Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук

Бакалаврська кваліфікаційна робота на тему:

ПРОГРАМНИЙ МОДУЛЬ НЕЙРОМЕРЕЖЕВОЇ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ СТЕНОЗУ КОРОНАРНИХ СУДИН СЕРЦЯ

Виконала: студентка 4 курсу, групи 2КН-21б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Яцків Д. О.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф.КН
Колесницький О.К.
(прізвище та ініціали)
«04» червня 2025 р.

Рецензент: к.т.н., проф. каф. КСУ
Биков М. М.
(прізвище та ініціали)
«05» червня 2025 р.

Допущено до захисту
Зав. кафедри Яровий А.А.
«06» червня 2025 р.

Вінниця ВНТУ - 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 122 Комп'ютерні науки
Освітньо - професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
проф., д.т.н. Яровий А.А.
«05» травня 2025 р.

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Яцків Діані Олександрівні

1. Тема роботи: Програмний модуль нейромережевої сегментації зображень стенозу коронарних судин серця.
керівник роботи: Колесницький Олег Костянтинович, к.т.н., проф.
затверджені наказом вищого навчального закладу "20" березня 2025 року № 97

2. Термін подання студентом роботи "30" травня 2025р.

3. Вихідні дані до роботи :
Формат вхідних файлів зображень – jpeg; вид вхідних зображень – рентгенівська коронарна ангіографія (РКА); обсяг набору даних для навчання та тестування – не менше 1000 зображень; використання об'єктноорієнтованої мови програмування.

4. Зміст текстової частини (перелік питань, які потрібно розробити):
проаналізувати існуючі методи сегментації зображень та вибрати найбільш ефективний;
вибрати нейронну мережу, спроектувати її структуру та метод навчання
розробити алгоритм роботи модуля сегментації зображень стенозу коронарних судин серця;
спроектувати та реалізувати модуль сегментації зображень стенозу коронарних судин серця за допомогою згорткової нейронної мережі;
протестувати роботу програми сегментації зображень стенозу коронарних судин серця на основі згорткової нейронної мережі.

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень):
Схема алгоритму роботи програми
Структура нейронної мережі
Схема запропонованої процедури аугментації даних
Результати роботи програми

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Колесницький О.К., к.т.н., проф. каф. КН		

7. Дата видачі завдання "05" травня 2025р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз предметної області	05.05.25	07.05.25	
2	Обгрунтування методу розв'язання задачі, проектування програмного модуля нейромережевої сегментації зображень стенозу коронарних судин серця	08.05.25	11.05.25	
3	Розробка алгоритму роботи програмного модуля сегментації зображень стенозу коронарних судин серця на основі згорткової нейронної мережі	12.05.25	26.05.25	
4	Програмна реалізація модуля нейромережевої сегментації зображень стенозу коронарних судин серця	27.05.25	29.05.25	
5	Аналіз результатів тестування модуля нейромережевої сегментації зображень стенозу коронарних судин серця	30.05.25	01.06.25	
6	Оформлення матеріалів до захисту БКР	02.06.25	04.06.25	

Студентка _____ **Яцків Д. О.**
(підпис) (прізвище та ініціали)

Керівник проекту (роботи) _____ **Колесницький О.К.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Яцків Д. О. Програмний модуль нейромережевої сегментації зображень стенозу коронарних судин серця. Бакалаврська кваліфікаційна робота скаладається з 72 сторінок формату А4, на яких є 17 рисунків, 1 таблиця, список використаних джерел містить 28 найменувань.

Метою роботи є підвищення точності сегментації стенозу коронарних судин серця по зображеннях рентгенівської коронарної ангіографії.

Розроблений програмний модуль призначений для сегментації стенотичних уражень у пацієнтів з ішемічною хворобою серця і працює на рентгенівській коронарній ангіографії (РКА). Було обгрунтовано вибір архітектури згорткової нейронної мережі Mask R-CNN для використання у сегментації зображень стенозу коронарних судин серця як найбільш перспективну. Розроблено метод сегментації зображень стенозу коронарних судин серця на основі Mask R-CNN, що реалізує підхід RoIAlign. Було розроблено процес аугментації даних для навчання згорткової нейронної мережі Mask R-CNN. Модуль розроблено на мові програмування Python з використанням бібліотек Pytorch, Pytorch Image Models, NumPy, Torchvision та Matplotlib. Для навчання та тестування модуля взято набір ARCADE, який містить приблизно 3 000 зображень, з яких 2000 навчальних, 400 валідаційних та 600 тестових. Оцінка F1 на тестовому наборі для запропонованого модуля має величину 0,51, а для кращого з аналогів – 0,38. Тобто точність сегментації стенозу коронарних судин серця розробленої програми підвищена в абсолютній величині на 0,13 для показника F1, а у відносних одиницях - на 34,2%.

Ключові слова: сегментація зображеь, стеноз коронарних судин, рентгенівська коронарна ангіографія, ішемічна хвороба серця, згорткова нейронна мережа.

ABSTRACT

Yatskiv D. O. Software module for neural network coronary artery stenosis images segmentation. The bachelor thesis consists of 72 pages of A4 format, on which there are 17 figures, 1 tables, the list of used sources contains 28 names.

The aim of the work is to improve the accuracy of segmentation of coronary artery stenosis using X-ray coronary angiography images.

The developed software module is designed for segmentation of stenotic lesions in patients with ischemic heart disease and works on X-ray coronary angiography (RCA). The choice of the architecture of the convolutional neural network Mask R-CNN for use in segmentation of coronary artery stenosis images was justified as the most promising. A method for segmentation of coronary artery stenosis images based on Mask R-CNN was developed, which implements the RoIAlign approach. A data augmentation process was developed for training the convolutional neural network Mask R-CNN. The module was developed in the Python programming language using the Pytorch, Pytorch Image Models, NumPy, Torchvision and Matplotlib libraries. The ARCADE set was used for training and testing the module, which contains approximately 3,000 images, of which 2,000 are training, 400 are validation and 600 are test. The F1 score on the test set for the proposed module is 0.51, and for the best of the analogues - 0.38. That is, the accuracy of segmentation of coronary artery stenosis of the developed program is increased in absolute value by 0.13 for the F1 indicator, and in relative units - by 34.2%.

Keywords: image segmentation, coronary artery stenosis, X-ray coronary angiography, ischemic heart disease, convolutional neural network.

ЗМІСТ

ВСТУП

Актуальність досліджень. Ішемічна хвороба серця (ІХС) – це серцево-судинний стан, спричинений накопиченням атеросклеротичної бляшки в коронарних артеріях. ІХС є однією з основних причин смерті в усьому світі, вражаючи близько 5% населення світу. Накопичення бляшок викликає звуження цих судин, яке називається стенозом, що може загрожувати пацієнту, зменшуючи кровопостачання серця. Одним із найбільш часто використовуваних інструментів для діагностики ішемічної хвороби серця є рентгенівська коронарна ангіографія (РКА), при якій контрастну речовину вводять у коронарні артерії пацієнта через катетер, а потім отримують рентгенівське зображення для візуалізації артерій. Хоча деякі стверджують, що комп'ютерна томографічна коронарна ангіографія (КТКА) є більш цінним інструментом для оцінки ішемічної хвороби серця через її здатність захоплювати 3D-структури, у цій роботі вирішено працювати з РКА через її чудову діагностичну здатність у важких і обструктивних випадках ІХС.

Традиційні методи сегментації судин і стенотичних уражень є трудомісткими та забирають багато часу, що робить їх неможливими для великих обсягів даних. Було розроблено декілька методів глибокого навчання для загальної сегментації коронарної артерії, однак, наскільки відомо, у клінічних умовах не використовуються такі усталені методи для автоматизації сегментації стенозу на зображеннях рентгенівської коронарної ангіографії (РКА) за допомогою глибокого навчання. Таким чином, створення системи комп'ютерного зору глибокого навчання для покращення сегментації стенозу з високою точністю та обчислювальною ефективністю для клінічного застосування було б надзвичайно цінним. Така автоматична система виявлення стенозу може слугувати помічником радіологів, виконуючи «перший прохід» над величезною кількістю РКА -зображень, створених для аналізу, щоб зменшити рутинне навантаження на радіологів. Це допомогло б оптимізувати клінічні конвеєри, а також потенційно зменшити високі

показники вигорання, які відчувають радіологи, в основному через суттєве збільшення робочого навантаження в останні роки. З цією метою в цій роботі розробляється алгоритм, який приймає РКА-зображення будь-якого розміру як вхідні дані та використовує модель згорткової нейромережі Mask R-CNN для створення прогнозованої бінарної маски сегментації, що вказує на імовірні ділянки стенозу.

Метою дослідження є підвищення точності сегментації стенозу коронарних судин серця по зображеннях рентгенівської коронарної ангіографії. На відміну від аналогів, які використовують такі архітектури згорткових нейромереж як YOLO та U-Net, у цій роботі пропонується застосувати згорткову штучну нейронну мережу Mask R-CNN.

Об’єктом дослідження є процес комп’ютеризованої сегментації стенозу коронарних судин серця по зображеннях рентгенівської коронарної ангіографії на основі нейромережі.

Предметом дослідження є алгоритми та програмні засоби сегментації стенозу коронарних судин серця по зображеннях рентгенівської коронарної ангіографії на основі згорткової нейронної мережі та точність їх роботи.

Задачі дослідження:

1. Проаналізувати відомі методи сегментації стенозу коронарних судин серця та обрати напрямок досліджень;
2. Обгрунтувати вибір архітектури згорткової нейромережі;
3. Розробити метод сегментації зображень стенозу коронарних судин серця на основі згорткової нейронної мережі;
4. Розробити алгоритм роботи програмного модуля сегментації зображень стенозу коронарних судин серця;
5. Здійснити програмну реалізацію модуля сегментації зображень стенозу коронарних судин серця;
6. Провести тестування програмного модуля сегментації зображень стенозу коронарних судин серця.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ СТЕНОЗУ КОРОНАРНИХ СУДИН СЕРЦЯ

1.1 Постановка задачі

Сегментація зображення – це завдання пошуку груп пікселів, кожна з яких характеризує один смисловий об'єкт. У статистиці ця проблема відома як кластерний аналіз і є широко вивченою областю із сотнями різних алгоритмів. У комп'ютерному зорі сегментація зображення є однією з найстаріших проблем, що широко вивчаються.

У комп'ютерному зорі, сегментація - це процес поділу цифрового зображення на декілька сегментів (множину пікселів, також званих суперпікселями). Мета сегментації полягає у спрощенні та/або зміні уявлення зображення, щоб його було простіше та легше аналізувати. Сегментація зображень зазвичай використовується для того, щоб виділити об'єкти та межі (лінії, криві, тощо) на зображеннях. Більш точно, сегментація зображень - це процес присвоєння міток кожному пікселю зображення так, що пікселі з однаковими мітками мають загальні візуальні характеристики.

Сегментація зображень стенозу коронарних судин серця – це виділення на зображенні рентгенівської коронарної ангіографії – РКА (X-ray Coronary Angiography - XCA) або комп’ютерної томографічної коронарної ангіографії – КТКА (Computed Tomography Coronary Angiography - CTCA) ділянок, де зображені судини із стенозом (звуження судин, що з`явились внаслідок ішемічної хвороби серця – ІХС (Coronary artery disease - CAD)).

Стеноз - це звуження кровоносних судин. Одна з найбільш частих його причин — атеросклероз. Небезпека цього стану в тому, що воно може не проявлятися ніякими симптомами до того моменту, поки у людини не виникне інсульт. Тому дуже важливим є виявлення стенозу на ранніх стадіях.

Завдання полягає в розробці програми, яка буде визначати на зображенні ділянки, на яких знаходяться стенозовані судини. Вхідними даними програмиє

є файл зображення рентгенівської коронарної ангіографії (РКА) будь-якого розміру. Буде використано модель згорткової нейромережі Mask R-CNN для створення прогнозованої бінарної маски сегментації у вигляді обмежувальної рамки, що вказує на імовірні ділянки стенозу.

1.2 Огляд відомих методів нейромережевої сегментації зображень стенозу коронарних судин серця

1.2.1 Традиційні методи сегментації зображень [1, 2].

Спочатку сегментація зображень почалася із цифрової обробки зображень у поєднанні з алгоритмами оптимізації. Ці примітивні алгоритми використовували такі методи, як вирощування областей та алгоритм змій, де вони встановлювали початкові області, а алгоритм порівнював значення пікселів, щоб отримати уявлення про карту сегментів.

Ці методи брали локальне уявлення про функції зображення та фокусувалися на локальних відмінностях та градієнтах у пікселях. Алгоритми, які використовували глобальне уявлення вхідного зображення, з'явилися набагато пізніше, коли серед класичних методів обробки зображень було запропоновано такі методи, як адаптивна гранична обробка, алгоритм Оцу та алгоритми кластеризації.

• Методи на основі порогу.

Порогове визначення – один із найпростіших методів сегментації зображення, при якому встановлюється порогове значення для розділення пікселів на два класи. Пікселі, значення яких перевищують граничне значення, встановлюються рівними 1, а пікселі, значення яких менше граничного значення, встановлюються рівними 0.

Таким чином, зображення перетворюється на двійкову карту, що призводить до процесу, який часто називають бінаризацією. Порогове значення зображення дуже корисне, якщо різниця у значеннях пікселів між

двома цільовими класами дуже велика, і легко вибрати середнє значення як поріг.

Порогове значення часто використовується для бінаризації зображення, щоб можна було використовувати додаткові алгоритми, такі як виявлення контуру та ідентифікація, які працюють лише з бінарними зображеннями.

• Сегментація у регіонах.

Алгоритми сегментації на основі областей працюють, шукаючи подібність між сусідніми пікселями та групуючи їх у загальний клас. Як правило, процедура сегментації починається з того, що деякі пікселі встановлюються як вихідні пікселі, а алгоритм працює, виявляючи безпосередні межі вихідних пікселів і класифікуючи їх як схожі або несхожі. Потім розглядаються безпосередні сусіди і кроки повторюються до того часу, поки все зображення не буде сегментоване. Прикладом подібного алгоритму є популярний алгоритм вододілу для сегментації, який працює, починаючи з локальних максимумів карти евклідових відстаней і зростає за умови, що жодні два початкові числа не можуть бути класифіковані як такі, що належать до однієї області або карти сегмента.

• Сегментація країв.

Сегментація країв, також звана виявленням країв, є завданням виявлення країв на зображеннях. З точки зору сегментації можна сказати, що виявлення країв відповідає класифікації пікселів зображення, які є крайовими пікселями, і виділення цих крайових пікселів в окремий клас.

Виявлення країв зазвичай виконується за допомогою спеціальних фільтрів, які видають краї зображення після згортки. Ці фільтри розраховуються за допомогою спеціальних алгоритмів, що оцінюють градієнти зображення в координатах x та y просторової площини.

• Сегментація з урахуванням кластеризації.

Сучасні процедури сегментації, які залежить від методів обробки зображень, зазвичай використовують алгоритми кластеризації для сегментації.

Алгоритми кластеризації працюють краще, ніж їхні аналоги, і можуть надавати досить якісні сегменти за невеликий проміжок часу. Популярні алгоритми, такі як алгоритми кластеризації K-середніх, є неконтрольованими алгоритмами, які працюють шляхом кластеризації пікселів із загальними атрибутами разом як такі, що належать до певного сегменту.

Кластеризація K-середніх, зокрема, враховує всі пікселі та групує їх у k класів. На відміну від методів нарощування регіонів, методи на основі кластеризації не вимагають вихідної точки для початку сегментації.

1.2.2 Методи, що ґрунтуються на глибокому навчанні

В даний час найкращі результати в області сегментації зображень показують методи, що базуються на машинному навчанні. У загальному випадку моделі сегментації надають карти сегментів як вихідні дані, відповідні вхідним даним. Ці карти сегментів часто є n-канальними, де n – кількість класів, які модель повинна сегментувати. Кожен із цих n-каналів є бінарним за своєю природою, при цьому розташування об'єктів «заповнені» одиницями, а порожні області складаються з нулів (рис. 1.1) [3].

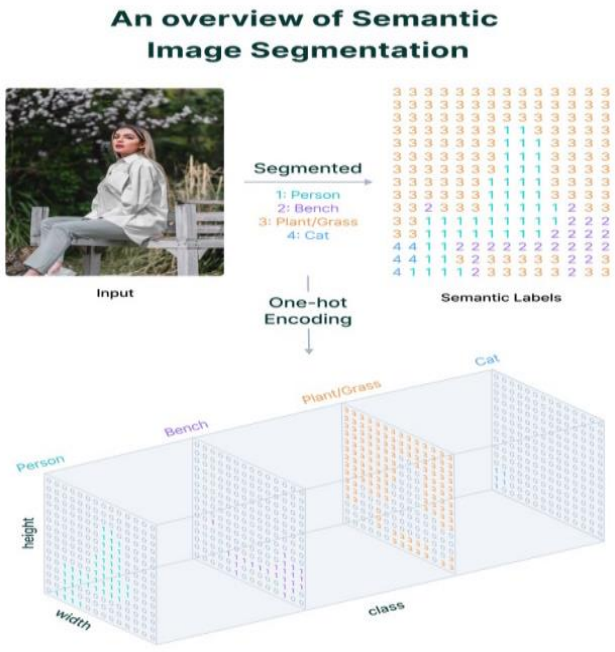


Рисунок 1.1 – Сегментація зображення

Нейронні мережі, що виконують сегментацію, зазвичай використовують структуру кодер-декодер, в якій за кодером слідує вузьке місце, а декодер або шари підвищення дискретизації прямують безпосередньо після вузького місця (рис. 1,2).

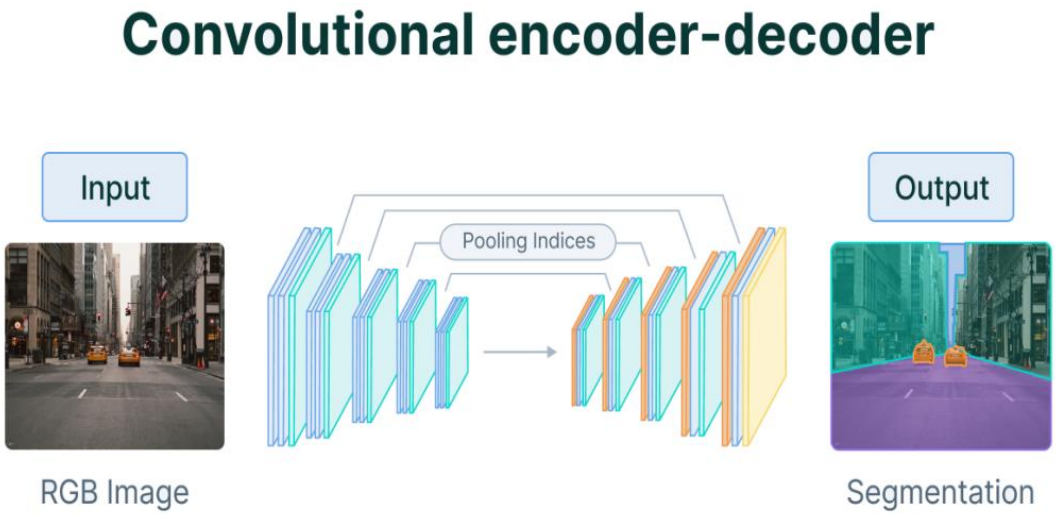


Рисунок 1.2 – Структура кодер-декодер для сегментації зображень

Архітектури кодер-декодер для семантичної сегментації стали популярними з появою такої архітектури, як SegNet у 2015 році [4]. SegNet пропонує використовувати комбінацію блоків згортки та знижувальної дискретизації, щоб сформувати подання вхідних даних. Потім декодер реконструює вхідну інформацію, щоб сформувати карту сегментів, що виділяє регіони на вході та групує їх за класами. Нарешті декодер має сигмоподібну функцію активації в кінці, яка стискає вихідні значення в діапазоні (0,1). На рис. 1.2 показано архітектуру кодер-декодер.

SegNet супроводжувався випуском ще однієї незалежної роботи з сегментації в той же час, U-Net [5], яка вперше використовувала пропуски з'єднань у глибокому навчанні як вирішення проблеми втрати інформації, що спостерігається в шарах зниження дискретизації типових мереж кодер-декодер.

Пропущені з'єднання (Skip Connections) – це з'єднання, які йдуть від кодера безпосередньо до декодера, минаючи вузьке місце [6].

Іншими словами, карти ознак на різних рівнях закодованих уявлень фіксуються та об'єднуються у карти ознак у декодері. Це допомагає зменшити втрату даних за рахунок агресивного об'єднання та зниження частоти дискретизації, як це робиться в блоках кодувальника архітектури кодер-декодер.

Skip Connections має великий успіх, особливо в галузі медичної візуалізації, оскільки U-Net надала найсучасніші результати в області сегментації клітин для діагностики захворювань (рис. 1.3). На рис. 1.3 зображено архітектуру U-Net.

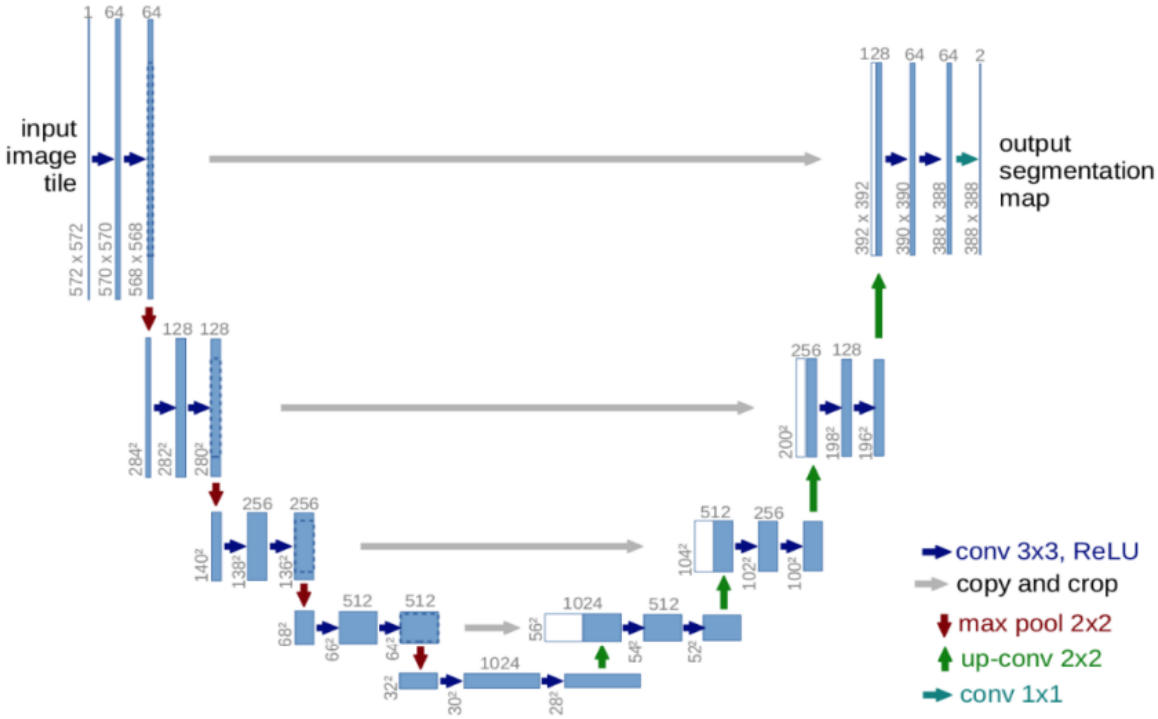


Рисунок 1.3 – Структура U-Net для сегментації зображень

Після U-Net трендом стала архітектура DeepLab, що надала найсучасніші результати з семантичної сегментації. [7]. DeepLab використовує складні згортки, замінивши прості операції об'єднання та запобігши значній втраті інформації при даунсемплінгу. Крім того, реалізовано багатомасштабне

вилучення ознак за допомогою Atrous Spatial Pyramid Pooling, що допомагає виявляти сегмент незалежно від його розміру [8].

Щоб відновити інформацію про межі, одну з найважливіших частин семантичної сегментації, а також сегментації екземплярів, використовується пов'язані умовні випадкові поля (CRF). Поєднання високої точності локалізації CRF і здатності розпізнавання згорткових нейронних мереж (CNN) призвело до того, що DeepLab надає високоточні карти сегментів, що значно перевершують такі методи, як FCN і SegNet.

1.2.3 Обгрунтування напрямку досліджень.

Із огляду відомих методів сегментації можна зробити висновок, що найбільш перспективним є метод на основі використання згорткових нейронних мереж. Саме тому у даній роботі будемо використовувати сегментацію зображень стенозу коронарних судин серця на основі згорткових нейронних мереж.

Було проведено пошук наукових джерел з відомими випадками застосування згорткових нейронних мереж до задачі сегментації зображень стенозу коронарних судин серця.

Попередні роботи продемонстрували високий потенціал нейромереж глибокого навчання для завдання локалізації коронарної артерії та стенозу за допомогою даних комп’ютерної томографічної коронарної ангіографії (КТКА). Наприклад, у роботі [9] змогли досягти оцінки F1 0,775 у завданні сегментації стенозу, тоді як у роботі [10] повідомляють про оцінку Dice 0,71 для сегментації артерії. В обох випадках використано архітектуру згорткової нейромережі на основі U-Net.

Однак, дуже мало робіт використовували зображення рентгенівської коронарної ангіографії (РКА), і більшість із цих робіт досягли успіху лише щодо сегментації тільки артерій [11] [12].

Існує дуже мало опублікованих методів виявлення стенозу, один з яких був розроблений у роботі [13] і називається DeepDiscern. DeepDiscern

використовує дві паралельні глибокі нейронні мережі — одну, яка виявляє сегменти артерій, а іншу, яка виявляє ураження на артеріях, — для створення діагностичної інформації високого рівня. Модель досягла оцінки F1 0,829 у завданні виявлення стенозу. Однак ця робота створює обмежувальні рамки як вихідні дані, які є набагато менш точними, ніж маски сегментації, і можуть бути більш неоднозначними.

Таким чином, огляд літературних джерел показав, що існує мало робіт по сегментації зображень стенозу коронарних судин серця саме по зображеннях рентгенівської коронарної ангіографії (РКА), а ті роботи, що є, показують недостатню точність сегментації. Тому метою данної роботи визначено саме підвищення точності сегментації стенозу коронарних судин серця по зображеннях рентгенівської коронарної ангіографії (РКА).

1.3 Обгрунтування вибору аналогів розробленого модуля сегментації зображень стенозу коронарних судин серця

Було обрано 2 аналоги для розробленого модуля сегментації зображень стенозу коронарних судин серця на основі згорткової нейронної мережі:

1) модель на основі YOLOv8, навчена у [14] на наборі для навчання стенозу ARCADE (n=1000) протягом 50 епох

2) модель на основі YOLOv8, названу YOLOv8n-seg і навчену в цій роботі.

Аналог 1 (Базова лінія 1) - було налаштовано YOLOv8 nano, добре відому модель виявлення об’єктів і сегментації зображення, на необробленому наборі даних стенозу ARCADE. Зокрема, ми точно налаштували YOLOv8n-seg, який було попередньо навчено на базі зображень COCO 2017 і чия архітектура ідентична YOLOv8, за винятком того, що голову мережі для виявлення об’єктів YOLOv8 замінено на голову мережі для сегментації.

Аналог 2 (Базова лінія 2) - модель на основі YOLOv8.

В якості базової лінії 2 ми налаштували YOLOv8n-seg на наборі для навчання стенозу ARCADE (n=1000) протягом 50 епох. Автори ARCADE зазначили, що їх базова модель YOLOv8 досягла оцінки F1 0,38 [14]. Трохи вища продуктивність команди ARCADE, ймовірно, пов’язана з використанням більшої тестової вибірки YOLO (порівняно з «нано» тестовою вибіркою, яку ми використовували), а також більш ретельним процесом налаштування гіперпараметрів.

1.4 Висновок до розділу 1

У розділі описано детальну постановку задачі сегментації зображень стенозу коронарних судин серця по зображеннях рентгенівській коронарній ангіографії (РКА). Було розглянуто як традиційні методи сегментації зображень, так і методи, що ґрунтуються на глибокому навчанні. Як найбільш перспективний і застосовний до даної задачі було обрано метод на основі згорткових нейромереж. Було оцінено застосовність до завдання сегментації зображень стенозу коронарних судин серця таких згорткових нейромереж як SegNet, U-Net, YOLO. Було показано, що з різних типів нейронних мереж найбільше підходить для вирішення поставленої задачі згорткова нейронна мережа Mask R-CNN. Крім цього, було обгрунтовано вибір аналогів розробленого модуля сегментації зображень стенозу коронарних судин серця.

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО МОДУЛЯ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ СТЕНОЗУ КОРОНАРНИХ СУДИН СЕРЦЯ НА ОСНОВІ ЗГОРТКОВОЇ НЕЙРОННОЇ МЕРЕЖІ

2.1 Обгрунтування вибору архітектури згорткової нейронної мережі для сегментації зображень стенозу коронарних судин серця

Для сегментації різних зображень (зокрема, зображень стенозу коронарних судин серця підходить багато різних архітектур згорткових нейронних мережі, у тому числі такі

- RetinaNet,
- U-Net;
- YOLO;
- Fast R-CNN,
- Mask R-CNN,

та інші.

У відомих роботах по сегментації зображень стенозу коронарних судин серця найчастіше використовуєтьтся YOLO та U-Net. Але найбільш перспективним представляється використання Mask R-CNN. Тому розглянемо і порівняємо тут архітектури згорткових нейромереж YOLO та Mask R-CNN.

2.1.1 Згорткова нейронна мережа YOLO.

YOLO — це одноетапний детектор об’єктів, і замість використання пропозицій регіонів для визначення місцезнаходження об’єктів, як у R-CNN, YOLO запускає єдину згорткову мережу на повному зображенні, щоб безпосередньо передбачити обмежувальні прямокутники та ймовірності класу в єдиному уніфікованому конвеєрі виявлення [15]. Зокрема, YOLO ділить зображення на сітку S × S. Потім кожна з цих комірок сітки передбачає кілька обмежувальних рамок і пов’язаних балів достовірності, а також умовні ймовірності класу для цієї комірки. Показник достовірності показує, наскільки

ймовірно, що в рамці є об’єкт, а також наскільки точні межі рамки, і визначається як:

$$\text{Conf} = Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

Під час логічного висновку оцінки довіри для кожного блоку множаться на ймовірності класу, що дає оцінки довіри класу для кожного блоку (рис. 2.1). Однією з важливих сильних сторін моделі YOLO є те, що, оскільки вона приймає та розглядає все зображення, вона може глобально міркувати про зображення та отримувати контекст поза межами регіону чи клітини інтересу.

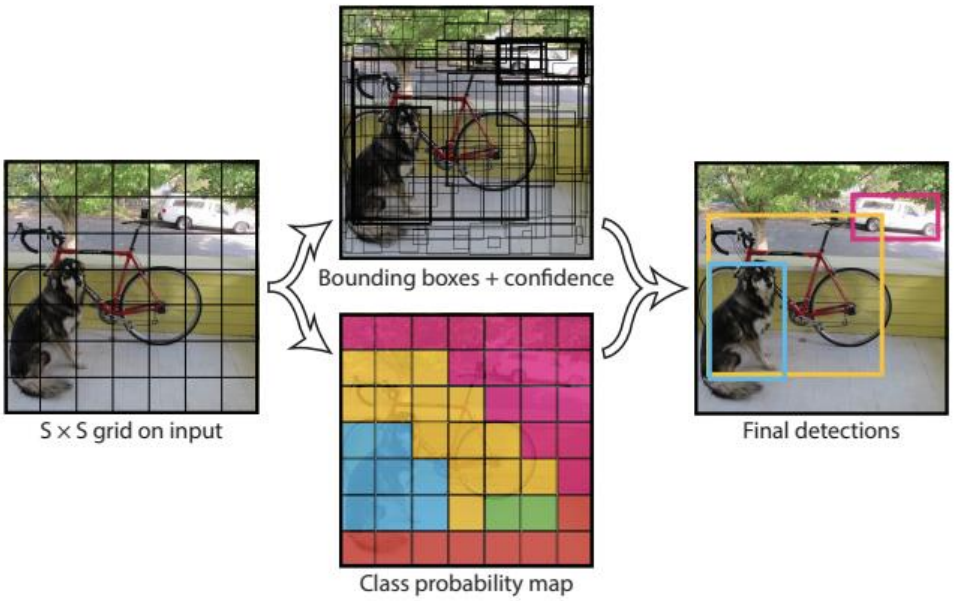


Рисунок 2.1 - Діаграма уніфікованої мережі виявлення YOLO, яка демонструє синтез балів достовірності обмежувальної рамки та ймовірностей класу для кожної з комірок сітки S × S.

Наприклад, сегментація за допомогою YOLOv8n-seg, архітектура YOLO трохи модифікована, щоб включити невелику повністю підключену мережу під назвою Proto, яка генерує маски сегментації [16].

2.1.2 Згорткова нейронна мережа Mask R-CNN.

Mask RCNN — глибока нейронна мережа, призначена для вирішення проблеми сегментації екземплярів у машинному навчанні або комп’ютерному зорі. Іншими словами, вона може розділяти різні об’єкти на зображенні чи відео. На вхід мережі надається зображення, а на виході мережа видає обмежувальні рамки об’єкта, класи та маски.

Існує два етапи роботи Mask RCNN. По-перше, вона генерує пропозиції щодо регіонів, де може бути об’єкт на полі вхідного зображення. По-друге, вона передбачає клас об’єкта, уточнює обмежувальну рамку та генерує маску на рівні пікселів об’єкта на основі пропозиції першого етапу. Обидва етапи пов'язані з хребтовою структурою (хребтом).

Хребет — це глибока нейронна мережа в стилі FPN (Feature Pyramid Network). Вона складається з шляху «знизу-до-гори», шляху «зверху-до-низу» і бічних з’єднань (див. рис. 2.2). Шляхом «знизу-до-гори» може бути будь-який ConvNet, зазвичай ResNet або VGG, який витягує ознаки з необроблених зображень. Шлях «зверху-до-низу» генерує пірамідну карту ознак, подібну за розміром до шляху «знизу-до-гори». Бічні зв’язки — це операції згортання та додавання між двома відповідними рівнями двох шляхів. FPN перевершує інші одиночні ConvNets головним чином через те, що він підтримує сильні семантичні характеристики в різних масштабах роздільної здатності.

Тепер розглянемо перший етап. Легка нейронна мережа, яка називається RPN (Region Proposal Network), сканує весь шлях «зверху-до-низу» FPN (далі – карта ознак) і пропонує області, які можуть містити об’єкти. Хоча сканування карти об’єктів є ефективним способом, нам потрібен метод прив’язування об’єктів до розташування необробленого зображення. У цьому допомагають прив’язки. Прив’язки — це набір прямокутників із заздалегідь визначеними розташуваннями та масштабами відносно зображень. Основні класи істинності (тільки двійковий файл об’єкта або фону, класифікований на цьому етапі) і обмежувальні прямокутники призначаються окремим

прив’язкам відповідно до певного значення IoU (перетин над об`єднанням). Оскільки прив’язки з різними масштабами прив’язуються до різних рівнів карти об’єктів, RPN використовує ці прив’язки, щоб визначити, де на карті об’єктів «повинен» отриматись об’єкт і який розмір його обмежувальної рамки. Тут ми можемо погодитися, що згортання, зменшення та підвищення дискретизації збережуть об’єкти в тих самих відносних розташуваннях, що й об’єкти на вхідному зображенні, і не зіпсують їх.

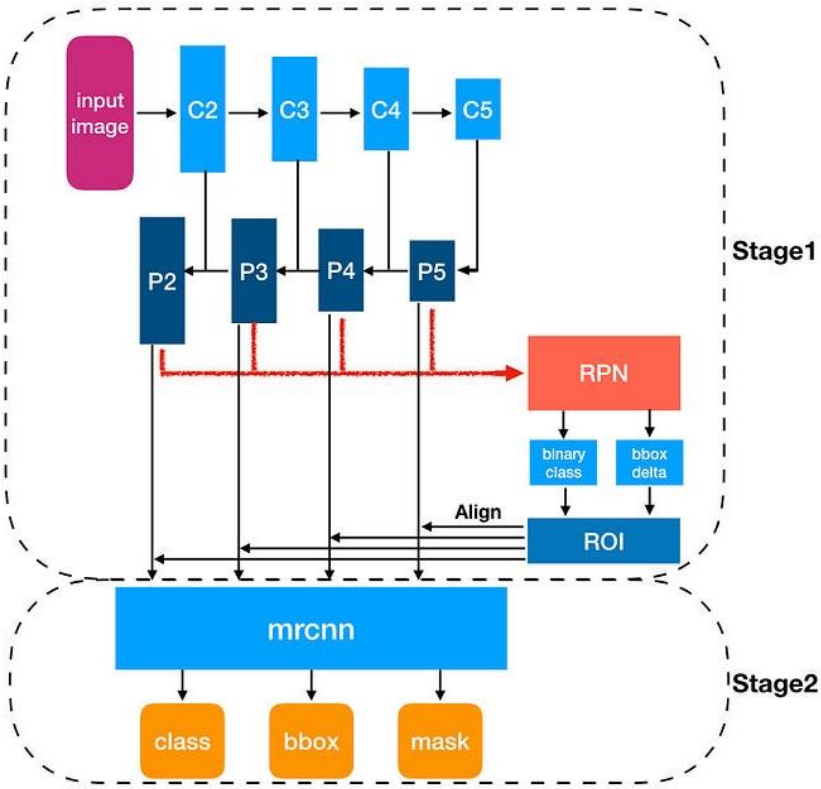


Рисунок 2.2 – Логічна структура нейронної мережі Mask RCNN

На другому етапі інша нейронна мережа бере запропоновані на першому етапі регіони та призначає їх кільком конкретним областям рівня карти ознак, сканує ці області та генерує класи об’єктів (багатокатегорійну класифікацію), обмежувальні рамки та маски. Процедура схожа на RPN. Відмінності полягають у тому, що без допомоги прив’язок другий етап використовував трюк під назвою ROIAlign, щоб знайти відповідні області карти об’єктів, і існує гілка, яка створює маски для кожного об’єкта на рівні пікселів.

Найцікавіше у Mask RCNN, це те, що можна було б змусити різні шари нейронної мережі вивчати функції з різними масштабами, так само як прив’язки та ROIAlign, замість того, щоб розглядати шари як чорний ящик.

Сітка об’єднання для області інтересу (ROI);

Використання мережі Feature Pyramid Network (FPN), яка розширює можливості Mask R-CNN, надаючи багатомасштабне представлення ознак, дозволяючи ефективне повторне використання функцій і вирішуючи коливання масштабу в об’єктах.

Що відрізняє Mask R-CNN, так це її здатність правильно сегментувати та ідентифікувати піксельні межі кожного елемента всередині зображення. Ця можливість дрібної сегментації досягається шляхом додавання додаткової гілки «голова маски» до моделі Faster R-CNN (рис. 2.3).

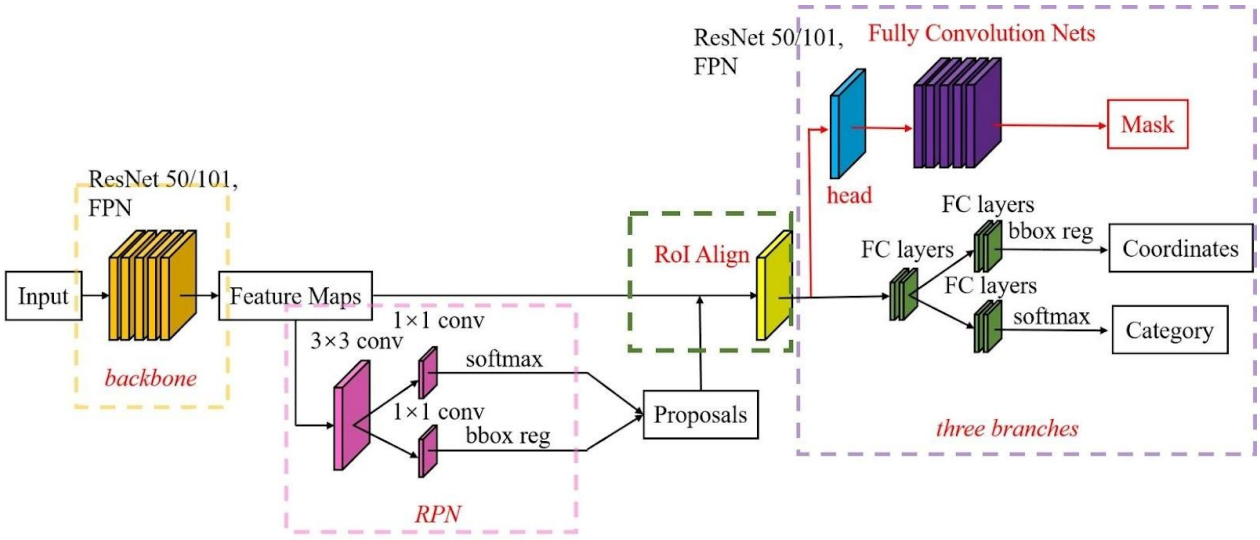


Рисунок 2.3 – Пошарова структура нейронної мережі Mask RCNN

Основною новинкою Mask R-CNN є її здатність робити попіксельну сегментацію екземплярів із розпізнаванням об’єктів. Це досягається шляхом включення додаткової гілки «mask head», яка створює точні маски сегментації для кожного розпізнаного елемента. Увімкнено чіткі межі на рівні пікселів, що забезпечує точну та ретельну сегментацію екземплярів.

ROIAlign і Feature Pyramid Network (FPN) — дві важливі інновації, вбудовані в Mask R-CNN. ROIAlign долає обмеження класичного підходу до об’єднання ROI, включаючи білінійну інтерполяцію в процес об’єднання. Це зменшує труднощі зі зміщенням і забезпечує правильне отримання просторової інформації з вхідної карти об’єктів, що забезпечує кращу точність сегментації, особливо для крихітних об’єктів.

Створюючи багатомасштабну піраміду функцій, FPN відіграє вирішальну роль у вилученні функцій. Ця піраміда об’єднує інформацію з багатьох масштабів, дозволяючи моделі отримати більш повне розуміння контексту об’єкта та забезпечуючи покращене розпізнавання об’єктів і сегментацію в широкому діапазоні розмірів об’єктів.

Конструкція Mask R-CNN базується на архітектурі Faster R-CNN із додаванням додаткової гілки «mask head», що забезпечує попіксельну сегментацію. Загальна архітектура складається з багатьох основних компонентів (рис. 2.3).

Магістральна мережа (хребет).

Основна мережа Mask R-CNN часто є попередньо навченою згортковою нейронною мережею, такою як ResNet або ResNeXt. Ця магістраль відповідає за обробку вхідного зображення та вилучення інформації високого рівня. Потім на вершині цієї магістральної мережі будується FPN, щоб сформувати піраміду функцій.

FPN призначені для вирішення проблеми роботи з елементами різних розмірів і масштабів на зображенні. Шляхом об’єднання функцій з кількох рівнів магістральної мережі проект FPN утворює багатомасштабну піраміду функцій. Ця піраміда має функції з різною просторовою роздільною здатністю, починаючи від функцій високої роздільної здатності з великою семантичною інформацією до функцій низької роздільної здатності з дрібнішими просторовими деталями.

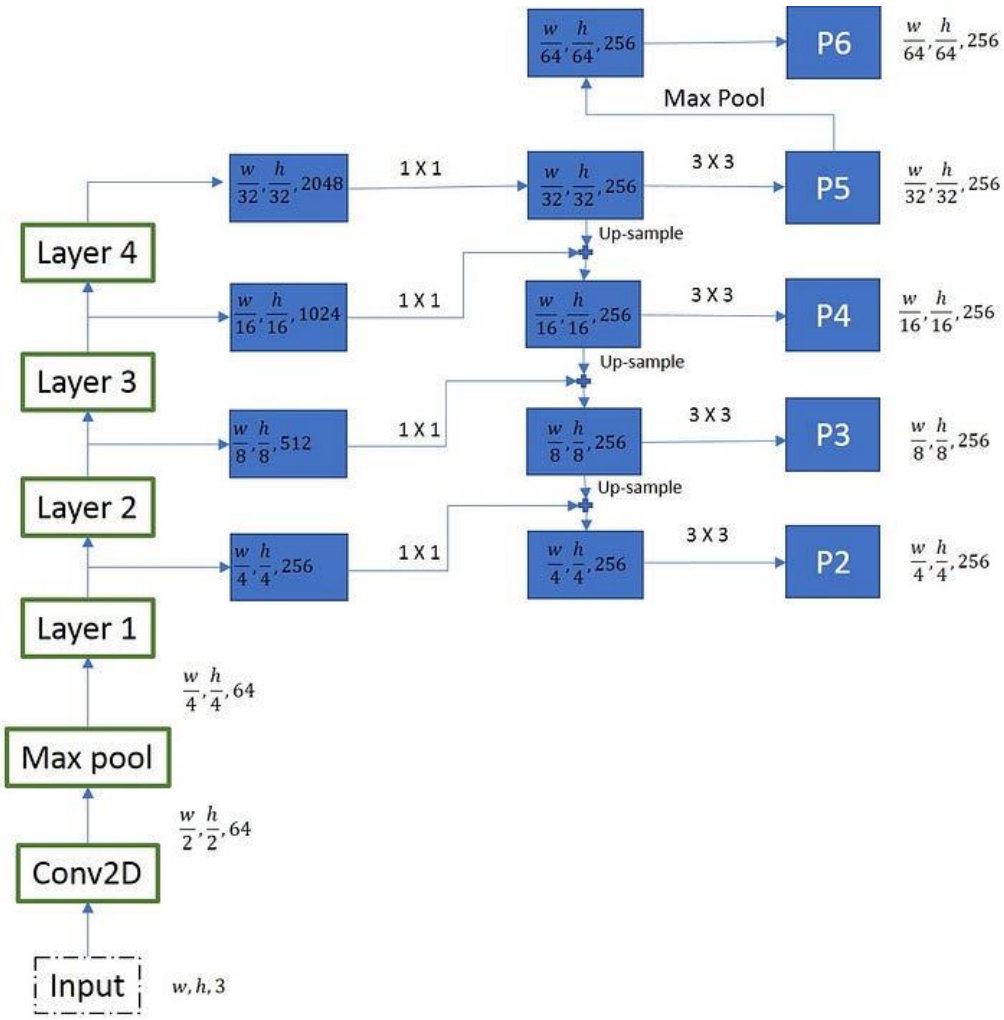


Рисунок 2.4 – Пірамідна мережа ознак (FPN, хребет)

У Mask R-CNN робота FPN складається з таких кроків:

1. Вилучення характеристик високого рівня: магістральна мережа отримує характеристики високого рівня з вхідного зображення.

2. Об’єднання функцій: щоб побудувати шлях зверху вниз, FPN з’єднує кілька рівнів магістральної мережі. Цей низхідний підхід інтегрує семантичну інформацію високого рівня з картами функцій нижчого рівня, що дозволяє моделі повторно використовувати функції різних розмірів.

3. Піраміда функцій: процес злиття формує багатомасштабну піраміду функцій, де кожен рівень піраміди представляє різну роздільну здатність ознак. Об’єкти з найвищою роздільною здатністю знаходяться у верхній частині піраміди, а об’єкти з найнижчою – у нижній частині.

Mask R-CNN може успішно обробляти об'єкти різного розміру завдяки піраміді функцій, утвореній FPN. Завдяки багатомасштабному представленню модель може збирати контекстну інформацію та надійно розпізнавати елементи в кількох масштабах зображення.

Мережа регіональних пропозицій (RPN) – рис. 2.5.

RPN відповідає за створення регіональних пропозицій або потенційних обмежувальних рамок, які можуть містити об’єкти на зображенні. Він працює на карті функцій магістральної мережі та пропонує потенційні цікаві місця.

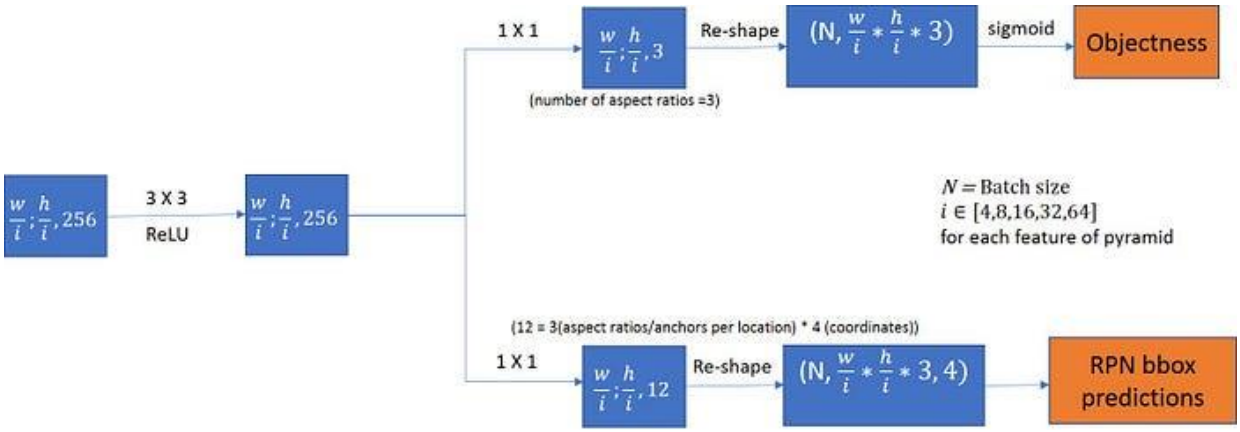


Рисунок 2.5 – Приклад RPN мережі (Region Proposal Network)

ROIAlign.

Рівень ROIAlign (Region of Interest Align) вставляється після того, як RPN створить пропозиції регіону. Цей крок допомагає подолати проблему неузгодженості в об’єднанні ROI.

ROIAlign має вирішальне значення для отримання ознак із вхідної карти ознак точно для кожної пропозиції області, гарантуючи ідеальну сегментацію по пікселях у завданнях сегментації екземплярів.

Основною метою ROIAlign є вирівнювання об’єктів у межах цікавої області (ROI) з просторовою сіткою вихідної карти об’єктів. Це вирівнювання має вирішальне значення для уникнення втрати інформації, яка може статися, коли просторові координати досліджуваної області квантуються до найближчого цілого числа (як у об’єднанні досліджуваної області).

Процес ROIAlign включає такі кроки:

1. Карта вхідних ознак: Карта вхідних ознак, яка зазвичай збирається з магістральної мережі, є першим кроком у процесі. Ця карта ознак пропонує повну семантичну інформацію про все зображення.

2. Регіональні пропозиції: RPN створює регіональні пропозиції (кандидати, обмежувальні рамки), які можуть містити цікаві елементи всередині зображення.

3. Поділ на сітки: кожна пропозиція області розбивається на певну кількість просторових ящиків або сіток однакового розміру. Ці сітки використовуються для вилучення об’єктів, пов’язаних із регіоном інтересу, із вхідної карти об’єктів.

4. Білінійна інтерполяція: на відміну від об’єднання ROI, яке квантує просторові координати сіток до найближчого цілого числа, ROIAlign обчислює внески об’єднання для кожної сітки за допомогою білінійної інтерполяції. Ця інтерполяція гарантує точніше вирівнювання функцій у ROI.

5. Вихідні об’єкти: Об’єкти з вхідної карти об’єктів, вирівняні з кожною сіткою на вихідній карті об’єктів, служать репрезентативними об’єктами для кожної пропозиції території. Дрібнозерниста просторова інформація фіксується цими вирівняними функціями, що є критично важливим для ефективної сегментації.

ROIAlign значно підвищує точність виділення ознак для кожної запропонованої області, застосовуючи білінійну інтерполяцію під час фази об’єднання, зменшуючи проблеми з розбіжністю.

Завдяки цьому ідеальному вирівнюванню Mask R-CNN може створювати більш точні маски сегментації, що особливо корисно для крихітних об’єктів або областей, де необхідно зберегти дрібні особливості. Як наслідок, ROIAlign додає виняткову продуктивність Mask R-CNN у завданнях сегментації екземплярів.

Голова маски.

Mask Head — це нова гілка в Mask R-CNN, яка відповідає за створення масок сегментації для кожної пропозиції області. Вирівняні функції, отримані ROIAlign, використовуються керівником для прогнозування бінарної маски для кожного об’єкта, окреслюючи піксельні межі екземплярів. Як правило, голова маски складається з кількох згорткових шарів, за якими йдуть шари підвищення дискретизації (деконволюція або транспоновані згортки) – рис. 2.6.

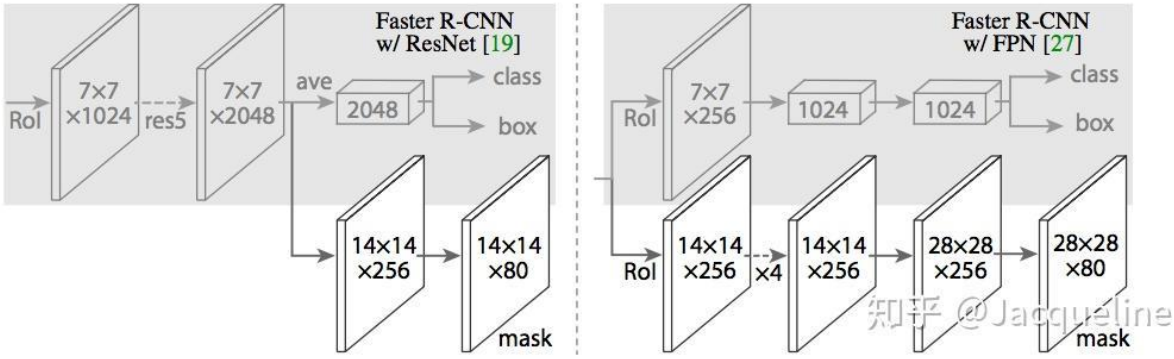


Рисунок 2.6 – Структура голови маски (Mask Head)

Модель спільно оптимізується під час навчання з використанням суміші втрат класифікації, втрат регресії обмежувальної рамки та втрат сегментації маски. Це дає змогу моделі навчитися розпізнавати об’єкти, а також уточнювати їх обмежувальні рамки та створювати точні маски сегментації.

Mask R-CNN чудово працює в різних областях, що робить її ефективною моделлю для широкого спектру програм комп’ютерного зору, таких як розпізнавання об’єктів, сегментація екземплярів, сегментація кількох об’єктів і складна обробка сцен.

Однак у Mask R-CNN є кілька недоліків, які слід враховувати:

1. Сегментація дрібних об’єктів: через недостатню інформацію про пікселі, Mask R-CNN може не відокремлювати дуже дрібні об’єкти.

2. Обчислювальна складність: Навчання та логічні висновки можуть вимагати обчислень, потребуючи значних ресурсів, особливо для зображень із високою роздільною здатністю або великих наборів даних.

3. Вимоги до даних: Навчання Mask R-CNN потребує величезної кількості анотованих даних, отримання яких може зайняти багато часу та дорого коштувати.

4. Обмежене узагальнення: здатність моделі до узагальнення категорій елементів, які зустрічалися раніше, обмежена, особливо коли дані розріджені.

Mask R-CNN поєднує ідентифікацію об’єктів і сегментацію екземплярів, дозволяючи розпізнавати об’єкти, а також точно окреслювати їхні межі на рівні пікселів. Mask R-CNN забезпечує високу продуктивність і точність завдяки поєднанню функції пірамідної мережі (FPN) із вирівнюванням регіону інтересу (ROIAlign).

Проведений аналіз показав, що згорткова нейромережа Mask R-CNN має більше переваг, ніж YOLO, і краще підходить функціонально до задачі сегментації зображень, тому ми обрали для реалізації завдання сегментації зображень стенозу коронарних судин серця саме згорткову нейронну мережу Mask R-CNN.

2.2 Розробка методу сегментації зображень стенозу коронарних судин серця на основі згорткової нейронної мережі Mask R-CNN

При огляді науково-технічної літератури було знайдено декілька робіт, в яких вирішувалось аналогічне завдання, тобто завдання сегментації стенозу, зокрема [17], [18] та [19]. Найкращим є метод сегментації зображень стенозу [19], в якому отримано результат на тестовому наборі показник F1 0,34. Однак багато з цих моделей використовують архітектуру на основі згорткової нейромережі YOLO. У цій роботі досліджується використання згорткової

нейромережі Mask R-CNN для сегментації стенозу, про що не було знайдено інформації у відомих наукових роботах.

Скоріш за все попіксельна сегментація стенозних артерій забезпечить більшу клінічну цінність завдяки більшій точності, тому було обрано попіксельну сегментацію стенозу як засіб досягнення мети. Інші цілі DeepDiscern (виявлення об’єктів проти сегментації екземплярів) у поєднанні з тим фактом, що ані їх модель, ані набір даних не є загальнодоступними, означає, що ми не можемо безпосередньо порівняти наші результати з їхніми показниками оцінки.

На додаток до базової моделі YOLO ми пропонуємо два методи на основі Mask R-CNN. Спочатку ми пропонуємо просто точно налаштувати попередньо підготовлену Mask RCNN ResNet-50 FPN [20] на наборі даних стенозу ARCADE. Ми називаємо цю модель StenSeg-base.

Mask R-CNN — це двоступеневий детектор об’єктів, який використовує початкову магістраль ResNet з наступною мережею регіональних пропозицій і рівнем об’єднання ROI для визначення регіонів карти функцій для виконання прогнозів. Роблячи прогнози та обчислюючи втрати, Mask R-CNN розширює Faster R-CNN, додаючи невелику мережеву гілку, яка прогнозує маску сегментації для кожного регіону інтересу (RoI) паралельно з прогнозом обмежувальної рамки, який спочатку обчислює Faster R-CNN [20]. Зокрема, ми вважаємо, що Mask R-CNN перевершує архітектуру YOLO через її вищезгадану здатність генерувати пропозиції щодо регіонів, яка помітно відсутня в YOLO, звідки й назва [15]. Хоча поділ зображень на комірки та швидке створення класифікацій добре підходять для обробки в режимі реального часу, коли у випадку сегментації стенозу можна дозволити собі більше обчислень і часу, ми вважаємо, що пропозиції складніших регіонів мають перевагу.

Крім того, Mask R-CNN реалізує підхід RoIAlign, який допомагає відокремити процес передбачення маски від класифікації. Крім того, у такому завданні, як сегментація зображення, точне відображення масок на дрібних

об’єктах, що відображаються на оригінальному зображенні, яке забезпечує RoIAlign, є критичним.

Зокрема, RoIAlign не квантує регіони інтересу (ROI) з плаваючою комою в дискретну карту функцій, що може призвести до помилки. Натомість RoI Align зберігає точність із плаваючою комою, а потім виконує білінійну інтерполяцію, щоб «заповнити» дискретні поля максимальним або середнім значенням [20]. З рис. 2.7,

$$f(x,y) = \sum_{i,j=1}^{2} f_{i,j} \max(0, 1-|x-x_i|) \max(0, 1-|y-y_i|)$$

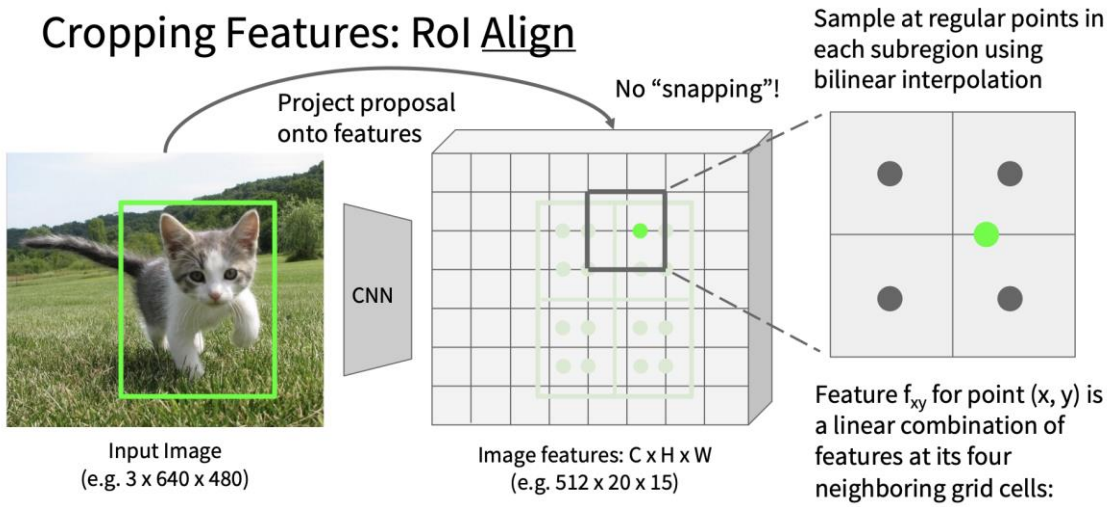


Рисунок 2.7 - Ілюстрація методології RoIAlign

Розмір вхідного зображення для архітектури Mask R-CNN становить 224x224, тому перед навчанням моделі ми змінюємо розмір наших зображень із 512x512 до 224x224. Було виявивлено, що ця проста процедура навчання призводить до значного переповнення протягом 50 епох, але все ще не перевершує найкращий аналог ARCADE на валідаційному наборі. Тому було запропоновано модернізацію процесу навчання на основі процедури аугментації даних.

2.3 Аугментація даних для навчання згорткової нейронної мережі Mask R-CNN

Набір даних ARCADE [15] складається з двох різних наборів із 1500 навчальних зображень ішемічної хвороби серця (IXC), один з яких позначено масками сегментації стенозу (набір даних «Stenosis»), а інший — лише масками сегментації різних гілок коронарних артерій (набір даних «Syntax»).

Наша базова модель StenSeg-base використовувала лише набір даних «Stenosis», але, як помітила команда SSASS (SSASS: Semi-supervised approach for stenosis segmentation [19]), хоча набір синтаксичних даних явно не позначено для стенозу, стенози все таки присутні в кожному прикладі набору Syntax синтаксичних даних. Таким чином, ми також використовуємо набір синтаксичних даних, генеруючи «псевдомітки» стенотичних областей, запускаючи виведення (інференс) нейромережі із StenSeg-base (див. рис. 2.8), дотримуючись подібної процедури, як автори моделі SSASS [19].

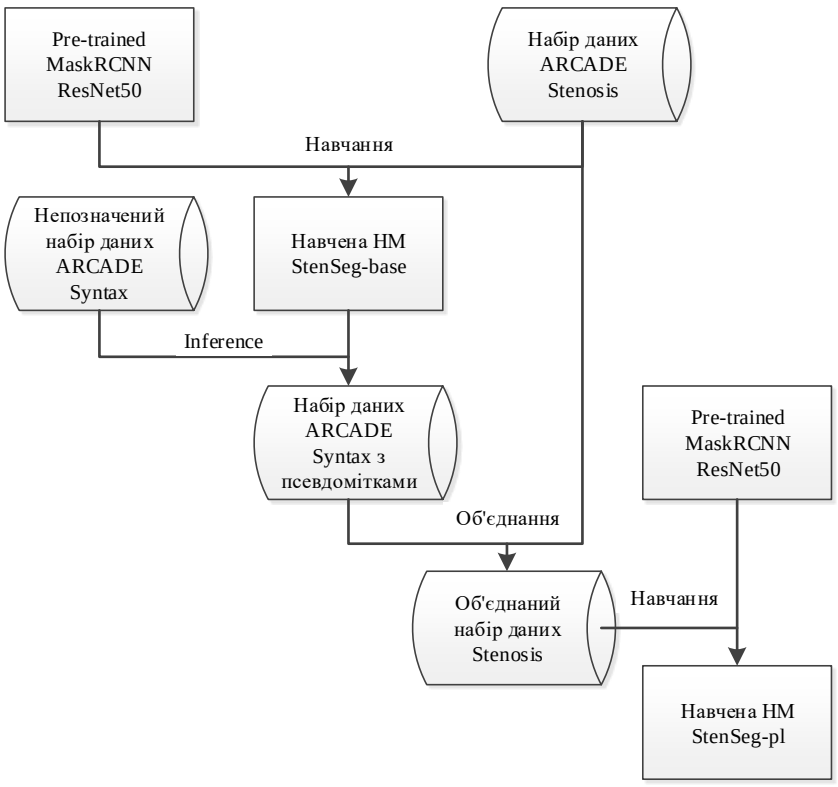


Рисунок 2.8 - Схема запропонованої процедури аугментації даних.

Синтаксична маска та комбінації зображень із виведення (інференсу) нейромережі потім об’єднуються з початковим набором даних стенозу. Потім ми спільно навчаємо іншу модель Mask R-CNN на комбінованому наборі синтаксичних даних стенозу та псевдо-міток і називаємо цю модель StenSeg-pl (див. рис. 2.8).

2.4 Розробка алгоритму роботи програмного модуля сегментації зображень стенозу коронарних судин серця на основі згорткової нейронної мережі

Відповідно до мети роботи та постановки задачі було розроблено алгоритм програмного модуля сегментації зображень стенозу коронарних судин серця на основі згорткової нейронної мережі Mask R-CNN, представлений на рис. 2.9.

Першим кроком в програмному модулі сегментації зображень стенозу коронарних судин серця буде завантаження необхідних бібліотек (блок 1), а другим кроком буде завантаження набору даних ARCADE зображень стенозу коронарних судин серця (блок 2).

Далі переходимо до завантаження попередньо натренованої моделі згорткової нейронної мережі MaskRCNN ResNet50 (блок 3).

Потім відбувається навчання нейронної мережі MaskRCNN ResNet50 на розмічеій частині набору ARCADE – ARCADE Stenosis протяго 50 епох. Така кількість епох обрана як оптимальна після кількох спроб навчити нейромережу на цьому наборі даних (блоки 4 і 5). Далі ми зберігаємо натреновану нейронну мережу MaskRCNN ResNet50 (блок 6) під іменем StenSeg-base для подальшого використання.

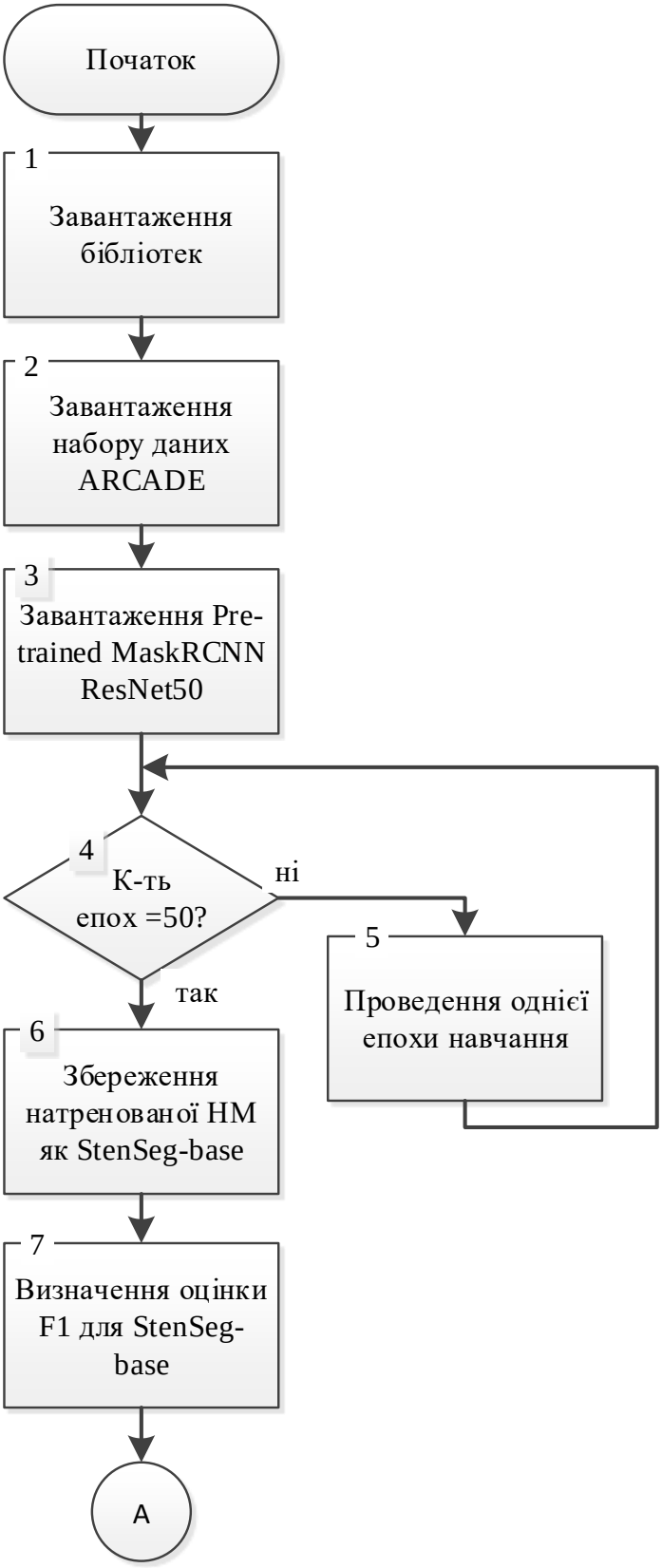


Рисунок 2.9 – Схема алгоритму роботи програмного модуля сегментації зображень стенозу коронарних судин серця

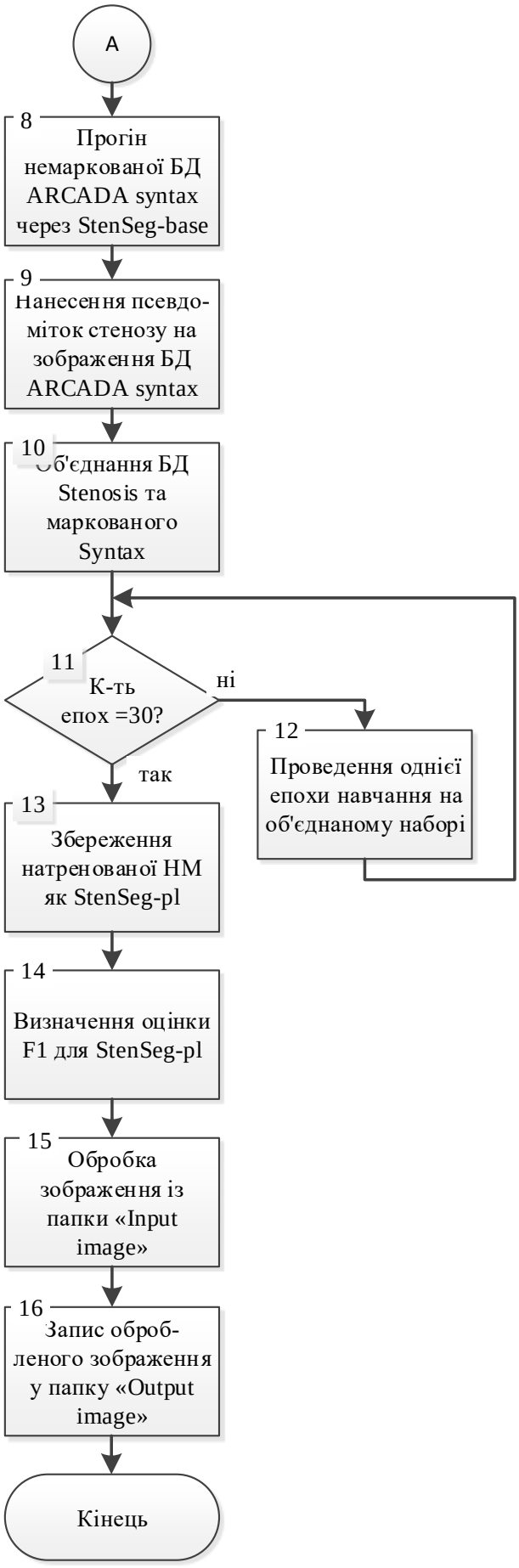


Рисунок 2.9 – (Продовження)

Потім переходимо до визначення оцінки F1 для StenSeg-base (блок 7). Далі ми проганяємо зображення немаркованої частини БД ARCADA Syntax через StenSeg-base (блок 8) з тим, щоб вона визначила ділянки стенозу на цих немаркованих зображеннях.

Ці визначенні ділянки стенозу і будуть псевдомітками стенозу, які наносяться на зображення (блок 9) із частини БД ARCADA Syntax і вони становляться вже розміченними (маркованими). Термін «псевдо» застосований тому, що ділянки стенозу визначено не лікарем, а навченою нейронною мережею.

Після цього відбувається об'єднання двох частин БД ARCADA : Stenosis та маркованої Syntax (блок 10) в одну загальну БД, яка тепер є розширеною (аугментованою), а значить на її основі вже можна більш якісно навчити попередньо натреновану модель згорткової нейронної мережі MaskRCNN ResNet50, що і відбувається далі у блоках 11 і 12 протягом 30 епох. Така кількість епох була обрана як оптимальна після кількох спроб навчити нейромережу на цьому наборі даних.

Далі ми зберігаємо натреновану нейронну мережу MaskRCNN ResNet50 (блок 13) під іменем StenSeg-pl для подальшого використання.

Потім переходимо до визначення оцінки F1 для StenSeg- pl (блок 14). Результати оцінювання метрик для аналога і розробленого модуля наведено у табл. 4.1.

Після цього програма обробляє зображення, розташоване у папці «Input image» (блок 15), наносить на зображення жовтий прямокутник, що обмежує область стенозу (якщо вона там є) і зафарбовує зеленим ділянку судини із стенозом. Оброблене зображення із жовтим прямокутником і зафарбованою ділянкою стенозу записується у папку «Output image» (блок 16).

2.5 Висновок до розділу 2

У розділі було обгрунтовано вибір архітектури згорткової нейронної мережі для сегментації зображень стенозу коронарних судин серця. Із двох проаналізованих архітектур: YOLO та Mask R-CNN, було обрано згорткову нейромережу Mask R-CNN для використання у сегментації зображень стенозу коронарних судин серця як найбільш перспективну. Розроблено метод сегментації зображень стенозу коронарних судин серця на основі Mask R-CNN, що реалізує підхід RoIAlign, який допомагає відокремити процес передбачення маски від класифікації. Було розроблено процес аугментації даних для навчання згорткової нейронної мережі Mask R-CNN. Також було розроблено алгоритм роботи програмного модуля сегментації зображень стенозу коронарних судин серця на основі згорткової нейронної мережі.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МОДУЛЯ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ СТЕНОЗУ КОРОНАРНИХ СУДИН СЕРЦЯ НА ОСНОВІ ЗГОРТКОВОЇ НЕЙРОННОЇ МЕРЕЖІ

3.1 Обґрунтування вибору мови програмування та спеціалізованих бібліотек

Для реалізації програмного модуля сегментації зображень стенозу коронарних судин серця на основі згорткової нейронної мережі Mask R-CNN було обрано мову програмування Python [21].

Для створення коду навчання, оцінки та візуалізації для розробки Mask RCNN використовувалася бібліотека Pytorch [22], а саме - бібліотека Pytorch Image Models (timm),

PyTorch — відкрита бібліотека машинного навчання на основі бібліотеки Torch, що застосовується для задач комп'ютерного зору та обробки природної мови. Розробляє її переважно група дослідження штучного інтелекту компанії Facebook. Вона є вільною з відкритим програмним кодом, що випускають під ліцензією Modified BSD. PyTorch також має зовнішній інтерфейс і для C++.

PyTorch забезпечує дві високорівневі функціональності:

- Тензорні обчислення (як NumPy) із значним прискоренням через графічні процесори (ГП);
- Глибокі нейронні мережі, побудовані на системі автоматичного диференціювання.

Pytorch Image Models `timm` — це бібліотека глибокого навчання, створена Россом Вайтманом і являє собою набір моделей комп’ютерного зору SOTA, шарів, утиліт, оптимізаторів, планувальників, завантажувачів даних, доповнень, а також сценаріїв навчання/перевірки з можливістю відтворення результатів навчання ImageNet.

Також використовується бібліотека Torchvision. Пакет Torchvision складається з популярних наборів даних, архітектур моделей і типових перетворень зображень для комп’ютерного зору.

Оскільки штучні нейронні мережі використовують дуже багато операцій множення векторів та матриць, то у цій роботі стане у пригоді бібліотека NumPy [23]. NumPy (скорочено від Numerical Python) — це бібліотека з відкритим кодом для мови Python. Вона має такі можливості:

- підтримка багатовимірних масивів (включаючи матриці);
- підтримка математичних функцій високого рівня, які призначені для роботи з багатовимірними масивами.

Оскільки при проектуванні програмного модуля сегментації зображень стенозу коронарних судин серця на основі згорткової нейронної мережі ми виконуємо побудову прямокутних областей та зафарбовування масок стенозу, то нам знадобитьсч бібліотека Matplotlib [24].

Matplotlib - комплексна бібліотека для створення статичних, інтерактивних та анімованих візуалізацій на мові Python. Matplotlib має такі функціональні спроможності:

- Створення якісних сюжетів для публікацій.
- Створення інтерактивних фігур, котрі можна масштабувати, оновлювати, панорамувати.
- Налаштовування візуального макету і стилю.
- Експорт у велику кількість форматів файлів.
- Інтегрування у JupyterLab і у графічний користувацький інтерфейс.
- Використання широкого набору зовнішніх пакетів, побудованих на основі Matplotlib.

Matplotlib може створювати публікації цифрової якості у різних форматах друкованих копій та інтерактивних середовищах на різноманітних платформах. Matplotlib може бути використаний у сценаріях мови Python, серверах веб-додатків, оболонках Python/IPython та у різноманітних наборах інструментів графічного користувацького інтерфейсу.

3.2 Опис набору даних зображень стенозу коронарних судин для навчання та тестування модуля

Наразі найдоступнішим набором даних і еталонним показником для сегментації стенозу коронарних артерій із зображень РКА є публічно опублікований набір даних Automatic Region Coronary Disease Diagnostics using X-ray angiography images (ARCADE), який містить 1500 зображень РКА з анотаціями сегментації стенозних областей [15]. Анотації надаються у форматах YOLO та COCO, двох усталених форматах для представлення масок сегментації. Під час публікації набору даних автори також навчили модель YOLOv8 на наборі даних про стеноз як загальнодоступну базову лінію та змогли досягти оцінки F1 0,38.

Ряд інших моделей використовували набір даних ARCADE. Деякі, наприклад [25], використали частину набору даних ARCADE, яка містить мітки для різних сегментів коронарних артерій, а не для стенозу.

Інший опублікований набір даних для виявлення стенозу на зображеннях РКА [26] містить 8325 зображень від 100 пацієнтів, які пройшли РКА візуалізацію в науково-дослідному медичному закладі серцево-судинних захворювань. Хоча цей набір даних містить лише анотації обмежувальної рамки стенозних областей, що робить результати в [26] непрямими для порівняння з нашими, він є ще корисний як додаткове джерело зображень РКА, на яких наявні стенози. Ми вирішили включити цей набір даних у наш навчальний набір, як описано в Розділі 2.3, оскільки ми відчували, що це може бути корисним для узагальнення наших моделей.

Вивчення інших наборів даних.

Крім того, для подальшого розширення нашого набору даних і покращення можливості узагальнення наших моделей за межами набору даних ARCADE ми також досліджуємо включення іншого набору ангіографічних даних для виявлення стенозу, представленого в [26].

Було вирішено не тренуватися безпосередньо з анотаціями в цьому наборі даних, оскільки цей набір даних містить лише анотації обмежувальної рамки для стенозованих областей, а не маски сегментації, а наївне перетворення міток обмежувальної рамки на маски, позначаючи всі пікселі всередині поля як стенотичні, створить значну кількість шуму (багато пікселів, навіть не всередині судини, будуть помилково позначені як стенозовані). Замість цього було вирішено створити власні мітки, використовуючи ту саму процедуру псевдоміток, як описано в Розділі 2.3.

Набір даних і ознаки.

Основний використаний набір даних, набір даних ARCADE, — це набір зображень ХСА, позначений медичними експертами, що охоплює 1500 пацієнтів із середнім віком 60 років. Зображення були отримані від когорти дослідження в Науково-дослідному інституті кардіології та внутрішніх хвороб (Алмати, Казахстан). Зауважте, що хоча кожен пацієнт генерував 60-120 кадрів, 0-12 кадрів було вибрано на основі оптимального контрастного заповнення артерій і мінімальної розмитості серед інших властивостей для максимальної ефективності навчання. У прямому клінічному конвеєрі ця попередня обробка була б необхідною для встановлення прогнозів у реальному часі від машини для візуалізації ХСА [15]. Набір даних ARCADE складається з двох окремих частин: набору даних стенозу та набору синтаксичних даних. Набір даних про стеноз – це той, який ми в основному досліджували та тренували наші моделі. Він складається з 1000 навчальних зображень, 200 перевірочних зображень і 300 тестових зображень, усі з пов’язаною обмежувальною рамкою та анотаціями маски сегментації у форматі YOLO. Другий набір даних, синтаксичний набір, складається із зображень, позначених масками сегментації різних гілок/областей коронарних артерій. Цей набір даних також отримано від пацієнтів з ішемічною хворобою серця. Синтаксичний набір даних має такий самий розмір, як і набір даних стенозу, з 1000 тренувальних зображень, 200 валідаційних зображень, і 300

тестових зображень. Усі зображення в цих двох наборах даних мають розміри 512x512.

Щоб підготувати набір даних для навчання Mask R-CNN, попередньо було оброблено анотації YOLO, які складалися з багатокутних координат, у бінарні маски сегментації. Другий набір даних, включений у [26] мав 8325 зображень РКА. Хоча анотації обмежувальних рамок стенотичних областей є доступними, було використано необроблені зображення лише для конвеєра псевдоміток.

3.3 Програмна реалізація модуля сегментації зображень стенозу коронарних судин серця

Першим кроком в програмному модулі сегментації зображень стенозу коронарних судин серця на основі згорткової нейронної мережі Mask R-CNN буде завантаження бібліотек:

```
import timm
import torch
import torchvision
from torch.utils.data import DataLoader
from torchvision.transforms import functional as F
from torchvision.models.detection import MaskRCNN
from torchvision.models.detection.mask_rcnn import MaskRCNNPredictor
from torchvision.models.detection.backbone_utils import BackboneWithFPN
from torchvision.datasets.coco import CocoDetection
from torchvision.transforms import ToTensor, Normalize, Compose
import torchvision.transforms as tf
import matplotlib.pyplot as plt
from torchvision.transforms import v2
from torchvision.utils import draw_bounding_boxes, draw_segmentation_masks
from torchvision import tv_tensors
from torchvision.transforms.v2 import functional as F
import torch.nn as nn
from torchvision.datasets import wrap_dataset_for_transforms_v2
from torchvision.transforms.functional import pil_to_tensor
from torchvision.models.detection.mask_rcnn import MaskRCNN_ResNet50_FPN_Weights
```

```
import os
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
import matplotlib.patches as patches
import matplotlib.patheffects as PathEffects
import cv2
from torch.utils.data import Dataset
import warnings
```

Далі потрібно завантажити набори даних:

```
DATA_DIR = '/srv/submission/stenosis/'
SYNTAX_DIR = '/srv/submission/syntax/'
DANILOV_DIR = '/srv/danilov/dataset/'
```

Потім завантажуємо модель нейронної мережі `MaskRCNN_ResNet50_FPN_Weights` і проводимо її навчання

```
# Load model
model = torchvision.models.detection.maskrcnn_resnet50_fpn(weights=MaskRCNN_ResNet50_FPN_Weights.DEFAULT)
model.to(device)
params = [p for p in model.parameters() if p.requires_grad]
optimizer = torch.optim.SGD(params, lr=0.005, momentum=0.9, weight_decay=0.0005)

num_epochs = 30  # Define the number of epochs
save_every = 250   # Save checkpoint every certain amount of iterations

start_epoch = 0 # Check this!
start_iteration = 0
checkpoint_path = 'checkpoints/checkpoint_batch4_epoch3_iter750_pseudo_other.pth' # Modify this to '' if no checkpoint
if checkpoint_path:
    model, optimizer, start_epoch, start_iteration = load_checkpoint(model, optimizer, checkpoint_path)
```

Із фрагменту коду вище видно, що використовувався метод навчання (оптимізатор) «Стохастичний градієнтний спуск» - Stochastic Gradient Descent (SGD). Швидкість навчання була обрана 0,005, імпульс = 0,9 і розпад ваги 5×10^{-4}. Було задано 30 епох навчання. Навчання проводилось із збереженням чекпоінту через кожні 250 ітерацій.

Після кожної епохи навчання програма підраховує такі параметри: TP - вірно позитивний результат (True Positive), FP - хибно позитивний результат (False Positive), FN - хибно негативний результат (False Negative). Precision – влучність, Recall – повнота, показник F1:

```
    # Calculate True Positives (TP), False Positives (FP), and False Negatives (FN)
    tp = torch.logical_and(master_mask, target_mask).sum().item()
    fp = torch.logical_and(master_mask, torch.logical_not(target_mask)).sum().item()
    fn = torch.logical_and(torch.logical_not(master_mask), target_mask).sum().item()

    return tp, fp, fn


def compute_f1(tp, fp, fn):
    precision = float(tp / (tp + fp + 1e-8))  # Adding a small epsilon to avoid division by zero
    recall = float(tp / (tp + fn + 1e-8))  # Adding a small epsilon to avoid division by zero
    f1_score = 2 * (precision * recall) / (precision + recall + 1e-8)  # Adding a small epsilon to avoid division by zero
    return f1_score
```

Потім, коли мережа вже навчена, можемо перевірити будь-яке зображення РКА на наявність стенозу коронарних судин серця.

```
for image_name in tqdm(os.listdir(image_dir)):
    image_path = image_dir + image_name
    image = Image.open(image_path)
    image = np.asarray(image)
    adapt_eq_img = exposure.equalize_adapthist(image, clip_limit=0.03)
    frangi_adapt = frangi(adapt_eq_img)
    frangi_adapt_diff = adapt_eq_img - frangi_adapt
    frangi_adapt_diff = (frangi_adapt_diff * 255).astype(np.uint8)
    processed_img = Image.fromarray(frangi_adapt_diff)
    processed_img.save(out_dir + image_name)
```

Основна метрика якості роботи модуля сегментації зображень стенозу коронарних судин серця - це показник F1. Результати оцінювання метрики для аналога і розробленого модуля наведено у табл. 4.1.

3.4 Висновок до розділу 3

У розділі було обґрунтовано вибір мови програмування Python та спеціалізованих бібліотек Pytorch, Pytorch Image Models, NumPy, Torchvision та Matplotlib для програмної реалізації модуля сегментації зображень стенозу коронарних судин серця на основі згорткової нейронної мережі. Проведено аналіз набору ARCADE для навчання та тестування модуля, який містить приблизно 3 000 зображень, з яких 2000 навчальних, 400 валідаційних та 600 тестових. Описано основні етапи програмної реалізація та функціонування модуля сегментації зображень стенозу коронарних судин серця на основі згорткової нейронної мережі, що складається з завантаження бібліотек, завантаження набору даних із зображеннями, запуску роботи моделі аналогу і запропонованої нейромережі, візуалізації результатів роботи, підрахунку показників якості (метрик) аналога та розробленої нейромережі.

4 ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ ПРОГРАМНОГО МОДУЛЯ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ СТЕНОЗУ КОРОНАРНИХ СУДИН СЕРЦЯ НА ОСНОВІ ЗГОРТКОВОЇ НЕЙРОННОЇ МЕРЕЖІ

Розроблений програмний модуль є консольною програмою без інтерфейсу. У програмі передбачено наявність папок, в які записані зображення рентгенівської коронарної ангіографії. Причому є три папки: в одній зберігаються зображення навчальної множини, в другій - зображення валідаційної множини і в третій - зображення тестової множини. Після запуску програми, створюється модель нейронної мережі, яка навчається на зображеннях із папки з навчальною множиною, після кожної епохи навчання перевіряються параметри продуктивності (похибка та показник F1) на навчальній множині і валідаційній множині. Після закінчення навчання визначаються параметри продуктивності на тестовій множині.

Для аналізу стенозу на конкретному зображенні в програмі прописується шлях до цього зображення, яке має зберігатися у папці «Input image» і після запуску програми воно аналізується, на цьому зображенні малюється рамка жовтого кольору з передбачуваною областю стенозу і зеленим кольором зафарбовується маска стенозованої ділянки судини. Це оброблене зображення з намальованими рамками і маскуванням записується у папку вихідних зображень «Output image», де ми потім можемо його подивитись, відкривши за допомогою будь-якого переглядача зображень.

При тестуванні роботи розробленого програмного модуля використовувалось пакетне навчання з розміром пакету 4 і методом навчання «Стохастичний градієнтний спуск» - Stochastic Gradient Descent (SGD). Швидкість навчання була обрана 0,005, імпульс = 0,9 і розпад ваги 5×10^{-4}. Виконувалось все навчання з одним графічним процесором NVIDIA T4. Зазначені гіперпараметри були обрані саме такими, тому що було виявлено, що всі досліджувані при тестуванні моделі досягали дуже низьких втрат

точності на тренувальній вибірці та дуже високого показника F1 при досягненні 50 епох навчання (що тривало протягом кількох годин), використовуючи ці гіперпараметри. У програмному модулі використовувалась загальнодоступна попередньо навчена згорткова нейронна мережа Mask R-CNN ResNet-50-FPN для всіх експериментів з тонким налаштуванням Mask R-CNN.

Щоб оцінити продуктивність розроблених моделей, спочатку визначалися порогові значення масок сегментації, створених розробленими моделями на основі Mask-RCNN, які не є двійковими, а натомість мають м'які значення в діапазоні від 0 до 1, що відображає впевненість моделі в тому, що певний піксель містить стенозовану судину. Для побудови бінарної маски прогнозованих стенотичних ділянок для пікселів, які отримали оцінку вище порогу достовірності (який налаштовується за допомогою тестового набору даних), встановлюється значення 1, тоді як для всіх інших пікселів встановлюється значення 0. Потім об'єднуємо всі прогнозовані маски стенозу, виконуючи логічну операцію АБО, щоб створити одну агреговану прогнозовану маску. Використовуємо ту саму процедуру, щоб отримати агреговану маску дійсної ділянки стенозу. Потім порівнювалася агрегована дійсна ділянка стенозу та прогнозовані маски попіксельно, щоб обчислити попіксельні оцінки F1 (так звані коефіцієнти подібності Dice) [27].

$$F_1 = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}} = \frac{\text{TP}}{\text{TP} + 0.5(\text{FP} + \text{FN})}$$

де TP - **вірно позитивний результат** (True Positive), FP - **хибно позитивний результат** (False Positive), FN - **хибно негативний результат** (False Negative). Precision – влучність, Recall – повнота. Більш наочно визначення влучності та повноти показано на рис. 4.1.

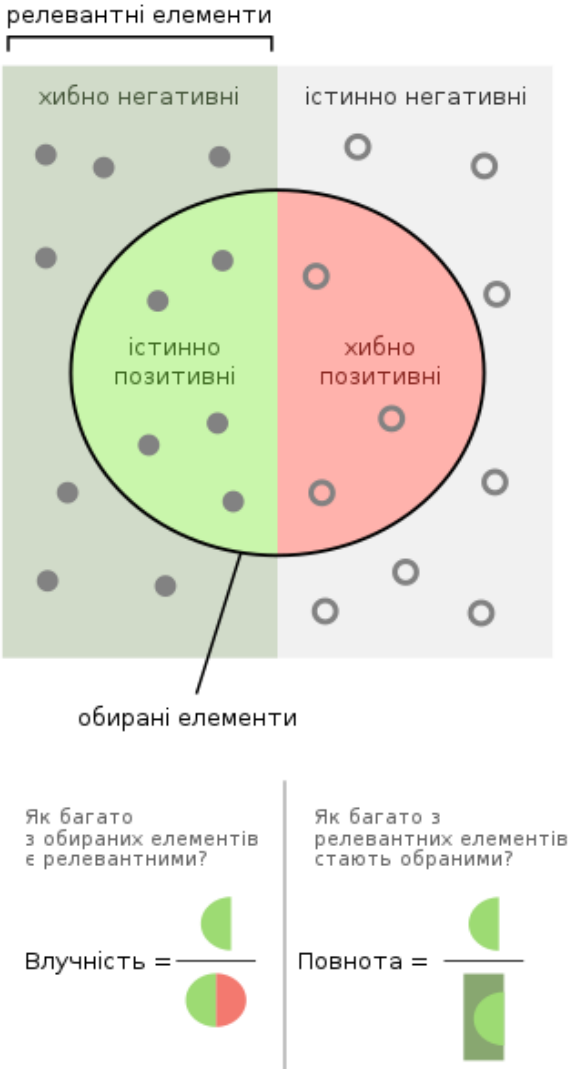


Рисунок 4.1 – Визначення влучності та повноти

Ми обираємо поріг довіри для кожної моделі, оцінюючи кожну модель з усіма можливими порогами довіри (0-1) з кроком 0,05, і вибираючи поріг довіри, який дає найвищу оцінку F1 на тестовому наборі. Запропоновані методи порівнювалися з базовими результатами YOLOv8 [15], а також власними базовими показниками.

Аналог (Базова лінія) - модель на основі YOLOv8.

В якості базової лінії ми налаштували YOLOv8n-seg на наборі для навчання стенозу ARCADE (n=1000) протягом 50 епох і досягли оцінки F1 0,35 у валідаційному наборі та оцінки F1 0,31 у тестовому наборі (табл. 4.1). Автори ARCADE зазначили, що їх базова модель YOLOv8 досягла оцінки F1 0,38 [15]. Трохи вища продуктивність команди ARCADE, ймовірно, пов’язана з

використанням більшої тестової вибірки YOLO (порівняно з «нано» тестовою вибіркою, яку ми використовували), а також більш ретельним процесом налаштування гіперпараметрів.

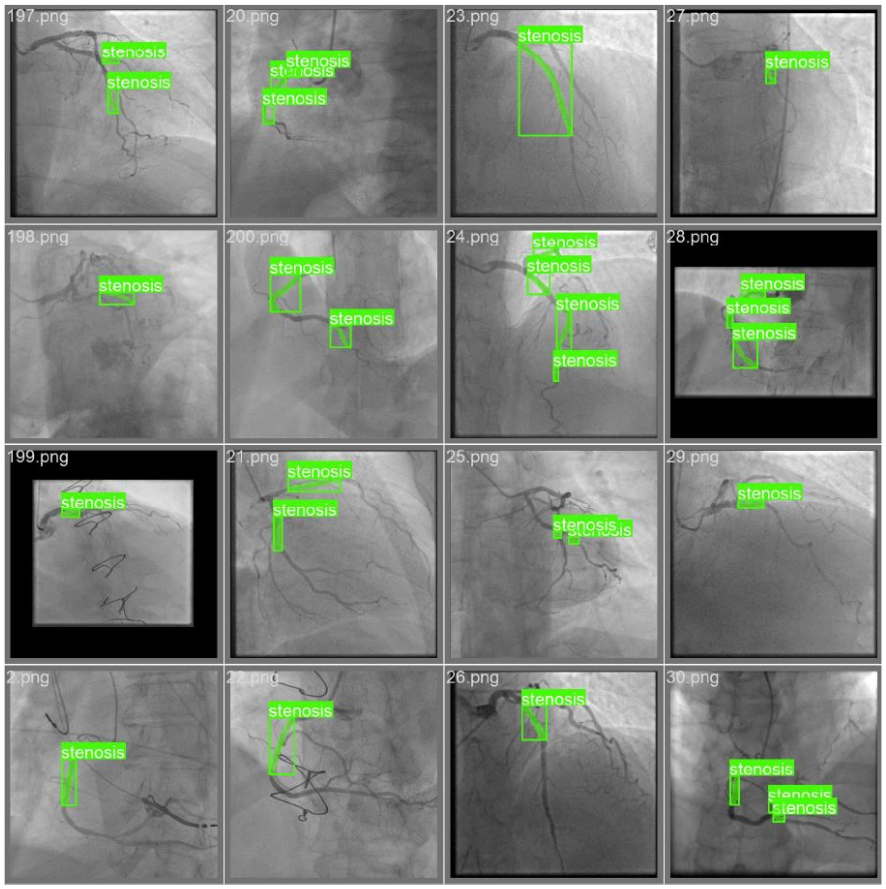
Таблиця 4.1 - Оцінка F1 для валідаційного та тестового набору аналогів та розробленого модуля.

	Model	Оцінка F1 на валідаційному наборі	Оцінка F1 на тестовому наборі
Аналоги	YOLOv8 ([15])	-	0.38
	YOLOv8n-seg	0.35	0.31
Розроблені моделі	StenSeg-base	0.54	0.51
	StenSeg-pl	0.55	0.51

У табл. 4.1 наведено також показники для двох базових методів: моделі YOLOv8, навченої у [15], і YOLOv8n-seg, яка була навчана в ході цієї роботи. Хоча деякі зображення позначені точно, модель не змогла сегментувати деякі зображення та надала більше прогнозів, ніж очікувалося, для інших (рис. 4.2).

Конвеєр розширення даних із псевдо-маркуванням (StenSeg-pl) продемонстрував потенціал порівняно з моделлю vanilla Mask R-CNN (StenSeg-base). У той час як оцінки тесту F1 були дуже схожими, якщо округлити їх до двох десяткових знаків (0,51 проти 0,51), StenSeg-pl досяг вищого результату F1 у наборі даних валідації (0,55 проти 0,54). Хоча покращення завдяки доповненню даних наразі здаються незначними, зауважте, що ми включили лише 1000 додаткових зображень із непозначеного набору даних Syntax, отриманого з того самого загального набору даних ARCADE. Існує величезна кількість немаркованих даних ангіографії XCA, і можливість отримати до них доступ для навчання, швидше за все, створить ефективніші та узагальнені моделі.

Усі моделі StenSeg значно перевершили базову модель YOLOv8-seg, яка досягла оцінки F1 0,35 та 0,31 відповідно для валідації та тестування.



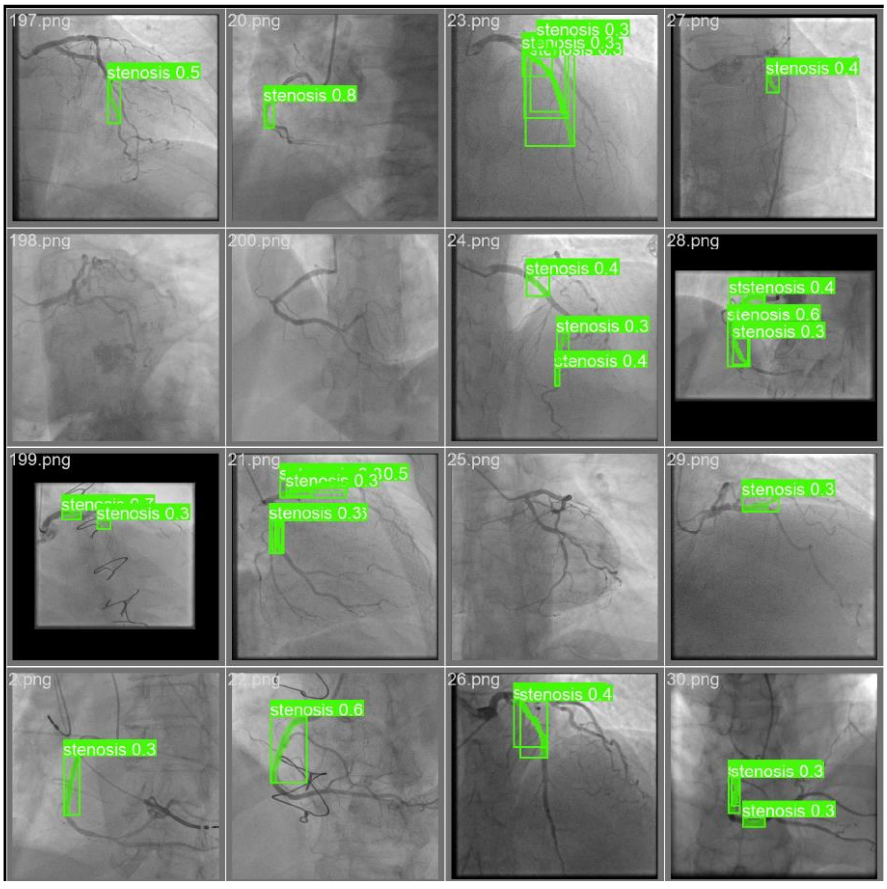
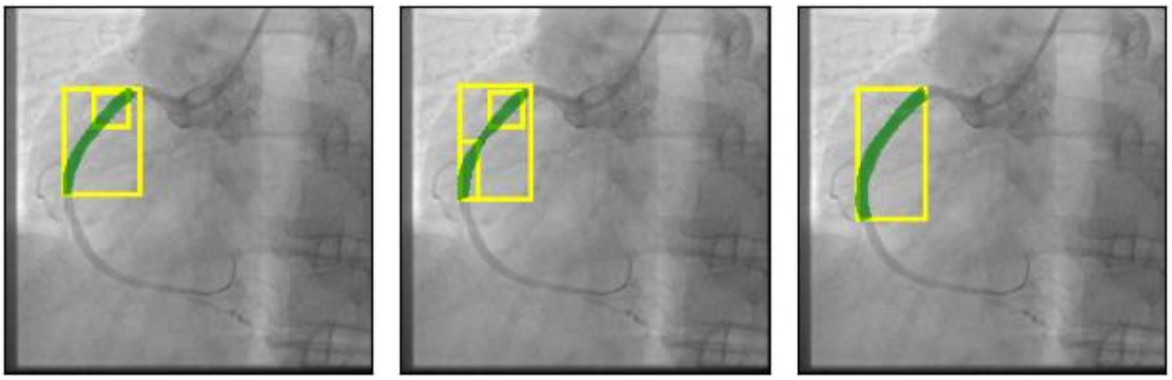


Рисунок 4.2 - Угорі: дійсні мітки стенозу для групи тестових зображень. Внизу: прогнози аналога YOLOv8 для тих самих тестових зображень

Результати для розробленої моделі StenSeg-base.

Ми тренували маск R-CNN на навчальному наборі протягом 50 епох. Було виявлено, що, незважаючи на значне перевищення достатньої кількості епох до 50-ї епохи (оцінка навчання F1 була значно вищою, близько 0,80), модель на 50-й епосі все одно показала кращі результати на валідаційному наборі, ніж усі попередні контрольні точки. Цей контрольний пункт отримав найкращу оцінку валідації F1 0,54 при порозі достовірності 0,55. При оцінці на тестовому наборі з таким самим порогом довіри модель досягла оцінки F1 0,51 (таблиця 4.1).

З якісного боку було також виявлено, що StenSeg-base виводить досить високоякісні маски стенозу (рис. 4.3), підтверджуючи наше рішення використовувати цю модель для створення псевдоміток для подальшого навчання.



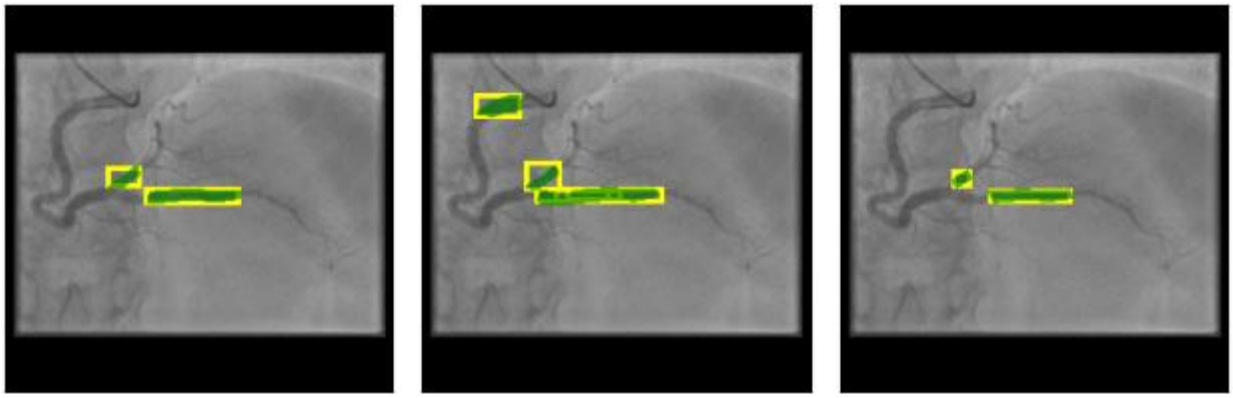
StenSeg-pl StenSeg-base Правильна область стенозу

Рисунок 4.3 – Прогнозовані (ліворуч та посередині) та базові істинні (праворуч) мітки стенозу.

Однак потрібно зазначити, що StenSeg-base далекий від досконалості, що видно з рис. 4.4 і 4.5, де StenSeg-base неправильно визначає здорові судини

як стенозовані або взагалі не визначає правильну область стенозу відповідно. Крім того, цікаво відзначити, що іноді, коли StenSeg-base правильно визначає стенотичну область на зображенні, вона визначає кілька випадків стенозу, коли є лише один (рис. 4.3).

На рис. 4.3 представлено прогнозовані маски стенозу на базі StenSeg-base (посередині), які якісно досить подібні до базових істинних міток стенозу (праворуч), що підтверджує використання прогнозів на базі StenSeg-base як псевдоміток. StenSeg-pl (ліворуч) також працює дуже добре якщо порівнювати з базовими істинними мітками. Кожна жовта обмежувальна рамка в основних мітках істинності представляє інший прогнозований стеноз.



StenSeg-pl StenSeg-base Правильна область стенозу

Рисунок 4.4 - Прогнозовані маски стенозу на базі StenSeg-base (посередині) та StenSeg-pl (ліворуч) порівняно з базовими істинними ділянками (праворуч).

На рис. 4.4 представлено прогнозовані маски стенозу на базі StenSeg-base (посередині), яка передбачила помилково додаткову стенотичну ділянку порівняно з реальнии ділянками (праворуч), а StenSeg-pl (ліворуч) правильно визначає стенотичні ділянки.

На рис. 4.5 представлено випадок, коли прогнозовані маски стенозу на базі StenSeg-base (посередині) не зафіксували правильну область стенозу порівняно з базовою істинною ділянкою (праворуч). Крім того, StenSeg-pl

(ліворуч) у даному випадку помилково передбачив додаткову стенотичну ділянку.

StenSeg-pl StenSeg-base Правильна область стенозу

Рисунок 4.5 - Прогнозовані маски стенозу на базі StenSeg (посередині) можуть не зафіксувати правильну область стенозу (праворуч). Крім того, StenSeg-pl (ліворуч) має тенденцію передбачати додаткові стенотичні ділянки.

Результати збільшення (аугментація) даних - StenSeg-pl.

Як описано в Розділі 3.2, ми навчили ще одну згорткову нейромережу Mask R-CNN на комбінованому наборі даних, який складався з оригінального набору даних стенозу, а також синтаксичного набору даних, псевдопозначеного StenSeg-base (див. Рис. 2.8). Було виявлено, що ця модель почала переналаштовуватися набагато раніше, ніж базова модель Mask R-CNN, ймовірно, через те, що розмір набору даних був удвічі більшим, ніж у StenSeg-base. Після оцінки кількох різних контрольних точок у різні епохи навчання у валідаційному наборі стенозу ARCADE було встановлено, що контрольна точка епохи 30 має найкращу продуктивність. Потім було виконано таку саму перевірку порогових значень довіри на цій контрольній точці, щоб виявити, що ця модель досягла найкращого результату перевірки F1 0,55 при порозі довіри 0,5. При оцінці на тестовому наборі модель досягла оцінки F1 0,51 (табл. 4.1).

Із табл. 4.1 видно, що оцінка F1 на тестовому наборі для запропонованого модуля має величину 0,51, а для кращого з аналогів – 0,38. Тобто точність сегментації стенозу коронарних судин серця розробленої програми вища, ніж у програми аналогу. В абсолютній величині вища на 0,13 (0,51 проти 0,38) для показника F1. У відносних одиницях можна сказати, що точність сегментації стенозу коронарних судин серця розробленого модуля вища на 34,2% ((0,51-0,38)/0,38=0,342), ніж у аналога.

Таким чином, можна зробити висновок, що розроблений модуль сегментації стенозу коронарних судин серця на основі згорткової нейронної мережі має порівняно з аналогом збільшену на 34,2% точність сегментації стенозу коронарних судин серця по показнику F1. Тобто мета роботи досягнута – точність сегментації стенозу коронарних судин серця підвищена.

Перспективи подальшої роботи.

Через несподіваний результат, що диверсифікація навчального набору даних зображеннями, зібраними з різних джерел, завадила продуктивності в наборі тестів ARCADE, пропонуємо до розробки тестів сегментації стенозу ХСА долучати зображення з різних джерел як шлях майбутньої роботи. Ми припускаємо, що це посилить майбутній розвиток моделі, оскільки дані будуть більш різноманітними та відображатимуть клінічні умови.

Крім того, потрібно тримати комунікацію з рентгенологами, щоб отримати їх якісні гіпотези щодо того, чому наша модель може видавати неправильні сегменти стенозу в наших випадках невдачі. Якщо є повторювані візерунки (тобто артефакти під час візуалізації, загальні анатомічні структури, візуально схожі на стеноз), можливо, можна застосувати певну форму контрастної втрати, щоб зафіксувати ці особливості та мінімізувати помилку.

Нарешті, було б доцільним ще більше посилити конвеєр аугментації даних. Це може включати додавання геометричних перетворень, перетворень колірного простору, розмивання або шуми.

Іншим помітним майбутнім удосконаленням може бути використання «м’яких» [7] масок замість бінарних масок класифікації, що може забезпечити

менш чорно-білий підхід до сегментації стенозу на зображеннях ХСА, де є притаманна невизначеність.

Висновок до розділу 4

У розділі в результаті тестування програмного модуля було доведено його працездатність та відповідність поставленому завданню. Для оцінки точності роботи модуля використовувалась така метрика як F1. Оцінка F1 на тестовому наборі для запропонованого модуля має величину 0,51, а для кращого з аналогів – 0,38. Тобто точність сегментації стенозу коронарних судин серця розробленої програми вища, ніж у програми аналогу в абсолютній величині на 0,13 для показника F1, а у відносних одиницях - вища на 34,2%. Таким чином, мета роботи досягнута – точність сегментації стенозу коронарних судин серця по показнику F1 підвищена на 34,2%.

ВИСНОВКИ

У роботі було розглянуто задачу сегментації зображень стенозу коронарних судин серця на основі згорткової нейронної мережі. Розроблений програмний модуль призначений для сегментації стенотичних уражень у пацієнтів з ішемічною хворобою серця і працює на рентгенівській коронарній ангіографії (РКА) замість комп’ютерної томографічної коронарної ангіографії (КТКА), оскільки РКА зберігає найкращу діагностичну здатність у важких випадках В ході аналізу предметної області було розглянуто детальну постановку задачі сегментації зображень стенозу коронарних судин серця. Також розглянуто різні методи розв'язання задачі сегментації зображень і оцінено їх застосовність до завдання сегментації зображень стенозу коронарних судин серця. Як найбільш перспективний і застосовний до даної задачі було обрано метод на основі згорткових нейромереж. Було оцінено застосовність до завдання сегментації зображень стенозу коронарних судин серця таких згорткових нейромереж як SegNet, U-Net, YOLO. Крім цього, було обгрунтовано вибір аналогів розробленого модуля сегментації зображень стенозу коронарних судин серця.

Було обгрунтовано вибір архітектури згорткової нейронної мережі для сегментації зображень стенозу коронарних судин серця. Із двох проаналізованих архітектур: YOLO та Mask R-CNN, було обрано згорткову нейромережу Mask R-CNN для використання у сегментації зображень стенозу коронарних судин серця як найбільш перспективну. Розроблено метод сегментації зображень стенозу коронарних судин серця на основі Mask R-CNN, що реалізує підхід RoIAlign, який допомагає відокремити процес передбачення маски від класифікації. Було розроблено процес аугментації даних для навчання згорткової нейронної мережі Mask R-CNN. Також було розроблено алгоритм роботи програмного модуля сегментації зображень стенозу коронарних судин серця на основі згорткової нейронної мережі.

Було обґрунтовано вибір мови програмування Python та спеціалізованих бібліотек Pytorch, Pytorch Image Models, NumPy, Torchvision та Matplotlib для програмної реалізації модуля сегментації зображень стенозу коронарних судин серця на основі згорткової нейронної мережі. Проведено аналіз набору ARCADE для навчання та тестування модуля, який містить приблизно 3 000 зображень, з яких 2000 навчальних, 400 валідаційних та 600 тестових. Описано основні етапи програмної реалізація та функціонування модуля сегментації зображень стенозу коронарних судин серця на основі згорткової нейронної мережі, що складається з завантаження бібліотек, завантаження набору даних із зображеннями, запуску роботи моделі аналогу і запропонованої нейромережі, візуалізації результатів роботи, підрахунку показників якості (метрик) аналога та розробленої нейромережі.

У результаті тестування програмного модуля було доведено його працездатність та відповідність поставленому завданню. Для оцінки точності роботи модуля використовувалась така метрика як F1. Оцінка F1 на тестовому наборі для запропонованого модуля має величину 0,51, а для кращого з аналогів – 0,38. Тобто точність сегментації стенозу коронарних судин серця розробленої програми вища, ніж у програми аналогу в абсолютній величині на 0,13 для показника F1, а у відносних одиницях - вища на 34,2%. Таким чином, мета роботи досягнута – точність сегментації стенозу коронарних судин серця по показнику F1 підвищена на 34,2%.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 URL http://home.iitj.ac.in/~manpreet.bedi/btp/documents/sar.pdf

2 URL https://towardsdatascience.com/understanding-k-means-clustering-in-machinelearning-6a6e67336aa1

3 URL https://www.v7labs.com/blog/image-segmentation-guide

4 Vijay Badrinarayanan, Alex Kendall, SegNet: A Deep Convolutional EncoderDecoder Architecture for Image Segmentation, 2016. – С 14.

5 Olaf Ronneberger, Philipp Fischer, U-Net: Convolutional Networks for Biomedical Image Segmentation, 2015. – С. 8.

6 Liang-Chieh Chen, George Papandreou, DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs, 2017. – С. 14.

7 XuhangLiana, YanweiPang, Cascaded hierarchical atrous spatial pyramid pooling module for semantic segmentation, 2020.

8 Jonathan Long, Evan Shelhamer, Fully Convolutional Networks for Semantic, 2015. – С. 10.

9 Y. Li, Y. Wu, J. He, W. Jiang, J. Wang, Y. Peng, Y. Jia, T. Xiong, K. Jia, Z. Yi, et al. Automatic coronary artery segmentation and diagnosis of stenosis by deep learning based on computed tomographic coronary angiography. European Radiology, 32(9):6037–6045, 2022.

10 W. Huang, L. Huang, Z. Lin, S. Huang, Y. Chi, J. Zhou, J. Zhang, R.-S. Tan, and L. Zhong. Coronary artery segmentation by deep learning neural networks on computed tomographic coronary angiographic images. In 2018 40th Annual international conference of the IEEE engineering in medicine and biology society (EMBC), pages 608–611. IEEE, 2018.

11 H. R. Fazlali, N. Karimi, S. R. Soroushmehr, S. Shirani, B. K. Nallamothu, K. R. Ward, S. Samavi, and K. Najarian. Vessel segmentation and catheter detection in x-ray angiograms using superpixels. Medical & biological engineering & computing, 56:1515–1530, 2018.

12 H. Zhang, Z. Gao, D. Zhang,W. K. Hau, and H. Zhang. Progressive perception learning for main coronary segmentation in x-ray angiography. IEEE Transactions on Medical Imaging, 42(3):864–879, 2022.

13 T. Du, L. Xie, H. Zhang, X. Liu, X. Wang, D. Chen, Y. Xu, Z. Sun, W. Zhou, L. Song, et al. Training and validation of a deep learning architecture for the automatic analysis of coronary angiography: Automatic recognition of coronary angiography. EuroIntervention, 17(1):32, 2021.

14 M. Popov, A. Amanturdieva, N. Zhaksylyk, A. Alkanov, A. Saniyazbekov, T. Aimyshev, E. Ismailov, A. Bulegenov, A. Kuzhukeyev, A. Kulanbayeva, et al. Dataset for automatic region-based coronary artery disease diagnostics using x-ray angiography images. Scientific Data, 11(1):20, 2024.

15 J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. CoRR, abs/1506.02640, 2015. 2,

16 G. Jocher, A. Chaurasia, and J. Qiu. Ultralytics YOLO, Jan. 2023.

17 H. Lin, T. Liu, A. Katsaggelos, and A. Kline. Stenunet: Automatic stenosis detection from x-ray coronary angiography. arXiv preprint arXiv:2310.14961, 2023.

18 M. Bilal, D. Martinho, R. Sim, A. Qayyum, H. Vohra, M. Caputo, T. Akinosho, S. Abioye, Z. Khan,W. Niaz, et al. Multivessel coronary artery segmentation and stenosis localization using ensemble learning. arXiv preprint arXiv:2310.17954, 2023.

19 I. K. Lee, J. Shin, Y.-H. Lee, J. Ku, and H.-W. Kim. Ssass: Semi-supervised approach for stenosis segmentation. arXiv preprint arXiv:2311.10281, 2023.

20 K. He, G. Gkioxari, P. Doll´ar, and R. Girshick. Mask r-cnn. In Proceedings of the IEEE international conference on computer vision, pages 2961–2969, 2017.

21 Python [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Python

22 A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. K¨opf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. CoRR, abs/1912.01703, 2019.

23 NumPy [Електронний ресурс] – Режим доступу: https://numpy.org/

24 Matplotlib: Visualization with Python [Електронний ресурс] – Режим доступу: https://matplotlib.org/

25 S. Pokhrel, S. Bhandari, E. Vazquez, Y. R. Shrestha, and B. Bhattarai. Data augmentation through pseudolabels in automatic region based coronary artery segmentation for disease diagnosis. arXiv preprint arXiv:2310.05990, 2023.

26 V. Danilov, K. Klyshnikov, O. Gerget, A. Kutikhin, V. Ganyukov, A. Frangi, and E. Ovcharenko. Real-time coronary artery stenosis detection based on modern neural networks. Scientific Reports, 11, 2021.

27 D. M¨uller, I. Soto-Rey, and F. Kramer. Towards a guideline for evaluation metrics in medical image segmentation. BMC Research Notes, 15(1):210, 2022.

28 Методичні вказівки до виконання бакалаврської кваліфікаційної роботи для студентів денної та заочної форм навчання спеціальності 122 - «Комп’ютерні науки» / Укладачі А.А.Яровий, О.В.Сілагін, І.В.Богач. – Вінниця: ВНТУ, 2022. – 47 с.

Додаток А (обов'язковий)
ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Програмний модуль нейромережевої сегментації зображень стенозу коронарних судин серця

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра комп'ютерних наук
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 11,91 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

- ✓ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

- ☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

- ☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Яровий А.А., зав. каф. КН _____ _____
(прізвище, ініціали, посада) (підпис)

Колесницький О.К., проф. каф КН _____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Озеранський В.С.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Колесницький О.К.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Яцків Д.О.
(підпис) (прізвище, ініціали)

Додаток Б (обов'язковий)

Лістинг програми

Фрагмент коду

```
import timm
import torch
import torchvision
from torch.utils.data import DataLoader
from torchvision.transforms import functional as F
from torchvision.models.detection import MaskRCNN
from torchvision.models.detection.mask_rcnn import MaskRCNNPredictor
from torchvision.models.detection.backbone_utils import BackboneWithFPN
from torchvision.datasets.coco import CocoDetection
from torchvision.transforms import ToTensor, Normalize, Compose
import torchvision.transforms as tf
import matplotlib.pyplot as plt
from torchvision.transforms import v2
from torchvision.utils import draw_bounding_boxes, draw_segmentation_masks
from torchvision import tv_tensors
from torchvision.transforms.v2 import functional as F
import torch.nn as nn
from torchvision.datasets import wrap_dataset_for_transforms_v2
from torchvision.transforms.functional import pil_to_tensor
from torchvision.models.detection.mask_rcnn import
MaskRCNN_ResNet50_FPN_Weights

import os
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
import matplotlib.patches as patches
import matplotlib.patheffects as PathEffects
import cv2
from torch.utils.data import Dataset
import warnings


def load_yolo_annotations(file_path, img_width, img_height):
    """
    Load YOLO annotations from a file and convert them to bounding boxes and
polygons.
    Returns: a list of (class_id, polygon_mask) tuples
    """
    boxes = []
    labels = []
    masks = []
    with open(file_path, 'r') as f:
        for line in f:
            parts = line.strip().split()

            # record class label
            label = int(parts[0])
            labels.append(label)

            # create pixelwise mask from YOLO polygon annotation
            polygon = np.array([float(p) for p in parts[1:]]).reshape(-1, 2)
            polygon[:, 0] *= img_width
            polygon[:, 1] *= img_height
            polygon_mask = np.zeros((img_height, img_width), dtype=np.uint8)
```

```
            polygon = polygon.astype(np.int32)
            cv2.fillPoly(polygon_mask, [polygon], color=1)
            masks.append(polygon_mask)

            # create bounding box based on polygon coordinates
            x_coords = polygon[:, 0]
            y_coords = polygon[:, 1]
            x_min = np.min(x_coords)
            x_max = np.max(x_coords)
            y_min = np.min(y_coords)
            y_max = np.max(y_coords)
            coordinates = [x_min, y_min, x_max, y_max]
            bbox = torch.tensor([x_min, y_min, x_max, y_max],
dtype=torch.float32)
            boxes.append(bbox)

    boxes = torch.stack(boxes, dim=0)
    labels = torch.tensor(labels, dtype=torch.int64)
    masks = torch.from_numpy(np.stack(masks, axis=0))

    return boxes, labels, masks


def plot(imgs, row_title=None, **imshow_kwargs):
    if not isinstance(imgs[0], list):
        # Make a 2d grid even if there's just 1 row
        imgs = [imgs]

    num_rows = len(imgs)
    num_cols = len(imgs[0])
    fig, axs = plt.subplots(nrows=num_rows, ncols=num_cols, squeeze=False)
    for row_idx, row in enumerate(imgs):
        for col_idx, img in enumerate(row):
            boxes = None
            masks = None
            if isinstance(img, tuple):
                img, target = img
                if isinstance(target, dict):
                    boxes = target.get("boxes")
                    masks = target.get("masks")
                elif isinstance(target, tv_tensors.BoundingBoxes):
                    boxes = target
                else:
                    raise ValueError(f"Unexpected target type:
{type(target)}")
            img = F.to_image(img)
            if img.dtype.is_floating_point and img.min() < 0:
                # Poor man's re-normalization for the colors to be OK-ish.
This
                # is useful for images coming out of Normalize()
                img -= img.min()
                img /= img.max()

            img = F.to_dtype(img, torch.uint8, scale=True)
            if boxes is not None:
                img = draw_bounding_boxes(img, boxes, colors="yellow",
width=3)
            if masks is not None:
                img = draw_segmentation_masks(img, masks.to(torch.bool),
colors=["green"] * masks.shape[0], alpha=.65)

            ax = axs[row_idx, col_idx]
            ax.imshow(img.permute(1, 2, 0).numpy(), **imshow_kwargs)
```

```
            ax.set(xticklabels=[], yticklabels=[], xticks=[], yticks=[])

    if row_title is not None:
        for row_idx in range(num_rows):
            axs[row_idx, 0].set(ylabel=row_title[row_idx])

    plt.tight_layout()
    plt.show()
    fig.savefig("plot.png")


def save_checkpoint(model, optimizer, epoch, iteration, loss,
filename="checkpoint.pth"):
    checkpoint = {
        'epoch': epoch,
        'iteration': iteration,
        'model_state_dict': model.state_dict(),
        'optimizer_state_dict': optimizer.state_dict(),
        'loss': loss,
    }
    torch.save(checkpoint, filename)
    print(f"Checkpoint saved at epoch {epoch}")

def load_checkpoint(model, optimizer, filename="checkpoint.pth"):
    checkpoint = torch.load(filename)
    model.load_state_dict(checkpoint['model_state_dict'])
    optimizer.load_state_dict(checkpoint['optimizer_state_dict'])
    epoch = checkpoint['epoch']
    loss = checkpoint['loss']
    # iteration = checkpoint['iteration']
    print(f"Checkpoint loaded from epoch {epoch} with loss {loss:.4f}")
    return model, optimizer, epoch, loss


def get_metrics(master_mask, target_mask):
    # Ensure both masks are binary and have the same shape
    assert master_mask.shape == target_mask.shape, "Masks should have the
same shape"

    # Convert to binary (if not already)
    master_mask = (master_mask > 0).to(torch.uint8)
    target_mask = (target_mask > 0).to(torch.uint8)

    # Flatten the masks
    master_mask = master_mask.view(-1)
    target_mask = target_mask.view(-1)

    # Calculate True Positives (TP), False Positives (FP), and False
Negatives (FN)
    tp = torch.logical_and(master_mask, target_mask).sum().item()
    fp = torch.logical_and(master_mask,
torch.logical_not(target_mask)).sum().item()
    fn = torch.logical_and(torch.logical_not(master_mask),
target_mask).sum().item()

    return tp, fp, fn


def compute_f1(tp, fp, fn):
    precision = float(tp / (tp + fp + 1e-8))  # Adding a small epsilon to
avoid division by zero
    recall = float(tp / (tp + fn + 1e-8))  # Adding a small epsilon to avoid
division by zero
```

```
    f1_score = 2 * (precision * recall) / (precision + recall + 1e-8)  #
Adding a small epsilon to avoid division by zero
    return f1_score

class CustomImageDataset(Dataset):
    def __init__(self, data_dir, split='train', transform=None):
        """
        Args:
            data_dir (string): Root directory with all the images and labels.
            split (string): 'train' or 'val' or 'test'
            transform (callable, optional): Optional transform to be applied
on a sample.
        """
        self.data_dir = data_dir
        img_dir = os.path.join(data_dir, f'{split}/images')
        self.image_filenames = [f for f in os.listdir(img_dir) if
f.endswith('.png')]
        self.split = split
        self.transform = transform

    def __len__(self):
        return len(self.image_filenames)

    def __getitem__(self, idx):
        img_name = str(idx + 1) + '.png'
        ann_name = str(idx + 1) + '.txt'
        image_path = os.path.join(self.data_dir, f'{self.split}/images',
img_name)  # Replace 'example.jpg' with your image file
        annotation_path = os.path.join(self.data_dir, f'{self.split}/labels',
ann_name)  # Replace 'example.txt' with your annotation file

        image = Image.open(image_path).convert('RGB')
        image = image.resize((224, 224))
        image = pil_to_tensor(image)
        image = image.to(torch.float32)
        image = image / 255.0

        boxes, labels, masks = load_yolo_annotations(annotation_path, 224,
224)
        targets = {'image_id': torch.tensor([idx + 1]), 'boxes': boxes,
'masks': masks, 'labels': labels}

        return image, targets

class PseudoSyntaxDataset(Dataset):
    def __init__(self, data_dir, checkpoint_file, split='train',
transform=None):
        """
        Args:
            data_dir (string): Root directory with all the images and labels.
            split (string): 'train' or 'val' or 'test'
            transform (callable, optional): Optional transform to be applied
on a sample.
        """
        self.data_dir = data_dir
        img_dir = os.path.join(data_dir, f'{split}/images')
        self.image_filenames = [f for f in os.listdir(img_dir) if
f.endswith('.png')]
        self.split = split
        self.transform = transform

        # init model
```

```
        model =
torchvision.models.detection.maskrcnn_resnet50_fpn(weights=MaskRCNN_ResNet50_
FPN_Weights.DEFAULT)
        params = [p for p in model.parameters() if p.requires_grad]
        optimizer = torch.optim.SGD(params, lr=0.005, momentum=0.9,
weight_decay=0.0005)
        model, optimizer, _, _ = load_checkpoint(model, optimizer,
filename=checkpoint_file)
        self.model = model

    def __len__(self):
        return len(self.image_filenames)

    def __getitem__(self, idx):
        img_name = str(idx + 1) + '.png'
        image_path = os.path.join(self.data_dir, f'{self.split}/images',
img_name)  # Replace 'example.jpg' with your image file

        image = Image.open(image_path).convert('RGB')
        image = image.resize((224, 224))
        image = pil_to_tensor(image)
        image = image.to(torch.float32)
        image = image / 255.0

        boxes, labels, masks = pseudo_eval(self.model, image,
split=self.split, conf=0.55)
        targets = {'image_id': torch.tensor([idx + 1]), 'boxes': boxes,
'masks': masks, 'labels': labels}

        return image, targets


def pseudo_eval(model, image, split='train', conf=0.80, k=None):
    # load model
    if not isinstance(image, torch.Tensor):
        raise ValueError("The 'image' parameter should be a tensor.")

    device = torch.device("cuda") if torch.cuda.is_available() else
torch.device("cpu")
    model.to(device)
    model.eval()

    imgs = []
    imgs.append(image.to(device))

    with torch.no_grad():
        # We only need imgs for inference
        outputs = model(imgs)

    boxes = []
    masks = []
    labels = []
    # Process the outputs as needed
    for i, output in enumerate(outputs):
        output['masks'] = output['masks'].squeeze(1)
        output['masks'][output['masks'] > conf] = 1
        output['masks'][output['masks'] <= conf] = 0
        masks.append(output['masks'])

        boxes.append(output['boxes'])
        labels.append(output['labels'])

    boxes = torch.stack(boxes, dim=0).reshape((-1, 4))
```

```
    labels = torch.cat(labels).to(torch.int64)
    masks = torch.cat(masks, dim=0).squeeze(1)

    # boxes = torch.stack(boxes, dim=0).reshape((-1, 4))
    # labels = torch.cat(labels, dim=0)
    # masks = torch.cat(masks, dim=0).squeeze(1)
    # print(masks.shape)

    return boxes, labels, masks

class DanilovDataset(Dataset):
    def __init__(self, data_dir, checkpoint_file, stopping_num=1000,
split='train', transform=None):
        """
        Args:
            data_dir (string): Root directory with all the images and labels.
            split (string): 'train' or 'val' or 'test'
            transform (callable, optional): Optional transform to be applied
on a sample.
        """
        self.data_dir = data_dir
        img_dir = data_dir
        all_bmp_files = [f for f in os.listdir(img_dir) if
f.endswith('.bmp')]
        self.stopping = stopping_num
        self.image_filenames = all_bmp_files[:self.stopping]
        self.split = split
        self.transform = transform

        # init model
        model =
torchvision.models.detection.maskrcnn_resnet50_fpn(weights=MaskRCNN_ResNet50_
FPN_Weights.DEFAULT)
        params = [p for p in model.parameters() if p.requires_grad]
        optimizer = torch.optim.SGD(params, lr=0.005, momentum=0.9,
weight_decay=0.0005)
        model, optimizer, _, _ = load_checkpoint(model, optimizer,
filename=checkpoint_file)
        self.model = model

    def __len__(self):
        return len(self.image_filenames)

    def __getitem__(self, idx):
        img_name = str(idx + 1) + '.bmp'
        image_path = os.path.join(self.data_dir, img_name)  # Replace
'example.jpg' with your image file

        image = Image.open(image_path).convert('RGB')
        image = image.resize((224, 224))
        image = pil_to_tensor(image)
        image = image.to(torch.float32)
        image = image / 255.0

        boxes, labels, masks = pseudo_eval(self.model, image,
split=self.split, conf=0.55)
        targets = {'image_id': torch.tensor([idx + 1]), 'boxes': boxes,
'masks': masks, 'labels': labels}

        return image, targets
```

Додаток В (обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

«ПРОГРАМНИЙ МОДУЛЬ НЕЙРОМЕРЕЖЕВОЇ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ СТЕНОЗУ КОРОНАРНИХ СУДИН СЕРЦЯ»

Виконала: студентка 4 курсу,групи 2КН-21б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Яцків Д. О.
(прізвище та ініціали)

Керівник: к.т.н., проф. каф.КН
Колесницький О.К.
(прізвище та ініціали)
«03» 06. 2025 р.

Вінниця ВНТУ - 2025 рік

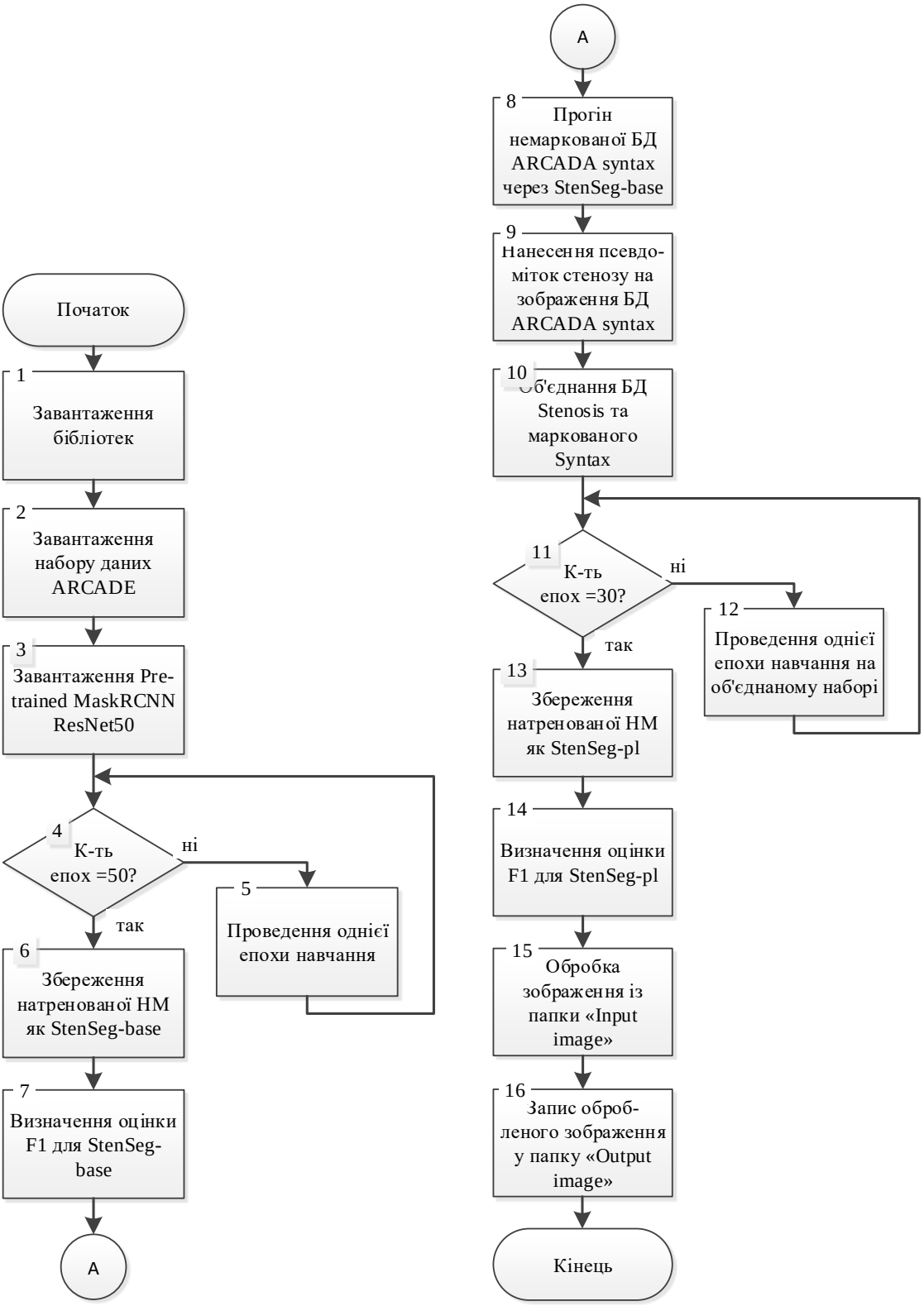


Рисунок В.1– Схема алгоритму роботи програмного модуля сегментації зображень стенозу коронарних судин серця

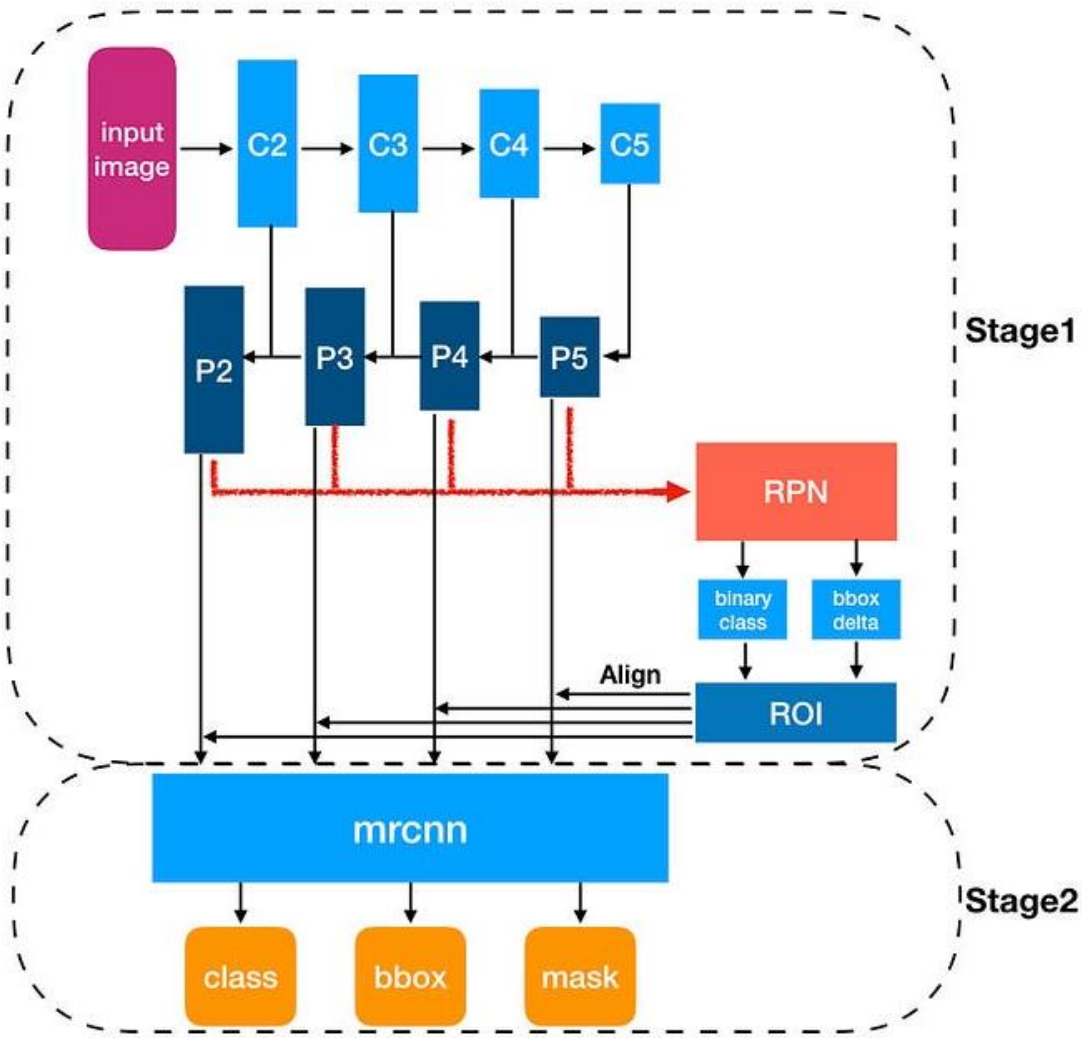


Рисунок В.2– Логічна структура нейронної мережі Mask RCNN

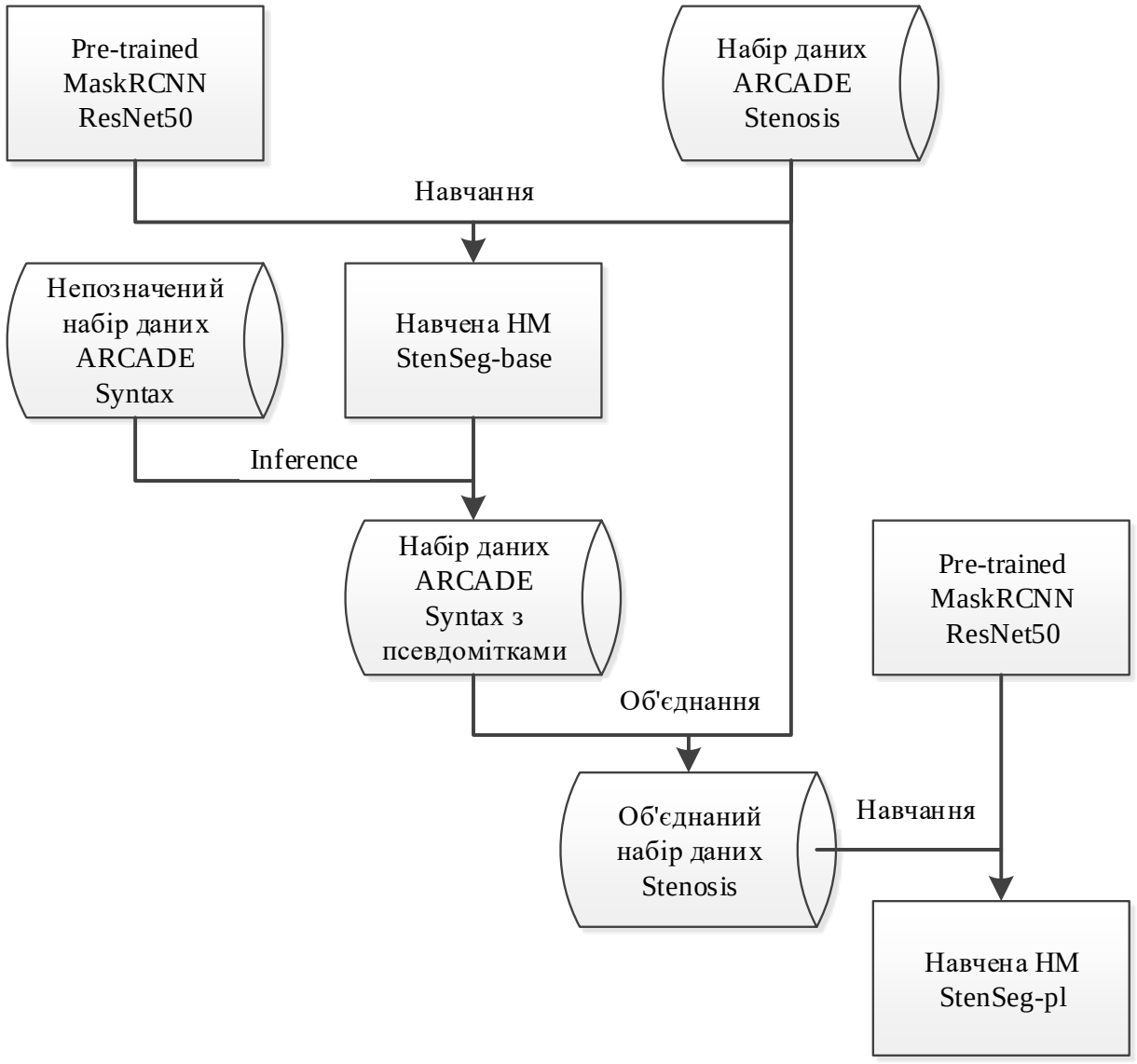


Рисунок В.3– - Схема запропонованої процедури аугментації даних.

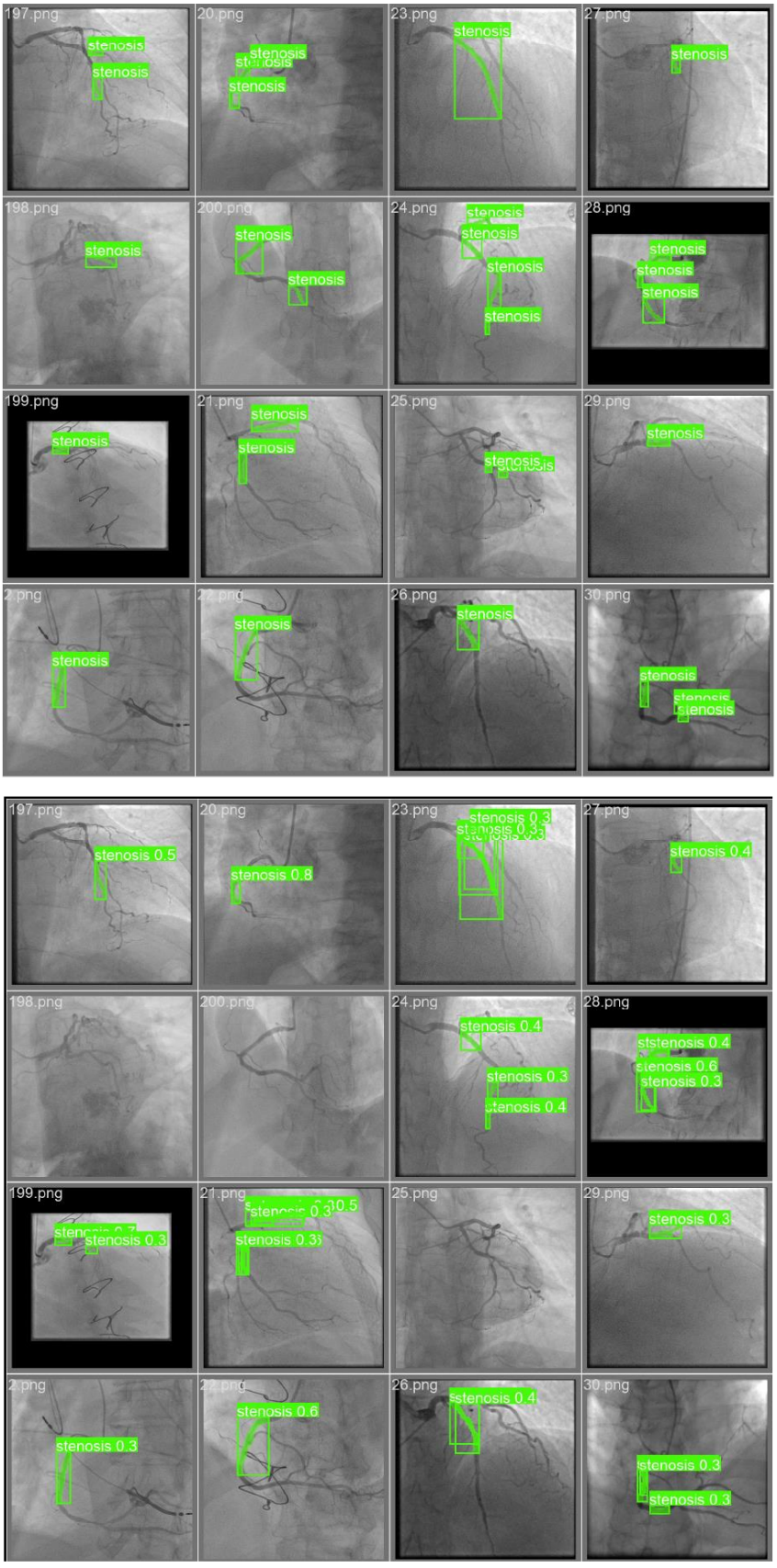
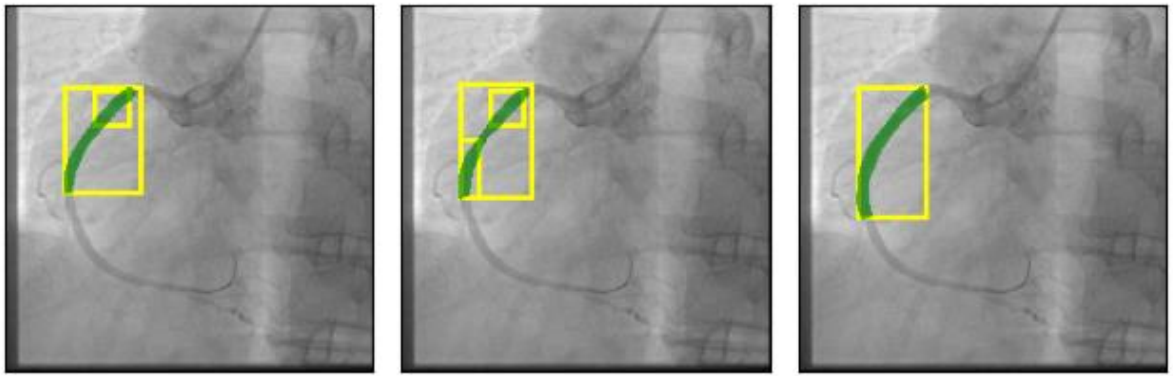


Рисунок В.4 – Результати роботи програми. Угорі: дійсні мітки стенозу для групи тестових зображень. Внизу: прогнози аналога YOLOv8 для тих самих тестових зображень



StenSeg-pl StenSeg-base Правильна область стенозу

StenSeg-pl StenSeg-base Правильна область стенозу

Рисунок В.5 – Результати роботи програми

Таблиця В.1 - Оцінка F1 для валідаційного та тестового набору методів.

	Model	Оцінка F1 на валідаційному наборі	Оцінка F1 на тестовому наборі
Аналоги	YOLOv8 ([15]) YOLOv8n-seg	- 0.35	0.38 0.31
Розроблені моделі	StenSeg-base StenSeg-pl	0.54 0.55	0.51 0.51

Додаток Г (необов'язковий)
Довідка про впровадження

ТОВ «Айбекс айті»
Україна, 21050, м. Вінниця , вул. Театральна 39

3 травня 2025 року

ДОВІДКА

Дана Яцків Діані Олександрівні у тому, що результати, одержані нею в процесі виконання бакалаврської кваліфікаційної роботи, а саме програмний модуль нейромережевої сегментації зображень стенозу коронарних судин серця, планується використати в розробках ТОВ «Айбекс айті».

Директор ТОВ «Айбекс айті» Бойко Р. В.