Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

СЕГМЕНТАЦІЯ ЗОБРАЖЕНЬ З ВИКОРИСТАННЯМ ЗГОРТКОВИХ МЕРЕЖ
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-20-2

Ковальова Л.Г.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник проф. Машталір В.П
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Ковальовій Людмилі Геннадіївні
(прізвище, ім'я, по батькові)

1. Тема роботи Сегментація зображень з використанням згорткових мереж

затверджена наказом університету від 20 травня 2024 року № 464 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 01 червня 2024 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, бібліотека комп'ютерного зору з відкритим кодом Detectron2.

4. Перелік питань, що потрібно опрацювати в роботі

1. Огляд методів сегментації зображень та аналіз їх можливостей.
2. Математична модель сегментації зображень.
3. Розробка комп'ютерної моделі сегментації зображень з архітектурою Mask R-CNN.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми сегментації зображень, постановка задачі, основні архітектури згорткових нейронних мереж для сегментації зображень, характеристики CNN (гіперпараметри, регуляризація, нормалізація вхідних даних, функції втрат), програмна реалізація (налаштування середовища, підготовка даних, налаштування конфігурації та навчання моделі, оцінка результатів навчання), тестування розробленої моделі, перспективи подальшого використання, висновки.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	08.04.2024	
2	Аналіз завдання, підбір літератури	08.04.24-15.04.24	
3	Аналіз літератури з досліджуваної проблеми	16.04.24-19.04.24	
4	Аналіз технічних засобів	20.04.24-26.04.24	
5	Розробка методу	27.04.24-15.05.24	
6	Програмна реалізація	16.05.24-26.05.24	
7	Оформлення пояснювальної записки	27.05.24-01.06.24	
8	Перевірка на плагіат	02.06.24	
9	Рецензування	03.06.24	
10	Підготовка презентації та доповіді	04.06.24-09.06.24	
11	Занесення роботи в електронний архів	10.06.24	
12	Попередній захист кваліфікаційної роботи	10.06.24	

Дата видачі завдання 8 квітня 2024 р.

Студент _____
(підпис)

Керівник роботи _____ ____ проф. Машталір В.П._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 65 с., 1 табл., 28 рис., 1 дод., 31 джерело.

СЕГМЕНТАЦІЯ ЗОБРАЖЕНЬ, ГЛИБОКЕ НАВЧАННЯ, ЗГОРТКОВІ НЕЙРОНІ МЕРЕЖІ, MASK R-CNN, DETECTRON2, GOOGLE COLAB.

Об’єктом роботи є створений набір даних з зображеннями.

Метою роботи є розробка моделі сегментації зображень з використанням глибоких нейронних мереж, яка дозволяє точно визначати контури об’єктів на зображеннях.

Для реалізації мети використано згорткову мережу, зокрема архітектуру Mask R-CNN. Проведено аналіз методів сегментації зображень. Робота включає огляд основних методів сегментації зображень та використання згорткових нейронних мереж у сегментації зображень. Налаштовано конфігурацію моделі та навчено модель на власному наборі даних.

У результаті роботи здійснена програмна реалізація моделі сегментації зображень, яка здатна визначати контури об’єктів на зображеннях.

IMAGE SEGMENTATION, DEEP LEARNING, CONVOLUTIONAL NEURAL NETWORKS, MASK R-CNN, DETECTRON2, GOOGLE COLAB.

The object of the work is a created dataset with images.

The aim of the work is to develop an image segmentation model using deep neural networks, which allows for accurate delineation of object contours in images.

To achieve this aim, a convolutional network, specifically the Mask R-CNN architecture, was used. The work includes a study of image segmentation methods, a review of basic image segmentation techniques, and the use of convolutional neural networks for image segmentation. The model’s configuration was adjusted, and the model was trained on a custom dataset.

As a result of the work, a software implementation of an image segmentation model was developed, capable of accurately delineating object contours in images.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

CNN – Convolutional Neural Networks (згорткові нейроні мережі)

RPN – Region Proposal Network (мережа пропозицій регіонів)

FPN – Feature Pyramid Network (мережа піраміди ознак)

RoI – Region of Interest (область інтересів)

IoU – Intersection over Union (перетин через об’єднання)

ВСТУП

Зі стрімким розвитком у галузях комп’ютерного зору та штучного інтелекту, сегментація зображень набуває все більшої ваги у численних застосунках та системах. Сегментація зображень – це процес розділення зображення на окремі частини або об’єкти для подальшого аналізу та розуміння зображення в цілому. Цей процес є вирішальним для широкого спектру задач, включаючи медичну діагностику, автономне керування транспортними засобами, контроль якості у виробництві, розпізнавання осіб, моніторинг дорожнього руху, а також у сферах сільського господарства та екології [1, 2].

З появою глибокого навчання та згорткових нейронних мереж, сегментація зображень зазнала значних змін. Згорткові мережі забезпечують високу точність та швидкість обробки, використовуючи свою здатність до автоматичного визначення відмінностей у структурі та текстурі об’єктів на зображеннях. Ці методи сприяли автоматизації багатьох завдань, які раніше потребували значних зусиль та часу від фахівців.

Наприклад, у медичній сфері, згорткові нейронні мережі відкривають нові можливості для підвищення ефективності діагностики. Вони можуть асистувати лікарям у виявленні патологій на рентгенівських знімках або МРТ, виділяючи проблемні ділянки для подальшого аналізу. Існують також системи, які здатні автоматично визначати ці патології, хоча вони все ще потребують перевірки медичними фахівцями для остаточного діагнозу. Окрім цього, сегментація зображень відкриває нові можливості в екологічних дослідженнях. Згорткові мережі можуть допомогти в ідентифікації та відстеженні окремих видів у дикій природі, що може сприяти збереженню біорізноманіття та екосистем. Це може бути особливо корисним у випадках, коли потрібно відстежувати популяції тварин або рослин для оцінки впливу змін клімату або людської діяльності. Також, ці технології можуть бути

використані для моніторингу стану сільськогосподарських угідь та вказувати на необхідність їх обробки.

Актуальність роботи з сегментації зображень за допомогою згорткових мереж полягає у потребі поділу цифрового зображення на декілька частин, для подальшої роботи з цим зображенням. Основна мета сегментації – це спрощення або зміна способу представлення зображення для полегшення його аналізу. Це дозволяє виділити ключові об’єкти на зображенні. Використання моделей глибоких нейронних мереж для сегментації зображень надає системі можливість самостійного навчання та використання отриманих знань для обробки даних. Наприклад, такі мережі можуть бути корисними для виявлення або ідентифікації зникаючих видів тварин, що може допомогти у їх збереженні в майбутньому.

1 ОГЛЯД ОСНОВНИХ МЕТОДІВ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ

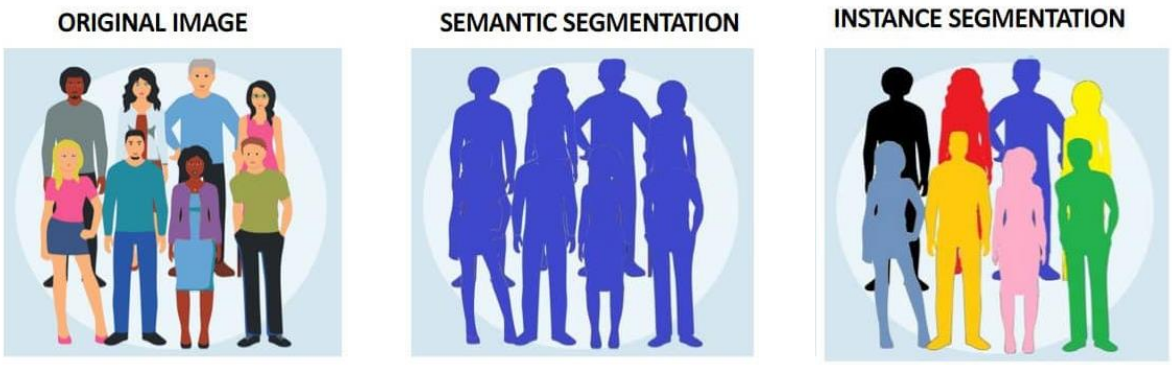
1.1 Загальний огляд сегментації зображень

Сегментація зображень – це процес розділення зображення на окремі сегменти для подальшого аналізу чи обробки. Вона є потужнім інструментом, який застосовується в різних сферах, наприклад, аналізі медичних зображень, створенні карт на основі супутникових знімків, обробці зображень, автоматизованому керуванні автомобілем.

Традиційні методи сегментації використовують різні характеристики пікселів або об’єктів, такі як яскравість, текстура, контраст і т. д., щоб виділяти сегменти на зображенні. На основі цих атрибутів пікселям із схожими візуальними характеристиками присвоюється мітки певних класів. Однак ці методи можуть бути обмеженими в складних сценах або при наявності шуму.

Існує два основних типи сегментації зображень, які широко використовуються на практиці та в дослідженнях. Перший тип – це семантична сегментація. Вона відносить кожен піксель зображення до одного з класів та надає тільки загальний контекст класів. Тобто, наприклад, якщо на зображенні знаходиться група людей (рис. 1.1 а) [3]), за допомогою семантичної сегментація можна виділити фон та цих людей в окремі класи як показано на рис. 1.1 б).

Другий тип – це сегментація екземплярів. Вона вже дозволяє відрізняти окремі екземпляри тих самих об’єктів. Однак сегментація екземплярів не відносить кожен піксель зображення до якогось класу, вона зосереджується саме на виявлені об’єктів класів. Наприклад, на тому ж зображенні з групою людей сегментація екземплярів допоможе нам виділити кожну людину окремо та позначити її як людину, це можна побачити на рис. 1.1 в). Фон в такому випадку сегментуватися вже не буде.



а) б) в)

Рисунок 1.1 – Різниця між семантичною сегментацією та сегментацією екземплярів:
а) оригінальне зображення; б) семантична сегментація;
в) сегментація екземплярів

Ще одним типом сегментації але менш використаним є паноптична сегментація – це поєднання перелічених вище типів. Такий вид сегментації відносить всі пікселі зображення до одного з класів та виділяє окремо різні екземпляри класів.

1.2 Класичні методи сегментації зображень

Перш ніж розглядати сучасні підходи до сегментації зображень на основі глибокого навчання, звернемося до деяких класичних методів, які проклали шлях для розвитку цієї галузі. Ці традиційні підходи використовують різноманітні техніки та алгоритми [4-12], щоб розділити зображення на окремі сегменти, визначити області інтересу та виділити об’єкти на зображеннях. І хоча сучасні згорткові нейронні мережі стали домінуючими у цій сфері, класичні методи все ще залишаються важливими для розуміння та оцінки відмінностей між різними підходами.

1.2.1 Порогова сегментація

Порогова сегментація – це один з найпростіших і найпоширеніших методів сегментації зображень. Цей метод розділяє зображення на дві або більше областей відповідно до заданого порогового значення (рис. 1.2 [4]). Порогове значення визначає деяку яскравість пікселя. Для кожного пікселя зображення знаходиться його яскравість та порівнюється з пороговим значенням: якщо яскравість перебільшує порогове значення, піксель відноситься до одного класу, а якщо переменшує – до іншого [9].

Здебільшого такий метод сегментації використовується до чорно-білих зображень, але також може бути адаптован до кольорових. У такому випадку сегментація застосовується до кожного каналу кольору окремо або до середнього значення інтенсивності всіх каналів.

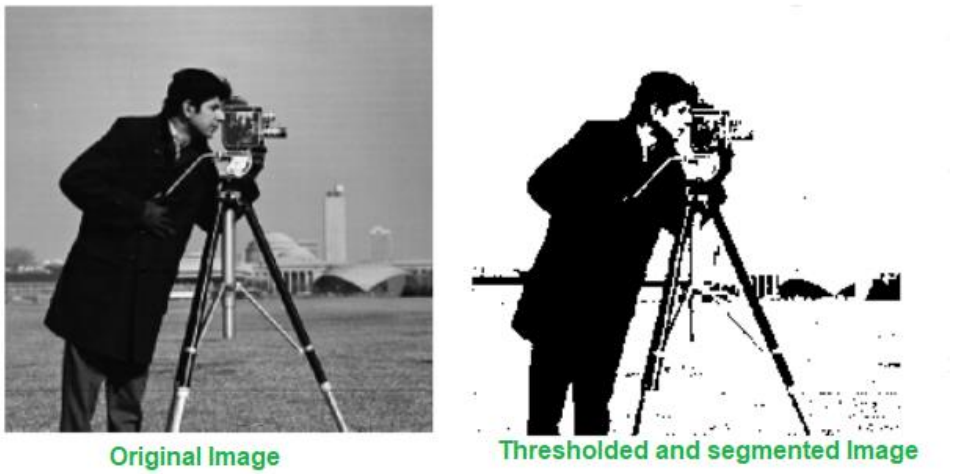


Рисунок 1.2 – Приклад порогової сегментації

Незважаючи на простоту реалізації та швидкість виконання такої сегментації, вона не є дуже ефективною для зображень з нерівномірним освітленням та великим рівнем шуму.

Одним з прикладів застосування порогової сегментації є медична діагностика [10]. І хоча цей метод сегментації вже є досить застарілим, він все ще використовується.

1.2.2 Метод вододілу

Сегментація вододілом – це метод сегментації зображень, що дозволяє виділити схожі об’єкти,. Зображення в цьому методі розглядається як топографічна карта, де пікселі з високою інтенсивністю – вершини (білі області), а з низькою – долини (чорні області). Долини наповнюються водою, що утворює водозбірні басейни (рис. 1.3 [11]). Лінії вододілу визначають межі сегментів.

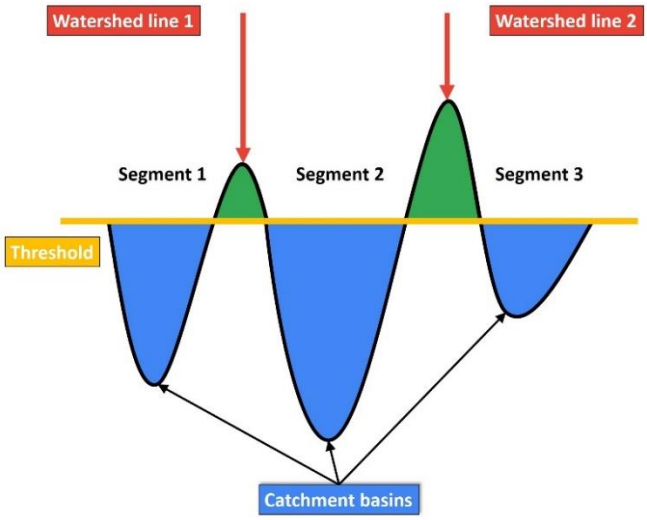


Рисунок 1.3 – Метод вододілу

Як можна побачити на рисунку 1.3, підняття порогу води (жовта лінія) в «басейнах» призведе згодом до об’єднання першого та другого сегментів. Через це додається перша лінія вододілу «watershed line». Відповідно додається і друга лінія вододілу, там де об’єднуються другий та третій сегменти. В результаті отримуються три сегменти.

Цей метод ефективно розбиває зображення на сегменти з однаковою інтенсивністю. Він особливо корисний для відокремлення об’єктів на зображеннях з нерівномірним освітленням або шумом.

1.2.3 Розрізи графів

Алгоритм розрізів графів – це метод сегментації зображень, який базується на моделі зображення у вигляді зваженого неорієнтованого графу. Кожен піксель зображення представляється як вершина, а вага ребер визначається як відмінність між сусідніми пікселями. Алгоритм шукає оптимальний розріз графа, який мінімізує сумарну вагу перерізаних ребер. Після обчислення розрізу графа, пікселі розділяються на дві групи відповідно до цього розрізу. Одна група представляє один клас або область, а інша – інший. Приклад розрізу графа можна побачити на рисунку 1.4 [12].

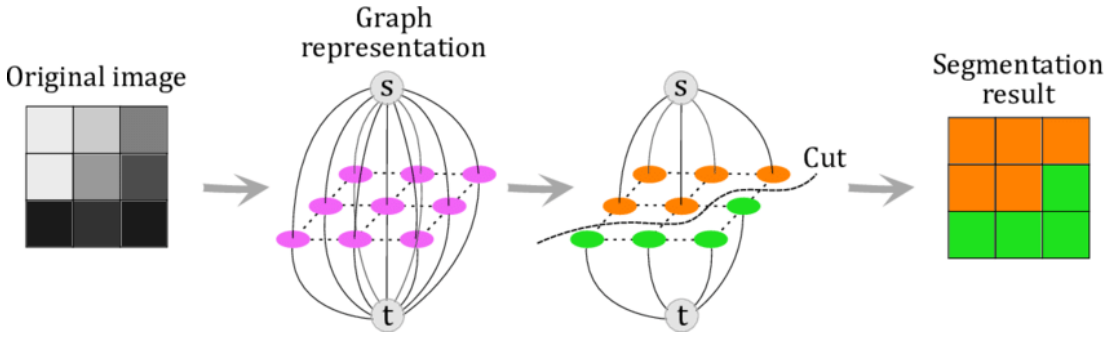


Рисунок 1.4 – Розріз графу

1.3 Використання згорткових мереж у сегментації зображень

Розвиток глибокого навчання, а зокрема згорткових нейронних мереж значно полегшили процес сегментації зображень. Більшість традиційні алгоритмів сегментації зображень базуються на створених вручну функціях машинного навчання таких як HOG або SIFT. Однак створення таких функцій (інженерія функцій) потребує експертних знань предметної області. На відміну від цих алгоритмів, методи глибокого навчання дозволяють забезпечити високу продуктивність, потребуючи для роботи тільки дані [13]. Згорткові мережі автоматично вивчають властивості зображень та роблять

висновки про сегментацію зображень без необхідності вручну вибирати атрибути.

1.3.1 Принцип роботи згорткових нейронних мереж

Одною з ключових операцій в згорткових нейронних мережах є згортка [14]. Процес згортки представляє собою проходження по зображенню та знаходження скалярних добутків ядра згортки та частини зображення відповідного розміру. Ядро згортки є фільтром, який представляється у вигляді невеликої матриці ваг довільного розміру з глибиною відповідною до глибини вхідного зображення. Наприклад для кольорових зображень глибина буде дорівнювати 3, бо зображення має 3 канали які позначають насиченість кожного з кольорів RGB. Результатом такої згортки є карти активації або карти ознак, які відображають присутність певних ознак на вхідному зображенні. Кількість таких карток відповідає кількості фільтрів, які було застосовано до вхідного зображення. Під кінець згортки карти ознак поєднуються до одного образу (рис. 1.5 [15]).

Розмірність виходу під час такої операції зменшується. Для її розрахунку для вхідних даних розміром $W_1 \times H_1 \times D_1$ можна скористатися наступними формулами:

$$W_2 = (W_1 - F + 2P)/S + 1, \quad (1.1)$$

$$H_2 = (H - F + 2P)/S + 1, \quad (1.2)$$

$$D_2 = K, \quad (1.3)$$

де W – ширина;

H – висота;

D – глибина;

F –розмір фільтру;

P – падінг;

S – крок змішення ядра;

K – кількість фільтрів.

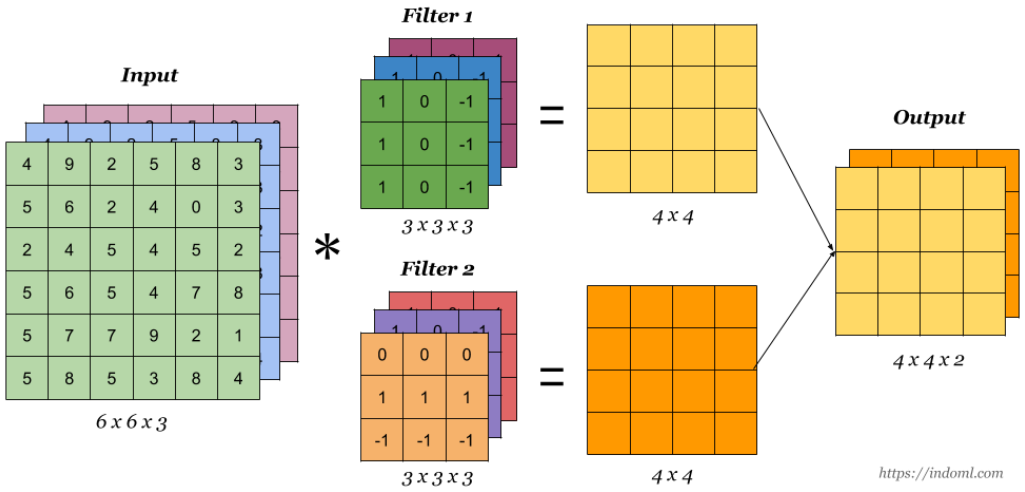


Рисунок 1.5 – Операція згортки

Важливо також зазначити, що крок переміщення фільтру (крок згортки) можна зазначити довільно, наприклад якщо він дорівнює 1 – ядро переміщується на 1 піксель. Це впливає на розмір вихідної карти ознак (збільшення кроку призводить до зменшення карти) та чутливість до локальних ознак, бо чим крок менший, тим детальніше буде досліджуватися зображення.

Ще одним важливим поняттям, яке необхідно розглянути є падінг. Необхідність використання падінгу виникає коли на краях зображення є важлива інформація, яка втрачається під час згортки. Це процес додавання додаткових пікселів навколо зображення, які заповнюються нулями. Таким чином ці пікселі не несуть ніякої інформації але допомагають зберегти розмірність вхідного зображення під час згортки. Падінг можна застосовувати як до вхідного зображення, так і до результату попередньої згортки.

Для того щоб запобігти перенавчанню та збільшити розрахункову

швидкість після шарів згортки застосовується операція пулінгу. Вона зменшує розмірність зображення при цьому зберігаючи виділені ознаки. Процес пулінгу схож із згортанням: по вхідній матриці також проходить фільтр, але цей фільтр вже не містить ваг, замість цього він використовує агрегатні функції.

Існує три основних типи пулінгу:

– максимальний пулінг;
– середній пулінг;
– мінімальний пулінг.

На рисунку 1.6 [16] представлено приклад максимального пулінгу з фільтром 2×2 та кроком 2. Зазвичай крок в пулінгу дорівнює розміру фільтру.

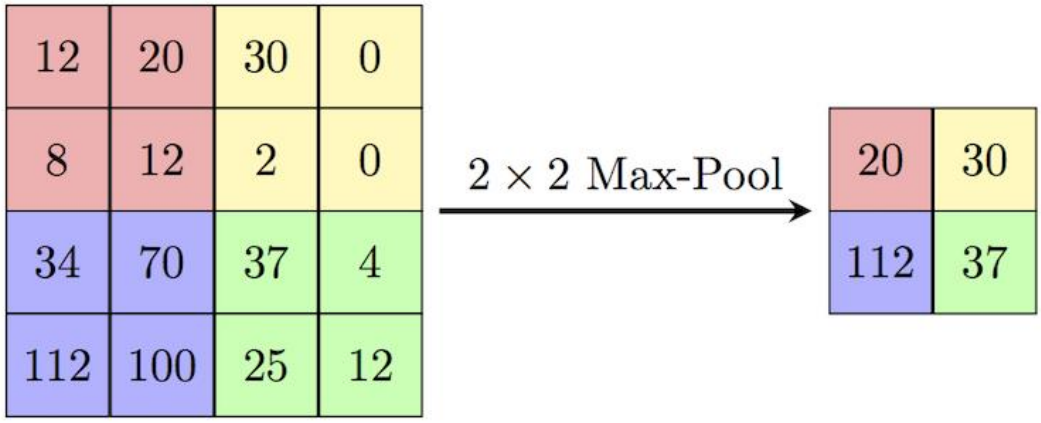


Рисунок 1.6 – Максимальний пулінг з фільтром 2×2

Як можна побачити з цього прикладу для кожної ділянки вхідної матриці розміром відповідним до розміру фільтра знаходиться найбільше значення, яке передається до вихідної матриці. Аналогічно знаходяться середній та мінімальний пулінги, але з використанням відповідних агрегатних функцій.

Після кожного згорткового або повнозв’язного шару використовується функція активації, яка додає нелінійності мережі, що дозволяє їй виконувати більш складні задачі. До найбільш розповсюджених функцій активації відносяться ReLU, Sigmoid, Tanh, Softmax.

У згорткових нейронних мережах зазвичай використовують функцію ReLU, бо вона прискорює навчання та покращує можливість мережі узагальнювати. Ця функція повертає 0 для всіх від'ємних значень та не змінює позитивні. Це дозволяє відкинути всі виходи які не несуть в собі ніяких ознак і враховувати виходи з ознаками [14].

Останнім часом на передній план також виходять налаштовуванні функції активації як спосіб покращення навчання нейронних мереж. Ці функції, як наприклад, налаштовуванні поліноміальні функції активації, дозволяють моделям регулювати параметри активації під час навчання, що може призвести до швидшої збіжності та точнішого навчання [17]. Ця адаптивність підвищує продуктивність мережі в різних завданнях глибокого навчання.

В кінці згорткової нейронної мережі зазвичай розташовуються повнозв'язні шари, які використовуються для остаточної сегментації зображення. Отримані на попередніх шарах карти ознак векторизуються та передаються на повнозв'язні шари у вигляді векторизованих ознак. Ці шари можуть використовувати різні методи сегментації та робити передбачення для кожного пікселя.

Навчання згорткових нейронних мереж – це процес, під час якого мережа навчається визначати та розпізнавати певні патерни та ознаки у вхідних зображеннях. Основна мета навчання полягає у встановленні оптимальних значень ваг, які дозволять мережі ефективно виконувати такі завдання як сегментація зображень.

Процес навчання включає наступні кроки:

Крок 1. Ініціалізація ваг: Спочатку ваги нейронів у всіх шарах мережі ініціалізуються випадковими значеннями.

Крок 2. Прямий прохід: Вхідні дані (в нашому випадку зображення) проходять через всі шари мережі від початку до кінця. Кожен шар обчислює вихід на основі вхідних даних та поточних ваг.

Крок 3. Обчислення втрат: Далі оцінюється, наскільки вихід моделі

відрізняється від бажаного виходу.

Крок 4. Зворотній прохід: Використовуючи алгоритм зворотнього поширення помилки, розраховується градієнт функції втрат для кожного параметра мережі. Цей градієнт показує наскільки зміна параметра вплине на зміну функції втрат.

Крок 5. Оновлення ваг: Ваги кожного нейрона оновлюються з урахуванням градієнтів, обчислених під час зворотного проходу, у напрямку який зменшує витрати. Це здійснюється з використанням методу оптимізації, такого як стохастичний градієнтний спуск або його варіанти.

Крок 6. Повторення: Процес прямого та зворотного проходу (Кроки 2 – 5) повторюється для кожного пакету зображень вхідних даних (батча) у наборі даних. Це триває до досягнення встановленої кількості епох (повних проходів через весь набір даних) або до досягнення критерію зупинки.

Такий цикл навчання допомагає зменшити втрати мережі та підвищити її точність у розпізнаванні та класифікації зображень.

1.3.2 Переваги та обмеження використання згорткових нейронних мереж у сегментації

Згорткові нейронні мережі стали одними з найбільш ефективних інструментів у сегментації зображень. Їх здатність автоматично вивчати та розпізнавати важливі ознаки на зображеннях робить CNN ідеальними для завдань сегментації, де необхідно виділити та класифікувати області на зображенні. Проте, разом з перевагами згорткові нейроні мережі мають і деякі обмеження, які варто враховувати при використанні цих моделей. Розглянемо деякі з них.

До переваг CNN належать:

– автоматизоване вивчення ознак: CNN можуть самостійно вивчати корисні ознаки зображень, що дозволяє їм адаптуватися до різноманітних

завдань сегментації;

– ієрархічна інформація: згорткові шари нейронних мереж автоматично виділяють ознаки на різних рівнях абстракції, що дозволяє їм розпізнавати об’єкти на різних рівнях складності;

– гнучкість: можливість змінювати архітектуру мережі та налаштовувати параметри дозволяє досягати кращої точності для конкретних завдань сегментації.

До обмежень CNN належать:

– вимоги до обчислювальних ресурсів: CNN вимагають значних обчислювальних ресурсів для навчання, що може бути проблематичним для великих зображень або обсягів даних;

– необхідність великого обсягу даних: для досягнення високої точності потрібно мати велику кількість позитивних та негативних прикладів для навчання моделі, що неможливо для деяких галузей використання (наприклад, медицина);

– обробка різних розмірів зображень: нейронні мережі зазвичай вимагають фіксованого розміру вхідних зображень, тому обробка зображень різних розмірів може бути проблематичною.

Незважаючи на ці обмеження, згорткові нейронні мережі залишаються потужним інструментом у завданнях сегментації, особливо коли застосовуються належні стратегії підготовки даних та архітектурні оптимізації.

1.4 Постановка задачі

Таким чином, знаходження ключових об’єктів на зображеннях є однією з актуальних та важливих задач у сфері обробки зображень та комп’ютерного зору. Тому ставиться завдання розробки методу сегментації зображень на основі нейронних згорткових мереж.

Об’єктом роботи є створений набір даних з зображеннями.

Метою роботи є розробка моделі сегментації зображень з використанням глибоких нейронних мереж, яка дозволяє точно визначати контури об’єктів на зображеннях.

Для досягнення мети необхідно вирішити такі завдання:

– провести аналіз існуючих методів сегментації зображень з використанням CNN;

– реалізувати алгоритм сегментації на основі нейронних згорткових мереж;

– реалізувати комп’ютерну модель для сегментації зображень з використанням CNN.

2 МАТЕМАТИЧНІ МОДЕЛІ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ

2.1 Архітектури згорткових нейронних мереж для сегментації зображень

2.1.1 Загальна архітектура CNN для сегментації зображень

Більшість архітектур згорткових нейронних мереж, що використовуються для сегментації зображень, базуються на структурі енкодер-декодер або її варіаціях (рис. 2.1 [18]). Енкодер в цьому контексті відповідає за виділення важливих ознак зображення та його зменшення. Для цього використовуються згорткові та пулінгові шари. Згорткові шари служать для виявлення різних характеристик на зображенні, а пулінгові допомагають зменшити розмірність, зберігаючи при цьому важливі ознаки.

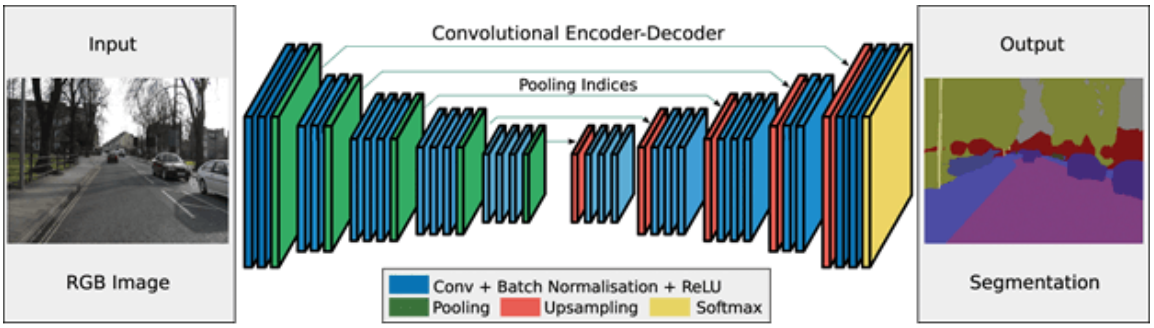


Рисунок 2.1 – Структура згорткового енкодер-декодера

Після енкодера слідує декодер, який відповідає за відновлення розмірності зображення та реконструкцію сегментованих областей. Декодер використовує транспоновані згорткові шари (деконволюція) (рис. 2.2 [19]) або об’єднання для збільшення розмірності зображення, а також згорткові шари для подальшої обробки та реконструкції зображення.

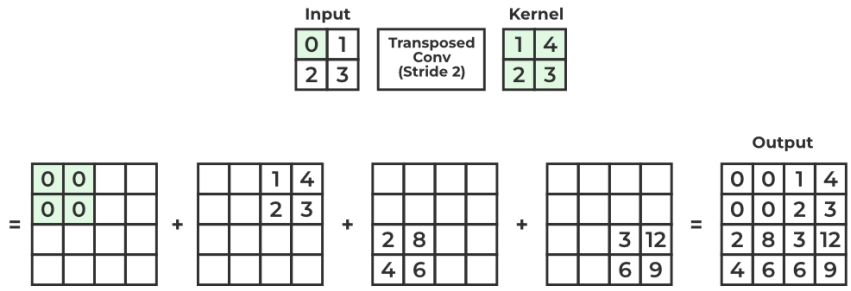


Рисунок 2.2 – Транспонована згортка

Деякі архітектури також використовують пропускні з’єднання, які передають інформацію від енкодера до декодера через пропускні шари. Це сприяє покращенню якості сегментації та збереженню важливих деталей зображення.

Останнім етапом є визначення функції втрат, яка використовується для оцінки різниці між передбаченими та реальними значеннями сегментації. Ця функція допомагає покращити точність моделі та визначити, наскільки ефективно вона виконує сегментацію.

2.1.2 U-Net

Однією з найпопулярніших архітектур для семантичної сегментації є U-Net. Ця архітектура була спеціально розроблена для біомедичних зображень [15], але згодом знайшла широке застосування і в інших областях. U-Net відома своєю здатністю точно виділяти контури об’єктів на зображенні, навіть при обмеженому обсязі даних для навчання. Назва цієї архітектури походить від її структурної схожості з буквою «U» (рис. 2.3 [20]).

Структура U-Net складається з двох основних блоків: енкодера та декодера. Енкодер містить послідовність з двох згорткових шарів з фільтром 3×3, а також максимальні пулінгові шари 2×2. Після кожного згорткового

шару використовується функція активації ReLU для нелінійної зміни ознак. У процесі зменшення розміру зображення, інформація про контекст зберігається за допомогою проміжних ознак.

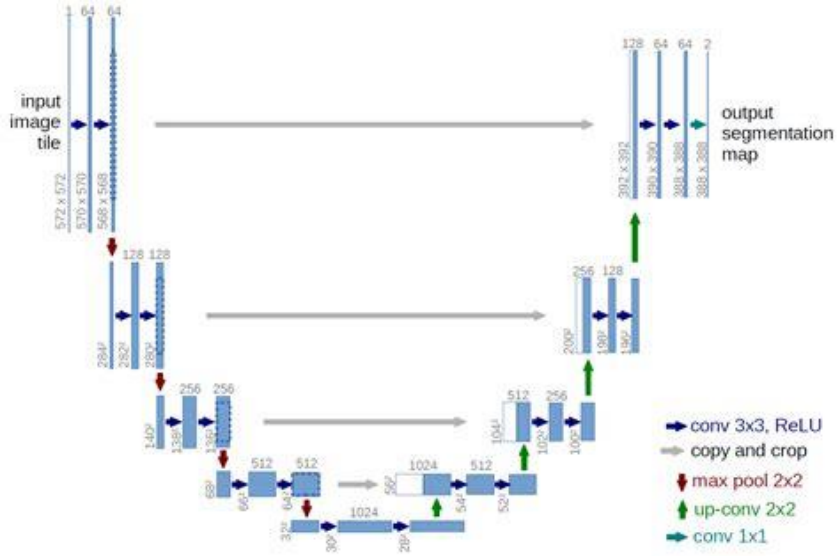


Рисунок 2.3 – Архітектура U-Net

Декодер складається з послідовності транспонованих згорткових шарів та конкатенації ознак з відповідних шарів енкодера. Цей блок поступово збільшує розмір зображення та відтворює його структуру. Після кожного транспонованого згорткового шару також використовується ReLU. Останній шар декодера використовується для отримання кінцевого відтвореного зображення, яке відповідає розміру та структурі вхідного зображення. Для зменшення глибини під кінець робиться згортка з ядром 1×1. В декодері також використовуються «мости» або пропускні з’єднання, які передають інформацію від енкодера до декодера. Ці пропускні з’єднання допомагають зберегти просторову інформацію, яка може бути втрачена в процесі енкодування. Це допомагає декодеру точніше локалізувати особливості на зображенні.

Під час навчання U-Net використовує функцію втрати, таку як перехресна ентропія або кошти втрати Дайса, для порівняння прогнозованих піксельних значень зі справжніми мітками сегментації. Оптимізація

виконується за допомогою алгоритму зворотного поширення помилки з використанням методу стохастичного градієнтного спуску.

2.1.3 SegNet

SegNet – це ще одна популярна архітектура, призначена для сегментації зображень. Вона була розроблена з метою ефективного використання обчислювальних ресурсів та отримання точних результатів в задачах сегментації [18] (рис. 2.4 [21]).

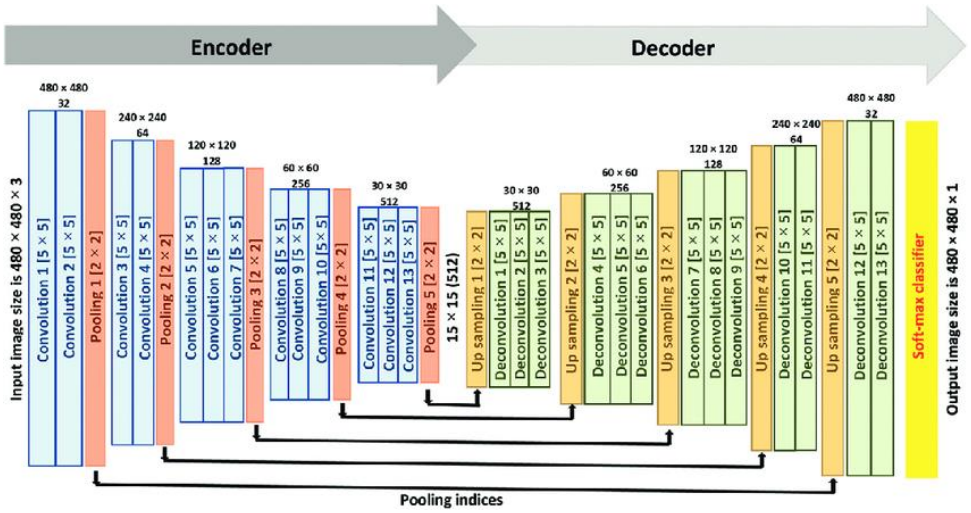


Рисунок 2.4 – Архітектура SegNet

Енкодер в SegNet складається з 13 згорткових шарів, які відповідають першим 13 шарам VGG16. Саме мережа VGG показала, що глибина мережі є особливо важливим компонентом при розв'язанні задач в CNNs [22]. Ці шари використовуються для знаходження ознак на різних рівнях деталізації. Після кожного згорткового шару використовується функція активації ReLU.

Декодер в SegNet використовує транспоновані згорткові шари для відновлення розміру зображення. Однак, на відміну від U-Net, SegNet не використовує пропускні з’єднання між енкодером та декодером. Замість цього, SegNet зберігає індекси максимального пулінгу в енкодері та

використовує їх в декодері для відновлення розмірності. Це дозволяє зберегти просторову інформацію, але зменшує кількість параметрів моделі, що робить SegNet більш ефективною з точки зору обчислень.

Після декодера використовується вихідний шар, який відповідає за відображення кінцевого сегментованого зображення. Він складається з одного згорткового шару, який використовується для відображення кінцевого сегментованого зображення.

Під час навчання SegNet, як і U-Net, використовує такі функції втрати, як перехресна ентропія або кошти втрати Дайса. Оптимізація виконується за допомогою алгоритму зворотного поширення помилки з використанням методу стохастичного градієнтного спуску.

SegNet демонструє гарну продуктивність в задачах сегментації зображень, особливо в медичних застосунках та в сценах з обмеженим обсягом даних для навчання. Ефективність цієї архітектури полягає у здатності точно виділяти області об’єктів на зображенні та відтворювати їх структуру.

2.1.4 DeepLab

DeepLab – це сучасна архітектура для семантичної сегментації зображень, яка була розроблена командою Google AI. Ця архітектура базується на ідеї використання декількох згорткових шарів для ефективного виділення ознак на різних рівнях деталізації та використання модифікованого пулінгу для збереження просторової інформації (рис. 2.5 [23]).

Однією з важливих особливостей архітектури DeepLab є використання згортки з пропусками. Цей метод згортки включає в себе введення так званих “пропусків”, або нулів, в ядро згортки.

На прикладі згортки з 2 пропусками (рис. 2.6 [24]), можна побачити, що під час згортки з фільтром 3×3, фільтр фактично проходить по частинам

зображення розміром 5×5. Це означає, що під час розрахунку враховуються пікселі через один.

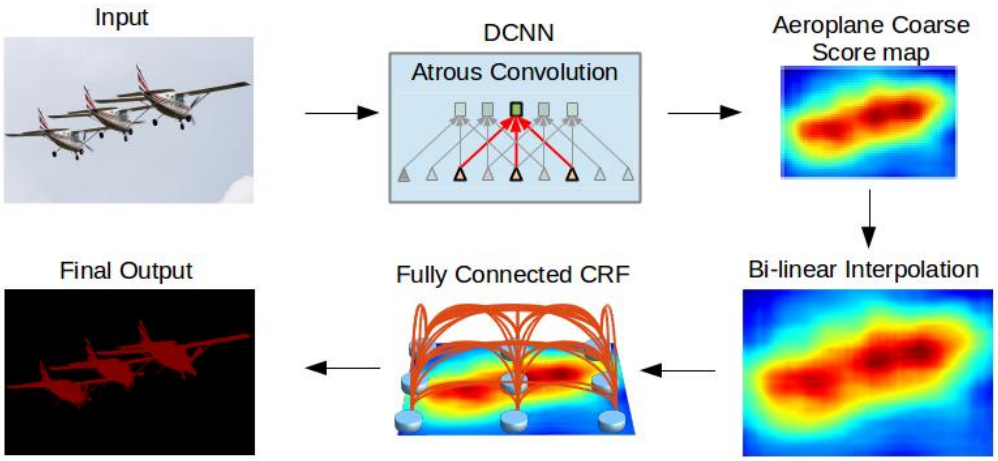


Рисунок 2.5 – DeepLab

Цей підхід дозволяє збільшити поле огляду без збільшення кількості параметрів та обчислювальної складності. Таким чином, згортка з пропусками дозволяє моделі DeepLab ефективно виділяти ознаки на різних масштабах без втрати просторової інформації.

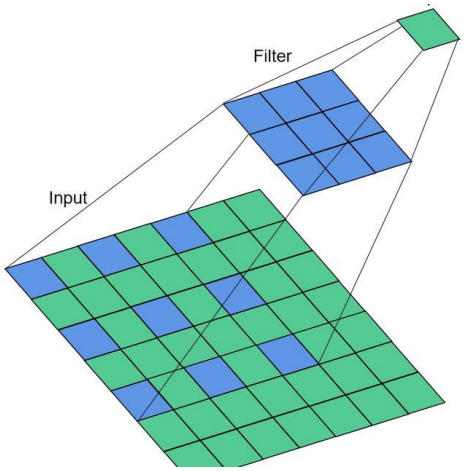


Рисунок 2.6 – Згортка з двома пропусками

Atrous Spatial Pyramid Pooling (ASPP) – це модуль який застосовує до вхідного зображення декілька паралельних пропускних згорток з різними коефіцієнтами пропусків. Використання ASPP є важливою складовою

архітектури DeepLab, що дозволяє моделі враховувати об’єкти різних масштабів та виконувати сегментацію з урахуванням контексту на різних рівнях.

Крім того, для видалення шуму та поліпшення точності сегментації DeepLab використовує метод пост-обробки Fully Connected CRF.

Загальною метою DeepLab є забезпечення точної сегментації об’єктів на зображеннях з високою швидкістю та ефективністю. Ця архітектура знаходить широке застосування у великих проєктах з обробки зображень, таких як аналіз зображень з великою кількістю об’єктів.

2.1.5 FastFCN

FastFCN – це ще одна сучасна архітектура для семантичної сегментації зображень (рис. 2.7 [25]).

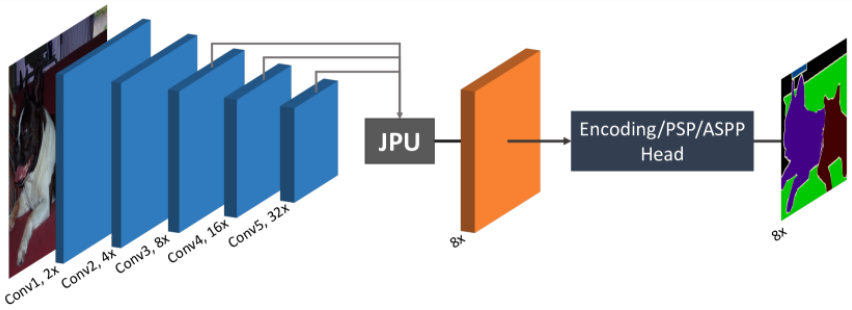


Рисунок 2.7 – FastFCN

Основним компонентом FastFCN є модуль Joint Pyramid Upsampling (JPU). Цей модуль замінює згортки з пропусками, які вимагають багато пам’яті та часу. JPU дозволяє проводити узагальнений апсемплінг на різних рівнях піраміди функцій, що допомагає зберегти більше інформації про контекст при збільшенні роздільної здатності.

FastFCN також використовує швидкі та ефективні згорткові шари та розгортання згорткових шарів. Це дозволяє FastFCN працювати з великими

зображеннями та великими розмірами кроку без значного збільшення обчислювальних витрат.

Замінюючи згортки з пропусками на запропонований модуль JPU, FastFCN досягає найкращих результатів на наборах даних Pascal Context (mIoU 53,13%) та ADE20K (кінцевий бал 0,5584), працюючи в три рази швидше [25].

Загалом, FastFCN – це потужна та ефективна архітектура для сегментації зображень. Вона забезпечує високу точність та швидкість обчислень. Ця архітектура може бути використана у багатьох задачах обробки зображень, включаючи медичну діагностику, аналіз зображень у реальному часі та інші задачі, де важливі швидкість та точність сегментації.

2.1.6 Gated-SCNN

Gated-SCNN (Gated Shape CNN) – це архітектура згорткової нейронної мережі, розроблена для виконання завдань сегментації зображень . Особливий акцент в цій архітектурі зроблено на відокремлення форм та структур об’єктів на зображенні (рис. 2.8 [26]).

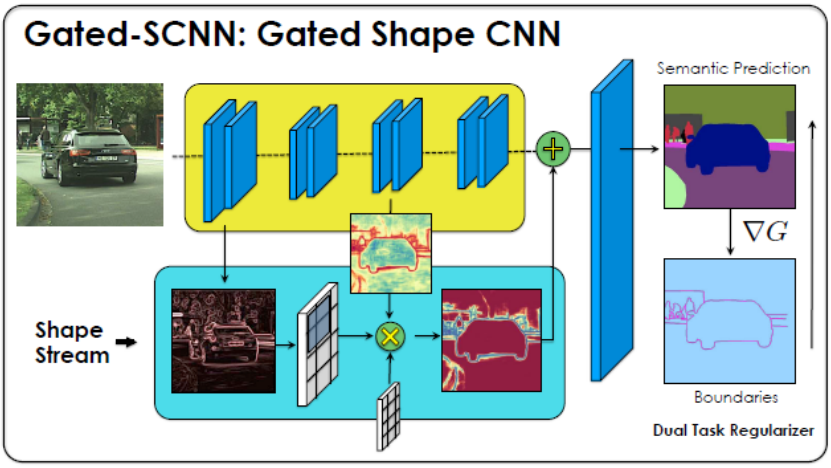


Рисунок 2.8 – Gated-SCNN

Gated-SCNN використовує два потоки обробки, які обробляють зображення паралельно. Один потік обробляє класичну інформацію, а другий – інформацію про форму.

Ключовим елементом цієї архітектури є воротні модулі, які з’єднують проміжні шари двох потоків. Вони використовують активації вищого рівня в класичному потоці для воріт активацій нижчого рівня в потоці форми. Це допомагає видалити шум і дозволяє потоку форми зосередитися тільки на обробці відповідної інформації про межі.

Крім того, Gated-SCNN використовує різноманітні методи оптимізації та регуляризації для забезпечення високої точності та стабільності під час навчання. Це включає в себе використання функцій втрати, таких як перехресна ентропія або кошти втрати Дайса, а також методів регуляризації, таких як виключення або L2 регуляризація.

Gated-SCNN демонструє високу продуктивність в задачах семантичної сегментації зображень, особливо на великих наборах даних. Її ефективність полягає у здатності точно виділяти області об’єктів на зображенні та відтворювати їх структуру, що необхідно для задач, де важливою є точність та стабільність сегментації.

2.1.7 Mask R-CNN

Mask R-CNN – це розширення архітектури Faster R-CNN, яке додає гілку для передбачення маски сегментації до існуючих можливостей виявлення об’єктів (рис. 2.9 [27]). Ця модель є передовою для задач сегментації екземплярів.

Для видобування ознак з вхідного зображення використовується згорткова мережа, відома як «backbone». Для класифікації, визначення обмежувальної рамки та масок використовується «head».

Модуль Region Proposal Network (RPN) відповідає за визначення областей інтересу на зображенні, які потенційно містять об’єкти. Наступним кроком ознаки, що потрапили в різні області, перетворюються до фіксованого розміру. На відміну від Faster R-CNN, де для цього використовувався RoIPool, в Mask R-CNN цей метод був модифікований в RoIAlign. Головна проблема була в тому, що карта ознак, отримана після згортки, має менший розмір, ніж вхідне зображення, і пропорціональні області картки ознак відповідні областям вхідного зображення неможливо відобразити з цілочисельною кількістю ознак. В RoIAlign всі числа залишаються дійсними, а для обчислення значень ознак використовується білінійна інтерполяція за чотирма найближчими цілочисельними точками.

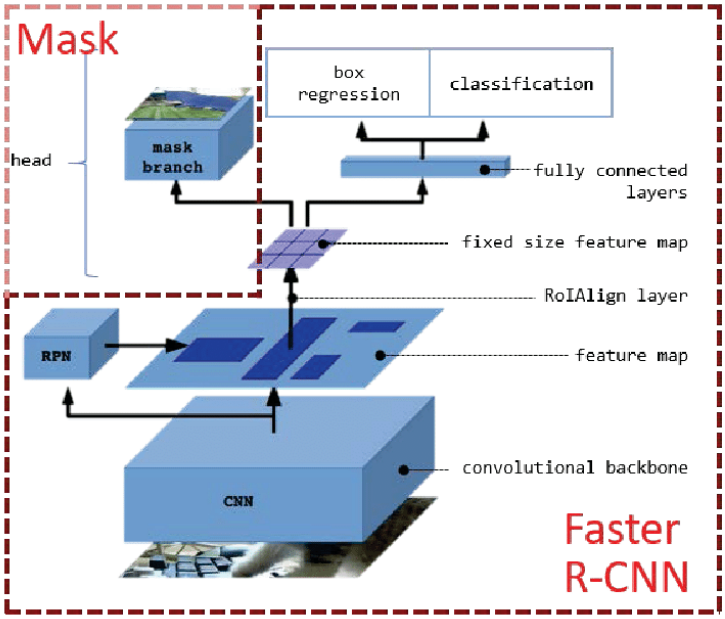


Рисунок 2.9 – Mask R-CNN

Отримана карта ознак фіксованого розміру передається на повнозв’язні шари. Є два паралельних шари, один з яких використовує SoftMax для класифікації, а другий регресію для передбачення обмежувальної рамки.

Процес навчання Mask R-CNN включає в себе використання функції втрати, яка враховує і як точність визначення меж об’єктів, так і точність

визначення масок. Це дозволяє моделі ефективно навчатися визначати як області інтересу, так і їхні маски на зображеннях.

Mask R-CNN демонструє високу продуктивність в задачах сегментації екземплярів та розпізнавання об’єктів, особливо на великих наборах даних. Її ефективність полягає у здатності точно виділяти області об’єктів на зображенні та відтворювати їх структуру.

2.2 Гіперпараметри

Гіперпараметри визначають структуру та параметри моделі, які не змінюються під час навчання і встановлюються перед початком процесу навчання. Вони грають критичну роль у визначенні швидкості та якості навчання моделі. Вони включають такі параметри, як кількість шарів, швидкість навчання, кількість епох та інші.

Кількість шарів визначає глибину мережі. Глибша мережа може виокремлювати більш складні ознаки та патерни, але також більш схильна до перенавчання.

Іншим важливим гіперпараметром є розмір ядра. Більше ядро захоплює більше інформації з вхідного зображення, але при цьому кількість параметрів в моделі збільшується. Коли ж ядро менше – кількість параметрів зменшується, але деякі ознаки можуть бути не виявлені.

Крок зміщення ядра зі збільшенням призводить до зменшення розміру вихідної карти ознак, але також призводить до втрати деяких деталей зображення.

Падінг може допомогти зберегти важливу інформацію на краях зображення, яка інакше могла б зникнути під час згортки. Однак при цьому збільшуються вимоги до обчислювальних ресурсів.

Швидкість навчання визначає, наскільки швидко модель оновлює ваги під час навчання. Занадто висока швидкість навчання може призвести до того,

що модель пропустить оптимальне рішення, тоді як занадто низька швидкість навчання може призвести до того, що модель навчатиметься дуже повільно або застрягне в локальному мінімумі.

Розмір батча визначає кількість прикладів, які модель використовує для одного оновлення ваг. Великі партії можуть прискорити процес навчання, але також можуть призвести до менш стабільного навчання і гіршої загальної продуктивності.

Кількість епох – це кількість разів, коли алгоритм навчання працює через весь набір даних. Занадто мало епох може призвести до недостатнього навчання, тоді як занадто багато епох може призвести до перенавчання.

2.3 Регуляризація

Регуляризація в нейронних мережах допомагає уникнути перенавчання та покращити загальну здатність моделі до узагальнення на нових даних. У контексті сегментації зображень, методи регуляризації можуть допомогти покращити роботу моделі на даних з великою кількістю шуму, невизначеністю або обмеженими обсягами навчальних даних. Розглянемо деякі з них.

L1 та *L2* регуляризації – це методи, які додають штраф до функції втрати, що залежить від величини ваг моделі. *L1* регуляризація додає штраф, пропорційний абсолютному значенню ваг, тоді як *L2* регуляризація додає штраф, пропорційний квадрату ваг.

$$L1 = \lambda \sum_{i=1}^{n} |w_i| , \tag{2.1}$$

$$L2 = \lambda \sum_{i=1}^{n} w_i^2 . \tag{2.2}$$

Метод виключення випадково “відключає” деякі нейрони під час навчання, що змушує мережу вчитися з меншою кількістю нейронів. Це

допомагає запобігти перенавчанню, оскільки мережа не може надто покладатися на окремі нейрони. Існує також варіація методу виключення – просторове виключення, який призначено спеціально для зображень. Воно вимикає випадкові канали зображень замість окремих нейронів, зменшуючи кореляцію між каналами та запобігаючи перенавчанню.

Метод нормалізації батча нормалізує активації нейронів в мережі для кожного міні-батчу. Це допомагає прискорити навчання та зменшити вплив початкової ініціалізації ваг.

Метод аугментації даних збільшує набір даних за допомогою створення модифікованих версій існуючих зображень. Для створення нових зображень використовуються різні техніки, такі як обертання, масштабування, зсув та інше. Це допомагає моделі краще узагальнювати, оскільки вона вчиться на більш різноманітному наборі даних.

Використання регуляризації може значно покращити продуктивність моделей.

2.4 Нормалізація вхідних даних

Нормалізація вхідних даних є важливим кроком у підготовці даних для навчання моделей глибокого навчання. Нормалізація допомагає моделі швидше збігатися і зменшує шанси отримання нестабільних та неефективних моделей. Розглянемо декілька поширених прикладів.

Мінімаксна нормалізація масштабує дані так, що всі значення лежать в діапазоні від 0 до 1. Це досягається шляхом віднімання мінімального значення в наборі даних від кожного значення, а потім ділення на різницю максимального та мінімального значень.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}. \quad (2.3)$$

Z-нормалізація (стандартизація) масштабує дані так, що середнє значення набору даних стає 0, а стандартне відхилення – 1. Це досягається шляхом віднімання середнього значення набору даних від кожного значення, а потім ділення на стандартне відхилення.

$$x' = \frac{x - \text{avg}(x)}{\sigma}. \tag{2.4}$$

В нормалізації за допомогою декількох масштабів кожен канал зображення (червоний, зелений, синій) нормалізується окремо.

2.5 Функції втрат

Функції втрат використовуються для оцінки того, наскільки добре модель виконує своє завдання. Вони вимірюють різницю між передбаченими моделлю виходами та істинними значеннями. Розглянемо функції втрат, які часто використовуються для задач сегментації зображень.

Перехресна ентропія використовується в задачах класифікації. Вона вимірює відстань між передбаченими мітками моделі та істинними мітками. Для задач сегментації зображень використовується її варіант – бінарна перехресна ентропія, яка вимірює відстань між передбаченими масками сегментації p та істинними масками y. Однак ця функція втрат не підходить для багатокласової сегментації .

$$L(y, p) = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)]. \tag{2.5}$$

Втрата Дайса вимірює схожість між передбаченими та істинними масками сегментації. Вона враховує обидва зображення, що порівнюються, і допомагає моделі краще визначати області об’єктів.

$$L(y,p) = 1 - 2\frac{y \cap p}{y+p}. \quad (2.6)$$

Функція втрат Джакарда (IoU) вимірює відношення області перетину передбачених та істинних масок сегментації до області їх об'єднання. Вона допомагає моделі краще визначати границі об'єктів.

$$IoU(y,p) = \frac{y \cap p}{y \cup p}. \quad (2.7)$$

Втрата Тверського – це узагальнення втрати Дайса, яке дозволяє контролювати важливість позитивних та негативних помилок. Це може бути корисним в задачах сегментації зображень, де об'єкти займають невелику частину зображення.

$$TL(y,p) = \frac{\sum_{i=1}^{N} y_i p_i}{\sum_{i=1}^{N} y_i p_i + \alpha \sum_{i=1}^{N} y_i (1-p_i) + \beta \sum_{i=1}^{N} (1-y_i) p_i}, \quad (2.8)$$

де α – параметр, який регулює важливість позитивних помилок;

β – параметр, який регулює важливість негативних помилок.

Вибір конкретної функції втрат залежить від конкретної задачі та даних. Наприклад, якщо важливо точно визначити границі об'єктів, може бути корисно використовувати втрату Дайса або IoU. З іншого боку, якщо важливо правильно класифікувати кожен піксель, може бути корисно використовувати перехресну ентропію.

2.6 Набори даних

У роботі з сегментацією зображень використовують різноманітні набори даних, які відповідають різним вимогам та цілям дослідження.

COCO (Common Objects in Context) містить більше мільйона зображень, які мають 80 класів об’єктів. COCO включає зображення різних розмірів та складності, що робить його універсальним набором даних для різних задач.

PASCAL VOC (Visual Object Classes) містить зображення 20 класів об’єктів, серед яких люди, автомобілі, тварини та інші. Він часто використовується для оцінки алгоритмів сегментації та детекції об’єктів.

Cityscapes містить зображення міських сцен, зокрема вулиць, будівель, людей та автомобілей. Він використовується для дослідження алгоритмів сегментації в умовах міського середовища.

ADE20K містить більше 20 тисяч зображень та 150 класів об’єктів, з різноманітними сценами, такими як природа, меблі, техніка тощо. Цей набір даних використовується для дослідження алгоритмів сегментації в різноманітних середовищах.

Незважаючи на чималу кількість наборів даних важливо зазначити, що багато з них складаються з зображень, які були анотовані вручну. І хоча ці анотації надають цінні дані для тренування та оцінки моделей сегментації, вони можуть містити помилки та недоліки через людський фактор. Для уникнення помилок ручної анотації створюють штучно згенеровані набори даних, проте їх використання може вплинути на здатність моделі працювати на реальних даних. Це зумовлено тим, що штучно створені набори даних можуть бути менш реалістичними порівняно з реальними фотографіями.

3 КОМП'ЮТЕРНА МОДЕЛЬ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ

3.1 Обґрунтування вибору середовища програмної реалізації

Для розробки та тестування моделі сегментації зображень у даній роботі було обрано хмарний сервіс Google Colab [28] та бібліотеку Detectron2 [29]. Цей вибір обґрунтований кількома важливими факторами, які роблять Google Colab і Detectron2 гарними інструментами для роботи із згортковими нейронними мережами. Модель, яка була використана для сегментації зображень – Mask R-CNN, одна з найбільш ефективних архітектур для завдань сегментації екземплярів. Для навчання моделі було створено власний набір даних (ДОДАТОК А) за допомогою інструменту для створення анотацій Make Sense [30].

Google Colab – це хмарний сервіс, який надає безкоштовний доступ до потужних обчислювальних ресурсів, таких як графічні процесори та тензорні процесори, що є важливим для навчання глибоких нейронних мереж. Використання графічного процесора дозволяє значно прискорити процес навчання моделей. Хоча ресурси, які надає Google Colab, є обмеженими, їх достатньо для навчання на власному наборі даних, оскільки він не є дуже великим. Це робить Google Colab ідеальним вибором для даної роботи, забезпечуючи необхідну обчислювальну потужність та зручність роботи.

Google Colab працює повністю у веббраузері на основі Jupyter Notebook, що забезпечує зручність і доступність для користувачів. Це дозволяє запускати і тестувати код без необхідності встановлення додаткового програмного забезпечення на локальному комп'ютері, що робить розробку більш гнучкою та мобільною. Крім того, інтеграція з Google Drive дозволяє легко завантажувати та зберігати дані. Це зручно для роботи з наборами зображень та анотацій, необхідних для навчання моделі.

Detectron2, розроблений Facebook AI Research, є одним з найсучасніших

і найпопулярніших фреймворків для комп’ютерного зору. Він підтримує різні архітектури для сегментації зображень, включаючи Mask R-CNN, яка була обрана для даної роботи через свою високу ефективність. Detectron2 відрізняється гнучкістю та модульністю, що дозволяє легко налаштовувати та розширювати моделі відповідно до специфічних потреб.

Інтеграція Detectron2 з PyTorch забезпечує доступ до інших інструментів та бібліотек для глибокого навчання. PyTorch є дуже популярним серед дослідників завдяки своїй простоті у використанні та широким можливостям. Крім того, Detectron2 надає доступ до багатьох попередньо натренованих моделей, що дозволяє швидко розпочати розробку і отримати хороші результати навіть на невеликих наборах даних.

Інструмент Make Sense, який було використано для анотації, надає зручний інтерфейс для ручного маркування зображень. Він підтримує різні типи анотацій, такі як бокси, полігони та маски, що дозволяє точно і детально маркувати об’єкти на зображеннях. Використання цього інструменту забезпечило якісне і швидке створення анотацій для навчання моделі, що є важливим для досягнення високої точності сегментації.

Таким чином, вибір Google Colab як середовища реалізації, Detectron2 як основного фреймворку та Make Sense для створення анотацій був зумовлений їх перевагами в доступності, продуктивності, зручності використання та підтримці сучасних технологій комп’ютерного зору. Використання моделі Mask R-CNN у цьому середовищі дозволяє ефективно реалізовувати та тестувати модель, забезпечуючи високу якість та точність сегментації.

3.2 Програмна реалізація

3.2.1 Налаштування середовища

Для початку роботи з Detectron2 необхідно клонувати репозиторій Detectron2 з GitHub та встановити залежності. Для цього використовується модуль distutils.core для запуску файлу setup.py, який визначає залежності

проєкту.

Лістинг 3.1 Встановлення Detectron2:

```
!python -m pip install pyyaml==5.1
import sys, os, distutils.core
!git clone 'https://github.com/facebookresearch/detectron2'
dist = distutils.core.run_setup("./detectron2/setup.py")
!python -m pip install {' '.join([f"'{x}'" for x in dist.install_requires])}
sys.path.insert(0, os.path.abspath('./detectron2'))
```

3.2.2 Підготовка даних

Першим кроком у процесі реалізації є підготовка даних. Для цього було створено набір даних з зображеннями качок та його анотацію. Набір даних містить 45 зображень: 36 в тренувальному та 9 у валідаційному наборі. Анотації включають полігони, що окреслюють контури качок на зображеннях. Сервіс Make Sense надає можливість зберігати анотації у форматі JSON, сумісному з Detectron2 завдяки якому було створено два файли «annotations.json». Інформація щодо масок сегментації в ньому зберігається в наступному вигляді.

Лістинг 3.2 Формат анотацій:

```
{  "info": {"description": "my-project-name"},
  "images": [
    {"id": 1, "width": 1072, "height": 1026, "file_name": "IMG_0001.jpg"},
    ..... ],
 "annotations": [
    { "id": 0,
      "iscrowd": 0,
```

"image_id": 1,

"category_id": 1,

"segmentation": [[x1, y1, x2, y2, ..., xn, yn]],

"bbox": [xmin, ymin, xmax, ymax] } ……],

"categories": [

{"id": 1, "name": "duck"}]}

Для завантаження даних в Google Colab використовується Google Drive, де зберігається набір даних.

Лістинг 3.3 Підключення до Google Drive:

```
drive.mount('/content/drive')
```

Для зручності роботи набір даних копіюється в робочу директорію.

Лістинг 3.4 Копіювання набору даних до робочої директорії:

```
drive_dataset_path = '/content/drive/MyDrive/MaskRCNN/duck_dataset'
local_dataset_path = '/content/images'
shutil.copytree(drive_dataset_path, local_dataset_path)
```

Наступним кроком необхідно зареєструвати набір даних в Detectron2, що дозволить бібліотеці коректно завантажувати та обробляти дані для тренування та валідації моделей. Для цього реєструємо набір даних у форматі COCO, використовуючи функцію register_coco_instances:

Лістинг 3.5 Реєстрація набору даних в Detectron2:

```
register_coco_instances("duck_train", {}, "images/train/annotations.json", "images/train")
register_coco_instances("duck_val", {}, "images/val/annotations.json", "images/val")
```

Після реєстрації набору даних отримуємо метадані, які містять інформацію про цей набір даних, такі як категорії об’єктів, кольори для візуалізації тощо.

Лістинг 3.6 Отримання метаданих:

```
duck_metadata = MetadataCatalog.get("duck_train")
```

Щоб переконатися що все правильно зареєстровано та анотації відображаються коректно, візуалізуються декілька зразків з набору даних (рис. 3.1).

Рисунок 3.1 – Приклад відображення зображення з анотацією

Лістинг 3.7 Виведення прикладів сегментованих зображень

```
dataset_dicts = DatasetCatalog.get("duck_train")
for d in random.sample(dataset_dicts, 3):
    img = cv2.imread(d["file_name"])
    visualizer = Visualizer(img[:, :, ::-1], metadata=duck_metadata, scale=0.5)
```

out = visualizer.draw_dataset_dict(d)

cv2_imshow(out.get_image()[:, :, ::-1])

На рисунку 3.1 можна побачити, що обмежувальна рамка та маска відповідають об’єкту на зображенні, тобто дані коректні.

3.2.3 Налаштування конфігурації та навчання моделі

Для успішного тренування моделі Mask R-CNN на власному наборі даних необхідно налаштувати конфігураційні параметри, які визначають архітектуру моделі, параметри тренування, оптимізатори та інші важливі налаштування. Конфігураційний файл включає наступні параметри:

- архітектура моделі;
- ваги моделі;
- параметри навчання.

Для задачі сегментації зображень вибираємо архітектуру Mask R-CNN з ResNet-50 як основною мережею. ResNet-50 має 50 згорткових шарів з активаційною функцією ReLU та батч нормалізацією після кожного шара згортки, що дозволяє їй ефективно вивчати високорівневі ознаки на зображеннях. Глибокі згорткові шари дозволяють моделі автоматично визначати складні зразки та ознаки, які можуть бути корисні для виявлення об’єктів. Крім того, модель використовує Feature Pyramid Network (FPN), що дозволяє створити піраміду функцій для виявлення об’єктів на різних масштабах. Це важливо для виявлення об’єктів різних розмірів на зображенні.

Лістинг 3.8 Завантаження моделі Mask R-CNN з model_zoo:

cfg = get_cfg()

cfg.merge_from_file(model_zoo.get_config_file("COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml"))

Модель ініціалізується вагами, натренованими на наборі даних COCO. Це дозволяє використовувати попередні знання моделі для покращення результатів тренування на власному наборі даних з меншим об’ємом даних.

Лістинг 3.9 Завантаження ваг для моделі Mask R-CNN з model_zoo:

cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml")

Основні параметри навчання налаштовуються для забезпечення ефективного тренування моделі.

Кількість ітерацій «cfg.SOLVER.MAX_ITER» встановлюється на 300 ітерацій. Це надає моделі достатньо часу для навчання на створеному наборі даних. Для більших та складніших наборів даних кількість ітерацій може бути збільшена.

Швидкість навчання «cfg.SOLVER.BASE_LR» дорівнює 0.00025. Цей параметр визначає, наскільки швидко модель оновлює свої ваги під час тренування. Було обрано помірне значення, щоб уникнути нестабільності навчання.

Розмір пакета «cfg.SOLVER.IMS_PER_BATCH» встановлюється на 2.

Кількість паралельних процесів для завантаження даних «cfg.DATALOADER.NUM_WORKERS» дорівнює 2. Використання паралельних процесів дозволяє значно прискорити процес завантаження та підготовки даних.

Параметр «cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE» встановлено як 128. Він визначає кількість зразків RoI на зображення для тренування голови моделі. Значення 128 забезпечує баланс між точністю та швидкістю тренування. Це допомагає моделі навчатися ефективно без надмірного збільшення часу на кожну ітерацію.

Параметр «cfg.MODEL.ROI_HEADS.NUM_CLASSES» дорівнює 1,

оскільки набір даних містить лише один клас об’єктів для сегментації.

Під час навчання виводяться деталі про втрати кожні 20 ітерацій, що допомагає зрозуміти як добре навчається модель. Ці дані наведено у таблиці 3.1.

Таблиця 3.1 – Втрати під час навчання

№ Ітер.	**Втрати**				**Час**		**Використання пам’яті**
	Загальні	**Обм. Рамки**	**Масок**	**Класифікації**	**Навчання**	**Обробки**	
19	2,121	0,7654	0,6882	0,6233	1,0302	0,5798	2072M
39	2,087	0,8707	0,6598	0,5504	1,1930	0,4639	2245M
59	1,901	0,7981	0,5968	0,4521	1,0860	0,7284	2246M
79	1,713	0,7906	0,5258	0,3701	1,0854	0,6030	2246M
99	1,593	0,8122	0,4453	0,3198	1,0865	0,6048	2247M
119	1,387	0,7959	0,3388	0,2464	1,0753	0,5175	2351M
139	1,283	0,7683	0,2878	0,2148	1,1077	0,8061	2351M
159	1,136	0,6943	0,2465	0,1828	1,0975	0,5293	2351M
179	0,9419	0,5767	0,1951	0,1421	1,0967	0,5878	2351M
199	0,8405	0,4674	0,1849	0,1249	1,0903	0,5425	2351M
219	0,6857	0,3379	0,1608	0,1262	1,0894	0,6088	2351M
239	0,5991	0,3235	0,1454	0,09945	1,0874	0,5826	2351M
259	0,5851	0,319	0,1617	0,1008	1,0824	0,5088	2351M
279	0,4823	0,2438	0,1447	0,0813	1,0824	0,5760	2351M
299	0,482	0,2492	0,152	0,08454	1,0858	0,6194	2351M

З результатів видно, що загальні втрати поступово знижуються з кожною ітерацією, що свідчить про успішне навчання моделі. Найбільший внесок у загальні втрати на початкових етапах вносять втрати класифікації та обмежувальних рамок. Втрати масок також значні, але вони знижуються швидше. Час на ітерацію стабільний, що свідчить про ефективне використання

обчислювальних ресурсів. Використання пам’яті поступово збільшується до 2351 МБ, що є прийнятним.

Результати вказують на те, що модель успішно навчається і поступово покращує свої передбачення. Подальші ітерації, можливо, вимагатимуть додаткового налаштування параметрів, щоб зменшити втрати ще більше та покращити загальну ефективність моделі.

3.2.4 Оцінка результатів навчання

Результати навчання моделі оцінюються на основі метрик середньої влучності (AP) та середньої повноти (AR) для задач детекції об’єктів та сегментації. Ці метрики дозволяють виміряти влучність та повноту моделі на різних рівнях інтерсекції об’єму (IoU).

Влучність P визначає, яка частка знайдених об’єктів є правильними. Вона обчислюється за формулою:

$$P = \frac{TP}{TP+FP}, \tag{3.1}$$

де TP – кількість правильно знайдених об’єктів;

FP – кількість неправильно знайдених об’єктів.

Повнота R визначає, яку частку правильних об’єктів було знайдено. Вона обчислюється за формулою:

$$R = \frac{TP}{TP+FN}, \tag{3.2}$$

де TP – кількість правильно знайдених об’єктів;

FN – кількість об’єктів, які не були знайдені.

Середня влучність є інтегралом кривої влучність-повнота. Вона

обчислюється як середнє значення точності при різних значеннях повноти. Для COCO середня влучність вимірюється на IoU від 0.5 до 0.95 з кроком 0.05, що дозволяє врахувати різні ступені відповідності між передбаченням та реальними об’єктами.

Середня повнота є середнім значенням повноти при різних значеннях IoU. Вона вимірюється для різної кількості максимально виявлених об’єктів і розміру об’єктів.

Для оцінки використовується COCOEvaluator та спеціально створений тестовий завантажувач даних.

Лістинг 3.10 Оцінка результатів навчання:

```
# Вибірка тестового датасету для оцінки моделі
evaluator = COCOEvaluator("duck_val", output_dir="./output/")
val_loader = build_detection_test_loader(cfg, "duck_val")
# Оцінка результатів моделі
evaluation_results = inference_on_dataset(trainer.model, val_loader, evaluator)
print(evaluation_results)
```

На рисунку 3.2 наведено метрики, отримані для детекції об’єктів.

AP	AP50	AP75	APs	APm	APl
60.129	79.719	70.012	0.000	49.785	71.548

Рисунок 3.2 – Середня влучність детекції об’єктів

На рисунку 3.3 наведено метрики, отримані для сегментації.

AP	AP50	AP75	APs	APm	APl
58.673	79.719	77.758	0.000	49.878	67.018

Рисунок 3.3 – Середня влучність сегментації

Загальне значення середньої влучності для задачі детекції склало 60,129%, а для сегментації – 58,673%. Це вказує на досить високу загальну точність моделі.

Високі показники AP при IoU = 0,50 (AP50) свідчать про високу точність моделі при менш строгих критеріях. Високі значення AP при IoU = 0,75 (AP75) свідчать про також хорошу продуктивність при строгих критеріях.

Далі розглянемо середні показники AP для малих (APs), середніх (APm) та великих (APl) об’єктів. Відсутність результатів для малих об’єктів (0.000%) зумовлена відсутністю таких об’єктів у наборі даних. Натомість високі значення для великих об’єктів (71,548% для детекції та 67,018% для сегментації) вказують на сильну продуктивність моделі для великих об’єктів.

Отримані результати свідчать про те, що модель має високу влучність та повноту для великих об’єктів та середні показники для середніх об’єктів. Модель може потребувати додаткового навчання або оптимізації для покращення продуктивності на середніх об’єктах. Загальна продуктивність моделі є достатньо високою для практичного використання у задачах детекції та сегментації об’єктів.

3.2.5 Візуалізація результатів сегментації на валідаційному наборі

Після проведення оцінки моделі, корисно візуалізувати результати роботи моделі на валідаційних зображеннях. Це дозволяє наочно побачити, як модель виконує сегментацію об’єктів. Для цього оновляємо модель, завантажуючи ваги натренованої моделі, встановлюємо поріг для сегментування та візуалізуємо 3 зображення (рис. 3.4, рис. 3.5).

Лістинг 3.11 Візуалізація отриманої сегментації зображень

cfg.MODEL.WEIGHTS = os.path.join(cfg.OUTPUT_DIR,

```
"model_final.pth")
        cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.7
predictor = DefaultPredictor(cfg)
        dataset_dicts = DatasetCatalog.get("duck_val")
        for d in random.sample(dataset_dicts, 3):
            im = cv2.imread(d["file_name"])
            outputs = predictor(im)
            v = Visualizer(im[:, :, ::-1],
                        metadata=duck_metadata,
                        scale=0.5,
                        instance_mode=ColorMode.IMAGE_BW
            )
            out = v.draw_instance_predictions(outputs["instances"].to("cpu"))
    cv2_imshow(out.get_image()[:, :, ::-1])
```

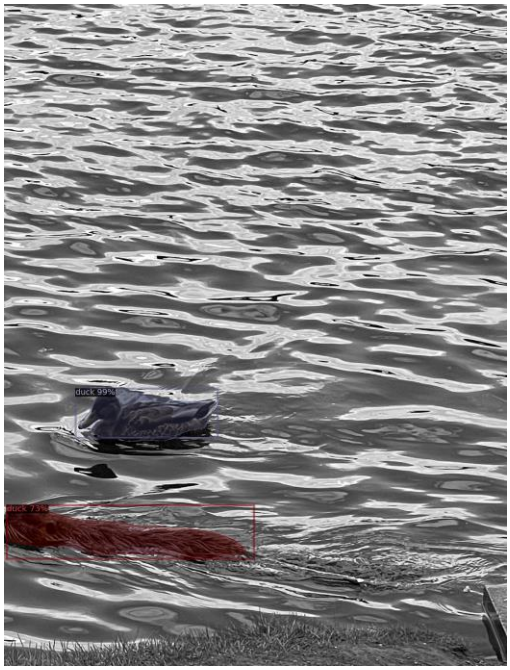


Рисунок 3.4 – Приклад сегментованого зображення з валідаційного набору

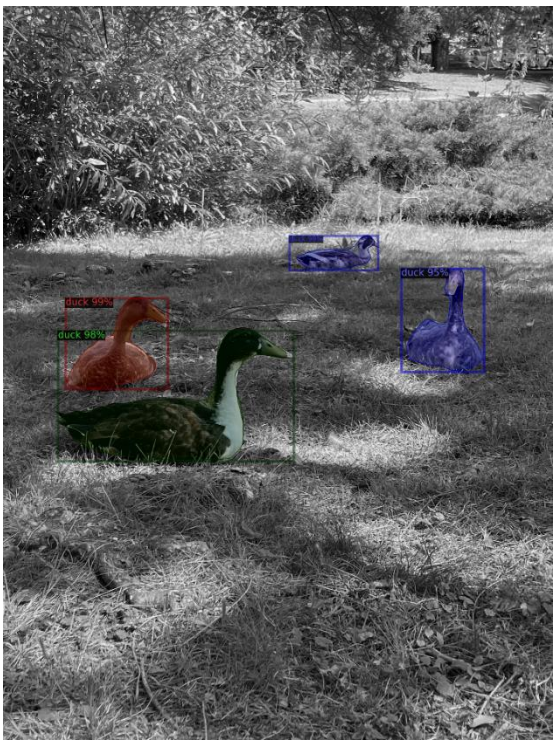


Рисунок 3.5 – Приклад сегментованого зображення з валідаційного набору

На рисунку 3.4 є два об’єкти: качка, яка була правильно класифікована та сегментована з високою точністю (99%), та бобер, який був неправильно розпізнаний як качка але точність вже складає 73%.

На рисунку 3.5 всі качки були вірно розпізнані та сегментовані з високою точністю (95-99%)

Візуалізовані результати підтверджують кількісні метрики оцінки моделі. Для вирішення проблеми з неправильною сегментацією бобра налаштуємо поріг для тестування.

Лістинг 3.12 Зміна порогу для тестування

cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.8

В результаті збільшення порогу, на рис. 3.6 можна побачити, що проблема з неправильною сегментацією, при якій бобра було класифіковано та сегментовано як качку (рис. 3.4), вирішилась.

Рисунок 3.6 – Приклад сегментованого зображення після збільшення порогу

3.2.6 Збереження моделі

Щоб зберегти та використовувати модель в подальшому, потрібно зберегти наступні елементи:

– ваги моделі (model_final.pth): файл з вагами натренованої моделі;

– метадані (duck_metadata.json): файл, що містить інформацію про класи та інші параметри набору даних;

– конфігураційний файл (config.yaml): файл, що містить конфігурацію моделі та параметри тренування.

Лістинг 3.13 Збереження натренованої моделі:

```
# Папка, де зберігаються ваги моделі
```

```
output_dir = cfg.OUTPUT_DIR
# Створення папки на Google Drive для збереження ваг
drive_output_dir = "./drive/MyDrive/MaskRCNN/model_output"
os.makedirs(drive_output_dir, exist_ok=True)
# Копіювання ваг моделі на Google Drive
shutil.copy(os.path.join(output_dir, "model_final.pth"), drive_output_dir)
# Збереження метаданих у файл
metadata = MetadataCatalog.get("duck_val").as_dict()
with open("./drive/MyDrive/duck_dataset/MaskRCNN/model_output/duck_metadata.json", "w") as f:
    json.dump(metadata, f)
# Збереження конфігурації у файл
cfg_path = "./content/drive/MyDrive/MaskRCNN/model_output/config.yaml"
with open(cfg_path, "w") as f:
    f.write(cfg.dump())
```

3.3 Тестування розробленої моделі

Після завершення навчання моделі важливо оцінити її продуктивність на нових, невідомих даних. Це дозволяє переконатися, що модель здатна адекватно розпізнавати та сегментувати об’єкти в умовах, відмінних від тих, що були використані під час навчання. Для цього було виконано тестування моделі на зображені (рис. 3.7), взяте зі стоку зображень [31].

Для тестування моделі було створено ще один блокнот в Google Colab, де було встановлено необхідні бібліотеки та залежності, а також підключено Google Drive для збереження і завантаження даних.

Наступний крок – завантаження конфігурації моделі та метаданих. Це забезпечує коректне налаштування середовища для подальшого використання

натренованої моделі.

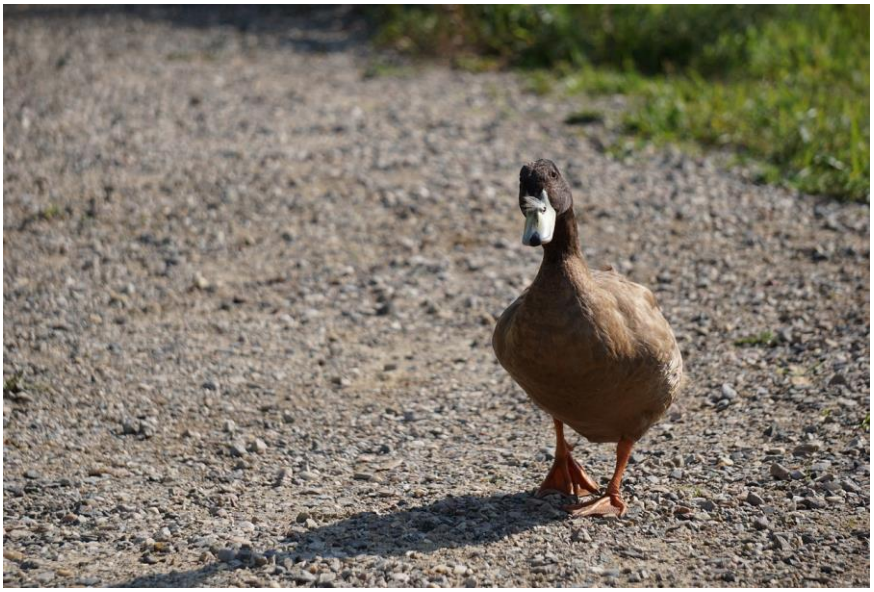


Рисунок 3.7 – Зображення для тестування

Наступний крок – завантаження конфігурації моделі та метаданих. Це забезпечує коректне налаштування середовища для подальшого використання натренованої моделі.

Лістинг 3.14 Завантаження натренованої моделі:

```
# Шлях до файлів на Google Drive
drive_output_dir = "/content/drive/MyDrive/MaskRCNN/duck_dataset/model_output"
weights_path = os.path.join(drive_output_dir, "model_final.pth")
metadata_path = os.path.join(drive_output_dir, "duck_metadata.json")
config_path = os.path.join(drive_output_dir, "config.yaml")
# Завантаження метаданих
with open(metadata_path, "r") as f:
    duck_metadata = json.load(f)
MetadataCatalog.get("duck_val").set(**duck_metadata)
# Завантаження конфігурації
from detectron2.config import get_cfg
```

```
    from detectron2.engine import DefaultPredictor
    cfg = get_cfg()
    cfg.merge_from_file(config_path)
    cfg.MODEL.WEIGHTS = weights_path
    cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.8
predictor = DefaultPredictor(cfg)
```

Далі для нового зображення робиться сегментація та візуалізується результат.

Лістинг 3.15 Передбачення сегментації

```
    # Шлях до нових зображень
    new_images_path = "/content/drive/MyDrive/MaskRCNN/duck_dataset/test"
    image_files = glob.glob(os.path.join(new_images_path, "*.jpg"))
    # Передбачення на нових зображеннях
    for image_path in image_files:
      im = cv2.imread(image_path)
      outputs = predictor(im)
      v = Visualizer(im[:, :, ::-1],
                     metadata=MetadataCatalog.get("duck_val"),
                     scale=0.5,
                     instance_mode=ColorMode.IMAGE_BW
      )
      out = v.draw_instance_predictions(outputs["instances"].to("cpu"))
      cv2.imshow("Prediction", out.get_image()[:, :, ::-1])
```

Результат сегментації показано на рисунку 3.8. Як можна побачити з цих результатів, модель показує задовільні результати, коректно сегментуючи качок на зображеннях. Це підтверджує можливість подальшого використання моделі в реальних умовах та її подальшого вдосконалення.

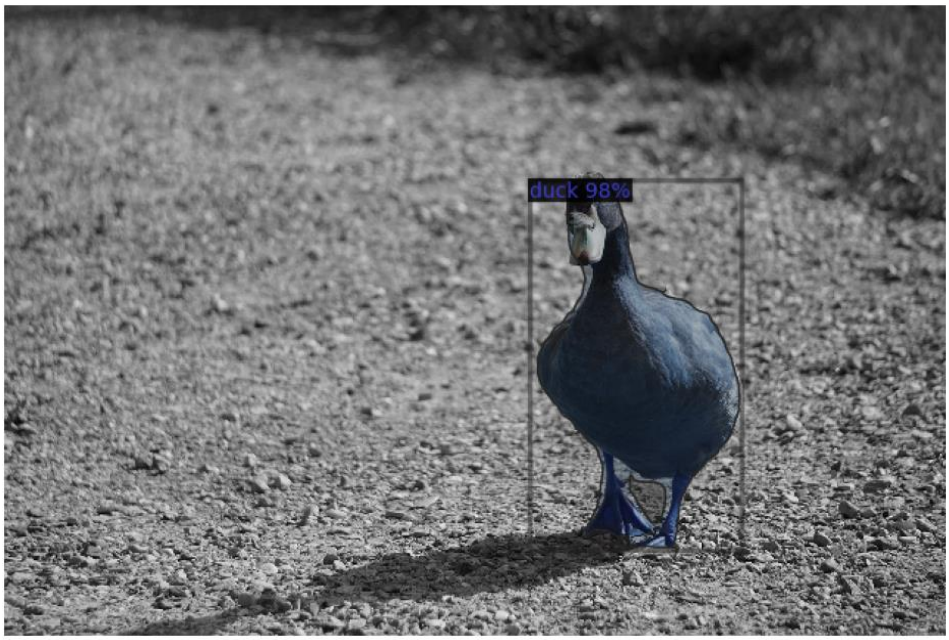


Рисунок 3.8 – Результат сегментації зображення

3.4 Перспективи подальшого використання

Розроблена модель відкриває широкі можливості для практичного використання в різних сферах. Завдяки точності та адаптивності, ця модель може бути ефективно застосована для автоматизації завдань, які раніше вимагали значних зусиль та часу.

Однією з ключових сфер застосування є екологічні дослідження та моніторинг біорізноманіття. Модель може використовуватися для автоматизованого моніторингу популяцій качок, а після вдосконалення й інших птахів у природних умовах, що значно спрощує та прискорює процес збору даних. Це також може дозволити вченим відстежувати міграційні маршрути птахів та аналізувати вплив змін у довкіллі на їхню поведінку. Такі дані можуть бути надзвичайно корисними для розробки програм з реабілітації та відновлення природних середовищ існування.

У міському плануванні та управлінні дикою природою розроблена модель може використовуватися для моніторингу популяцій птахів у міських

парках та водних об’єктах. Це допоможе підтримувати екологічний баланс у міських середовищах та забезпечувати належні умови для дикої природи. Крім того, такі дані можуть бути використані для прийняття рішень щодо управління міськими природними ресурсами та планування заходів з охорони довкілля.

Загалом, розроблена модель має великий потенціал для автоматизації завдань у різних сферах, сприяючи підвищенню ефективності та точності збору даних, що є важливим для прийняття обґрунтованих рішень у галузях екології та міського планування.

ВИСНОВКИ

У рамках кваліфікаційної роботи був розроблений і реалізований метод сегментації об’єктів на зображеннях з використанням сучасних технологій глибинного навчання та комп’ютерного зору. Використовуючи бібліотеку Detectron2, яка є потужним інструментом для задач комп’ютерного зору, було створено модель, здатну точно сегментувати качок на зображеннях.

Процес розробки починався з підготовки відповідного набору даних. Було зібрано зображення, на яких присутні качки, після чого ці зображення були анотовані для створення навчального та валідаційного набору даних. Анотація включала в себе виділення контурів об’єктів та їх класифікацію, що дозволило створити якісні дані для навчання моделі. Цей крок є критично важливим, оскільки якість анотацій безпосередньо впливає на результативність моделі.

Наступним етапом було навчено модель. Для цього було використано архітектуру Mask R-CNN, яка є однією з найбільш популярних та ефективних для задач сегментації об’єктів. Модель була налаштована та натренована на зібраному наборі даних.

Після завершення навчання модель була протестована на валідаційному наборі даних. Для оцінки якості роботи моделі використовувались метрики середньої влучності (AP) та середньої повноти (AR), які є стандартними в задачах детекції та сегментації об’єктів. Ці метрики дозволили оцінити точність та повноту моделі на різних рівнях інтерсекції об’єму (IoU), що дало можливість отримати детальну інформацію про продуктивність моделі. Результати показали, що модель досягла високих показників точності та повноти, що свідчить про її здатність ефективно вирішувати поставлену задачу.

Результати проведеної роботи свідчать про те, що розроблений метод є ефективним та може бути застосований для сегментації об’єктів на зображеннях в різних реальних умовах. Це відкриває широкі можливості для

подальшого використання моделі, зокрема в задачах моніторингу природного середовища та інших областях.

Підсумовуючи, можна сказати, що в рамках кваліфікаційної роботи було успішно реалізовано метод сегментації об’єктів на зображеннях з використанням сучасних технологій глибинного навчання. Розроблений метод продемонстрував високу ефективність і точність, що дозволяє його подальше використання та вдосконалення в різних практичних задачах.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Image Segmentation: 10 Concepts, 5 Use Cases and a Hands-on Guide. URL: https://www.basic.ai/blog-post/image-segmentation:-10-concepts,-5-use-cases-and-a-hands-on-guide-%7C-updated-2024#viewer-480aa (дата звернення 14.04.2024).

2. Borowiec, M. L., Dikow, R. B., Frandsen, P. B., McKeeken, A., Valentini, G., & White, A. E. (2022). Deep learning as a tool for ecology and evolution. Methods in Ecology and Evolution, 13(8), 1640-1660.

3. Human Pose Estimation using Keypoint RCNN in PyTorch. URL: https://learnopencv.com/human-pose-estimation-using-keypoint-rcnn-in-pytorch/ (дата звернення 14.04.2024).

4. Thresholding-Based Image Segmentation. URL: https://paimo.jodymaroni.com/thresholding-based-image-segmentation/ (дата звернення 14.04.2024).

5. Bogucharskiy, S., & Mashtalir, V. (2015, September). Image segmentation via X-means under overlapping classes. In 2015 Xth International Scientific and Technical Conference" Computer Sciences and Information Technologies"(CSIT) (pp. 45-47). IEEE.

6. Mashtalir, V., & Bogucharskiy, S. (2015). Covering image segmentation via matrix X-means and J-means clustering. Sensors & Transducers, 195(12), 56-61.

7. Bogucharskiy, S., Mashtalir, V., & Mikhnova, O. (2015, May). Fuzzy J-Means image segmentation. In Advances in Data Science: proc. International Workshop and Networking Event, Poland, Holny Mejera (pp. 6-8).

8. Mashtalir, S., & Mashtalir, V. (2020). Spatio-temporal video segmentation. Advances in Spatio-Temporal Segmentation of Visual Data, 161-210.

9. Кобилін, О. А., & Творошенко, І. С. (2021). Методи цифрової обробки зображень, 72-75.

10. Chowdhary, C. L., & Acharjya, D. P. (2020). Segmentation and feature extraction in medical imaging: a systematic review. Procedia Computer Science, 167, 26-29.

11. OpenCV projects – Image segmentation with Watershed algorithm. URL: https://datahacker.rs/007-opencv-projects-image-segmentation-with-watershed-algorithm/ (дата звернення 17.04.2024).

12. Graph-cut segmentation principle. URL: https://www.researchgate.net/figure/Graph-cut-segmentation-principle_fig2_285228259 (дата звернення 17.04.2024).

13. Sultana, F., Sufian, A., & Dutta, P. (2020). Evolution of image segmentation using deep convolutional neural network: A survey. Knowledge-Based Systems, 201, 106062.

14. Tvoroshenko, I., Gorokhovatskyi, V., Kobylin, O., & Tvoroshenko, A. (2023). Application of deep learning methods for recognizing and classifying culinary dishes in images, 59-62.

15. Thakur, N., Bhattacharjee, E., Jain, R., Acharya, B., & Hu, Y. C. (2024). Deep learning-based parking occupancy detection framework using ResNet and VGG-16. Multimedia Tools and Applications, 83(1), 1941-1964.

16. Max-pooling / Pooling. URL: https://computersciencewiki.org/index.php?title=Max-pooling_/_Pooling (дата звернення 20.04.2024).

17. Bilonoh, B., Bodyanskiy, Y., Kolchygin, B., & Mashtalir, S. (2021, May). Tunable Activation Functions for Deep Neural Networks. In International Scientific Conference “Intellectual Systems of Decision Making and Problem of Computational Intelligence” (pp. 624-633). Cham: Springer International Publishing.

18. Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE transactions on pattern analysis and machine intelligence, 39(12), 2481-2495.

19. What is Transposed Convolutional Layer? URL: https://www.geeksforgeeks.org/what-is-transposed-convolutional-layer/ (дата звернення 20.04.2024).

20. Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18 (pp. 234-241). Springer International Publishing.

21. Boonpook, W., Tan, Y., & Xu, B. (2021). Deep learning-based multi-feature semantic segmentation in building extraction from images of UAV photogrammetry. International Journal of Remote Sensing, 42(1), 1-19.

22. Pomazan, V., Tvoroshenko, I., & Gorokhovatskyi, V. (2023). Handwritten character recognition models based on convolutional neural networks, 65-66.

23. Chen, L. C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2017). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE transactions on pattern analysis and machine intelligence, 40(4), 834-848.

24. He, J., Wu, P., Tong, Y., Zhang, X., Lei, M., & Gao, J. (2021). Bearing fault diagnosis via improved one-dimensional multi-scale dilated CNN. Sensors, 21(21), 7319.

25. Wu, H., Zhang, J., Huang, K., Liang, K., & Yu, Y. (2019). Fastfcn: Rethinking dilated convolution in the backbone for semantic segmentation. arXiv preprint arXiv:1903.11816.

26. Takikawa, T., Acuna, D., Jampani, V., & Fidler, S. (2019). Gated-scnn: Gated shape cnns for semantic segmentation. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 5229-5238).

27. Viswanath, S., Krishnamurthy, R. J., & Suresh, S. (2021, November). Terrain surveillance system with drone and applied machine vision. In Journal of Physics: Conference Series (Vol. 2115, No. 1, p. 012019). IOP Publishing.

28. Google Colab. URL: https://colab.research.google.com/ (дата звернення 13.05.2024).

29. GitHub - facebookresearch/detectron2: Detectron2 is a platform for object detection, segmentation and other visual recognition tasks. URL: https://github.com/facebookresearch/detectron2 (дата звернення 13.05.2024).

30. Make Sence. URL: https://www.makesense.ai/ (дата звернення 13.05.2024).

31. Pixabay. URL: https://pixabay.com/photos/duck-nature-water-detroit-wildlife-936651/ (дата звернення 19.05.2024).