнт розміщується на спеціальному столі і поміщується в томограф, що складається з рентгенівського джерела та детекторів, які обертаються навколо пацієнта. Рентгенівське випромінювання проникає через органи та тканини пацієнта, а детектори реєструють його пропускання через тіло [4][5][6].

Отримані дані передаються до комп'ютера, де застосовуються спеціальні алгоритми для обробки інформації. За допомогою математичних розрахунків, комп'ютер створює детальне перерізне зображення органів та тканин у формі сіро-білих плям.

Основним недоліком виявлення не тільки COVID-19, а й будь якого іншого захворювання за допомогою КТ є те, що це задача вимагає високої кваліфікації та

досвіду лікаря-радіолога, оскільки зображення КТ є просторовим і виявлення, зазвичай, невеликої кількості інфікованих клітин є важкої та трудоємкою задачею [7][8].

Вищезазначені складнощі спонукали багатьох дослідників звернутися за допомогою до штучного інтелекту щоб, якщо но вирішити, то хоч значно спростити цю задачу для лікарів.

Але хоча ШІ вже мав деякий успіх в сегментації інших захворювань, в контексті задачі сегментації COVID-19 традиційні методи згорткових нейронних мереж досі не мали помітного успіху [9]. Ця відносна невдача традиційних методів наводить на думку звернутися до альтернативних методів, особливо сьогодні, коли нейронні мережі засновані на архітектурі трансформер проводять справжню революцію в області обробки природної мови та привертають до себе увагу не тільки наукової спільноти, а й всього світу [10][11].

1.2. Типи алгоритмів сегментування

1.2.1. Аналіз повністю згорткових алгоритмів сегментування

Революцію в області сегментації зробила архітектура U-Net, яка завдяки своїй простоті та ефективності стала найпопулярнішою моделлю сегментації та однією з найбільш відомих нейронних мереж в цілому.

Архітектура U-Net, запропонована Роннебергером та співавторами у 2015 році в [12], стала значним проривом у медичній сегментації зображень. U-Net є нейронною мережею, заснованою на архітектурі FCN [13]. Основним призначенням цієї мережі є сегментація медичних зображень, та протягом довгого часу вона була “state of the art” інструментом для багатьох задач цієї галузі [12].

Популярність мережі U-Net зберігалась протягом довгого часу і досі активно використовується в багатьох системах діагностики. Подібно FCN, структура U-Net стала основою цілого роду мереж, які тим чи іншим чином модифікують або поєднують з іншими методиками оригінальну архітектуру. Яскравими прикладами таких мереж можна назвати Dense U-Net, U-Net++, FlexibleU-Net [14][15][16].

Серед недоліків U-Net, мабуть найбільш помітним є складність та нетривіальність обробки просторових зображень, так як оригінальна структура призначена для обробки двовимірних зображень [17]. Для вирішення цієї проблеми була розроблена низка моделей, які модифікують оригінальну архітектуру для роботи з багатовимірними масивами. Найпопулярнішими з них є VNet та 3D U-Net. Більшість з цих моделей по більшій мірі лише використовують тривимірні згорткові шари, та кардинально не змінюють оригінальну структуру [14].

1.2.2 Аналіз алгоритмів сегментування на основі трансформерів

До недавнього часу в галузі обробки природної мови домінували рекурентні нейронні мережі (RNN), які рекурентно обробляють послідовні дані беруть до уваги вже обчисленні значення для підвищення контексту [18].

У якості альтернативи 2017-го року у [19] була презентована архітектура трансформер, яка швидко стала стандартом у своїй галузі, обійшовши у популярності RNN. З того часу вийшло багато модифікацій та покращень цієї архітектури, та багато фахівців з інших галузей штучного інтелекту стали використовувати трансформери у своїх роботах. Популярність цієї архітектури та інструментів які були створені на її основі сягнули такого високого рівня, що здобули всесвітнє відомість [20].

Найпопулярніша модифікація архітектури трансформер, генераторний попередньо навчений трансформатор (GPT) та програма ChatGPT побудований на її основі у березні 2023 року мала понад 1 мільярд користувачів [20].

Суть архітектури трансформер полягає у використанні блоків уваги, які обчислюють глибокі зв’язки між елементами вхідної послідовності та вирішують проблему вузькості вихідних даних кодувальника, яка обмежувала роботу RNN [19].

Ефективність трансформеру в галузі обробки природної мови надихнула багатьох дослідників використовувати їх у своїх галузях. Першою успішною спробою використання архітектури трансформер в обробці зображень є модель ViT (Vision Transformer) [21], робота якої зображена на рис. 1.1.

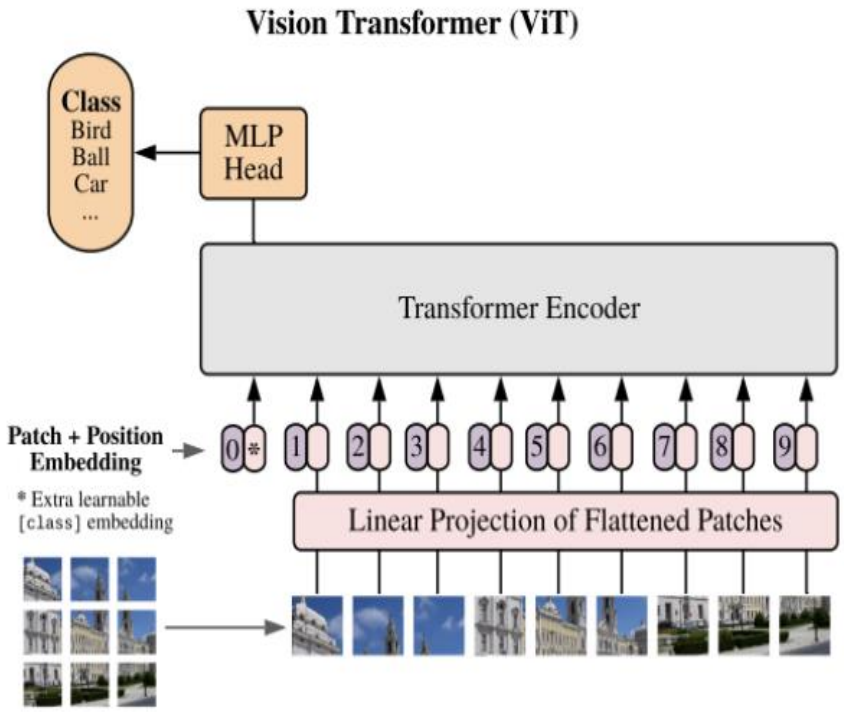

Рисунок 1.1 - Візуалізація роботи моделі ViT [21]

Спочатку розроблений для задачі класифікації, ця модель набула великої популярності та стала своєрідним доказом ефективності трансформерів у інших сферах окрім обробки природної мови. Основною відмінністю цієї моделі від своїх аналогів є використання зображень у якості послідовності, перед подачею даних у модель, вони розбиваються на певну кількість зображень розміром 16x16 [21].

В області сегментації медичних зображень трансформери ще не набули значної популярності, хоча поступово починають з'являтися моделі, які показують перспективні результати. Такі моделі, як наприклад UNETR, SegFormer, Segmenter показали вищі результати за традиційні U-Net та FCN методи на деяких задачах [22][23][24].

Виходячи з надзвичайної популярності та багаторазово доведеної ефективності трансформеру в різних областях використання штучного інтелекту, можна сказати те, що застосування цієї структури в області медичної сегментації має високий потенціал і дослідження та розвиток моделей заснованих на

трансформерах, має сенс [25]. Особливо є сенс в застосуванні цих моделей для сегментації COVID-19. Сегментації цього захворювання є доволі новою та складною задачею, яка вимагає точного обчислення глибоких просторових ознак, особлива ефективність у вилученні яких є однією з головних переваг моделі трансформер [2][26]. Більше того, в останні роки традиційні методи, засновані на повністю згоркових мережах, не мали особливого успіху у вирішенні цієї задачі, що робить трансформери ще більш привабливим варіантом [10].

Хоча дослідження нейронних мереж заснованих на трансформерах в контексті задачі сегментації COVID-19 не було надто активним в останні роки, успіхи таких мереж як UNETR та Segmenter в інших задачах медичного сегментування та відносна застарілість традиційних згорткових мереж, робить перспективу успішного застосування архітектури трансформер до цієї задачі як ніколи ймовірною [27][28].

Висновки до розділу 1

За часи розвитку сегментації, її методи пройшли значну трансформацію і до недавнього часу були зосереджені довкола повністю згорткових мережах. Зокрема в області сегментації медичних зображень мережа U-Net вже багато років є одноосібним лідером по популярності та частоті використання. Однак вражаюча ефективність трансформерів в області обробки природної мови та доведена ефективність в інших областях спонукає перевірити їх ефективність і для задачі сегментації COVID-19, особливо звертаючи увагу на те що ці методи вже мали деякий успіх в інших задачах сегментації.

РОЗДІЛ 2
ТЕОРЕТИЧНА ЧАСТИНА

2.1 Згорткові нейронні мережі

В області обробки зображень вже багато років найефективнішим методом вважається згорткова нейронна мережа, яка була створена надихаючись людським мозком та способом, яким він сприймає візуальну інформацію. На відміну від звичайних нейронних мереж, згорткова мережа опрацьовує інформацію поступово, подібно оку людини вона має так зване “рецептивне поле” (рис. 2.1), тобто поле уваги, на якому робиться основний акцент той чи інший момент [29][30].

Як видно з рис. 2.1 рецептивне поле приймає лише частино інформації на зображення, ця інформація потім обробляється операцією згортки для отримання релевантних ознак з обробленої частини зображення. Ці ознаки, в свою чергу, подаються на наступний шар мережі і той самий процес повторюється знову.

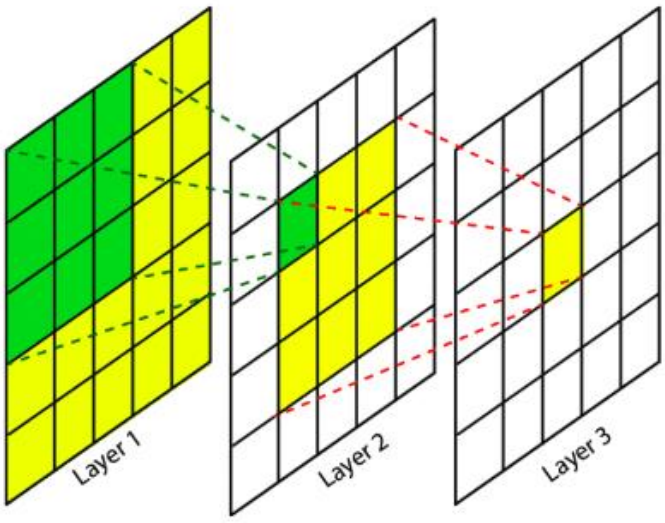

Рисунок 2.1 - Візуалізація рецептивного поля згорткової нейронної мережі [29]

Основними компонентами згорткових нейронних мереж є:

1. Згортковий (convolutional) шар: цей шар являється основною складовою цього типу мереж, саме в ньому відбуваються всі процеси обробки візуальної інформації. Залежно від типу шару, на його вхід може подаватися двовимірний або багатовимірний масив, який проходить через операцію згортки, що дозволяє вилучати локальні візуальні ознаки з різних частин зображення. Після згортки отримується карта ознак, яка відображає активацію кожного фільтра на різних місцях вхідного зображення. Результат роботи цього шару передається наступному шару, подібно відповідним нейронам в зоровій корі головного мозку людини [29]. Операція згортки зображена на рис. 2.2.

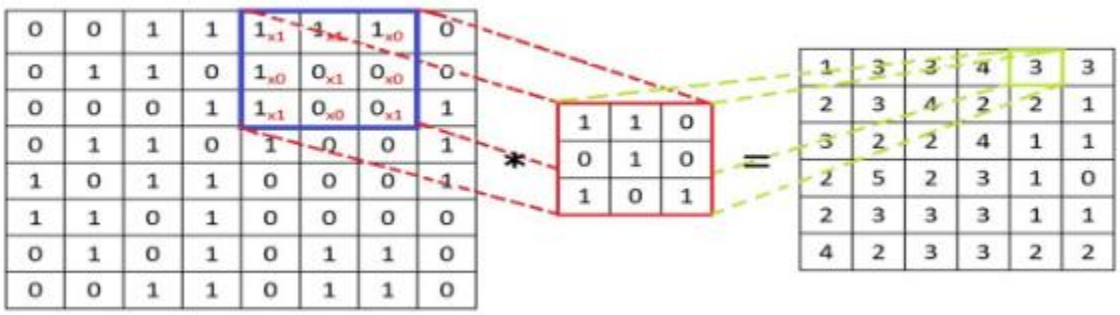

Рисунок 2.2 - Візуалізація операції згортки [30]

2. Функція активації: після кожної операції згортки на кожному шарі використовується нелінійна функція активації, найчастіше використовується функція ReLU (Rectified Linear Unit), однак вибір активації має залежати від типу задачі та формату даних. Метою функції активації є введення нелінійності в модель і отримання необхідного рівня узагальнення [31].

Функція активації ReLU обчислюється за наступною формулою:

$$ReLU(z) = \begin{cases} z, z > 0 \\ 0, z \leq 0 \end{cases} \quad (2.1)$$

Як видно з формули 2.1, всі від’ємні значення, які подаються на вхід функції ReLU будуть дорівнювати нулю, в той час як додатні значення залишаються без

змін. Такий підхід був доведений ефективним і наразі є стандартом для мереж обробки зображень [31]. Графік функції ReLU продемонстрований на рис. 2.3.

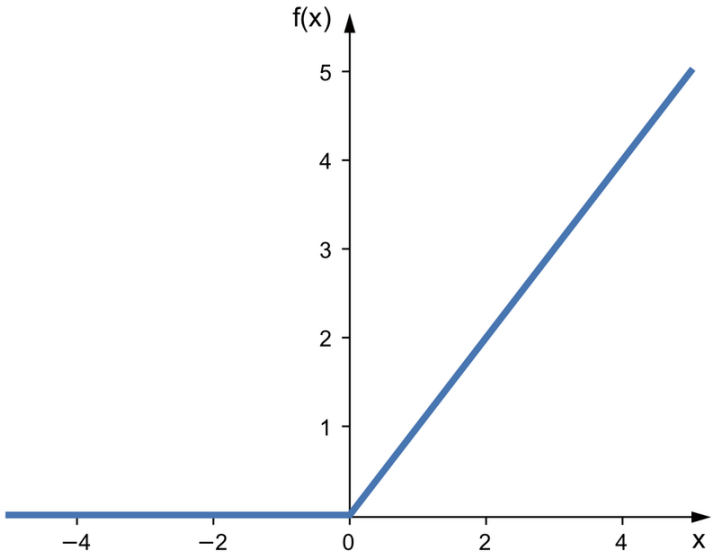

Рисунок 2.3 - Графік функції ReLU [31]

3. Шар пулінгу: цей шар призначений зменшує розмір карти ознак шляхом підвищення просторової інваріантності. Зазвичай застосовується пулінг максимуму, де найбільше значення вибирається з певного регіону, однак середній пулінг також інколи використовується. Пулінг дозволяє знизити кількість параметрів і обчислювальну складність моделі, а також зберегти важливі візуальні ознаки [32]. Шар пулінгу показаний на рис. 2.4.

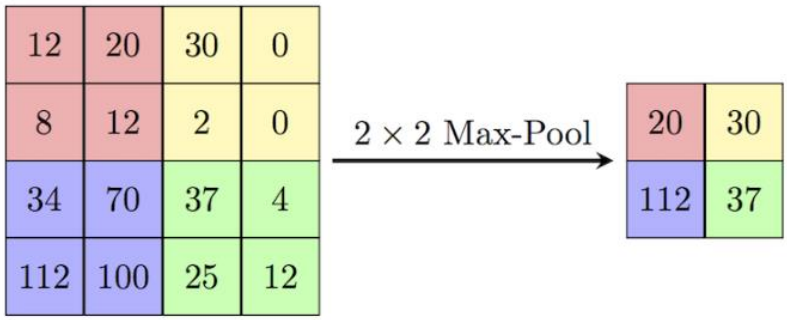

Рисунок 2.4 - Ілюстрація роботи шару пулінгу максимуму з полем розміру 2x2 [32]

4. Повнозв'язний (Fully Connected) шар: в кінці згорткової нейронної мережі зазвичай додається один або декілька повнозв’язних шарів, які об'єднують вилучені ознаки для класифікації або регресії. Ці шари розпізнають вищий рівень ознак, які відповідають конкретним класам або категоріям.

Найбільш розповсюджена структура згорткової мережі зображена на рис. 2.5.

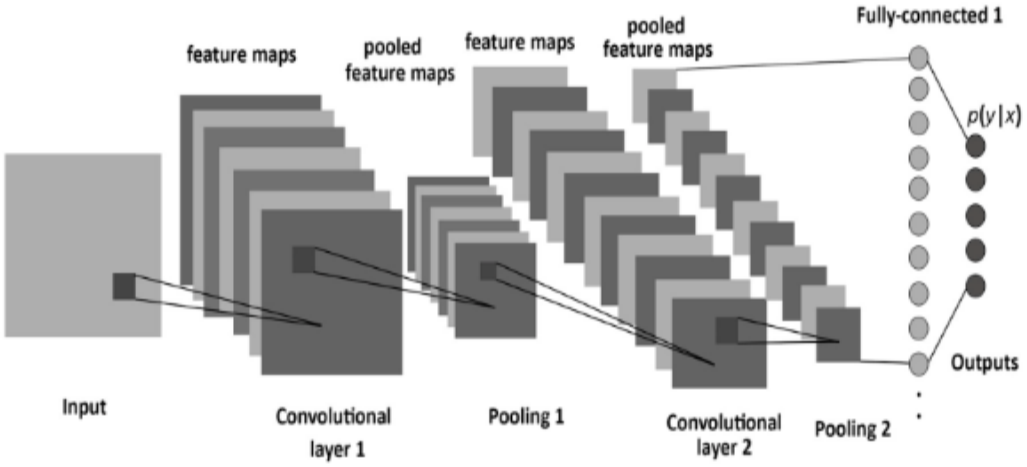

Рисунок 2.5 - Приклад структури згорткової мережі [30]

2.2 Повністю згорткові мережі

Як було згадано вище, згорткова нейронна мережа, у своїй стандартній формі, передбачає використання декількох повнозв’язних шарів, які допомагають обчислити загальні ознаки, необхідні для задач класифікації та регресії. Однак, як було показано у нейронній мережі під назвою FCN (Fully-Connected Network), для задач, де вихідними даними моделі є багаторозмірний масив, повнозв’язні шари можуть бути відкинуті [13]. Найчастіше такий метод використовується в задачах сегментації. В області сегментації медичних зображень найчастіше використовується мережі U-Net (рис. 2.6), яка наслідує структуру кодер-декодер. Суть цієї мережі полягає в тому, щоб вилучити з зображення найважливіші ознаки та побудувати з них, дзеркальний до кодера шляхом, маску сегментації [12].

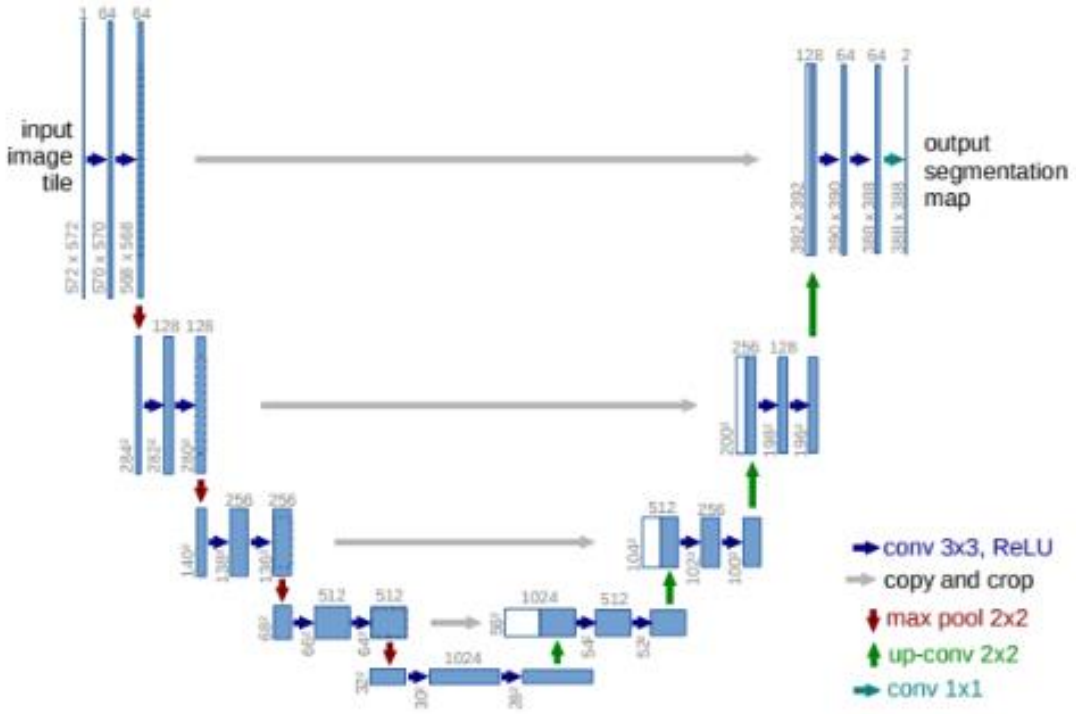

Рисунок 2.6 - Архітектура U-Net (приклад для 32x32 пікселів у найнижчій роздільній здатності) [12]

2.3 Архітектура трансформер

2.3.1 Оригінальна архітектура

З’явившись у 2017 році у [19], модель трансформер (рис. 2.7) підкорила всю область штучного інтелекту та дозволила продукувати неймовірні результати, які були до цього недосяжні. Від генерації фотореалістичних зображень до чат-ботів з глибоким розуміння контексту та лінгвістичних нюансів, трансформери привернули до себе увагу всього світу своєю незвичайною ефективністю [10]. Причина цієї ефективності полягає у своїх комплексній та багаторівневій структурі. Ця архітектура, яка як і багато інших сучасних моделей наслідує структуру кодер-декодер, однак що відрізняє її від інших, це модифікований блок уваги, який присутній у обох частинах архітектури і відіграє роль обчислювача глибоких контекстуальних ознак, таких які ніяка інша модель до цього не мала змоги обчислити [33].

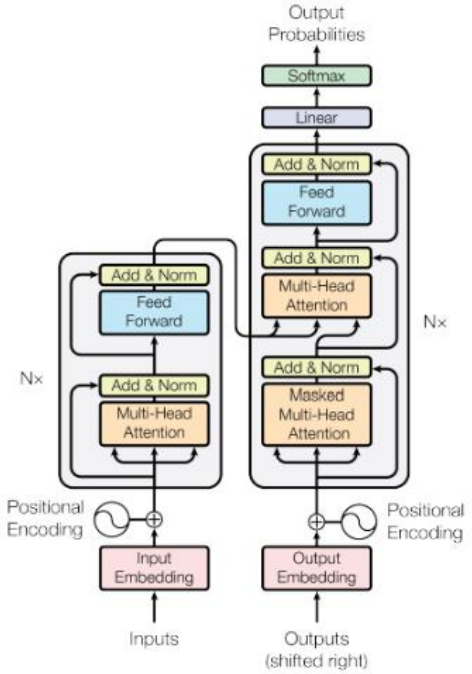

Рисунок 2.7 - Архітектура моделі трансформер [19]

Як видно з рисунку вище, основною частиною моделі є механізму уваги (attention mechanism), який дозволяє моделі приділяти увагу різним частинам вхідного тексту під час обробки. Вхідний текст розбивається на послідовність токенів (слів або символів), які проходять через серію шарів у трансформері.

Кожен шар трансформера складається з двох підшарів: механізму уваги та позиційно-пов'язаних фідбеків (positional feed-forward).

Механізм уваги використовується для знаходження ваги, яка приділяється кожному токену залежно від його відношення до інших токенів у вхідному тексті. Це дозволяє моделі враховувати широкий контекст і здійснювати більш точні прогнози.

Після кожного шару в трансформері використовується функція "резидуального з'єднання" (residual connection), що додає вихід підшару до початкового вхідного сигналу. Це допомагає уникнути проблеми з втратою інформації під час глибокого прямого поширення (feed-forward) [28].

Загальна архітектура трансформера базується на декількох повторюваних шарах, які дозволяють моделі розуміти контекст на різних рівнях абстракції.

2.3.2 Механізм уваги

Як було зазначено вище, механізм уваги є основним елементом, який робити трансформер настільки потужним. Внутрішня структура механізму включає в себе два фундаментальні елементи: масштабований скалярний добуток уваги (scaled dot-product attention) та багатошарова паралельна увага (рис. 2.8).

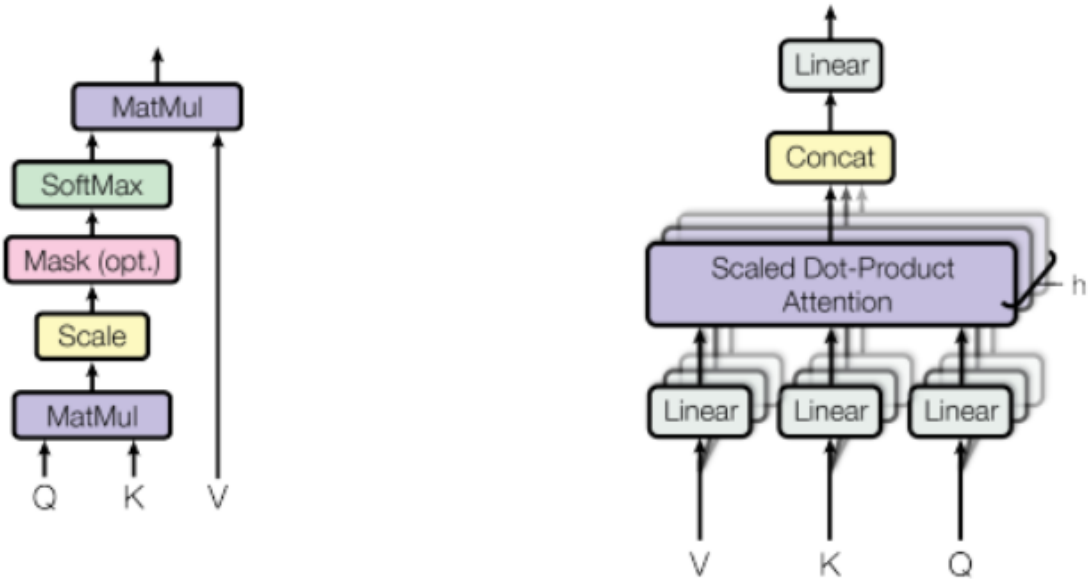

Рисунок 2.8 - Масштабований скалярний добуток уваги (зліва), багатошарова паралельна увага (справа) [19]

Масштабований скалярний добуток уваги, як можна зрозуміти з назви, обчислює скалярний добуток запиту Q з усіма ключами K. До кожного результат добутку застосовується функція softmax. Результати обчислення масштабуються значеннями V.

Для виконання всіх операцій одночасно значення Q,K,V подаються на вхід функції уваги, яка має наступний вигляд:

$$\text{attention}\ (\mathrm{Q}, \mathrm{K}, \mathrm{V}) \ = \ \text{softmax}(\mathrm{QKTdk})\mathrm{V} \qquad (2.2)$$

Коефіцієнт dk було введено для нейтралізації проблеми збільшення

скалярних добутків великих значень, де застосування softmax рузультувало надзвичайно малими значеннями градієнтів, що призвело б до сумнозвісної проблеми зникаючих градієнтів [21].

Свого роду надбудовою над масштабованим скалярним добутком уваги є багатошарова паралельна увага, суть якої полягає у паралельному обчисленню h результатів масштабованого добутку за тими самими вхідними значеннями. Ця операція дозволяє функції уваги не бути залежною від одного простору представлення та бути більш гнучкою.

Функція багатошарової уваги має наступний вигляд:

$$\text{multihead}(Q, K, V) = \text{concat}(\text{head1}, \ldots, \text{headh})W0, \quad (2.3)$$

де кожен *head* реалізує власну функцію уваги [21].

2.3.3 Swin Transformer

Найуспішнішою на даний момент моделью на базі трансформеру в обробці зображень є Swin (Shifted-Window) Transformer. Вона забезпечує ефективну обробку зображень за допомогою механізму уваги.

Основна ідея Swin Transformer полягає в поділі зображення на під-зображення, які називаються патчами. Замість безпосереднього застосування уваги до кожного пікселя, Swin Transformer працює з цими патчами, що дозволяє зменшити обчислювальну складність та пам'ять [28].

Архітектура Swin Transformer має дві ключові складові: Swin Block і Swin Transformer Layer. Swin Block використовує механізм уваги для взаємодії між патчами в різних областях зображення. Крім того, він також містить локальні свертки, які допомагають моделі виявляти локальні особливості [34].

Swin Transformer Layer поєднує кілька Swin Block для отримання більш глибоких репрезентацій зображень (рис. 2.9). Використовуючи шари Swin Transformer, модель може здійснювати рекурсивну обробку зображень на різних масштабах, що дозволяє здійснювати багатомасштабний аналіз [35].

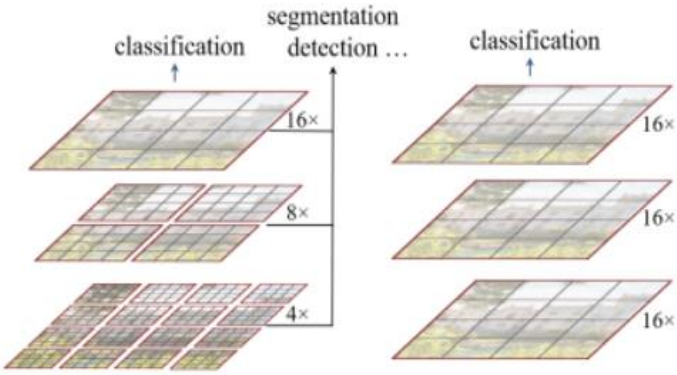

Рисунок 2.9 - Візуалізації роботи Swin Transformer (зліва) та візуального трансформера (справа) [28]

2.4 Трансформери для обробки медичних зображень

Певного успіху останнім часом досягли гібридні моделі на базі трансформерів та згорткових моделей. Такі архітектури зазвичай слідують стилістиці та загальній ідеї повністю згорткових моделей, але заміняють традиційні згорткові шари блоками уваги з архітектури трансформер. Першою помітно ефективною моделлю стала модель UNETR (UNEt TRansformer), відповідно до оригінальної статті цієї моделі вона досягає state of the art результату на BTCV датасеті [22]. На сьогоднішній момент найвижчих результатів досягає модифікація попередньої моделі з використанням вищезгаданого методу Swin, модель SwinUNETR [35] (рис. 2.10).

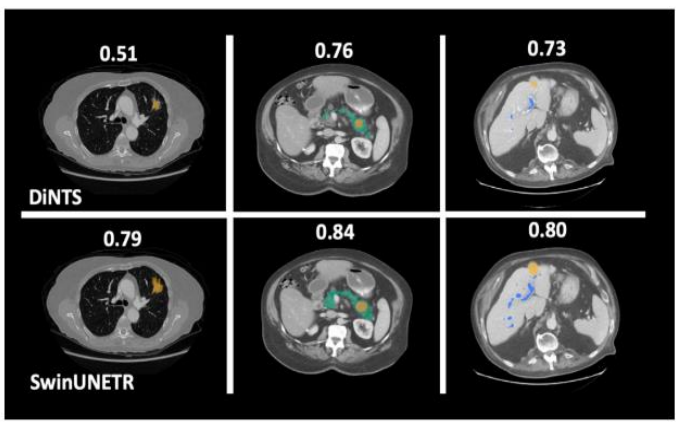

Рисунок 2.10 - Результати сегментації мереж DiNTS та SwinUNETR [36]

Висновки до розділу 2

У даному розділі було розглянуто два типи структур нейронних мереж, традиційних згорткових мереж та трансформерів. Проведений аналіз показав, що обидві ці архітектури є потужними інструментами у сфері обробки зображень.

Згорткові нейронні мережі, зокрема повністю згорткові мережі, виявилися ефективними у вирішенні задач класифікації, локалізації та семантичної сегментації зображень. Їх основною перевагою є здатність автоматичного виявлення різних ознак у зображеннях за допомогою фільтрів згортки та об'єднанням (пулінгом). Використання повністю згорткових мереж, які не мають повністю зв'язаних шарів, дозволяє зменшити кількість параметрів та спростити модель, зберігаючи при цьому високу точність класифікації.

Однак, останнім часом трансформери отримали значну популярність завдяки своїй здатності моделювати глибокі залежності та ефективно вирішувати завдання обробки послідовностей. Використання трансформерів у сфері обробки зображень стало особливо вартим уваги, оскільки вони можуть враховувати контекстуальну інформацію та залежності між різними областями зображень, що дозволяє отримати кращі результати у завданнях, пов'язаних з медичними зображеннями, особливо з просторовими медичними зображеннями, такими як КТ. Особливо варто відзначити swin transformers, які використовують блоки згортки у своїй архітектурі, що забезпечує кращу здатність моделі до розпізнавання об'єктів на зображеннях.

РОЗДІЛ 3
ПРАКТИЧНА ЧАСТИНА

3.1 Вибір бази даних

Складність вирішення будь-якої проблеми штучного інтелекту сильно залежить від якості та кількості вхідних даних, особливо це стосується таких нетривіальних задач як сегментація COVID-19 в просторових зображеннях. Тому дуже важливо приділити велику увагу вибору датасету для тренування нейронної мережі.

Для даної задачі цей набір даних має містити достатню кількість зображень від різних пацієнтів, різних джерел, та з різним ступенем інфікованості задля уникнення перенавчання мережі [4].

В даній роботі буде використовуватися набір даних з [37]. Цей датасет складається з 2729 слайсів КТ знімків грудної клітини від різних пацієнтів, у яких було виявлено SARS-CoV-2, та їх відповідні сегментаційні маски. Набір є попередньо розділений на тренувально, валідаційну, та тестову вибірку у співвідношенні 60%, 20%, 20% відповідно.

Дані КТ були зібрані з семи публічно доступних датасетів, та були приведені до спільного формату для зменшення можливості помилок через несумісність формату даних з різних джерел. Можливими обмеженням цього набору є потенційний шум в анотаціях, які в разі наявності будуть присутні в наборах для навчання, валідації та тестування оскільки вони всі мають зображення зі спільних джерел [37].

3.2 Завантаження та попередня обробка даних

Для завантаження зображень буде використовуватися бібліотека torchvision, яка спеціалізується на роботі з зображеннями. Після завантаження даних, вони

подаються на вхід класу CTDataset, який обробляє дані та приводить їх до стандартного формату. В середині класу дані переводяться у numpy массив, їх розмір приводиться до стандартного 224x224 та формат даних змінюється на float32. Приклад слайсу зображення КТ показано на рис. 3.1.

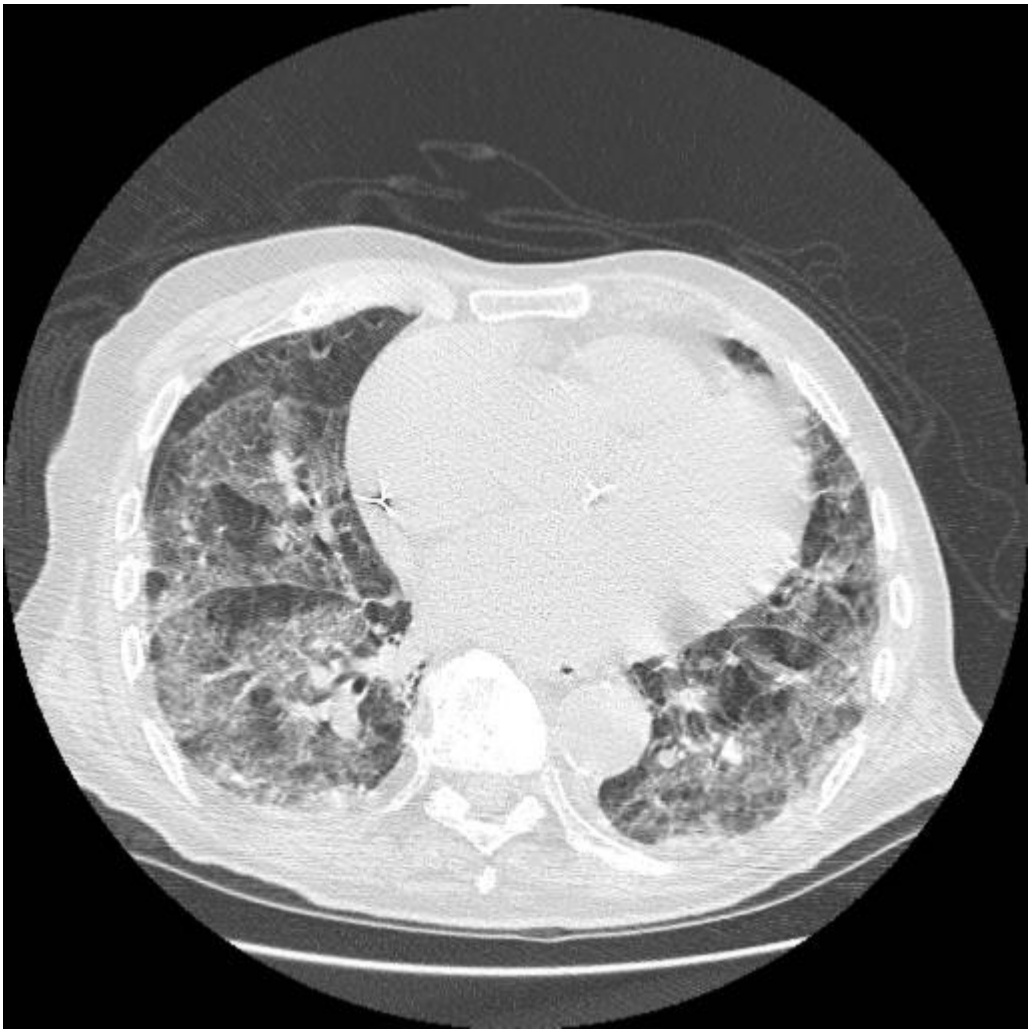

Рисунок 3.1 - Приклад слайсу зображення КТ [38]

На останньому етапі перед подачею даних на вхід мережі вони розбиваються на певне число партій, перемішуються, і на етапі тренування проходять аугментацію.

Аугментація необхідна для зменшення шансу попадання моделі у локальний максимум. В даній роботі використовується шість типів аугментації, які випадковим чином застосовуються до кожного окремого зображення. Зокрема

використовуються: випадкове обертання, випадкове афінне перетворення, еластичне перетворення, випадкове горизонтальне перевертання, випадкове вертикальне перевертання, та гаусівське розмиття.

3.3 Вибір нейронних мереж

Оскільки метою цієї роботи є перевірка ефективності мереж на базі трансформеру відносно традиційних повністю згорткових мереж, вибір мереж, які будуть порівнюватися є надзвичайно важливим. Обрані мережі обох типів мають бути доведено ефективними та бути спеціально розроблені для роботи з медичними зображеннями.

З типу повністю згорткових мереж було обрано три найбільш часто використовувані, які водночас має значні відмінності один від одного і зможуть репрезентативно показати рівень ефективності свого типу мережі.

Порівнюватися будуть наступні моделі:

1. U-Net: базова нейронна мережа для сегментації медичних зображень. Вона складається з енкодера, який зменшує розмір зображення, і декодера, який відновлює його розмір до початкового. U-Net має з'єднання, що дозволяють передавати інформацію з енкодера до декодера, що допомагає злагоджувати локальну та глобальну інформацію під час сегментації [12].

2. Basic U-Net: спрощена версія оригінальної архітектури U-Net. Вона зберігає загальну структуру U-Net з енкодером і декодером, проте має меншу кількість шарів і параметрів, що робить її більш ефективною для невеликих наборів даних або при обмежених обчислювальних ресурсах [9].

3. Flexible U-Net: нейронна мережа, призначена для сегментації тривимірних зображень. Вона розширює архітектуру U-Net, додаючи додаткові шари та модулі, що дозволяють зберігати просторову і контекстуальну інформацію під час сегментації об'єктів на медичних зображеннях [14].

4. UNETR: поєднання архітектури U-Net і трансформерних мереж. Вона використовує трансформерні блоки для обробки інформації з різних масштабних рівнів та використовує енкодер-декодерну структуру U-Net для здійснення сегментації. UNETR дозволяє ефективно обробляти великі зображення та досягати точності сегментації завдяки використанню трансформерних механізмів [22].

5. SwinUNETR: поєднує архітектуру Swin Transformer з енкодер-декодерною структурою U-Net для завдань семантичної сегментації. Вона використовує блоки Swin Transformer для моделювання глобальної контекстуальної інформації та енкодер-декодерну структуру U-Net для збереження локальної і детальної інформації. SwinUNETR поєднує переваги обох архітектур для досягнення високої точності сегментації зображень [35].

Реалізація всіх моделей розроблена пакетом роботи з моделями обробки медичних даних MONAI [39].

3.4 Оптимізація та налаштування моделі

3.4.1 Вибір параметрів для налаштування

Для отримання якомога точної моделі, вона спочатку має бути налаштована та оптимізована під конкретну задачу та тип даних. В даному випадку гіперпараметрами мережі, які будуть налаштовуватися є величини, що показані у табл. 3.1. Окрім налаштування числових значень необхідно також вибрати найбільш оптимальні функції, які використовуються для тренування мережі. Елементи які будуть налаштовуватися та функції які будуть перевірятися представлені у табл. 3.2. Варто зазначити що ці гіперпараметри є аналогічними для всіх моделей, які порівнюються в цій роботі.

Додатково є сенс налаштувати параметри аугментації зображень. Список параметрів, їх значення за замовчуванням, та діапазон можливих значень наведені в табл. 3.3.

Таблиця 3.1

Список числових гіперепараметрів

Гіперпараметр	Значення за замовчуванням	Діапазон можливих значень
Розмір партії	1	1-32
Крок навчання	0.01	0.0001-0.1
Коефіцієнт зниження ваги	0.1	0.001-0.9
Коефіцієнт регуляризації dropout	0.5	0.3-0.7
Параметр num_workers в класі DataLoader	4	1-4
Вага функції похибки DiceLoss	8	1-10

Таблиця 3.2

Список нечислових гіперепараметрів

Назва елементу	Варіант за замовчуванням	Альтернативні варіанти
Функція похибки	DiceLoss	BCELoss, DiceCELoss
Оптимізатор	Adam	Adagrad, RMSProp
Планувальник кроку навчання	-	ReduceLROnPlateau, LinearLR, ExponentialLR, PolynomialLR
Функція активації	Sigmoid	ReLU, Leaky ReLU

Таблиця 3.3

Список гіперепараметрів аугментації

Назва методу аугментації	Назва параметру	Значення за замовчуванням	Діапазон можливих значень
Поворот зобрадення	Кут нахилу	180	90-270
Афінне перетворення	Кут повороту	90	0-180
Гаусівське розмиття	Стандартне відхилення	0.5	0.1-2

3.4.2 Представлення результатів оптимізації

Згідно із зазначеними вище параметрами була проведена оптимізації гіперпараметрів. Для кожного із числових параметрів на кожній із ітерацій

налаштування було випадковим чином обрано значення із вказаного діапазону та проведено процес тренування протягом однієї епохи. Нижче (табл. 3.4 – табл. 3.6) представлені значення гіперпараметрів з найкращими результатами.

Таблиця 3.4

Результат оптимізації числових гіперпарметрів

Назва параметру	Оптимальне значення параметру	Значення IOU	Dice коефіцієнт
Розмір партії	32	0.173	0.265
Крок навчання	0.001	0.179	0.24
Коефіцієнт зниження ваги	0.004	0.174	0.257
Коефіцієнт регуляризації dropout	0.5	0.181	0.289
Параметр num_workers в класі DataLoader	4	0.173	0.269
Вага функції похибки DiceLoss	8	0.174	0.251

Таблиця 3.5

Результат вибору оптимального елементу тренування

Назва елементу	Оптимальний варіант	Значення IOU	Dice коефіцієнт
Функція похибки	DiceLoss	0.178	0.256
Оптимізатор	Adam	0.177	0.242
Планувальник кроку навчання	ReduceLROnPlateau	0.18	0.232
Функція активації	Sigmoid	0.177	0.243

Таблиця 3.6

Результат вибору оптимальних параметрів аугментації

Назва методу аугментації	Назва параметру	Оптимальне значення	Значення IOU	Dice коефіцієнт
Поворот зображення	Кут нахилу	180	0.173	0.243
Афінне перетворення	Кут повороту	90	0.174	0.25
Гаусівське розмиття	Стандартне відхилення	0.5	0.173	0.245

3.5 Вибір метрик оцінки точності моделі

Оскільки важливою частиною цієї роботи є порівняння точності моделей, надзвичайно важливим елементом є вибір інформативних та релевантних метрик. Для цієї роботи було обрано шість метрик точності моделі. Зокрема було обрано:

1. Recall: Ця метрика вимірює, який відсоток позитивних зразків був виявлений правильно. Вона розраховується як відношення позитивних зразків до загальної кількості позитивних зразків [4]. Recall обчислюється за наступною формулою:

$$Recall = \frac{TP}{TP + FN} \tag{3.1}$$

2. Precision: Ця метрика вимірює, який відсоток з виявлених моделлю позитивних зразків є справді позитивними. Вона розраховується як відношення кількості правильно виявлених позитивних зразків до загальної кількості зразків, виявлених як позитивні. Precision є важливим, коли важлива мінімізація невірно-позитивних результатів. Precision обчислюється за наступною формулою:

$$Precision = \frac{TP}{TP + FP} \tag{3.2}$$

3. Intersection over Union: Ця метрика вимірює схожість між прогнозованою сегментацією і правильною сегментацією, використовуючи відношення площ перетину і об'єднання цих областей. Вона розраховується як відношення площі перетину прогнозованої сегментації та правильної сегментації до площі їх об'єднання. Більше значення IOU вказує на більш точну сегментацію. IOU обчислюється за наступною формулою:

$$IoU = \frac{(Object \cap Dectected\ box)}{(Object \cup Detected\ box)} \tag{3.3}$$

4. Dice (коефіцієнт Соренсена): Ця метрика, яка використовується в якості функції похибки, також вимірює схожість між прогнозованою сегментацією і правильною сегментацією. Вона розраховується як подвійне значення площі перетину прогнозованої сегментації та правильної сегментації, поділене на суму їх площ. Dice обчислюється за наступною формулою:

$$Dice = \frac{2 \cdot TP}{(TP + FP) + (TP + FN)} \tag{3.4}$$

5. F1-показник: Це гармонічне середнє між точністю і повнотою (precision і recall). Він використовується для оцінки балансу між цими двома метриками. F1-показник досягає свого максимального значення в 1 (найкращий результат) і свого мінімального значення в 0 (найгірший результат), і він є хорошим показником загальної ефективності моделі. F1 обчислюється за наступною формулою:

$$F1 = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{3.5}$$

6. GDL (узагальнений коефіцієнт Соренсена): Ця метрика оцінює відхилення між прогнозованими значеннями моделі і правильними мітками. GDL обчислюється за наступною формулою:

$$GDL = 1 - 2\frac{\sum_{l=1}^{2} w_l \sum_n r_{ln} p_{ln}}{\sum_{l=1}^{2} w_l \sum_n r_{ln} + p_{ln}} \tag{3.6}$$

3.6 Тренування та валідація моделі

Наступним кроком після завантаження та обробки моделей є тренування моделі. Для проведення процесів тренування, валідації та тестування було створено класс Evaluation, який приймає на вхід об’єкт моделі та список з трьох вибірок. Кожен з етапів інкапсульовано в окремий метод класу. Процес тренування проходить у вигляді двох вкладених циклів. Зовнішній цикл є циклом епох, а зовнішній циклом партій набору. На кожному з тренувальних кроків обчислюються всі необхідні метрики та разом з поточними вихідними даними моделі записуються у TensorBoard. В кінці кожної епохи також обчислюється середнє значення кожної метрики. Валідаційний та тестовий процес проходять подібним чином.

3.6.1 Опис структури класу Evaluation

Клас Evaluation виконує процеси тренування, валідації та тестування моделі на основі переданих йому аргументів. Основна структура класу включає наступні методи:

1. __init__(self, model, loader_dict):

Цей метод виконує початкову ініціалізацію об'єкта класу Evaluation. Він приймає наступні аргументи:

- model: об'єкт моделі, який буде тренуватись та оцінюватись.
- loader_dict: словник, що містить об’єкти DataLoader трьох вибірок даних - тренувальної, валідаційної і тестувальної.

Крім ініціалізації класу та отримання аргументів класу, метод __init__ також визначає три атрибути:

- loss_fn: функція похибки, яка використовується для обчислення втрат під час тренування.
- optimizer: оптимізатор, який використовується для оновлення параметрів моделі під час тренування.
- scheduler: планувальник гіперпараметру кроку тренування, який змінює його при плато зміни функції похибки.

2. train(self, epochs=1, batch_size=1):

Цей метод виконує процес тренування моделі. Він проганяє дані через модель, обчислює втрати та оновлює параметри моделі за допомогою оптимізатора. Також він вимірює та записує метрики точності моделі на кожному кроці тренування. Аргументами цього методу є:

- epochs: кількість епох тренування моделі, значення за замовчуванням
- batch_size: розмір партій, на які ділиться вибірка, необхідно для запису метрик точності в TensorBoard, значення за замовчуванням 1.

3. evaluate(self, mode='val, batch_size=1):

Цей метод виконує процес валідації або тестування моделі. Він приймає вхідний загрузчик даних і проганяє дані через модель, обчислюючи вихідні значення моделі. Далі він викликає метод compute_metrics для обчислення метрик точності моделі на основі отриманих вихідних значень та реальних даних. Аргументами цього методу є:

- mode: режим оцінки моделі, який буде проводитись, значення за замовчуванням 'val'.
- batch_size: розмір партій, на які ділиться вибірка, необхідно для запису метрик точності в TensorBoard, значення за замовчуванням 1.

4. compute_metrics(self, y_pred, y):

Цей метод обчислює метрики точності моделі на основі переданих йому прогнозованих (y_pred) та реальних (y) даних. Він використовує вбудовані функції або власні реалізації метрик (наприклад, точність, чутливість, F1-показник і т. д.), щоб оцінити ефективність моделі.

3.6.2 Опис процесів тренування та оцінки

Як було сказано вище, клас Evaluate проводить процеси тренування, валідації та тестування, датальний опис цих процесів описано нижче.

Тренування моделі:

- Переведення моделі в режим тренування командою model.train() для запобігання відключення регуляризації dropout та інших функцій, залежних від режиму моделі.
- Перехід в цикл ітерації епох тренування.
- Створення об’єктів CumulativeAverage для запису середніх значень метрик на кожній епосі.
- Перехід в цикл ітерації партій тренувального набору даних.
- Перенесення даних на GPU або CPU командою .to(device)
- Анулювання попередніх обчислень градієнту командою optimizer.zero_grad()
- Прогін даних через модель та обчислення функції похибки.
- Виконання зворотного проходу (backward propagation) для обчислення градієнтів
- Оновлення значень параметрів моделі
- Збереження проміжних значень метрик
- Вихід з другого циклу
- Збереження середніх значень метрик
- Вихід з першого циклу

Валідація та тестування моделі:

- Переведення моделі в режим оцінки командою model.eval() для відключення регуляризації dropout та інших функцій, залежних від режиму моделі.
- Створення об’єктів CumulativeAverage для запису середніх значень метрик
- Відключення обчислень градієнтів переходячи в контекстну область командою with torch.no_grad()
- Перехід в цикл ітерації партій тренувального набору даних.
- Перенесення даних на GPU або CPU командою .to(device)
- Прогін даних через модель та обчислення функції похибки.
- Збереження проміжних значень метрик

- Вихід з циклу
- Збереження середніх значень метрик
- Вивід на екран значення метрик

3.6.3 Механізм запису та логування метрик

Логування та збереження метрик та станів під час процесу тренування моделі є важливими кроками для відстеження прогресу моделі та подальшого аналізу результатів. В даній роботі використовуються наступні аспекти:

1. Логування метрик:

- Використання логерів: За допомогою бібліотеки TensorBoard та logging, зберігаються проміжні та загальні значення метрик, які потім допомагають описати загальний стан рівня точності моделі. Зокрема TensorBoard дозволяє візуалізувати зміну метрик під час тренування ти зберегти проміжні приклади роботи моделі для більш наглядного аналізу точності моделі. Ці бібліотеки дозволяють легко створювати та оновлювати графіки метрик, які можна відслідковувати та порівнювати.

2. Збереження станів моделі:

- Збереження ваг моделі: Під час тренування можна зберігати ваги моделі після кожної епохи або для кращих результатів. Це досягнуто за допомогою методів torch.save() та model.state_dict() для збереження та завантаження ваг моделі.
- Використання контрольної точки: Для відновлення тренування моделі з попереднього стану в разі переривання або помилки можна створити контрольну точку, в якій зберігається стан моделі та оптимізатора. Це дозволяє продовжити тренування з цієї точки, а не з початку.

3. Збереження історії метрик та втрат:

- Використання структур даних: Створення списків або інших структур даних для збереження історії метрик та втрат на кожному кроці тренування. Після закінчення тренування можна використати ці дані для подальшого аналізу та візуалізації.

- Збереження у файл: можна зберегти метрики та втрати у текстовий або CSV-файл під час тренування для подальшого використання або аналізу.

Збереження метрик та станів моделі дозволяє зручно відстежувати прогрес моделі, порівнювати різні моделі та параметри тренування, а також легко повторювати та відновлювати тренування у разі потреби.

3.6.4 Налаштування та робота з TensorBoard

TensorBoard є потужним інструментом для візуалізації та аналізу результатів тренування моделей у фреймворку PyTorch. Він надає зручний інтерфейс для відстеження метрик, візуалізації моделей та аналізу експериментів. Етапи налаштування TensorBoard для даної роботи описані:

1. Інтеграція з PyTorch: TensorBoard легко інтегрується з фреймворком PyTorch. Для підключення TensorBoard в проект достатньо встановити torch.utils.tensorboard, щоб логувати дані прямо з коду PyTorch.

2. Візуалізація метрик: TensorBoard дозволяє логувати та відстежувати різні метрики під час тренування моделі, такі як точність, втрати, показники часу тощо. З його допомогою можна створювати графіки, діаграми та таблиці, щоб візуалізувати зміни цих метрик протягом тренування. Для цього достатньо створити об’єкт SummaryWriter та за допомогою його методів .add_scalar() та .add_images() записувати скалярні значення та зображення в TensorBoard.

3. Візуалізація графу моделі: TensorBoard дозволяє побудувати візуальне представлення графу моделі. Це дає змогу бачити структуру моделі, переглядати розмірність вхідних та вихідних шарів та більш детально аналізувати архітектуру. Для цього необхідно викликати .add_graph().

4. Візуалізація параметрів моделі: TensorBoard дозволяє відстежувати зміни параметрів моделі під час тренування. Він дозволяє відображати гістограми розподілу параметрів, порівнювати значення параметрів між різними епохами або експериментами. Для налаштування відображення гастограм достатньо викликати .add_histogram().

5. Візуалізація зображень та даних: TensorBoard дозволяє відображати зображення, відео та інші типи даних. Можна відстежувати вхідні дані, відтворювати прогнозовані результати, порівнювати реальні та прогнозовані зображення та багато іншого. Для запису зображень можна викликати .add_images().

6. Порівняння експериментів: TensorBoard дозволяє зберігати результати різних експериментів та порівнювати їх між собою. Можна відстежувати та порівнювати метрики, візуалізації та статистику між різними моделями, параметрами тренування або гіперпараметрами. Приклад такого порівняння зображено на рис 3.2.

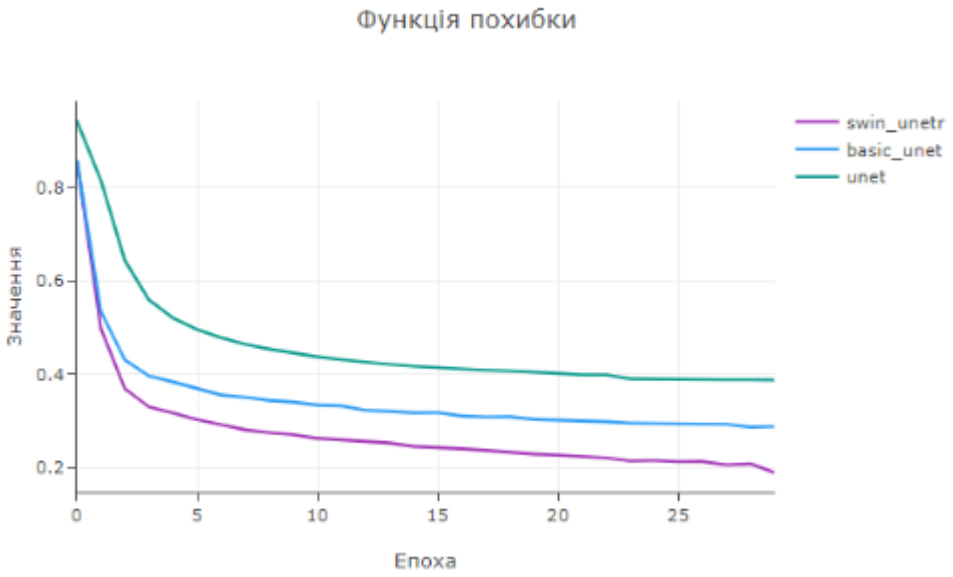

Рисунок 3.2 - Приклад порівняння результатів різних експериментів

Використання TensorBoard значно полегшує процес аналізу результатів тренування моделі. Дозволяє швидко відстежувати прогрес, виявляти проблеми, візуалізувати результати та зробити висновки щодо покращення моделі. Загалом, TensorBoard є потужним інструментом для розуміння та оптимізації моделі на основі візуальної зворотного зв'язку.

3.7. Оцінка та порівняння двох типів моделей

Провівши тренування моделей було отримано результати обраних метрик у числовому та графічному форматах. Результати точності кожної з моделей та загальне порівняння наведено нижче.

3.7.1 Модель U-Net

Графічне зображення тренувального процесу моделі U-Net наведено на рис. 3.3 – рис. 3.8 та на табл. 3.7:

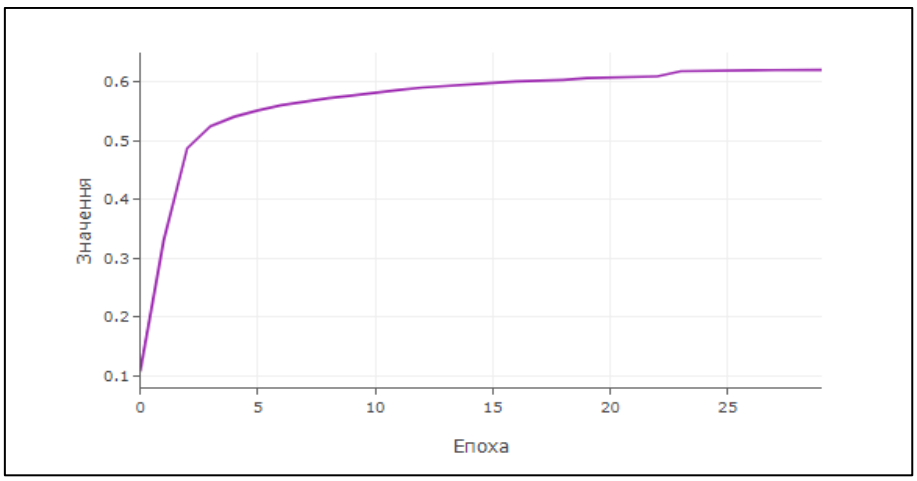

Рисунок 3.3 - Графік зміни метрики узагальнений коефіцієнт Соренсена (модель U-Net)

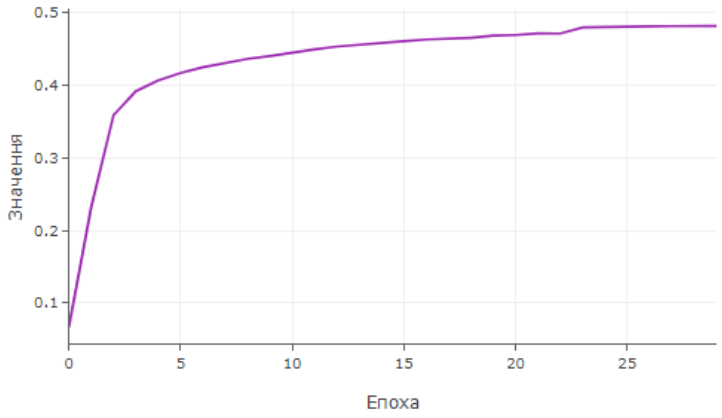

Рисунок 3.4 - Графік зміни метрики IOU (модель U-Net)

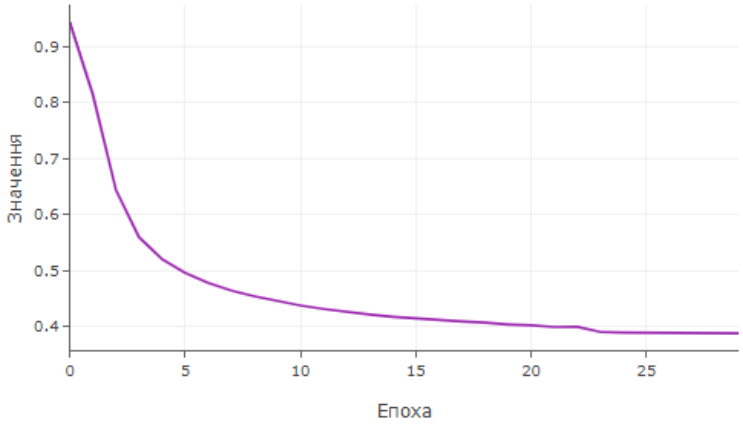

Рисунок 3.5 - Графік зміни функції похибки DiceLoss (модель U-Net)

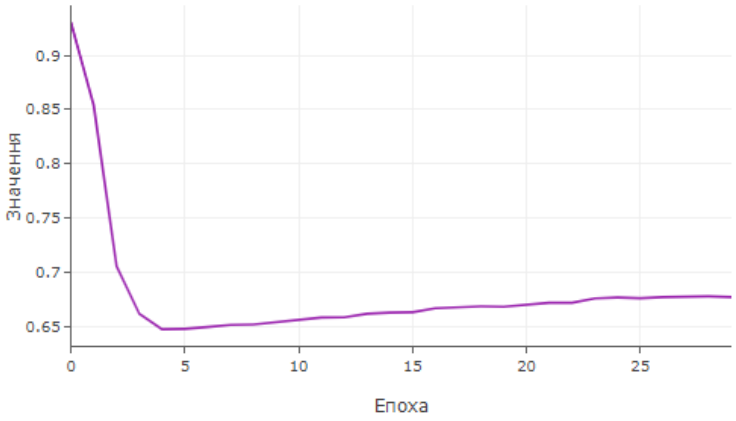

Рисунок 3.6 - Графік зміни метрики Recall (модель U-Net)

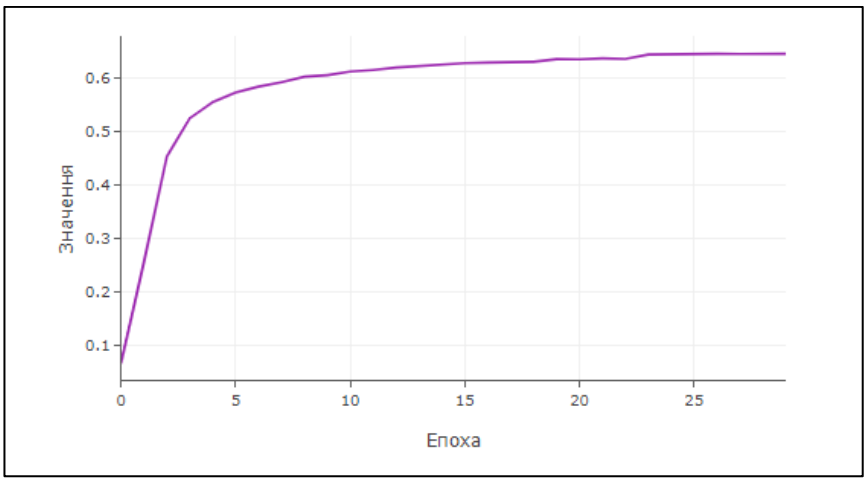

Рисунок 3.7 - Графік зміни метрики Precision (модель U-Net)

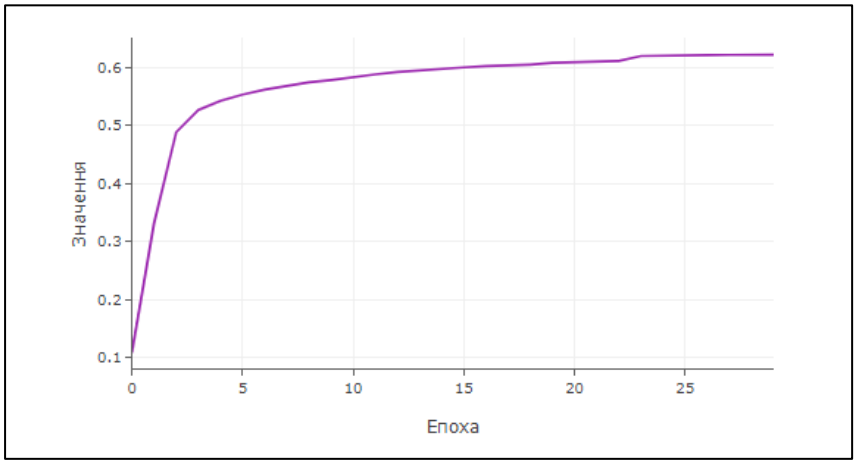

Рисунок 3.8 - Графік зміни F1 коефіцієнту (модель U-Net)

Таблиця 3.7

Результати метрик точності мережі та заміру часу виконання (модель U-Net)

Вибірка	Recall	Precision	F1	IOU	GDL	Dice	Час навчання (сек.)
Навчальна	0.64	0.63	0.56	0.45	0.59	0.43	180
Валідаційна	0.64	0.64	0.56	0.45	0.59	0.42	
Тестова	0.64	0.63	0.55	0.46	0.59	0.42	

3.7.2 Модель Basic U-Net

Графічне зображення тренувального процесу моделі Basic U-Net показане на рис. 3.9 – рис. 3.14 та на табл. 3.8.

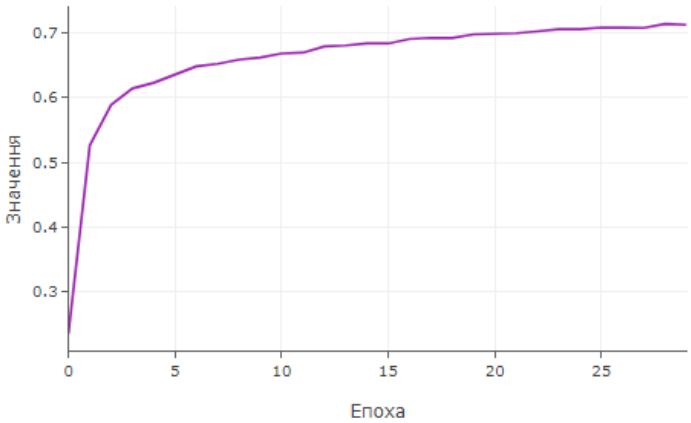

Рисунок 3.9 - Графік зміни метрики узагальнений коефіцієнт Соренсена (модель Basic U-Net)

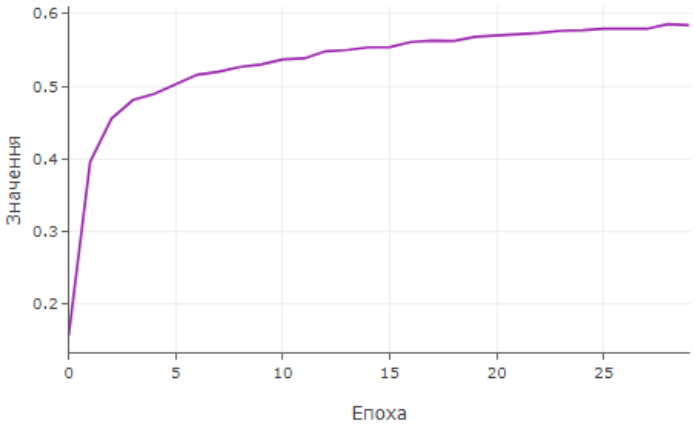

Рисунок 3.10 - Графік зміни метрики IOU (модель Basic U-Net)

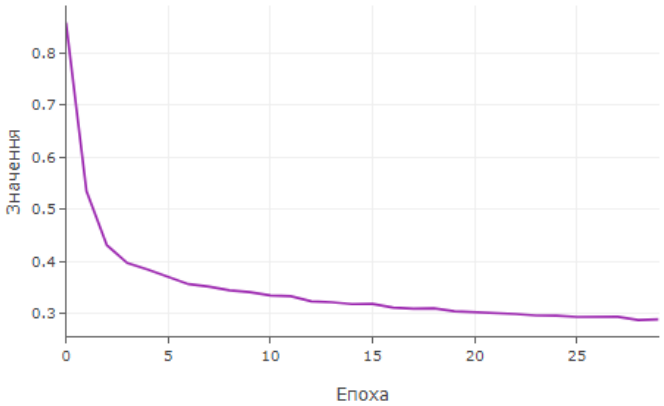

Рисунок 3.11 - Графік зміни функції похибки DiceLoss (модель Basic U-Net)

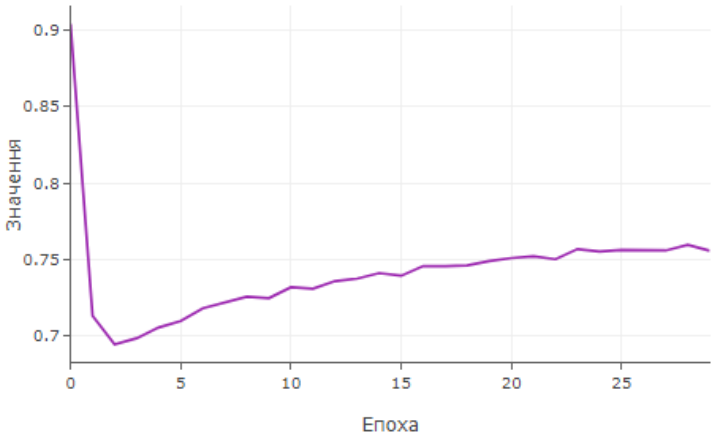

Рисунок 3.12 - Графік зміни метрики Recall (модель Basic U-Net)

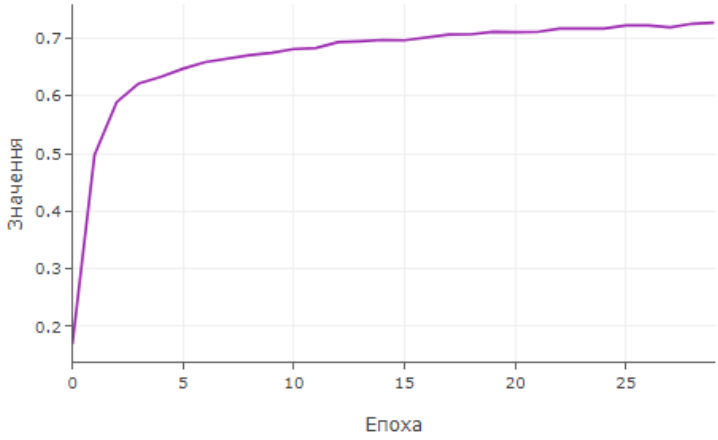

Рисунок 3.13 - Графік зміни метрики Precision (модель Basic U-Net)

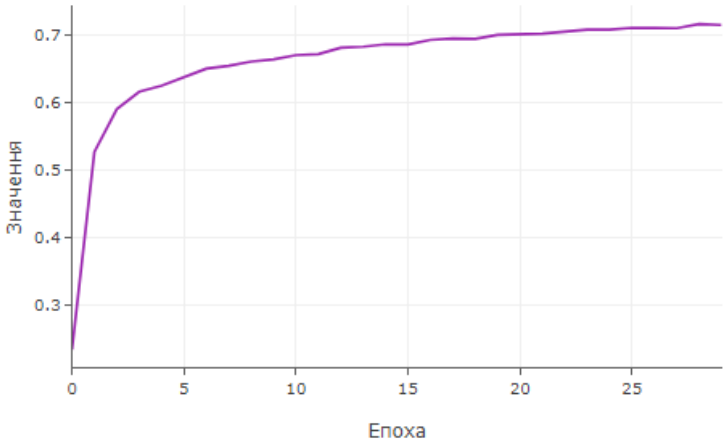

Рисунок 3.14 - Графік зміни F1 коефіцієнту (модель Basic U-Net)

Таблиця 3.8

Результати метрик точності мережі та заміру часу виконання (модель Basic U-Net)

Вибірка	Recall	Precision	F1	IOU	GDL	Dice	Час навчання (сек.)
Навчальна	0.83	0.68	0.72	0.59	0.72	0.28	1980
Валідаційна	0.81	0.69	0.72	0.59	0.72	0.29	
Тестова	0.81	0.69	0.72	0.58	0.72	0.29	

3.7.3 Модель Flexible U-Net

Графічне зображення тренувального процесу моделі Flexible U-Net наведено на рис. 3.15 – рис. 3.20 та на табл. 3.9.

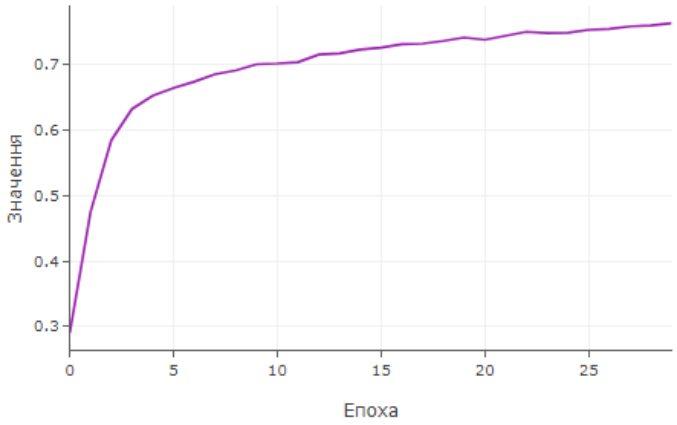

Рисунок 3.15 - Графік зміни метрики узагальнений коефіцієнт Соренсена (модель Flexible U-Net)

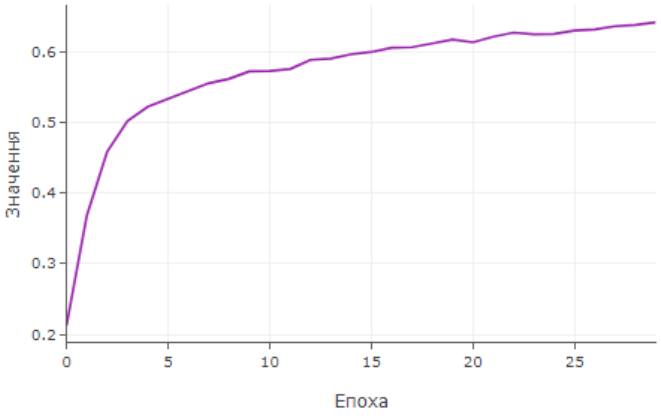

Рисунок 3.16 - Графік зміни метрики IOU (модель Flexible U-Net)

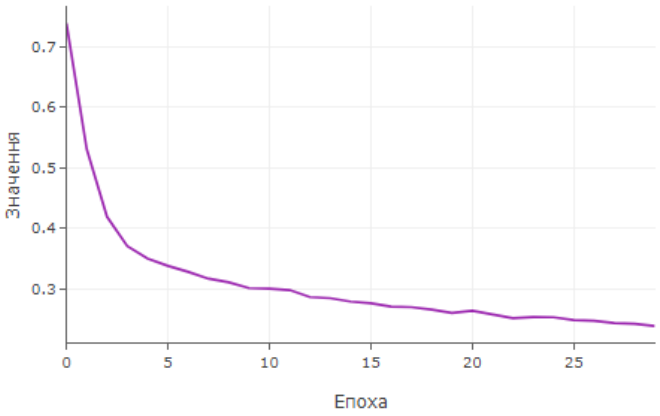

Рисунок 3.17 - Графік зміни функції похибки DiceLoss (модель Flexible U-Net)

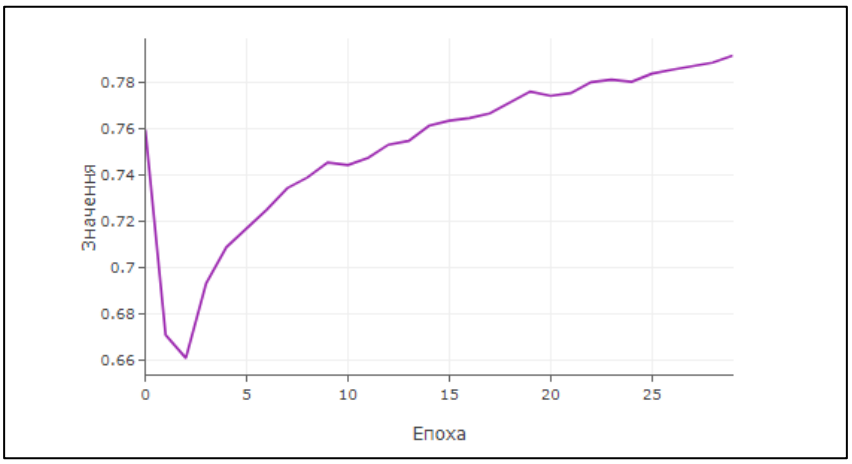

Рисунок 3.18 - Графік зміни метрики Recall (модель Flexible U-Net)

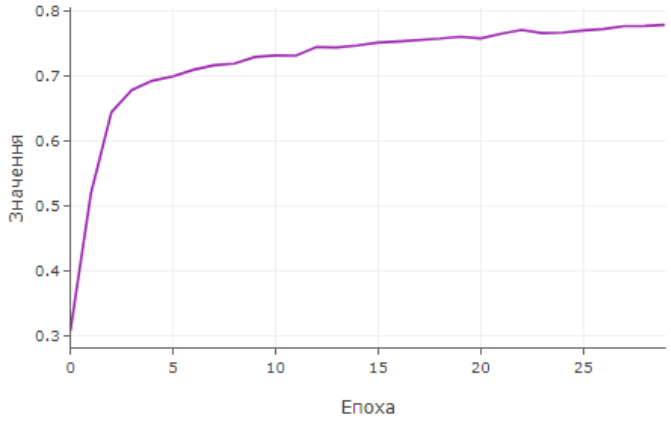

Рисунок 3.19 - Графік зміни метрики Precision (модель Flexible U-Net)

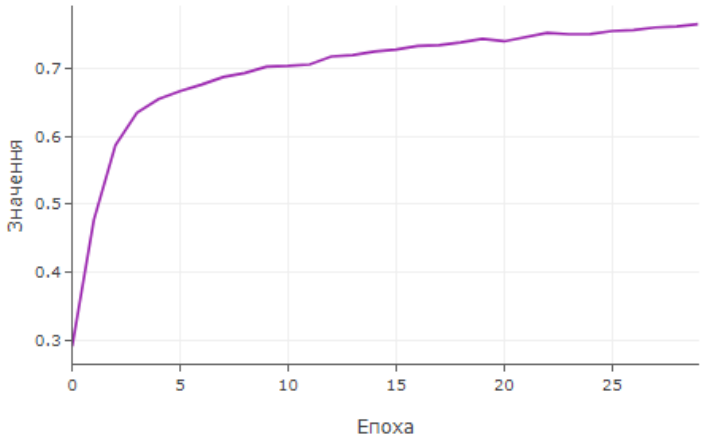

Рисунок 3.20 - Графік зміни F1 коефіцієнту (модель Flexible U-Net)

Таблиця 3.9

Результати метрик точності мережі та заміру часу виконання (модель Flexible U-Net)

Вибірка	Recall	Precision	F1	IOU	GDL	Dice	Час навчання (сек.)
Навчальна	0.75	0.79	0.74	0.62	0.74	0.25	2580
Валідаційна	0.75	0.78	0.73	0.6	0.73	0.27	
Тестова	0.74	0.77	0.72	0.59	0.72	0.28	

3.7.4 Модель UNETR

Графічне зображення тренувального процесу моделі UNETR наведено на рис. 3.21 – рис. 3.26 та на табл. 3.10.

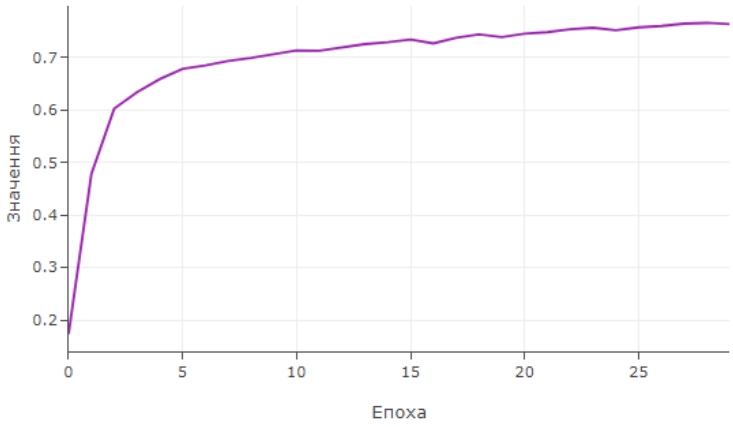

Рисунок 3.21 - Графік зміни метрики узагальнений коефіцієнт Соренсена (модель UNETR)

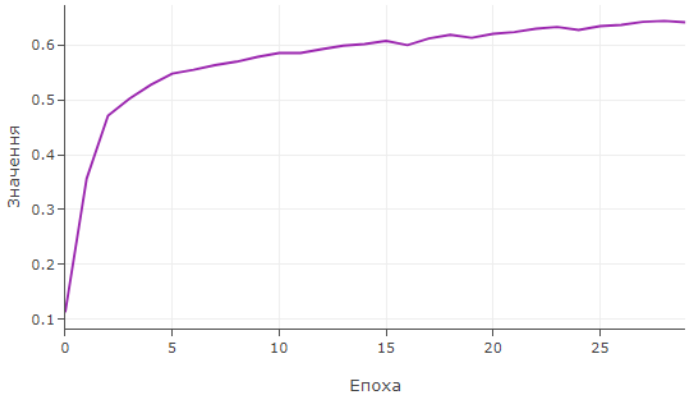

Рисунок 3.22 - Графік зміни метрики IOU (модель UNETR)

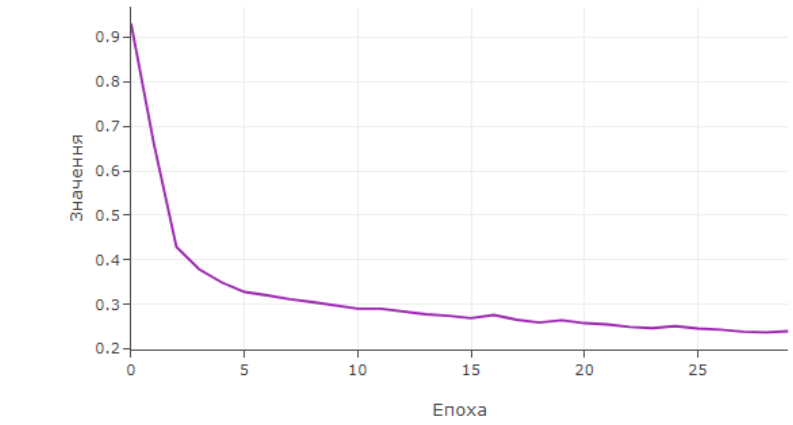

Рисунок 3.23 - Графік зміни функції похибки DiceLoss (модель UNETR)

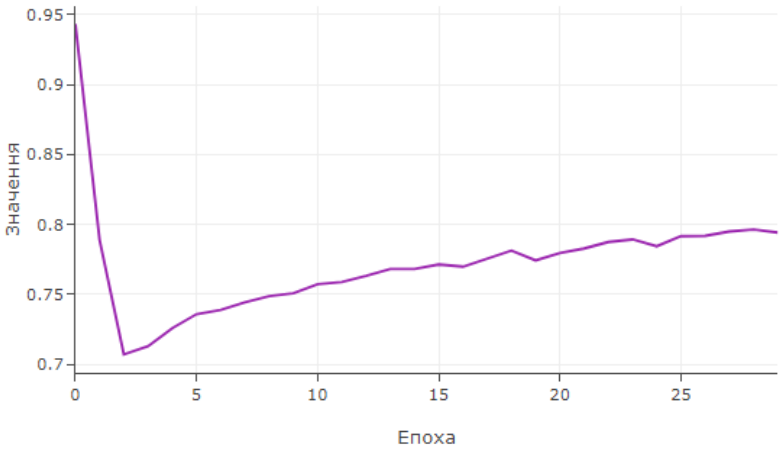

Рисунок 3.24 - Графік зміни метрики Recall (модель UNETR)

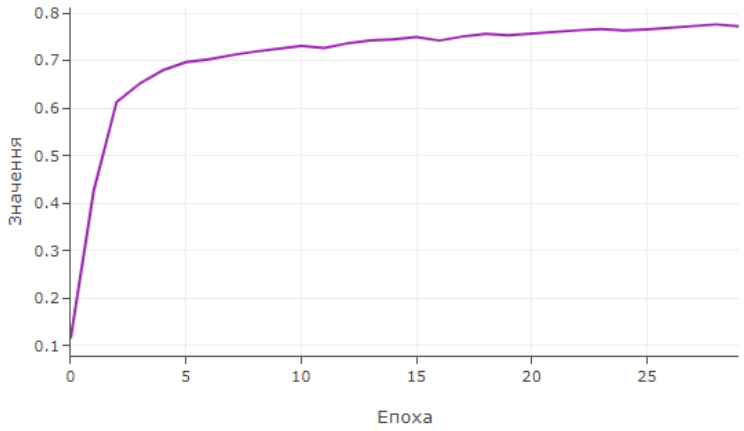

Рисунок 3.25 - Графік зміни метрики Precision (модель UNETR)

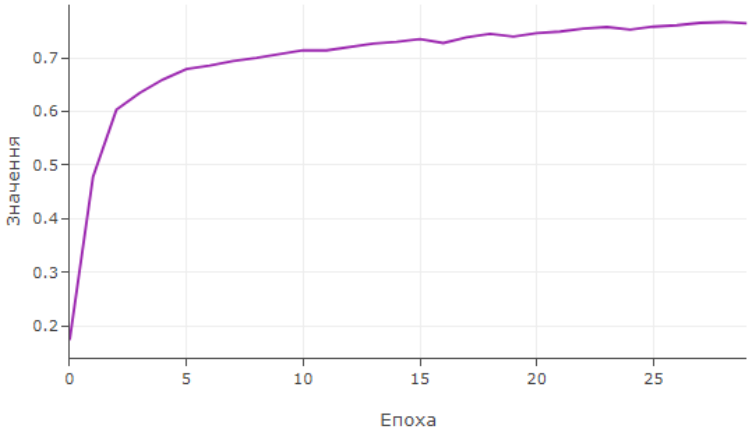

Рисунок 3.26 - Графік зміни F1 коефіцієнту (модель UNETR)

Таблиця 3.10

Результати метрик точності мережі та заміру часу виконання (модель UNETR)

Вибірка	Recall	Precision	F1	IOU	GDL	Dice	Час навчання (сек.)
Навчальна	0.82	0.77	0.78	0.66	0.78	0.22	3200
Валідаційна	0.79	0.76	0.76	0.63	0.76	0.25	
Тестова	0.79	0.76	0.76	0.64	0.76	0.24	

3.7.5 Модель SwinUNETR

Графічне зображення тренувального процесу моделі SwinUNETR наведено на рис. 3.27 – рис. 3.32 та на табл. 3.11.

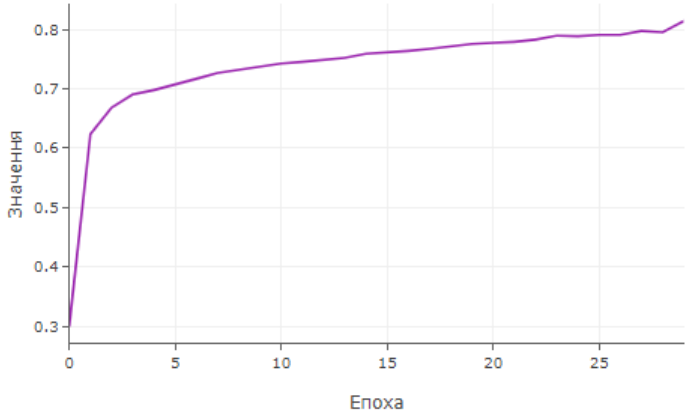

Рисунок 3.27 - Графік зміни метрики узагальнений коефіцієнт Соренсена (модель SwinUNETR)

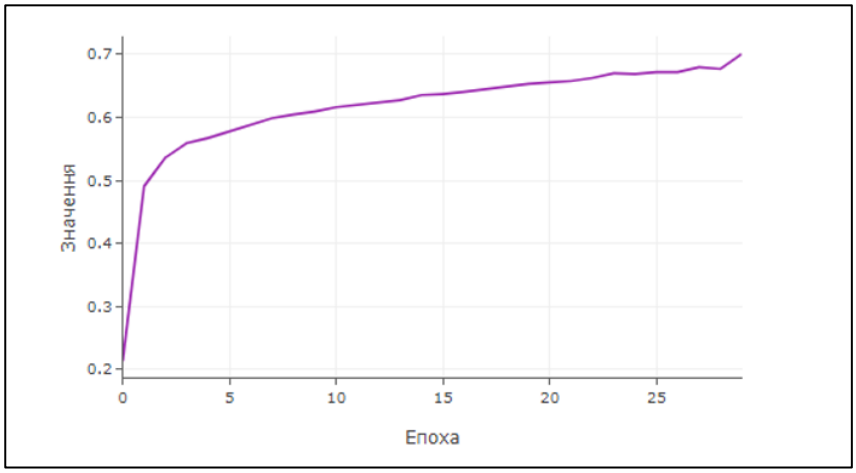

Рисунок 3.28 - Графік зміни метрики IOU (модель SwinUNETR)

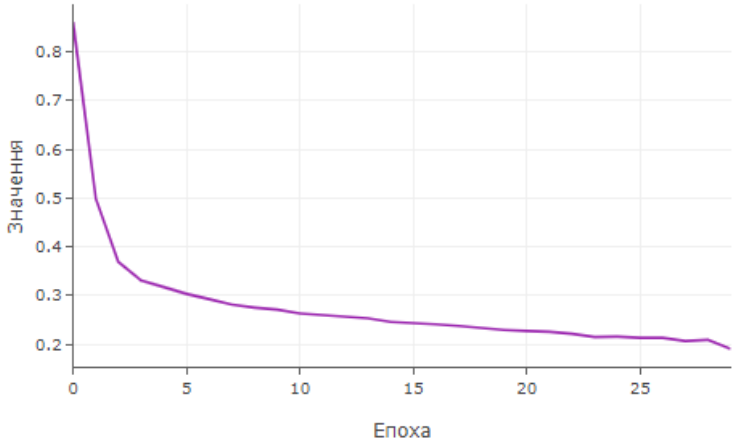

Рисунок 3.29 - Графік зміни функції похибки DiceLoss (модель SwinUNETR)

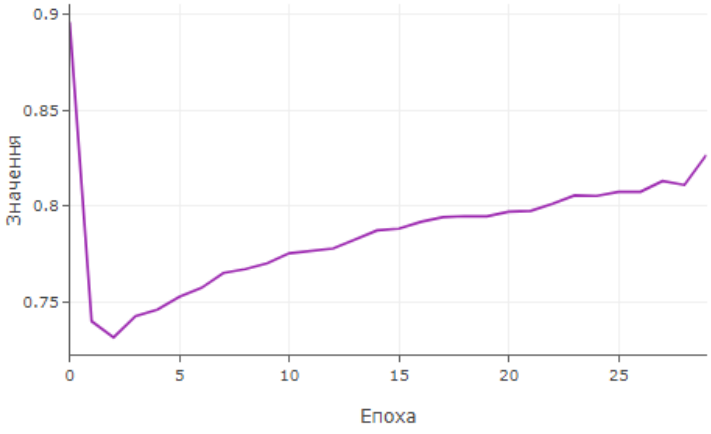

Рисунок 3.30 - Графік зміни метрики Recall (модель SwinUNETR)

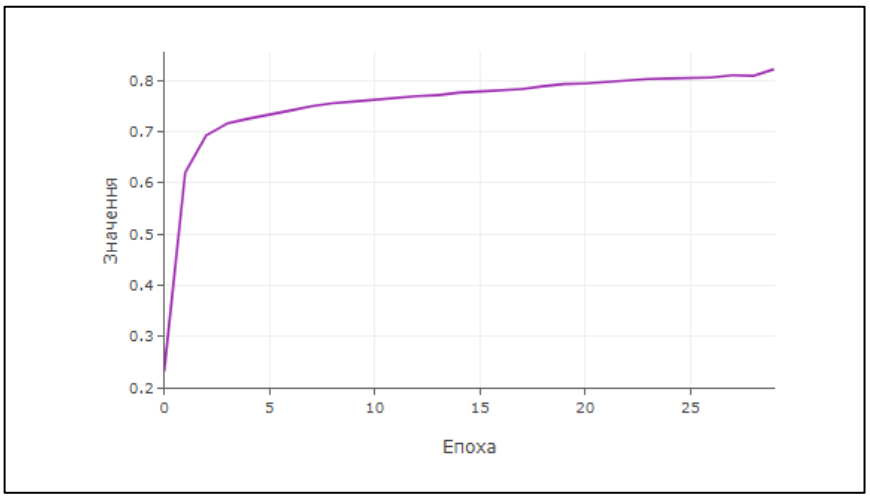

Рисунок 3.31 - Графік зміни метрики Precision (модель SwinUNETR)

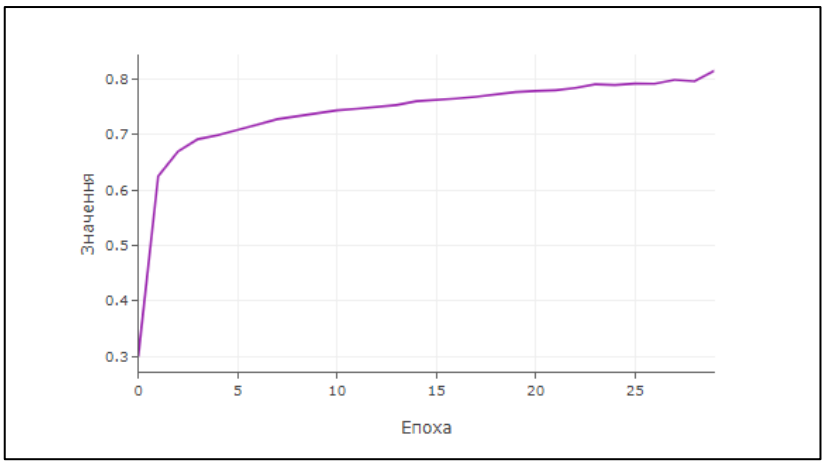

Рисунок 3.32 - Графік зміни F1 коефіцієнту (модель SwinUNETR)

Таблиця. 3.11

Результати метрик точності мережі та заміру часу виконання (модель SwinUNETR)

Вибірка	Recall	Precision	F1	IOU	GDL	Dice	Час навчання (сек.)
Навчальна	0.84	0.8	0.8	0.7	0.81	0.19	3600
Валідаційна	0.81	0.78	0.77	0.65	0.77	0.23	
Тестова	0.81	0.78	0.78	0.66	0.77	0.23	

3.7.6 Порівняння точності двох типів мереж

Отже, натренувавши п’ять моделей, та зафіксувавши їх метрики ефективності можна сформувати таблицю порівняння повністю згорткових мереж

та мереж на базі трансформеру. В табл. 3.12 та на рис. 3.33 представлено результати тестової вибірки моделей тренованих протягом 30 епох з однаковими параметрами:

Таблиця 3.12

Порівняння згорткових мереж та мереж на базі трансформеру

Назва моделі	Recall	Precision	F1	IOU	GDL	Dice	Час навчання (сек.)
U-Net	0.64	0.63	0.55	0.46	0.59	0.42	180
Basic U-Net	0.81	0.69	0.72	0.58	0.72	0.29	1980
Flexible U-Net	0.74	0.77	0.72	0.59	0.72	0.28	2580
UNETR	0.79	0.76	0.76	0.64	0.76	0.24	3200
SwinUNETR	0.81	0.78	0.78	0.66	0.77	0.23	3600

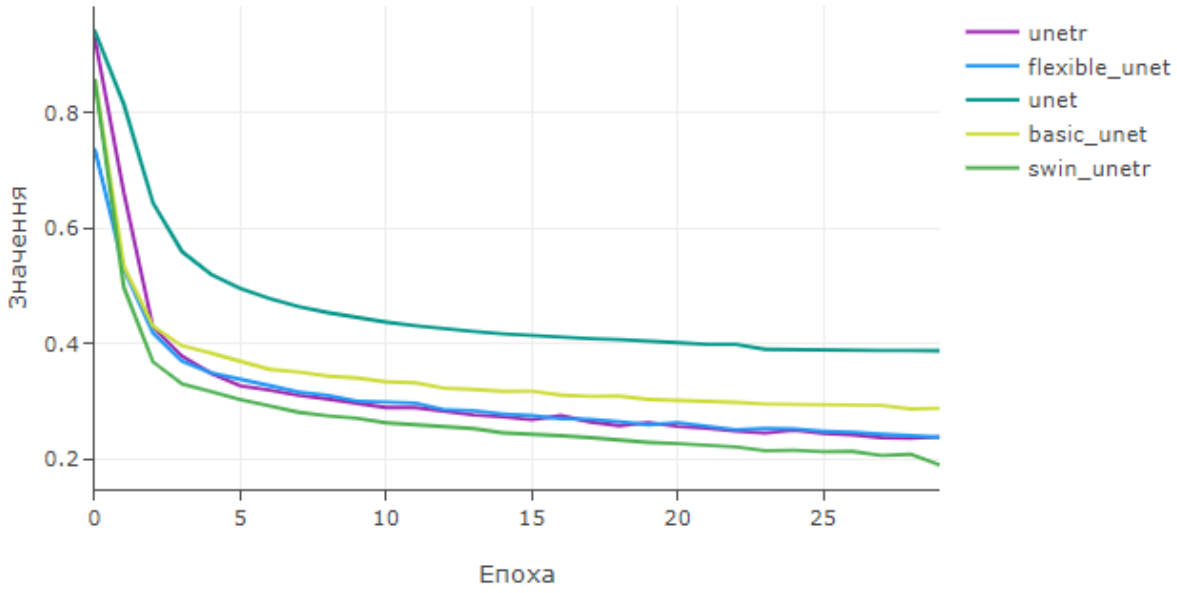

Рисунок 3.33 - Порівняння графіків зміни функції похибки моделей

Отже, як видно з таблиці вище, моделі на базі трансформеру мають перевагу по точності над методами повністю згорткових мереж. Найвижчу точність має мережа SwinUNETR і тому саме вона буде використовуватися у подальшій частині цієї роботи.

Загалом аналізуючи вище наведених таблиць та графіків можна зробити висновок що моделі на базі трансформеру є більш стабільними та не так часто піддаються проблемам перенавчання та локального мінімуму. За рахунок свого

механізму уваги ці мережі можуть розпізнати глибші зв’язки, які дозволяють їм не попадати у локальні мінімуми і які відсутні у повністю згорткових мереж. Мабуть головною перевагою повністю згорткових мереж є більша швидкість виконання за рахунок простішої структура та меншої кількості параметрів, однак різниця у швидкості не є достатньо високою щоб переважити нижчу точность цих мереж.

У висновку до проведеного аналізу можна сказати, що моделі на основі трансформерів є більш точними та більш обчислювально стабільними ніж повністю згорткові мережі, що робить їх очевидним вибором для використання в даній задачі. З двох порівняних моделей на базі трансформеру, кращою виявилась модель SwinUNETR, тому вона буде використовуватися у подальшій частині цієї роботи.

3.8. Переведення натренованої мережі у формат ONNX

ONNX (Open Neural Network Exchange) – це відкритий формат обміну моделями глибокого навчання. Він був розроблений спільно компаніями Microsoft та Facebook з метою забезпечити універсальну стандартну платформу для ефективного обміну моделями між різними фреймворками машинного навчання.

ONNX дозволяє експортувати та переносити моделі між різними середовищами без необхідності перекомпілювання або переписування коду. Він підтримує багато популярних фреймворків, таких як PyTorch, TensorFlow, Caffe, MXNet та інші.

За допомогою ONNX можна експортувати навчені моделі глибокого навчання до формату ONNX, який потім може бути імпортований і використаний в інших фреймворках для прогнозування, інференсу або налаштування моделі [36].

ONNX Runtime - це виконавча система для розгортання моделей глибокого навчання, заснована на стандарті ONNX. Вона надає оптимізовану та швидку інференцію моделей на різних платформах, включаючи ПК, сервери, мобільні пристрої та пристрої Інтернету речей (IoT).

Принцип роботи ONNX полягає у наступному:

1. Підготовка моделі: Модель глибокого навчання навчається у вибраному фреймворку, такому як PyTorch, TensorFlow або Keras.

2. Експорт у формат ONNX: Після навчання модель експортується у формат ONNX. Це означає, що модель конвертується до структурованого формату, який включає визначення архітектури моделі, параметрів та інших важливих відомостей.

3. Розгортання та інференція: ONNX Runtime надає можливість завантажувати та виконувати цей сконвертований ONNX-файл на різних платформах. Вона забезпечує оптимізацію та прискорення виконання моделі, використовуючи різні методи, такі як оптимізація графу, розпаралелювання, використання апаратного прискорення та інші.

Переваги ONNX та ONNX Runtime:

1. Переносимість моделей: ONNX дозволяє переносити моделі між різними фреймворками, що робить його корисним інструментом для співпраці між командами, які використовують різні фреймворки глибокого навчання.

2. Оптимізація та швидкість: ONNX Runtime надає оптимізовану та швидку інференцію моделей на різних платформах. Вона використовує різні техніки оптимізації для забезпечення максимальної продуктивності.

3. Гнучкість: ONNX підтримує широкий спектр операторів та функцій, що дозволяє моделювати різноманітні архітектури та завдання глибокого навчання.

4. Легкість використання: ONNX та ONNX Runtime мають зрозумілі та прості API, що робить їх досить легкими для використання та інтеграції в різні проекти.

5. Розширені можливості: ONNX підтримує розширення, що дозволяють додавати нові оператори та функції до стандарту. Це дозволяє використовувати спеціалізовані оператори, які можуть бути важливими для конкретних завдань або архітектур моделей.

6. Інтеграція з інструментами ML: ONNX і ONNX Runtime відкриті для використання з іншими інструментами машинного навчання, такими як

фреймворки для автоматизованого навчання (AutoML) або бібліотеки для візуалізації та аналізу моделей. Це спрощує інтеграцію та розширення функціональності з використанням існуючих інструментів.

7. Підтримка різних платформ: ONNX Runtime надає можливість розгортати моделі на різних платформах, включаючи ПК, сервери, мобільні пристрої та пристрої Інтернету речей (IoT). Це дозволяє використовувати моделі глибокого навчання в широкому спектрі сценаріїв та використовувати їх на різних пристроях.

8. Сумісність з популярними фреймворками: ONNX підтримує інтеграцію з відомими фреймворками глибокого навчання, такими як PyTorch, TensorFlow, Keras, MXNet та інші. Це означає, що моделі, створені в цих фреймворках, можуть бути легко експортовані в формат ONNX та розгорнуті за допомогою ONNX Runtime.

9. Масштабованість та продуктивність: ONNX Runtime використовує оптимізовані алгоритми та методи для досягнення максимальної продуктивності під час інференції моделей. Вона використовує паралельність, розподілення роботи та використання апаратного прискорення для забезпечення ефективної роботи з великими обсягами даних та складними моделями.

10. Активна спільнота та підтримка: ONNX та ONNX Runtime мають велику активну спільноту розробників, що підтримує стандарт та надає допомогу та відповіді на питання користувачів. Це дозволяє швидко розв'язувати проблеми та отримувати нові функції та покращення через постійний розвиток проекту.

11. Інтероперабельність: ONNX дозволяє обмінюватись моделями між різними фреймворками, незалежно від мови програмування чи платформи. Це дає змогу розробникам спрощувати процеси співпраці та спільної роботи над проектами, оскільки вони можуть використовувати різні фреймворки на основі стандарту ONNX без необхідності переконвертації моделей.

12. Оптимізація розміру моделі: ONNX дозволяє зменшувати розмір моделі шляхом видалення непотрібних операторів та оптимізації графу моделі. Це

особливо корисно при розгортанні моделей на обмежених ресурсах, наприклад, на мобільних пристроях чи пристроях Інтернету речей.

В даній роботі ONNX буде використовуватися для зменшення часу обробки зображень моделлю при роботі користувача з програмою.

Для швидкого переводу натренованої моделі було створено клас ONNX (рис. 3.34), який приймає на вхід об’єкт моделі та масив з розмірністю аналогічною до вхідних даних.

```
class ONNX:
  def __init__(self, model_path, dummy_input, onnx_path):
    self.model_path = model_path
    self.dummy_input = dummy_input
    self.onnx_path = onnx_path

  def to_onnx(self):
    model = torch.load(self.model_path)
    model.eval()

    torch.onnx.export(model, self.dummy_input, self.onnx_path, export_params=True)

  def check_onnx(self):
    onnx_model = onnx.load(self.onnx_path)
    onnx.checker.check_model(onnx_model)
```

Рисунок 3.34 - Реалізація класу ONNX

Після переведення моделі в формат ONNX вона зберігається в .onnx файл, який потім, разом з КТ зображеннями, подається на вхід класу ONNX_Inference (рис. 3.35), що реалізує, використовуючи бібліотеку onnx runtime, процес взаємодії з моделлю, збереженою у форматі ONNX.

```python
class ONNX_Inference:
  ''' Running onnx model on ORT
      Attributes:
        onnx_path: path to .onnx file
  '''

  def __init__(self, onnx_path):
    self.dummy_input = torch.zeros((3, 256, 256)).to('cuda')
    self.onnx_path = onnx_path

  def _to_standart_format(self, image):
    ''' transform image to standart format '''

    if image.shape[1] > 256:
      image = torchvision.transforms.Resize(256)(image)
    elif image.shape[1] < 256:
      image = torchvision.transforms.Pad(256-image.shape[1])(image)

    if torch.max(image) > 1:
      image = image/255

    if image.shape[0] == 1:
      image = torch.cat([image, image, image])

    return image.unsqueeze(0)

  def run_onnx(self, image):
    image = torchvision.io.read_image(image)
    image = self._to_standart_format(image)

    sess = onnxruntime.InferenceSession(self.onnx_path)
    input_name = sess.get_inputs()[0].name
    pred = sess.run(None, {input_name: image.numpy()})

    return pred[0]
```

Рисунок 3.35 - Реалізація класу ONNX_Inference

Клас ONNX містить два методи для перетворення моделі PyTorch у формат ONNX та для перевірки правильності створеного ONNX файлу. Розбір кожного методу детальніше:

1. Конструктор __init__(self, model_path, dummy_input, onnx_path): Ініціалізує об'єкт ONNX з шляхом до збереженого файлу моделі PyTorch (model_path), вхідним зразком (dummy_input) та шляхом до файлу ONNX (onnx_path), де буде збережено сконвертовану модель.

2. Метод to_onnx(self): Завантажує модель PyTorch з вказаного шляху за допомогою torch.load(). Встановлює модель в режим оцінювання (model.eval()). Потім використовує torch.onnx.export(), щоб експортувати модель у формат ONNX. Параметр export_params=True вказує на експорт параметрів моделі разом з моделлю. Результат зберігається за допомогою self.onnx_path.

3. Метод check_onnx(self): Завантажує ONNX модель з файлу за допомогою onnx.load(). Використовує onnx.checker.check_model(), щоб перевірити правильність структури ONNX моделі. Якщо модель правильна, нічого не повертається, а якщо є помилки, видається виняток.

Клас ONNX_Inference реалізує інференс моделі, яка збережена у форматі ONNX за допомогою ONNX Runtime (ORT). Розбір роботи класу описаний нижче:

1. Конструктор __init__(self, onnx_path): Ініціалізує об'єкт ONNX_Inference зі шляхом до файлу .onnx моделі. В ньому також створюється тензор-зразок dummy_input розміром (3, 256, 256) для підготовки вхідного зображення.

2. Метод _to_standart_format(self, image): Трансформує вхідне зображення до стандартного формату, що вимагає модель. Якщо розмір зображення більший за (3, 256, 256), він зменшується до цього розміру. Якщо розмір менший, то зображення доповнюється до потрібного розміру подушкою. Якщо значення пікселів більше 1, то вони нормалізуються до діапазону [0, 1]. Якщо зображення має один канал, він розширюється до трьох каналів, копіюючи значення по всіх каналах.

3. Метод run_onnx(self, image): Виконує інференс моделі на вхідному зображенні. Спочатку зображення читається за допомогою torchvision.io.read_image(). Потім викликається метод _to_standart_format() для підготовки зображення до формату, що очікується моделлю. Після цього створюється сесія onnxruntime.InferenceSession з файлом .onnx моделі. За допомогою sess.get_inputs()[0].name отримується ім'я вхідного тензора моделі. Виконується інференс за допомогою sess.run(None, {input_name: image.numpy()}), де вхідне зображення перетворюється в массив NumPy. Повертається прогнозований вихід моделі pred[0].

3.9 Розробка інтерфейсу користувача

Для зручної взаємодії було розроблено інтерфейс користувача. Вхідними даними інтерфейсу є файл КТ, який подається на вхід натренованої моделі. Вхідній дані та результат роботи моделі відобразиться у вікні програми. Також, для кращої візуалізації результату, буде відображено маску сегментації накладену на вхідне зображення. Приклад роботи інтерфейсу відображено на рис. 3.36.

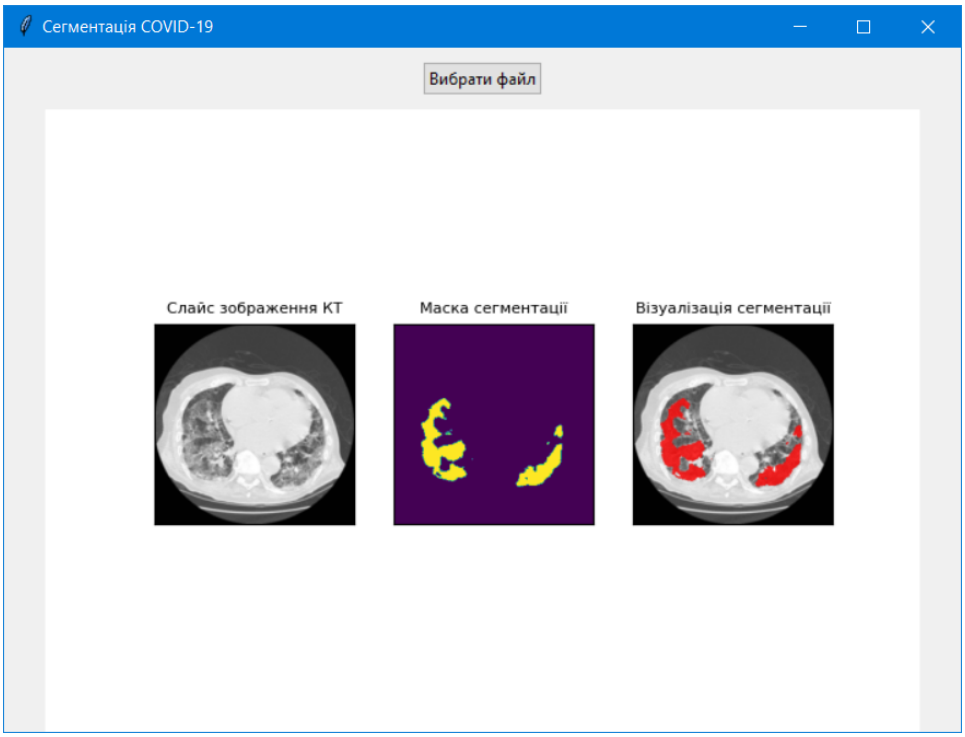

Рисунок 3.36 - Приклад роботи інтерфейсу користувача

3.9.1. Внутрішній інтерфейс взаємодії з моделлю

Внутрішній інтерфейс взаємодії з натренованою моделлю реалізований у файлі inference.py і складається з наступних функцій:

1. Inference(image_path): функція, яка приймає на вхід шлях до вхідного зображення, викликає функцію попередньої обробки зображення, завантажує стан моделі, проганяє зображення через модель та повертає результат.

2. image_preprocessing(image_path): функція, яка приймає на вхід шлях до вхідного зображення, завантажує його, стандартизує розмір та діапазон значень та повертає зображення.

Приклад взаємодії з внутрішнім інтерфейсом зображено на рис. 3.37.

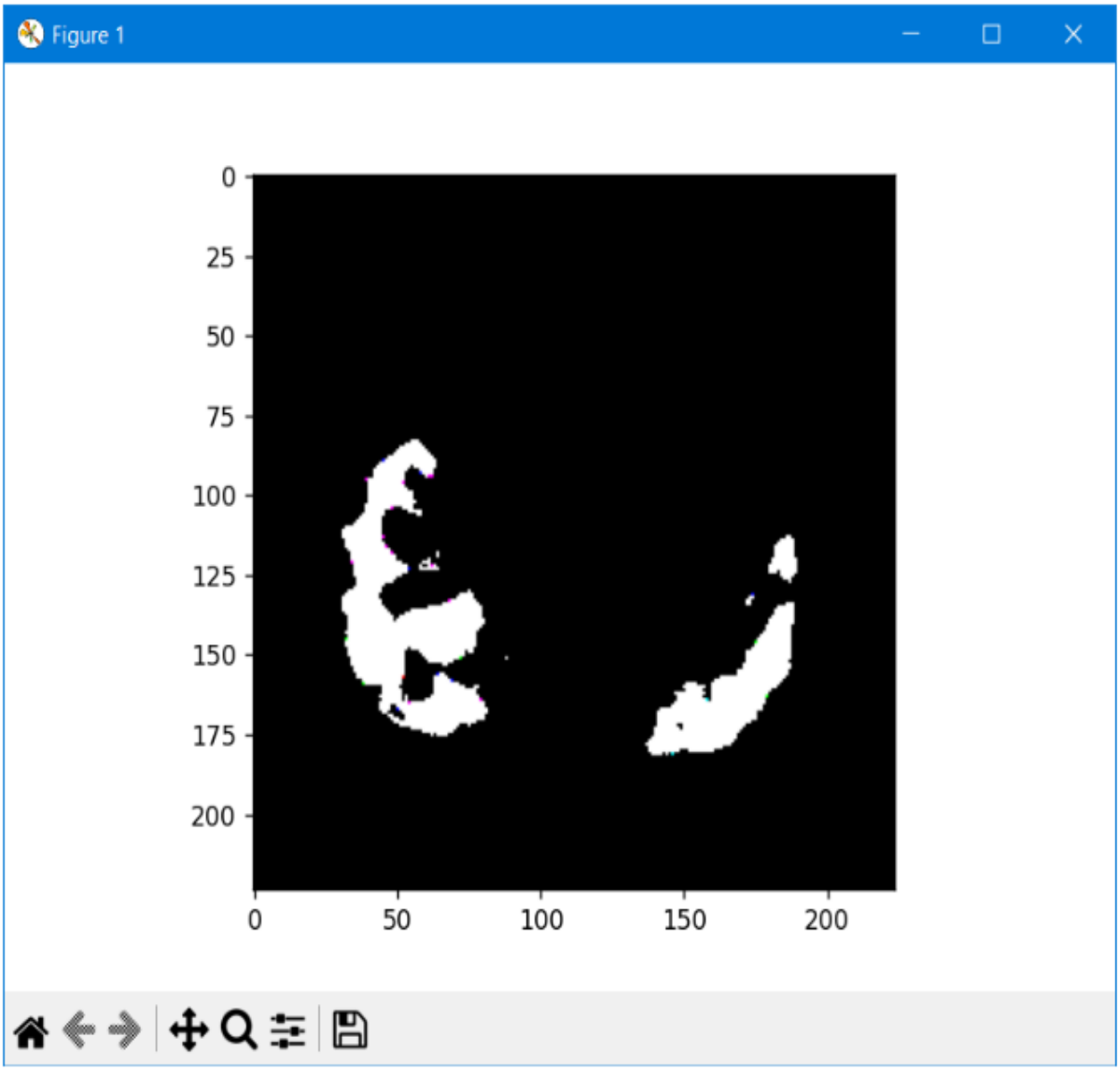

Рисунок 3.37 - Приклад взаємодії з внутрішнім інтерфейсом

3.9.2. Інтерфейс користувача

Інтерфейс користувача реалізовано у файлі ui.py і має наступну структуру:

1. Створення вікна інтерфейсу розміром 700 х 500 пікселів.

2. Відображення кнопки вибору файлу зображення.

3. При натисканні на кнопку викликається функція open_file()

4. Відкривається вікно вибору файлу

5. Файл відкривається бібліотекою torchvision

6. Викликається внутрішній інтерфейс та отримується результат роботи моделі

7. Отримана маска накладається на вхідне зображення

8. Налаштовується формат відображення зображень

9. Всі три зображення виводяться на панель інтерфейсу

10. При закритті вікна програма завершується

3.9.3. Налаштування бібліотеки Tkinter

Tkinter є стандартною бібліотекою для роботи з графічним інтерфейсом користувача (GUI) у Python. Вона надає можливість створювати вікна, кнопки, поля введення, список, меню та інші елементи інтерфейсу для взаємодії з користувачем. Нижче описаний процес налаштування бібліотеки Tkinter:

1. Імпортування Tkinter: Щоб почати використовувати Tkinter, необхідно імпортувати модуль tkinter у своєму Python скрипті.

2. Створення головного вікна: Першим кроком у роботі з Tkinter є створення головного вікна (вікна програми) за допомогою класу Tk.

3. Додавання віджетів: Tkinter надає різні віджети (елементи інтерфейсу), такі як кнопки, поля введення, мітки, списки і т.д., які можна додати до вікна.

4. Менеджер компонування: Tkinter використовує різні менеджери компонування, такі як pack(), grid() і place(), для розміщення віджетів у вікні. Менеджер pack() розташовує віджети один за одним у вертикальному або горизонтальному напрямку. Менеджер grid() дозволяє розміщувати віджети в сітку з рядками і стовпцями. Менеджер place() дозволяє точно розмістити віджети за вказаними координатами.

5. Реагування на події: Tkinter дозволяє прив'язувати функції (функції зворотного виклику) до подій, таких як натискання кнопки, переміщення миші або натискання клавіші. Це дозволяє виконувати певні дії при виникненні певних подій. Наприклад, для реагування на натискання кнопки можна використати метод bind().

3.10 Розрахунок економічного ефекту

В цьому розділі проведено оцінку характеристик розробленого програмного продукту для спрощення задачі діагностики та локалізації COVID-19 в легенях людини шляхом використання повністю згорткової нейронної мережі. Інтерфейс користувача для взаємодії з нейронною мережею був реалізований з використання фреймворку Flask мови програмування Python.

За результатом економічного розділу були закріплені знання в галузі організації виробництва та економіки. В цьому розділі було визначено чотири варіанти реалізації програми. Найефективнішим виявився третій варіант, величина витрат для якого становить 294 343 грн. Детальні розрахунки наведені у звіті з практики.

3.11 Безпека життєдіяльності та охорона праці

У цьому розділі виявлено небезпечні фактори, що виникають під час проведення комп'ютерної томографії, а також будуть розглянуті заходи для їх усунення.

За результатами розрахунку параметрів кабінету можна зробити висновок що даний кабінет відповідає нормам.

Згідно з ДСТУ 3855-99 в кабінеті комп'ютерної томографії правила пожежної безпеки дотримані. Детальний аналіз описаний у звіті з практики.

Висновки до розділу 3

В цьому розділі було проведено аналіз двох типів моделей, в результаті якого виявилося, що повністю згорткові мережі, хоч і мають дещо вищу швидкість, є менш ефективними відносно моделей на основі архітектури трансформер. Найефективнішою виявилась модель SwinUNETR, яка була використана у решті розділу.

ЗАГАЛЬНІ ВИСНОВКИ

В результаті виконання дипломної роботи для вирішення задачі сегментування COVID-19 було застосовано моделі на основі архітектури трансформер та порівняно їх ефективність відносно традиційних повністю-згорткових методів. Проаналізувавши заміри точності та швидкості обох типів моделей було виявлено, що мережі на основі трансформеру мають вищу точність (приблизно 15%), хоча повністю згорткові мережі мають деяку перевагу у швидкості виконання (приблизно 17 хвилин). Найефективнішою з моделей виявилась модель SwinUNETR, значення функції похибки якої становить 0.23 (на 0.01 більше UNETR та на 0.005 більше Flexible U-Net). Ця модель потім була використана для вирішення даної задачі та для взаємодії з нею було побудовано інтерфейс користувача.

Аналізуючи результати роботи можна сказати, що перевага архітектури трансформер над повністю згортковими мережами є очікуваною, оскільки трансфорер включає в себе механізм уваги, який дозволяє обчислити глибокі просторові ознаки, які недоступні повністю згортковим архітектурам, і які є дуже корисними для більш повного розуміння просторових залежностей між об’єктами. Однак, перевага в точності SwinUNETR та UNETR над іншими досліджуваними моделями не є переконливо високою і їх швидкість виконання є дещо нижчою, що говорить про те, що повністю згорткові моделі на даний момент досі є конкурентноспроможними та можуть бути використані для ряду задач. Але зважаючи на стрімкий розвиток архітектури трансформер, який не зупиняється і до сьогоднішнього дня, наводить на думку, що відрив у ефективності моделей на основі цієї архітектури від повністю-згорткових алгоритмів ставатиме все більшим і тоді вибір моделей типу U-Net над моделями на основі трансформеру ставатиме все менш і менш привабливим.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Булгар А. В. ОСОБЛИВОСТІ КЛІНІЧНОГО ПЕРЕБІГУ АДЕНОМІОЗУ ПІСЛЯ ПЕРЕНЕСЕНОГО ЗАХВОРЮВАННЯ COVID–19. *Актуальні питання педіатрії, акушерства та гінекології*. 2022. № 2. С. 80–84.
2. Xie X., Zhong Z., Zhao W., Zheng C., Wang F., Liu J. Chest CT for Typical Coronavirus Disease 2019 (COVID-19) Pneumonia: Relationship to Negative RT-PCR Testing. *Radiology*. 2020. Вип. 296, № 2. С. E41–E45.
3. Turnbull R. Enhanced detection of the presence and severity of COVID-19 from CT scans using lung segmentation. 2023.
4. Shan F., Gao Y., Wang J., Shi W., Shi N., Han M., Xue Z., Shen D., Shi Y. Lung Infection Quantification of COVID-19 in CT Images with Deep Learning. 2020.
5. МАЙСТРУК М. СУЧАСНІ АСПЕКТИ ВИКОРИСТАННЯ ДИХАЛЬНИХ ВПРАВ В ПРАКТИЦІ ФІЗИЧНОГО ТЕРАПЕВТА У ХВОРИХ НА ХРОНІЧНЕ ОБСТРУКТИВНЕ ЗАХВОРЮВАННЯ ЛЕГЕНЬ. *Physical Culture and Sport: Scientific Perspective*. 2022. № 2. С. 24–29.
6. МАЙСТРУК М. ІНФОРМАЦІЙНА СКЛАДОВА В РЕАБІЛІТАЦІЙНОМУ ПРОЦЕСІ ХВОРИХ НА ХРОНІЧНЕ ОБСТРУКТИВНЕ ЗАХВОРЮВАННЯ ЛЕГЕНЬ. *Physical Culture and Sport: Scientific Perspective*. 2022. № 1. С. 42–48.
7. Sun W., Chen J., Yan L., Lin J., Pang Y., Zhang G. COVID-19 CT image segmentation method based on swin transformer. *Frontiers in Physiology*. 2022. Вип. 13.
8. Ardila D., Kiraly A. P., Bharadwaj S., Choi B., Reicher J. J., Peng L., Tse D., Etemadi M., Ye W., Corrado G., Naidich D. P., Shetty S. End-to-end lung cancer screening with three-dimensional deep learning on low-dose chest computed tomography. *Nature Medicine*. 2019. Вип. 25, № 6. С. 954–961.
9. SN K., Fred A L., S M., H A. K., Varghese P S. A voyage on medical image segmentation algorithms. *Biomedical Research*. 2018.
10. Xia K., Wang J. Recent advances of Transformers in medical image analysis: A

comprehensive review. *MedComm – Future Medicine*. 2023. Вип. 2, № 1.

11. Enshaei N., Oikonomou A., Rafiee M. J., Afshar P., Heidarian S., Mohammadi A., Plataniotis K. N., Naderkhani F. COVID-rate: an automated framework for segmentation of COVID-19 lesions from chest CT images. *Scientific Reports*. 2022. Вип. 12, № 1. С. 3212.
12. Ronneberger O., Fischer P., Brox T. U-Net: Convolutional Networks for Biomedical Image Segmentation. 2015.
13. Long J., Shelhamer E., Darrell T. Fully Convolutional Networks for Semantic Segmentation. 2014.
14. Milletari F., Navab N., Ahmadi S.-A. V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. 2016.
15. Avilov O., Rimbert S., Popov A., Bougrain L. Deep Learning Techniques to Improve Intraoperative Awareness Detection from Electroencephalographic Signals. IEEE, 2020. ISBN 978-1-7281-1990-8.
16. Hou J., Xu J., Feng R., Zhang Y., Shan F., Shi W. CMC-COV19D: Contrastive Mixup Classification for COVID-19 Diagnosis. IEEE, 2021. ISBN 978-1-6654-0191-3.
17. Dar A. S., Padha D. Medical Image Segmentation A Review of Recent Techniques, Advancements and a Comprehensive Comparison. *International Journal of Computer Sciences and Engineering*. 2019. Вип. 7, № 7. С. 114–124.
18. Sherstinsky A. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network. 2018.
19. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser L., Polosukhin I. Attention Is All You Need. 2017.
20. OpenAI. GPT-4 Technical Report. 2023.
21. Dosovitskiy A., Beyer L., Kolesnikov A., Weissenborn D., Zhai X., Unterthiner T., Dehghani M., Minderer M., Heigold G., Gelly S., Uszkoreit J., Houlsby N. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. 2020.
22. Hatamizadeh A., Tang Y., Nath V., Yang D., Myronenko A., Landman B., Roth H.,

Xu D. UNETR: Transformers for 3D Medical Image Segmentation. 2021.

23. Xie E., Wang W., Yu Z., Anandkumar A., Alvarez J. M., Luo P. SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers. 2021.
24. Strudel R., Garcia R., Laptev I., Schmid C. Segmenter: Transformer for Semantic Segmentation. 2021.
25. Islam K. Recent Advances in Vision Transformer: A Survey and Outlook of Recent Work. 2022.
26. Carneiro G., Nascimento J., Bradley A. P. Deep Learning Models for Classifying Mammogram Exams Containing Unregistered Multi-View Images and Segmentation Maps of Lesions11This work is an extension of the paper published by the same authors at the Medical Image Computing and Computer-Assisted Inte. *Deep Learning for Medical Image Analysis*. Elsevier, 2017. С. 321–339.
27. Chen J., Lu Y., Yu Q., Luo X., Adeli E., Wang Y., Lu L., Yuille A. L., Zhou Y. TransUNet: Transformers Make Strong Encoders for Medical Image Segmentation. 2021.
28. Liu Z., Lin Y., Cao Y., Hu H., Wei Y., Zhang Z., Lin S., Guo B. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. 2021.
29. Adaloglou N. Understanding the receptive field of deep convolutional networks. 2020.
30. Saha S. A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way. 2018.
31. Agarap A. F. Deep Learning using Rectified Linear Units (ReLU). 2018.
32. Zafar A., Aamir M., Mohd Nawi N., Arshad A., Riaz S., Alruban A., Dutta A. K., Almotairi S. A Comparison of Pooling Methods for Convolutional Neural Networks. *Applied Sciences*. 2022. Вип. 12, № 17. С. 8643.
33. Touvron H., Cord M., Douze M., Massa F., Sablayrolles A., Jégou H. Training data-efficient image transformers & distillation through attention. 2020.
34. Chen L., Cruz A., Ramsey S., Dickson C. J., Duca J. S., Hornak V., Koes D. R., Kurtzman T. Hidden bias in the DUD-E dataset leads to misleading performance of

deep learning in structure-based virtual screening. *PLOS ONE*. 2019. Вип. 14, № 8. С. e0220113.

35. Hatamizadeh A., Nath V., Tang Y., Yang D., Roth H., Xu D. Swin UNETR: Swin Transformers for Semantic Segmentation of Brain Tumors in MRI Images. 2022.
36. Hatamizadeh Ali, Nath Vishwesh B. V. Novel Transformer Model Achieves State-of-the-Art Benchmarks in 3D Medical Image Analysis. 2022.
37. Morozov S. P., Andreychenko A. E., Pavlov N. A., Vladzymyrskyy A. V., Ledikhova N. V., Gombolevskiy V. A., Blokhin I. A., Gelezhe P. B., Gonchar A. V., Chernina V. Y. MosMedData: Chest CT Scans With COVID-19 Related Findings Dataset. *Medical Image Analysis*. 2020. Вип. 82. С. 102605.
38. Bell D., Greenway K. Hounsfield unit. *Radiopaedia.org*. Radiopaedia.org, 2015.
39. Cardoso M. J., Li W., Brown R., Ma N., Kerfoot E., Wang Y., Feng A. та ін. MONAI: An open-source framework for deep learning in healthcare. 2022.