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)

РОЗРОБКА ТА ДОСЛІДЖЕННЯ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ U-NET В KERAS

(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-18-2

Голубєв Д. С.
(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. Кузьомін О.Я.
(посада, прізвище, ініціали)

Допускається до захисту

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

2022 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)
«____»_____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Голубєву Данилу Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка та дослідження сегментації зображень U-Net в Keras

затверджена наказом університету від 16 травня 2022 року № 541Ст

2. Термін подання студентом роботи до екзаменаційної комісії 23 травня 2022 р.

3. Вихідні дані до роботи алгоритм сегментації хображень, теоретичні відомості про методи сегментації зображень, тестові зображення, середовище розробки Google Colaboratory.

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз існуючих засобів сегментації зображень.
2. Обгрунтувати обрані програмні застосунки та шляхирозробки програми.
3. Розробити програмне забеспечення.
4.Зробити висновки за результатами роботи.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) класифікація алгоритмів сегментації, робота алгоритму WaterShed, початкове зображення перед WaterShed, результат сегментації алгоритму WaterShed, розпізнавання бадилля моркви та бур'яну, операція згортки, сгментація FCN, архітектура U-Net, Dropout нейронної мережі з двома прихованими шарами, праклади графів, схема навчання моделі, схема роботи системи в

режимі передбачень, приклад аугментації, графік навчання, графік невдач, коректна сегментація, не коректне виявлення бур'яна, виявлення при сплутанні рослин, тестові зображення.

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
Консультант з дотримання діючих стандартів та норм	Доцент Бєлова Н.В.		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	18.04.2022	
2	Аналіз завдання, підбір літератури	18.04.22-21.04.22	
3	Аналіз літератури з досліджуваної проблеми	22.04.22-25.04.22	
4	Аналіз технічних і програмних засобів	26.04.22-30.04.22	
5	Розробка методу	01.05.22-14.05.22	
6	Програмна реалізація	15.05.22-23.05.22	
7	Оформлення пояснювальної записки	24.05.22-26.05.22	
8	Перевірка на плагіат	27.05.22	
9	Рецензування	28.05.22	
10	Підготовка презентації та доповіді	29.05.22-30.05.22	
11	Занесення роботи в електронний архів	09.06.22	
12	Попередній захист кваліфікаційної роботи	09.06.22	

Дата видачі завдання 18 квітня 2022 р.

Студент _____

(підпис)

Керівник роботи _____ ____проф. Кузьомін О.Я. _

(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи 48 с., 18 рис., 1 табл., 2 додатка, 20 джерел.

СЕГМЕНТАЦІЯ, КОМП'ЮТЕРНИЙ ЗІР, U-NET, KERAS, ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ.

Об'єктом роботи є методи сегментації зображень.

Метою роботи є дослідження сегментації зображень U-Net для розв'язання задачі сегментації на прикладі сегментації рослин.

Під час розробки було розглянуто алгоритми сегментації зображень. Для створення системи було використано архітектуру U-Net. Програмну модель архітектури для сегментації було реалізовано за допомогою Python та бібліотеки Keras. Було проведено дослідження поведінки зображень рослин у внутрішніх шарах мережі.

Результатом роботи є мережа для сегментації рослин на зображеннях поля з рослинами.

SEGMENTATION, COMPUTER VISION, U-NET, KERAS, CONVOLVED NEURAL NETWORKS.

The object of the work is image segmentation techniques

The aim of the work is to investigate the segmentation of U-Net images for solving the problem of segmentation on the application of vegetation segmentation.

Image segmentation algorithms were considered during the development. The U-Net architecture was used to create the system. The software architecture model for segmentation was implemented using Python and the Keras library. A study of the behavior of plant images in the inner layers of the network was conducted.

The result of the work is a measure for segmentation of plants on the images of the field of plants.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

CNN – Convolution Neural Network

IoU – Intersection over Union.

VGG – Visual Geometry Group

API – Application Programming Interface

GPU – Graphical User Interface

CPU – Central Processing Unit

ВСТУП

Стрімкий розвиток інформаційних технологій відбувається у всіх сферах одночасно, агропромисловість одна з них. Агропромисловість займає ключову роль у економіці України, впровадження технологій у цю галузь напряму впливає как кількість та якість врожаю, а отже і на прибуток господарству та подальшій сплати податків. Вже сьогодні не рідкість застосування дронів для обробки рослин. Для цієї системи необхідно володіти достатньою кількістю відомостей про навколишнє середовище для подальшого прийняття рішення. Для цього використовується інформація про оточуючі об’єкти, їх структуру та зміну їх положення в просторі. Однією із прикладних задач цієї галузі є семантична сегментація зображень з потоку відеокамери дрона. На основі отриманої інформації дрон обирає модель поведінки. Важливо точно розпізнати інформацію із зображення, щоб система керування могла вірно сформувати доцільну поведінку дрона, для вирішення поставленої задачі. Отже, задача сегментації інформації на зображеннях з дрону є актуальною і має практичне значення.

Існує багато методів та алгоритмів для вирішення задачі сегментації зображень, проте з появою згорткових нейронних мереж задача сегментації зображень перейшла на новий рівень.

Метою цієї роботи є розробка моделі на основі згорткової штучної нейронної мережі, що здатна сегментувати зображення рослин.

В даній роботі було використано архітектуру U-Net згорткових нейронних мереж для сегментації зображення рослин.

1 ПІДХОДИ І МЕТОДИ РОЗВ'ЯЗАННЯ ЗАДАЧІ СЕГМЕНТАЦІЇ

Існує багато алгоритмів і основних методів для сегментації зображень, але ця частина комп'ютерного зору все ще потребує нових методів для підвищення якості та швидкості розпізнавання.

В агропромисловості машинний зір може виконувати роль контролю за рослинами, а саме виявляти небажані рослини. В даній роботі розглядається задача сегментації рослин на зображеннях які були зроблені в полях.

1.1 Задача сегментації

Сегментація зображень – це підобласть комп'ютерного зору та цифрової обробки зображень, яка спрямована на групування подібних областей або сегментів зображення під відповідними мітками класів.

Оскільки весь процес цифровий, аналогове зображення відображається у вигляді пікселів, що робить задачу формування частин рівною задачі групування пікселів.

У комп'ютерному баченні більшість моделей сегментації зображень складаються з мережі кодер-декодер у порівнянні з однією мережею кодувальників у класифікаторах. Кодер кодує латентне представлення простору вхідних даних, яке декодер декодує, щоб сформувати карти сегментів, або іншими словами карти, що окреслюють розташування кожного об'єкта на зображенні [1,3].

1.2 Загальні підходи

Сегментація зображень відбувається на базі інформації яка наявна в самому зображенні, а саме це може бути інформація про відтінки кольорів, текстура зображення чи колір пікселів. Сегментацію можна поділити на такі групи:

- методи на основі границь;
- методи на основі подібності.

Кожна техніка має свої методи реалізації.

На рисунку 1.1 представлені основні методи сегментації.

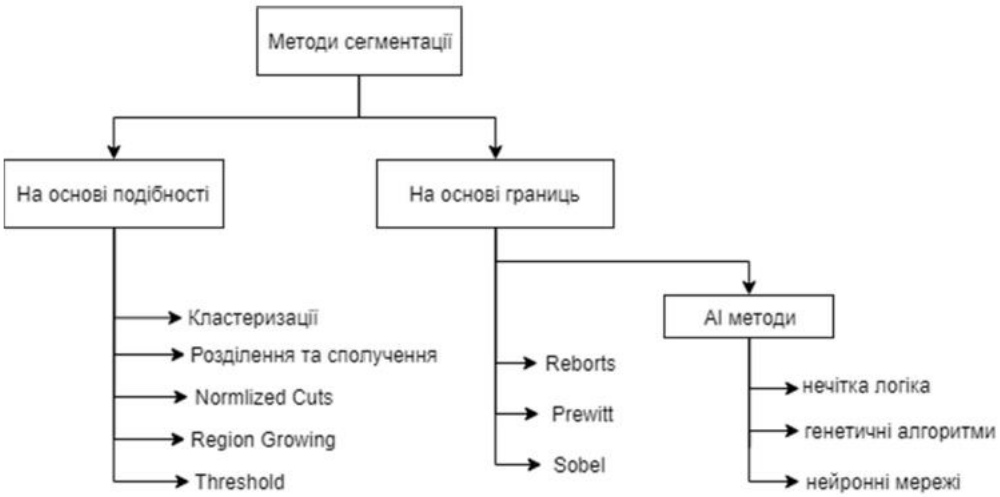

Рисунок 1.1 – Класифікація алгоритмів сегментації

Для сегментації на основі виявлення подібності областей використовуються методи кластерного аналізу. При роздільній кластеризації весь набір даних сприймається як кластер, який рекурсивно розщеплюється на менші кластери. При агломеративної кластеризації кластером вважається кожен елемент даних, а для отримання кращого представлення кластери рекурсивно зливаються. Основою для злиття (розщеплення) служить аналіз відстані між кластерами [2,4].

Найпростіші алгоритми кластеризації ґрунтуються на ітераційному (рекурсивному) злитті (розщепленні) пар кластерів на основі критерію мінімальної (максимальної) відстані між кластерами. Методи кластерного аналізу можуть застосовуватися у просторовій, і частотної області – для гістограми зображення.

Однак слід зазначити, що в такій комбінованій ситуації методи кластеризації вирішують не проблему фрагментації, а проблему структури зображення і в той же час характеризуються великою складністю застосування.

Водночас цінність від використання методів кластеризації, що застосовуються на початковій фазі аналізу зображення (до сегментації) полягає в тому, що із запровадженням певних обмежень ми можемо ефективно зробити: структуризацію однорідних областей на знімку незалежно від варіацій ТГФ параметрів зображень і потім, на цій основі, локалізацію зображень об’єктів для цілей сегментації. Обмеження при цьому можуть вводитися на область застосування, зв’язність кластерів, а також на кількість аналізованих рівнів ієрархії.

Для цілей практичного застосування в даний час поширеними є методи кластеризації, засновані на поданні зображення у вигляді графа, та працюють за принципом ітераційного вирощування батьківських областей із дочірніх областей за заданим критерієм близькості характеристик дочірніх областей. У цьому відношенні одним із ефективних методів зв’язування піраміди є комбінований метод Бюрт [5].

Багато експертів вважають, що найкращі результати розлучення отримують від комбінації поведінки для кластерного та дискримінантного аналізу, наприклад, критерію зменшення середньоквадратичної помилки в умовах, в яких визначаються центри кластерів для зображень аналізованих класів об'єктів.

Розбиття зображення проведенням кордонів. Підхід полягає у виявленні меж контурними операторами, у їхньому простеженні, зв’язуванні та складанні з них замкнених меж областей. Більшість алгоритмів виявлення контурних

кордонів засноване на фільтрації, що перетворює зображення на двомірний масив, значення якого відповідають ймовірності знаходження контуру у відповідній точці. Найбільш відомі використовують обчислення першої похідної або другий похідної. Результати їх застосування виглядають непогано з погляду візуальної оцінки, проте не задовольняють ряду формальних критеріїв, зокрема вимогам безперервності та мінімальної товщина контурних ліній. Альтернативою алгоритмів фільтрації є простежування контурів. Відомий так званий метод «zero crossing», заснований на виявленні точок переходу через нуль другої похідної він створює контурні лінії мінімальної товщини, проте при цьому з'являється багато хибних ліній, слабо відповідних реальних контурів. Цікаві результати досягаються алгоритмом Кенні [6].

1.3 Алгоритм сегментації WaterShed

Алгоритм сегментації по водорозділах (WatedShed) – працює із зображенням як із функцією від двох змінних $f=I(x,y)$, де x,y – координати пікселя. Значення функції може бути інтенсивність або модуль градієнта. Для найбільшого розмаїття можна взяти градієнт від зображення. Якщо по осі OZ відкладати абсолютне значення градієнта, то місцях перепаду інтенсивності утворюються хребти, а однорідних регіонах – рівнини. Після знаходження мінімумів функції f йде процес заповнення «водою», який починається з глобального мінімуму. Як тільки рівень води досягає значення чергового локального мінімуму, починається його наповнення водою. Коли два регіони починають зливатися, будується перегородка, щоб запобігти об’єднанню областей.

На рисунку 1.2 показано як вода продовжить підніматися доти, доки регіони не відокремлюватимуться лише штучно побудованими перегородками.

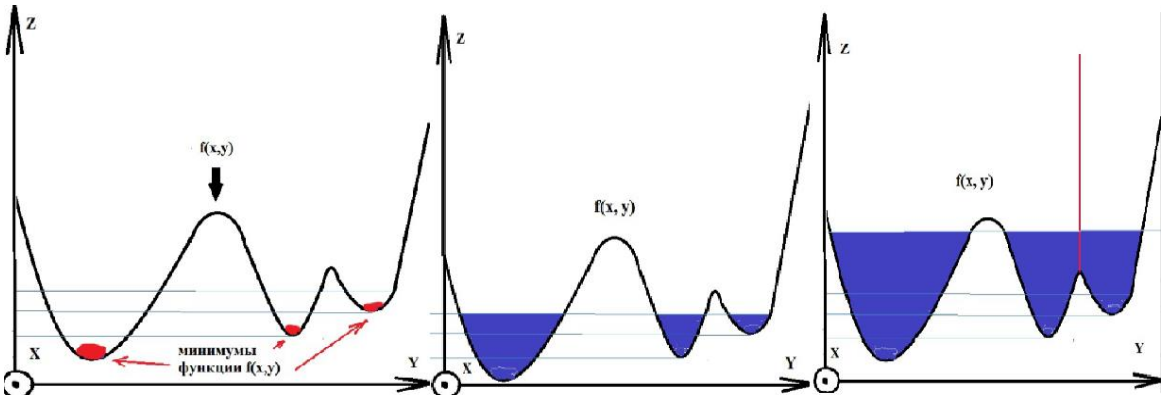

Рисунок 1.2 – Ілюстрація роботи алгоритму WaterShed [3]

Цей алгоритм може бути корисним, якщо зображення має невелику кількість локальних мінімумів, у разі великої кількості – надмірний розподіл частин [9].

Наприклад, якщо безпосередньо застосувати алгоритм на рисунку 1.3 та в результаті отримуємо багато дрібних деталей рисунку 1.4.

Рисунок 1.3 – Початкове зображення [3]

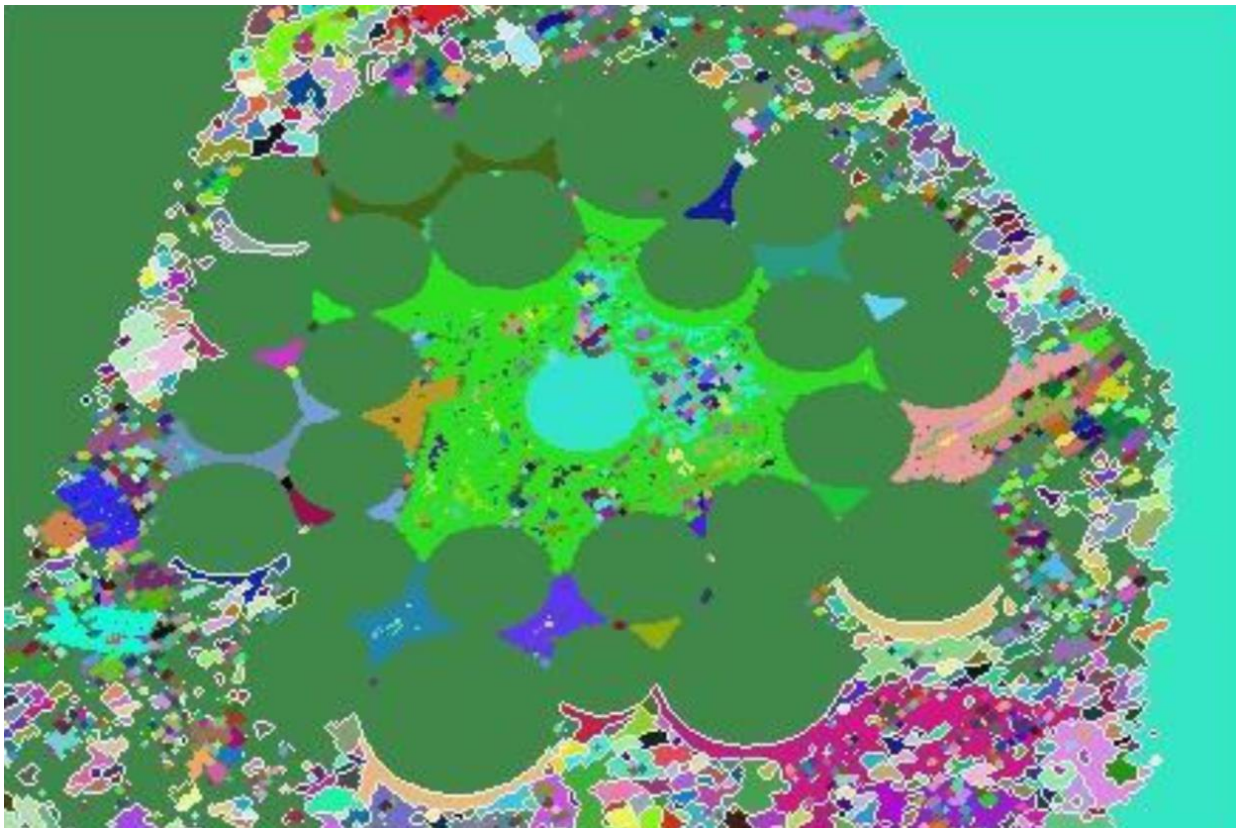

Рисунок 1.4 – Зображення після сегментації алгоритмом WaterShed [3]

В результаті роботи алгоритму отримуємо маску із зображенням компонента, в якій пікселі на одному компоненті позначені однаковою міткою і утворюють зв’язну область. Основним недоліком цього алгоритму є використання попередньої обробки для зображень з багатьма локальними мінімумами (зображення зі складною текстурою і різними кольорами) [9].

1.4 Сегментація рослин

Одною із ключових галузей економіки України є агропромисловість. Впровадження технологій в цю промисловість напряму впливає на зріст кількості вражаю та його якість. При обробці рослин використовується агрохімікати, зараз є тренд на зменшування використання хімікатів для зменшення забруднення навколишньої середи та зниження біорізноманіття.

Для вирішення цієї задачі можна використовувати сільськогосподарські дрони. При штатній обробці поля від бур’яну дрони або літаки обробляють все поле рівномірно. Якщо використовувати дрони можна здійснювати точкове оброблення, цей метод також економить хімічну рідину, але вимагає від системи класифікації рослин, за для поняття це є культура або бур’ян.

Сегментацію зображення рослин на полі можна охарактеризувати в такій послідовності:

- подання до системи зображення з відеокамери;
- визначення рослини та виділені його від фону;
- класифікація наявних рослин;
- маркування ділянки у системі/мапі/хмарному сховище.

Одну із систем вчені розробили на основі сегментації рослин на фоні ґрунту. Вони розробили алгоритм для вилучення рослинності на основі одержання зображення з інфрачервоної камери [10].

На рисунку 1.5 було запропоновано спосіб розпізнання бадилля моркви та бур’яну.

Рисунок 1.5 – Розпізнавання бадилля моркви та бур’яну [4]

Отримані результати згладжуються шляхом інтерполяції. Цей метод дає точність більше 90% [4].

1.5 Постановка задачі

Сегментація зображень за допомогою U-Net та Keras є актуальною задачею у роботі з нейронними мережами. Поставлено задачу розроблення методу виявлення агрокультури та бур’яну.

Об’єктом роботи є методи сегментації зображень.

Метою роботи є розроблення швидкісного методу сегментації зображень у системі комп’ютерного зору.

Для досягнення мети необхідно вирішити такі завдання:

- провести аналіз методів сегментації та вибрати найбільш придатний для нашої роботи;
- розробити алгоритм сегментації для множин зображень;
- реалізувати алгоритм сегментації;
- експериментально перевірити результативність методу сегментації.

2 РОЗРОБЛЕННЯ СЕГМЕНТАЦІЇ ЗГОРТКОВИМИ НЕЙРОННИМИ МЕРЕЖАМИ

Сегментація зображення може бути досягнута за допомогою нейронних мереж.

Нейронна мережа – це послідовність нейронів, з’єднаних між собою синапсами. Структура нейронної мережі прийшла у світ програмування прямо з біології. Завдяки такій структурі машина знаходить здатність аналізувати і навіть запам'ятовувати різну інформацію. Нейронні мережі також здатні не лише аналізувати інформацію, що входить, але й відтворювати її зі своєї пам'яті. Іншими словами, нейромережа це машинна інтерпретація мозку людини, в якому знаходяться мільйони нейронів, що передають інформацію у вигляді електричних імпульсів.

Нейронні мережі вже широко використовуються в різних сферах життя – розпізнають особи (у тому числі ловлять злочинців), діагностують хвороби, працюють як голосові помічники. У тому числі зростає їхнє застосування в бізнесі: оцінка ефективності співробітників, схвалення кредиту, чат-боти, управління кол-центрами [10].

Нейрон – це обчислювальна одиниця, яка отримує інформацію, здійснює над нею прості обчислення і передає її далі. Вони поділяються на три основні типи: вхідний, прихований та вихідний. Також є нейрон зміщення та контекстний нейрон. У разі, коли нейромережа складається з великої кількості нейронів, вводять термін шару. Відповідно, є вхідний шар, який отримує інформацію, n прихованих шарів (зазвичай їх не більше 3), які її обробляють та вихідний шар, який виводить результат. У кожного з нейронів є 2 основні параметри: вхідні дані (input data) та вихідні дані (output data). Що стосується вхідного нейрона: input=output. В інших, в поле input потрапляє сумарна інформація всіх нейронів з попереднього шару, після чого вона нормалізується за допомогою функції активації і потрапляє в поле output [11].

2.1 Згорткові нейронні мережі

Згорткова нейронна мережа (Convolutional Neural Network – ConvNet/CNN) – це Deep Learning –алгоритм, який може приймати вхідне зображення, надавати важливість (засвоювані ваги та зміщення) різним областям/об’єктам у зображенні та може відрізняти одне від одного. Попередня обробка в ConvNet вимагає значно менше в порівнянні з іншими алгоритмами класифікації. У той час як у примітивних методах фільтри сконструйовані вручну, ConvNets, при достатньому навчанні, здатні вивчати ці фільтри/характеристики [10].

Архітектура ConvNet подібна до нейронів у мозку людини і надихається організацією зорової кори. Окремі нейрони реагують лише на подразники в обмеженій області поля зору, відомому як рецептивне поле. Комбінація цих полів узгоджена, щоб охопити все поле зору.

Згорткові нейронні мережі працюють на основі фільтрів, які розпізнають певні характеристики зображення (наприклад, прямих ліній). Фільтр – це колекція кернелів; іноді у фільтрі використовується один кернел. Кернел – це звичайна матриця чисел, званих вагами, які «навчаються» (підлаштовуються, якщо вам так зручніше) з метою пошуку на зображеннях певних характеристик. Фільтр переміщується вздовж зображення і визначає, чи є певна шукана характеристика в конкретній його частині. Для отримання відповіді такого роду відбувається операція згортки, яка є сумою творів елементів фільтра та матриці вхідних сигналів [11].

На рисунку 2.1 показана операція згортки.

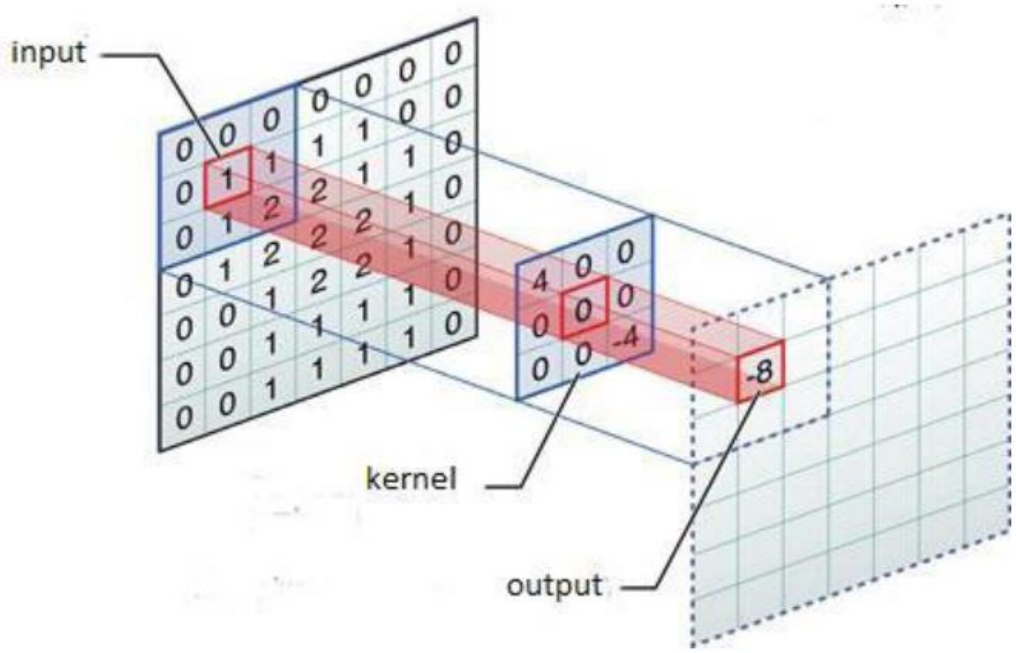

Рисунок 2.1 – Операція згортки [11]

Якщо у фрагменті зображення присутній будь-яка бажана характеристика, операція згортки створює число з відносно високим значенням. Якщо ж характеристика відсутня, вихідне число буде невеликим.

2.2 CNN архітектура для сегментації

В області комп’ютерного зору сегментація зображення відноситься до завдання призначення мітки кожному пікселя в зображенні, а також може розглядатися як завдання щільного прогнозування класифікації кожного пікселя в зображенні. На відміну від виявлення мети з використанням прямокутних кадрів-кандидатів, сегментація зображення має бути точною за позиціями на рівні пікселів, тому вона грає дуже важливу роль у таких завданнях, як медичний аналіз, виявлення об’єкта із супутникового зображення, розпізнавання райдужної оболонки та автомобілі з автоматичним. керуванням. З безперервним розвитком глибокого навчання технологія сегментації зображень також відкрила прориви у швидкості та точності останніми роками.

Сегментація семантичного зображення – це завдання призначення позначок семантичної категорії кожному пікселю в зображенні, яка не сегментує екземпляри об’єкта. Тепер основним методом обробки таких завдань є FCN[14].

На рисунку 2.2 показано FCN сегментація.

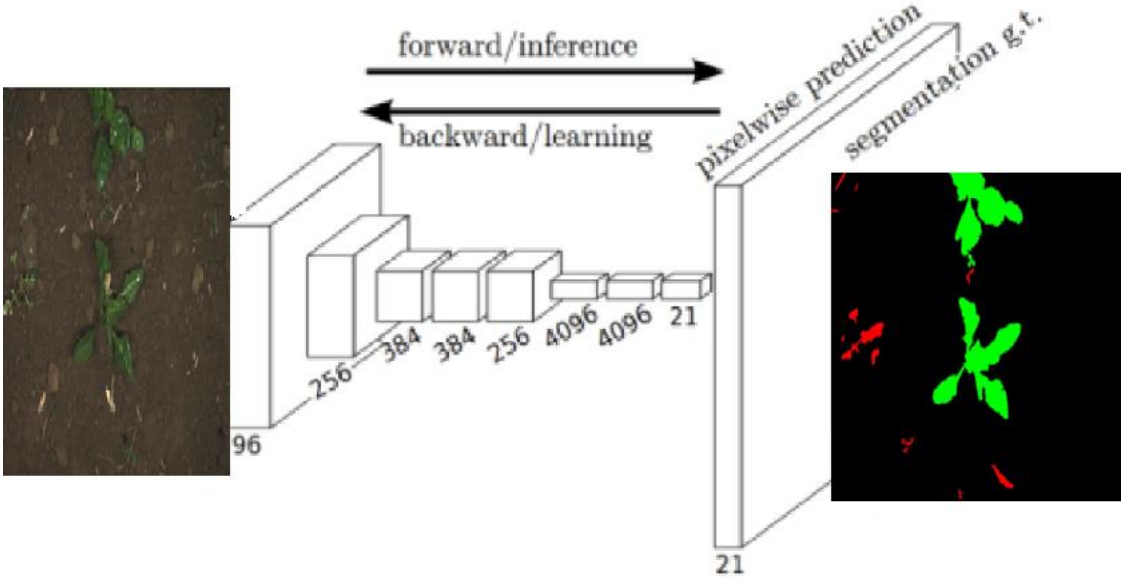

Рисунок 2.2 – Сегментація FCN

Метод створення FCN дуже простий: нам потрібно лише замінити всі повністю з’єднані шари в CNN згортковим шаром з шириною і висотою 1 × 1. У цей час число фільтрів згорткового шару дорівнює кількості нейронів (вихідних) повністю пов’язаного шару. вони називаються повними згортковими мережами. Причина цього полягає в тому, що просторове положення кожного пікселя дуже важливе для сегментації, і шар згортки може класифікувати окремі пікселі, які не можуть бути оброблені повністю підключеним шаром. В результаті обробки позиції у верхніх шарах нейронної мережі відповідатимуть позиціям на зображеннях, пов’язаних їх шляхами, тобто їх рецептивними полями.

Як показано на рисунку 2.2, архітектура FCN дуже проста, в основному складається з кодера CNN (візьмемо VGG як приклад), але останні три рівні, відповідні мережі класифікації, змінені на (4096, 1, 1) (4096, 1). , 1) (N + 1,1,1) згортковий шар (N представляє кількість категорій). За кодером знаходиться мережа декодерів, яка містить лише зворотний згортковий рівень (також званий

транспонованим згортком або деконволюцією). Його вихід має самі просторові розміри, як і вхідне зображення, і має $N + 1$ канал, кожен канал передбачає категорію [15].

2.3 Архітектура U-Net

U-net спочатку була винайдена та вперше використана для сегментації біомедичних зображень. Його архітектуру можна в цілому розглядати як мережу кодувальника, за якою слідує мережа декодера. На відміну від класифікації, де кінцевий результат глибокої мережі є єдиною важливою річчю, семантична сегментація вимагає не тільки розрізнення на рівні пікселя, але й механізму для проектування дискримінаційних ознак, засвоєних на різних етапах кодера, на простір пікселів.

Кодер є першою половиною архітектурної схеми (рис. 2.3). Зазвичай це попередньо навчена мережа класифікації, така як VGG / ResNet, де ви використовуєте блоки згортки для подальшого зменшення вибірки maxpool для кодування вхідного зображення в представлення об’єктів на різних рівнях.

Декодер є другою половиною архітектури. Мета полягає в тому, щоб семантично спроектувати дискримінаційні ознаки (нижча роздільна здатність), засвоєні кодером, на простір пікселів (вища роздільна здатність), щоб отримати щільну класифікацію. Декодер складається з підвищення дискретизації та конкатенації з наступними регулярними операціями згортки [14].

На рисунку 2.3 сині поля представляють багатоканальні карти об’єктів, тоді як поля представляють скопійовані карти об’єктів. Стрілки різного кольору позначають різні операції.

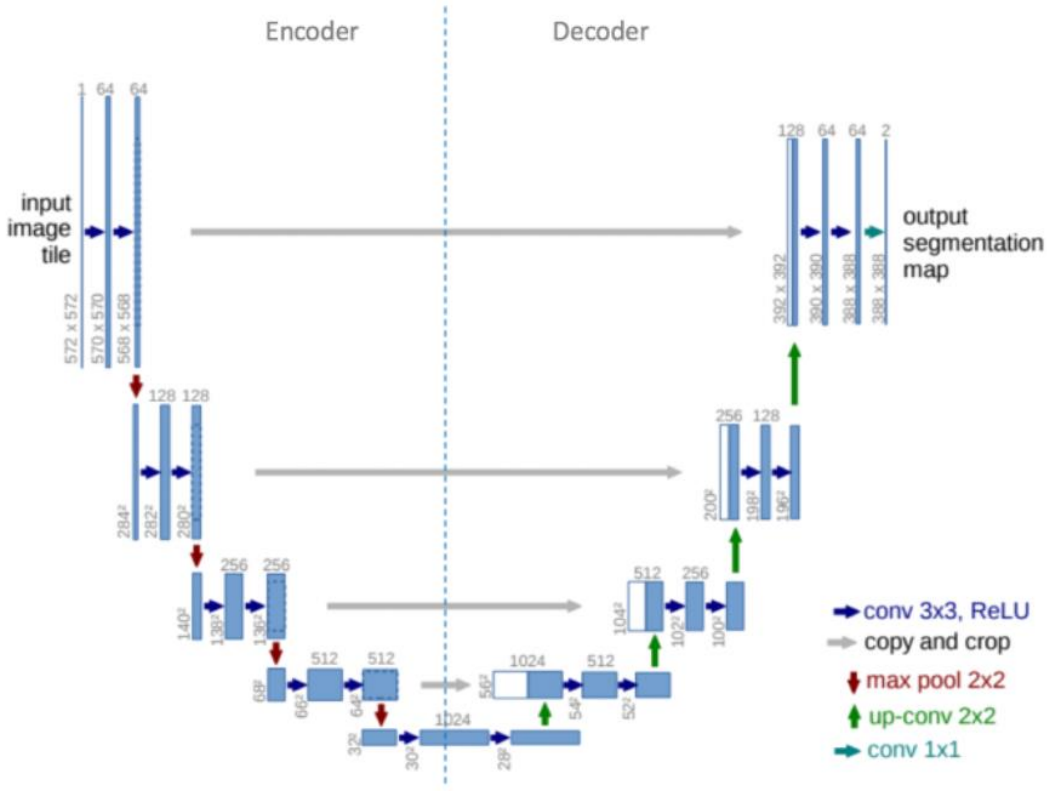

Рисунок 2.3 – Архітектура U-Net [14]

2.4 Keras

Keras – це бібліотека для Python, що дозволяє легко та швидко створювати нейронні мережі. Вона сумісна з TensorFlow, Theano, Microsoft Cognitive Toolkit та MXNet. Перші дві платформи найбільш використовуються розробки алгоритмів глибокого навчання, але досить складні в освоєнні. Keras ж, навпаки, є чудовим варіантом для тих, хто тільки починає вивчення нейронних мереж.

Автор бібліотеки Франсуа Шолле, інженер Google, розробив Keras для того, щоб максимально спростити процес створення нейронних мереж. Наголос був на розширюваності, модульності та мінімалізмі з підтримкою Python.

Розробка Keras дозволили Google зробити значний внесок у глибоке навчання та штучний інтелект. В основному це пов’язано з тим, що бібліотека містить сучасні алгоритми, яких раніше не було.

Keras розроблено для користувачів, а не для машин – він використовує спеціальні прийоми: забезпечує послідовний і простий API, який вимагає ряду дій користувача для вирішення найпоширеніших завдань. Якщо ви зробили помилку, ви завжди можете попросити відгук.

Keras модульний, під цим ми розуміємо послідовність або граф автономних, повністю налаштованих модулів, які можуть бути підключені без додаткових обмежень. Наприклад, це можуть бути нейронні шари, функції помилки, оптимізатори, схеми ініціалізації, функції активації та схеми регулювання – всі ці модулі можна комбінувати для створення моделі. Можливість легко додавати нові класи, модулі та функції робить Keras відмінним засобом для проведення різноманітних досліджень. Всі моделі були написані мовою програмування Python, тому код завжди компактний та легко читається [16].

Keras – це простий у використанні API, за допомогою якого легко створювати моделі нейронних мереж. Він підходить для реалізації алгоритмів глибокого навчання та обробки природної мови. Модель нейронної мережі можна побудувати за допомогою кількох рядків коду.

У Keras один із найкращих прикладів документації. Вона представляє кожну функцію послідовно та дуже докладно. Приклади коду також корисні та прості для розуміння. У Keras є чудова підтримка з боку спільноти. Багато розробників віддають перевагу Keras для участі у змаганнях з Data Science. Також багато дослідників публікують свій код та керівництва для широкої аудиторії [17].

Keras пропонує підтримку кілька бекенд-движків, включаючи Tensorflow, Theano та CNTK. Будь-який може бути обраний з урахуванням вимог проекту. Можна також тренувати модель Keras на основі одного двигуна, а перевіряти результати – на іншому. Поміняти двигун у Keras також дуже легко. Для цього його ім’я потрібно просто записати у конфігураційному файлі.

Keras надає кілька моделей глибокого навчання з натренованою вагою. Їх можна використовувати для передбачення або отримання ознак. У цих моделей є вбудовані ваги, які є результатами тренування моделі на даних ImageNet.

Ось деякі з представлених моделей:

- Xception;
- VGG16;
- VGG19;
- ResNet, ResNetV2;
- InceptionV3;
- InceptionResNeetV2;
- MobileNetV2;
- DenseNet;
- NASNet.

Keras дозволяє тренувати модель як на одному, так і на кількох GPU. Це забезпечує підтримку паралелізму даних та дозволяє обробляти великі обсяги[16].

Іноді виникають низькорівневі помилки бекенду. Це відбувається в тих випадках, коли намагаються виконати операції, для яких Keras не призначений. Однак він не дозволяє змінювати щось у бекенді. Отже, складно займатися налагодженням на основі логів із помилками.

Інструменти підготовки Keras не такі хороші, як інші пакети, такі як scikit-learn. Вони не підходять для створення стандартних алгоритмів машинного навчання: кластерний аналіз як метод базового компонента. Немає можливості динамічно створювати графіку.

Іноді Keras дуже повільно працює на графічному процесорі, і його операції займають більше часу, ніж бекенд. Тому швидкістю потрібно пожертвувати заради зручності використання.

2.5 Функція помилки та регуляризації

Більшість задач машинного навчання зводиться до задач оптимізації. Функції активації нейронної мережі є важливим компонентом глибокої нейронної мережі. Функції активізації визначають вихід моделі навчання, її точність, а також обчислювальну ефективність навчання моделі. При виборі вибору функції активації модель може значно покращити свою ефективність, при невдалому може перестати навчатися взагалі.

Функції активізації активації також мають великий вплив на здатність нейронні мережі до зближення і швидкість збіжності, або в деяких випадках, функції активації можуть перешкодити зближенню нейронних мереж.

Функції активації є математичними рівняннями, які визначають вихід нейронної. Функція прикріплена до кожного нейрона в мережі і визначає, чи повинен він бути активована (збудженим) чи ні, на основі того, чи є вхід кожного нейрона відповідним для прогнозування моделі. Функції активації також допомагають нормалізувати вихід кожного нейрона до діапазону між 1 та 0 або між -1 та 1.

Додатковим аспектом функцій активації є те, що вони повинні бути ефективними з точки зору обчислень, тому що вони розраховані на тисячі або навіть мільйони нейронів для кожного зразка даних. Сучасні нейронні мережі використовують методику, яку називають зворотним розповсюдженням, для навчання моделі, яка збільшує обчислювальну деформацію на функцію активації та її похідну функцію.

Є такі типи функцій активацій:

- бінарна функція;
- лінійна функція;
- нелінійна функція.

Бінарна функція кроку є функцією активації на основі порогу. Якщо вхідне значення вище або нижче певного порогу, нейрон активується і передає точно такий же сигнал на наступний шар.

Регулярність нейронних мереж важлива, оскільки мережа, в якій багато параметрів добре «запам'ятовуються» значеннями навчальної вибірки і, як наслідок, недостатньо прогнозує необхідні значення в мішках. -початкова точка. Сучасні нейронні мережі – це моделі з багатьма параметрами, навіть найскладніша архітектура може мати мільйони масштабів.

Є багато ідей щодо упорядкування. Ми намагаємося отримати менші значення з кожної змінної, яку ми оптимізуємо. Це можна зробити, додавши цільовий контролер функцій у будь-якій відповідній формі. У теорії нейронних мереж це називається втратою ваги, оскільки зменшує її абсолютні значення.

В бібліотеці Keras є можливість для кожного шару додати регуляризатор на три види зв’язків:

- kernel_regularizer – на матрицю вагів;
- bias_regularizer – на вектор вільних членів;
- activity_regularization – на вектор виходів.

Інша ідея регуляризації полягає в тому, щоб створити набір даних аутентифікації, і ми навчаємо мережу в навчальному наборі та обчислюємо помилку аутентифікації. Це ґрунтується на тому, що набір для перевірки помилок правильно оцінює помилку в нових точках (тестовому DataSet), оскільки він береться з даних тієї ж природи, але не з даних досліджуваних мереж. Це також дозволяє зупинити процес навчання не тоді, коли мережа досягає локального максимуму для навчального тесту, а коли помилка в наборі аутентифікації починає зменшуватися. У теорії нейронних мереж це називається передчасним припиненням.

Останніми роками був розроблений більш ефективний метод — dropout. В його основі лежить «вимкнення» нейрона з мережі. Для кожного нейрона встановлюється деяка ймовірність з якою він буде вимкнений. На кожному новому тренувальному прикладі ми спершу випадковим чином вирішуємо чи

буде нейрон використовуватися. Якщо ні, то вихід нейрона стає рівним 0. Це приводить до того, що нейрон фактично випадає з графу обчислення і ні пряме обчислення, ні зворотне поширення градієнту не відбувається.

На рисунку 2.4 зображен Dropout нейронної мережі.

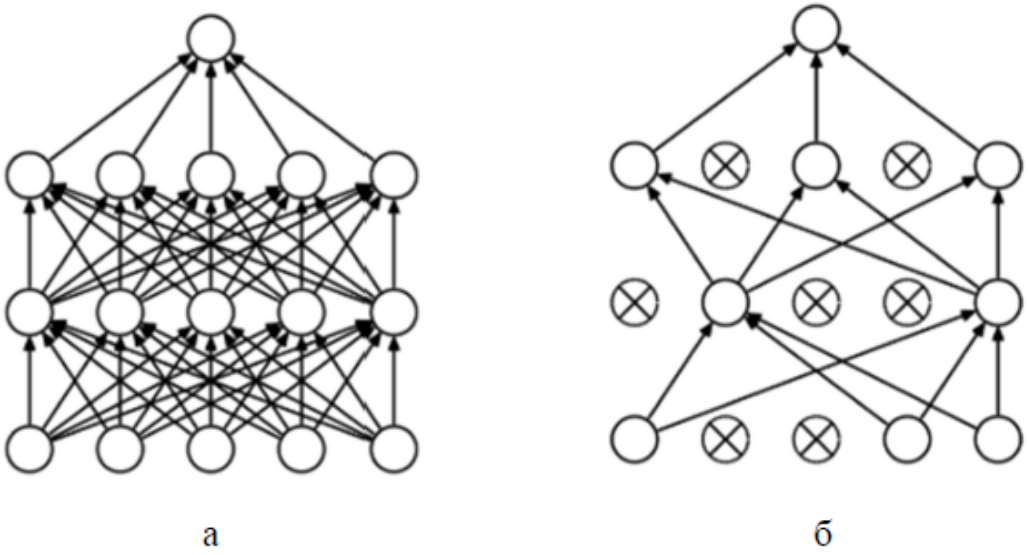

Рисунок 2.4 – Dropout нейронної мережі з двома прихованими шарами

На рисунку 2.4.а зображена звичайна нейронна мережа з двома прихованими шарами.

На рисунку 2.4.б та ж сама мережа з двома прихованими шарами та застосованою технікою dropout. Закреслені нейрони були викреслені з мережі.

З практичної точки зору, найкращої ефективності більшість архітектур досягає при використанні техніки dropout з ймовірністю 0.5. Техніка dropout – це спроба досягти узагальнення моделі. Це техніка має також еволюційне пояснення. На перший погляд може здатися, що для еволюції важливо, якщо гени навчилися «працювати» разом, то важливо передати їх наступному поколінню. Але статеве розмноження порушує адаптивні механізми генів, важливо зібрати стабільну комбінацію генів. А цього простіше досягти, якщо примушувати працювати гени наодинці, не розраховуючи на сусідні нейрони [5].

2.6 TensorFlow

TensorFlow – бібліотека для машинного навчання, розроблена компанією Google для побудови та тренування нейронних мереж.

TensorFlow чудово підходить для автоматичного знаходження та класифікації образів, оскільки якість розпізнавання наближається до людського сприйняття.

Відкриваючи вихідний код бібліотеки машинного навчання TensorFlow, Google спрощує процес створення та розгортання складних нейронних мереж. TensorFlow не дає кожному розробнику можливість використовувати результати машинного навчання, але він надає API для Python і C / C ++, які можна підключити до програми розробника [7].

Машинне навчання такого роду призначене виключно для дослідницьких цілей, але завдяки програмному забезпеченню з відкритим кодом на зразок TensorFlow підприємство отримує потужні засоби для використання своїх власних даних та їх обробки у дешевому хмарному середовищі.

Бібліотеки TensorFlow додатково спрощують інтеграцію програм, що самонавчаються, з елементами та функціями штучного інтелекту, призначеними для розпізнавання мови, комп'ютерного зору або обробки природної мови.

Звичайно, TensorFlow не єдина бібліотека глибинного навчання, але, як і пошуковий механізм Google, вона вважається найкращою у своєму класі. Альтернативами є програмне забезпечення Torch, створене швейцарськими дослідниками, та розробка Каліфорнійського університету в Берклі Caffe, остання версія якої Caffe2 була спроектована за участю Facebook.

Google дозволяє використовувати TensorFlow лише на одній машині з кількома графічними процесорами, що стримує масштаби застосування цього інструменту на підприємствах. Звичайно, існують шляхи обходу зазначеного обмеження, але на створення таких рішень знадобляться певні знання, час та гроші.

Згідно з інформацією, опублікованою на сайті TensorFlow, програмну бібліотеку використовує цілий ряд великих компаній, у тому числі Airbnb, Airbus, Dropbox, Snapchat та Uber[16].

Google хоче використовувати машинне навчання, щоб використовувати переваги своїх масивних наборів даних, щоб надати користувачам найкращий досвід. Три різні групи використовують машинне навчання: дослідники, вчені які працюють з даними, програмісти. Всі вони можуть використовувати один і той же набір інструментів для спільної роботи та підвищення ефективності.

Google не просто має будь-які дані, у них найпотужніший комп'ютер у світі, тому Tensor Flow був побудований у масштабі. TensorFlow – це бібліотека, розроблена Google Brain Team для прискорення машинного навчання та глибоких досліджень нейронних мереж.

Він був створений для роботи на декількох процесорах або графічних процесорах і навіть у мобільних операційних системах і має кілька оболонок кількома мовами, таких як Python, C++ або Java.

Принципи роботи тензорного потоку відносно прості. Нам потрібно створити графік операцій, потім передати дані на цей графік і ввести команду для виконання обчислень.

На рисунку 2.5 можем побачити 3 приклади таких графів.

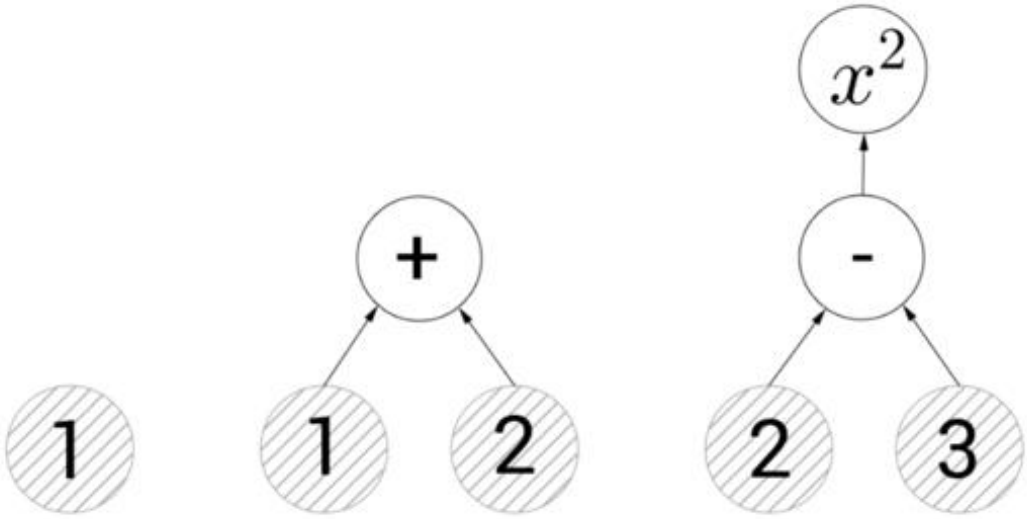

Рисунок 2.5 – Приклади графів

Граф ліворуч містить лише одну вершину, що представляє константу зі значенням 1. Тут і далі за текстом, у таких ілюстраціях, колами із сірим штрихуванням будуть позначатися вершини з константами, а без штрихування вершини з операціями. Центральний граф ілюструє операцію додавання. Якщо ми попросимо tensorflow обчислити значення вершини, що представляє операцію додавання, то він обчислить значення направлених до нього ребер графа і підсумує їх (тобто буде повернено 3). На правому ж графі у нас дві вершини з операціями — віднімання та зведення у квадрат. Якщо ми спробуємо обчислити вершину, що представляє зведення в квадрат, то tensorflow спочатку виконує віднімання [18].

2.7 Ініціалізація вагів та перенесення навчання

Навчання нейронної мережі – це велика та складна задача оптимізації в просторі великої розмірності, яка вирішується методами локального пошуку. Тому цілком природно, що одне з ключових запитань стоїть в тому, де почати локальний пошук. Залежно від якості початкового наближення можна потрапити в самі різні локальні оптимуми. Правильна ініціалізація вагів дозволяє нам навчати нейронні мережі краще та швидше.

Одним із варіантів вирішення проблеми ініціалізація вагів є до навчання без вчителя. Можна навчати окремі шари глибокої мережі без вчителя, послідовно, а потім початкові ваги отриманих шарів брати, як початкове наближення та до навчати вже на розміченому наборі даних.

Для навчання глибокої мережі потрібно велика кількість навчальних прикладів та багато часу на навчання. Оскільки для вирішення різного роду задач можуть застосовуватися одні й ті ж глибокі мережі, використовується підхід перенесення навчання. Підхід полягає у використанні попередньо вивченої нейронної мережі класифікації зображень. Останні шари будь-якої згорткової нейронної мережі, навченої на розв’язання задачі класифікації розпізнають

низько рівневі характеристики зображення, наприклад кути або лінії, які містять будь які зображення. Замість навчання нейронної мережі з самого початку, можна використати ваги з нижніх шарів вже навченої моделі, та зосередитися на навчанні глибших шарів, що відповідають на ознаки високого рівня.

До навчання відбувається послідовно, від нижніх до верхніх шарів. Це дозволяє уникнути проблеми затухаючих градієнтів та суттєво знизити кількість обчислень на кожному етапі. До навчання відбувається без вчителя, тобто, без врахування існуючих розмічених даних. Це часто дозволяє суттєво збільшити навчальну вибірку. В результаті до навчання отримуємо модель, яку потім потрібно до навчити на існуючих даних. Модель, навчена таким чином в кінцевому рахунку стабільно сходиться до найкращих існуючих рішеннях ніж при випадковій ініціалізації.

Донавчання – це складна та дорога операція. Фактично виходить, що потрібно навчити робити дві різні процедури навчання, налаштовувати їх, причому до навчання швидше за все буде більше складно процедурою навчання.

3 РЕЗУЛЬТАТИ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ

3.1 Навчання нейронної мережі

Центральний процесор є центром будь-якого обчислювального пристрою, що виконує інструкції програми, обробляє ввід та вивід даних. Перший процесор, Intel 4004, був розроблений компанією Intel в 1971 році. Більшість центральних процесорів потім були розроблені з одним ядром, це означає, що одночасно може бути виконана лише одна операція. Роком пізніше, завдяки великим поліпшенням у розробці, дослідженні та виробництві чіпів, обчислювальний ринок перейшов до двоядерних та багатоядерних процесорів, які були швидшими, оскільки тепер вони могли виконувати дві або більше операцій одночасно[16].

Сьогоднішні процесори мають лише кілька ядер, а його основна конструкція та призначення – обробка складних обчислень, не змінилось.

Графічний процесор (GPU), з іншого боку, має менші розміри, але багато інших логічних ядер, основна конструкція яких полягає в обробці набору більш простих та ідентичних обчислень в паралельному режимі. Наприклад, обробка тривимірної графіки та візуалізація.

Можна сказати, що процесори найкраще підходить для обробки поодиноких, більш складних обчислень послідовно, тоді як графічні процесори краще обробляти декілька але більш простих обчислень паралельно. Використання GPU дозволяє навчати нейронні мережі в кілька разів швидше, ніж CPU.

Всі навчальні матеріали були завантажені на хмарне сховище. В процесі навчання ваги моделей також зберігались на хмарне сховище.

Для реалізації нейронної мережі було обрано бібліотеку Keras, як обгортку над TensorFlow.

Keras – це бібліотека з відкритим кодом, написана на Python. Бібліотека надає високорівневий інтерфейс для роботи з бібліотеками TensorFlow та Theano. Бібліотека містить численні реалізації широко вживаних блоків нейронних мереж, таких як шари, цільові та передавальні функції, оптимізатори, та безліч інструментів для спрощення роботи із зображеннями та текстом.

Для навчання мережі U-Net спершу був використано хмарне середовище для навчання нейронних мереж Google Collaborator, навчання проводилось на Intel 8 го покоління та Nvidia GeForce GTX 1050 3GB.

3.2 Реалізована мережа U-Net

Для сегментації було реалізовано мережа U-Net. Перелік шарів мережі, розмірність виводу та кількість параметрів наведено в таблиці 3.1. Всього мережа містить 29 832 654 параметра. Входом мережі є зображення у вигляді тримірного масиву. Входом є двовимірна маска на зображення. Кожний елемент містить ймовірність належності пікселя зображення до шуканого класу.

Перша половина U-net – це кодування, а її функція – отримання ознак (отримання локальні особливості і виконання класифікації на рівні зображення). Теоретичне значення цієї понижуючої дискретизації полягає в тому, що воно може підвищити стійкість до деяких невеликих порушень вхідного зображення, таким як переведення зображення, поворот і т. д., знизити ризик перенавчання, зменшити обсяг обчислень та збільшити розмір рецептивного поля.

Лістинг 3.1 Реалізація кодера в U-Net:

```
img_input = tf.keras.layers.Input(shape=[img_h, img_w, 3])

x = tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same', name='block1_conv1')(img_input)
```

```
x = tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same', name='block1_conv2')(x)

x = tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block1_pool')(x)

down1 = x

x = tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same', name='block2_conv1')(x)

x = tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same', name='block2_conv2')(x)

x = tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block2_pool')(x)

down2 = x

x = tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name='block3_conv1')(x)

x = tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name='block3_conv2')(x)

x = tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name='block3_conv3')(x)

x = tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block3_pool')(x)

down3 = x

x = tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name='block4_conv1')(x)

x = tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name='block4_conv2')(x)

x = tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name='block4_conv3')(x)

x = tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block4_pool')(x)
```

```
down4 = x

x = tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name='block5_conv1')(x)

x = tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name='block5_conv2')(x)

x = tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name='block5_conv3')(x)

x = tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block5_pool')(x)
```

Друга половина U-net – це декодування, тобто процес використання раніше закодованих абстрактних ознак відновлення розміру вихідного зображення і, нарешті, отримання результату сегментації. Коли декодування відновлює дані зі зниженою дискретизацією (деконволюцією), масштаб ознак буде змінюватися, і неминуче буде втрата інформації.

Лістинг 3.1 Реалізація кодера в U-Net:

```
x = tf.keras.layers.Conv2D(512, (3, 3), activation=None, padding='same', kernel_initializer = 'he_normal')(x)

x = tf.keras.layers.LeakyReLU(alpha=leakyrelu_alpha)(x)

x = tf.keras.layers.UpSampling2D((2, 2), interpolation='bilinear')(x)

up4 = x

x = tf.keras.layers.concatenate([up4, down4], axis=-1)

x = tf.keras.layers.Conv2D(512, (3, 3), activation=None, padding='same', kernel_initializer = 'he_normal')(x)

x = tf.keras.layers.LeakyReLU(alpha=leakyrelu_alpha)(x)
```

```
x = tf.keras.layers.UpSampling2D((2, 2), interpolation='bilinear')(x)

up3 = x

x = tf.keras.layers.concatenate([up3, down3], axis=-1)

x = tf.keras.layers.Conv2D(256, (3, 3), activation=None, padding='same', kernel_initializer = 'he_normal')(x)

x = tf.keras.layers.LeakyReLU(alpha=leakyrelu_alpha)(x)

x = tf.keras.layers.UpSampling2D((2, 2), interpolation='bilinear')(x)

up2 = x


x = tf.keras.layers.concatenate([up2, down2], axis=-1)

x = tf.keras.layers.Conv2D(128, (3, 3), activation=None, padding='same', kernel_initializer = 'he_normal')(x)

x = tf.keras.layers.LeakyReLU(alpha=leakyrelu_alpha)(x)

x = tf.keras.layers.UpSampling2D((2, 2), interpolation='bilinear')(x)

up1 = x


x = tf.keras.layers.concatenate([up1, down1], axis=-1)

x = tf.keras.layers.Conv2D(64, (3, 3), activation=None, padding='same', kernel_initializer = 'he_normal')(x)

x = tf.keras.layers.LeakyReLU(alpha=leakyrelu_alpha)(x)

x = tf.keras.layers.UpSampling2D((2, 2), interpolation='bilinear')(x)

up0 = x
```

x = tf.keras.layers.Conv2D(32, (3, 3), activation=None, padding='same', kernel_initializer = 'he_normal')(x)

x = tf.keras.layers.LeakyReLU(alpha=leakyrelu_alpha)(x)

x = tf.keras.layers.Conv2D(num_classes, (1, 1), activation='softmax', padding='same')(x)

Розроблена мережа показана у таблиці 3.1

Таблиця 3.1 – Розроблена мережа U-Net.

Шар	**Вихідна розмірність**	**Кількість параметрів**
1	**2**	**3**
input_1(InpurLayer)	(None, 512, 512, 64)	0
block_conv1 (Conv2D)	(None, 512, 512, 64)	1792
block_conv2 (Conv2D)	(None, 256, 256, 64)	36928
block_pool(MaxPooling2D)	(None, 256, 256, 64)	0
Block2_conv1 (Conv2D)	(None, 256, 256, 128)	73856
block2_conv2 (Conv2D)	(None, 256, 256, 128)	147584
block2_pool (MaxPooling2D)	(None, 128, 128, 128)	0
block3_conv1 (Conv2D)	(None, 128, 128, 256)	295168
block3_conv2 (Conv2D)	(None, 128, 128, 256)	590080
block3_conv3 (Conv2D)	(None, 128, 128, 256)	590080
block3_pool (MaxPooling2D)	(None, 64, 64, 256)	0
block4_conv1 (Conv2D)	(None, 64, 64, 512)	1180160

Продовження таблиці 3.1

1	2	3
block4_conv2 (Conv2D)	(None, 64, 64, 512)	2359808
block4_conv3 (Conv2D)	(None, 64, 64, 512)	2359808
block4_pool (MaxPooling2D)	(None, 32, 32, 512)	0
block5_conv1 (Conv2D)	(None, 32, 32, 512)	2359808
block5_conv2 (Conv2D)	(None, 32, 32, 512)	2359808
block5_conv3 (Conv2D)	(None, 32, 32, 512)	2359808
block5_pool (MaxPooling2D)	(None, 16, 16, 512)	0
conv2d (Conv2D)	(None, 16, 16, 512)	2359808
leaky_re_lu (LeakyReLU)	(None, 16, 16, 512)	0
up_sampling2d (UpSampling2D)	(None, 32, 32, 512)	0
concatenate (Concatenate)	(None, 32, 32, 1024)	0
conv2d_1 (Conv2D)	(None, 32, 32, 512)	4719104
leaky_re_lu_1 (LeakyReLU)	(None, 32, 32, 512)	0
concatenate_1 (Concatenate)	(None, 64, 64, 768)	0
up_sampling2d_1 (UpSampling2D)	(None, 64, 64, 512)	0
conv2d_2 (Conv2D)	(None, 64, 64, 256)	1769728
leaky_re_lu_2 (LeakyReLU)	(None, 64, 64, 256)	0
up_sampling2d_2 (UpSampling2D)	(None, 128, 128, 256)	0
concatenate_2 (Concatenate)	(None, 128, 128, 384)	0
conv2d_3 (Conv2D)	(None, 128, 128, 128)	442496

Продовження таблиці 3.1

1	2	3
leaky_re_lu_3 (LeakyReLU)	(None, 128, 128, 128)	0
up_sampling2d_3 (UpSampling2D)	(None, 256, 256, 128)	0
concatenate_3 (Concatenate)	(None, 256, 256, 192)	0
conv2d_4 (Conv2D)	(None, 256, 256, 64)	110656
leaky_re_lu_4 (LeakyReLU)	(None, 256, 256, 64)	0
up_sampling2d_4 (UpSampling2D)	(None, 512, 512, 64)	0
conv2d_5 (Conv2D)	(None, 512, 512, 32)	18464
leaky_re_lu_5 (LeakyReLU)	(None, 512, 512, 32)	0
conv2d_6 (Conv2D)	(None, 512, 512, 3)	99

3.3 Структура системи

В ході роботи було реалізовано дві системи.

На рисунку 3.1 показана схема мережі для навчання.

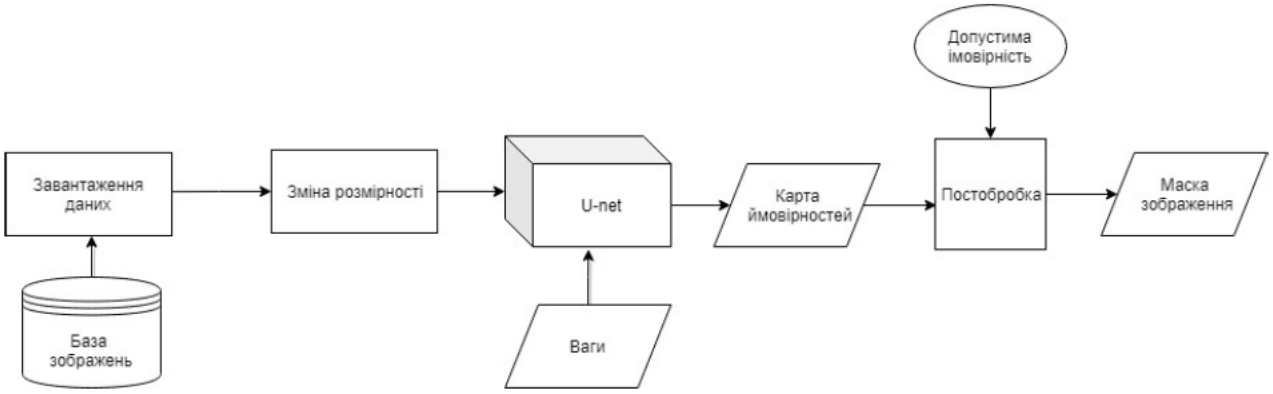

Рисунок 3.1 – Схема навчання моделі

Для навчання мережі модуль завантаження отримує з файлового сховища зображення та відповідну маску для нього. Далі зображення та маска приводяться до одного заданого розміру та передаються на вхід модулю аугментації даних, який виконує однакові операції по модифікаціям зображення та маски. Після чого зображення йде на вхід мережі на основі архітектури U-Net. Мережа обробляє зображення та видає згенеровану маску у вигляді карти ймовірностей. Сформована маска порівнюється з наявною маскою, проводиться обчислення похибки на основі якої мережа проводить корекцію вагів.

На рисунку 3.2 показа схема використання мережі.

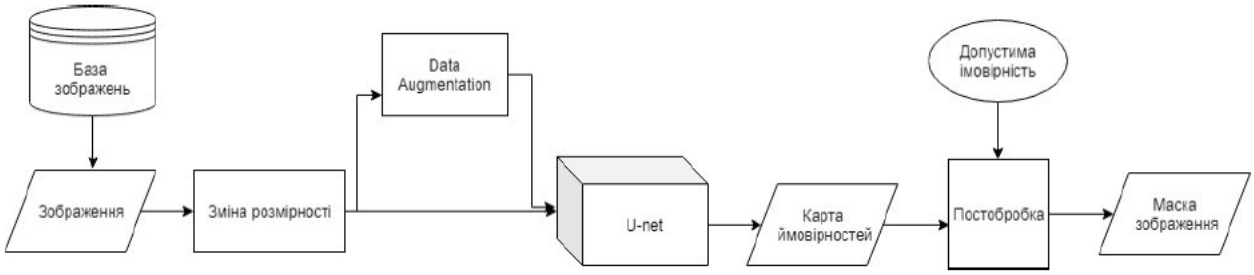

Рисунок 3.2 – Схема роботи системи в режимі передбачень

Система для передбачень відрізняється від системи для навчання. Ціллю роботи системи є генерація маски сегментації зображення, тому на вхід подається тільки зображення. Аугментація не проводиться. Результатом роботи мережі є карта ймовірностей належності пікселя зображення до шуканого сегменту. Для отримання маски в модулі пост обробки проводиться порівняння кожного значення карти ймовірностей до допустимого значення: якщо значення більше, то приймається, що піксель належить шуканому об’єкту. В результаті отримуємо маску сегментація для зображення.

3.4 Аугментація зображення

Для отримання коректних результатів тренування для нейронної мережі необхідна велика кількість вхідних даних. Для реалізації цілей задачі було здійснено штучне збільшення кількості даних, шляхом модифікації наявних зображень (рис 3.3). Для зображення було застосовано комбінацію методів обробки зображення:

- обертання;
- зсув;
- масштабування;
- відображення відносно осей.

Комбінація методів та їх ступінь інтенсивність в процесі обробки вибирається випадковим чином. Для задачі сегментації зображення та його маска мають відповідати один одному, тому одній й ті ж методи та їх інтенсивність застосовують для зображення та маски.

На рисунку 3.3 показан результат аугментації зображення.

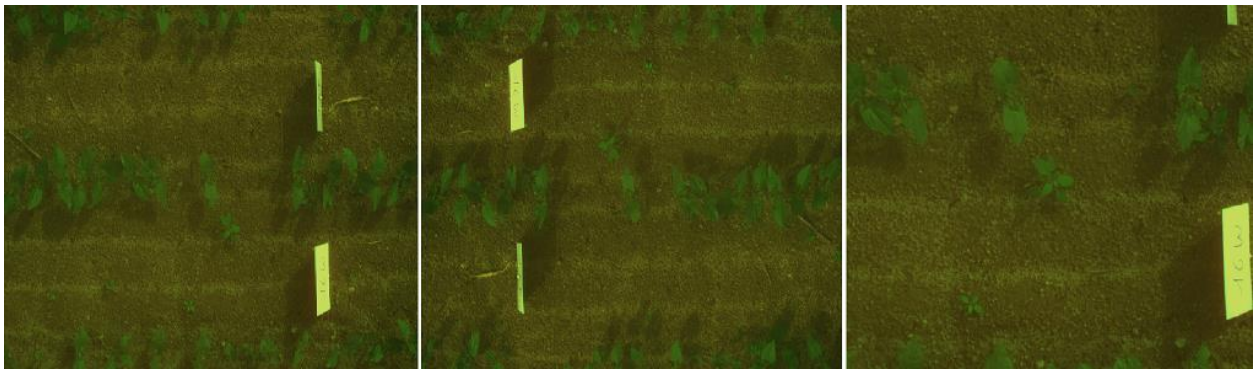

Рисунок 3.3 – Приклад аугментації (оригінальне зображення, повернення на 180 градусів, масштабування)

3.5 Зміна розмірності

Для правильної роботи необхідно ввести зображення однакового розміру. Також, від розміру навчання залежить час навчання мережі. При зображеннях великого розміру можуть виникнути проблеми з нестачею оперативної та відео пам’яті. Емпіричним шляхом було визначено, що оптимальний розмірність зображення є 512 на 512 пікселів. Для зміни розмірів використовується метод transform.resize пакету skimage.

3.6 Процесс навчання нейронної мережі

В Google Collaboratory є можливість використання TensorBoard. В процесі навчання программа записала логи як вона навчалась та час навчання. Після навчання автоматично виводиться графіки навчання та невдач.

На рисунку 3.4 показан графік навчання.

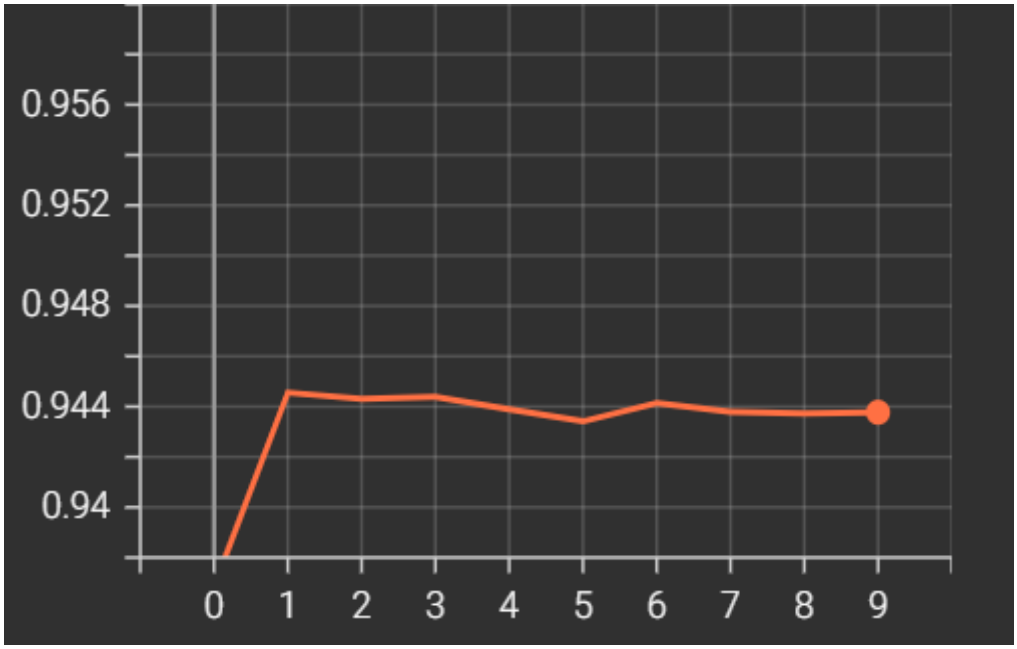

Рисунок 3.4 – Графік навчання

На рисунку 3.5 показано графік невдач при навчанні.

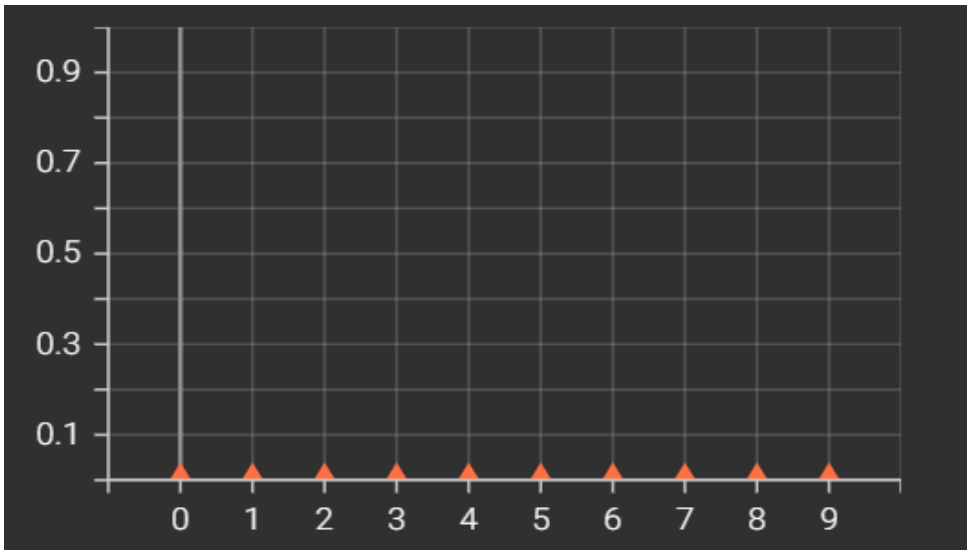

Рисунок 3.5 – Графік невдач

3.7 Результати сегментації зображення

Для демонстрації роботи програми було обрано декілька зображень з наявного набору даних.

На рисунку 3.6 показана коректна робота нейронної мережі, виявлені всі рослини та були індефіковани, де синім кольором показа культура рослини, а зеленим – бур’ян.

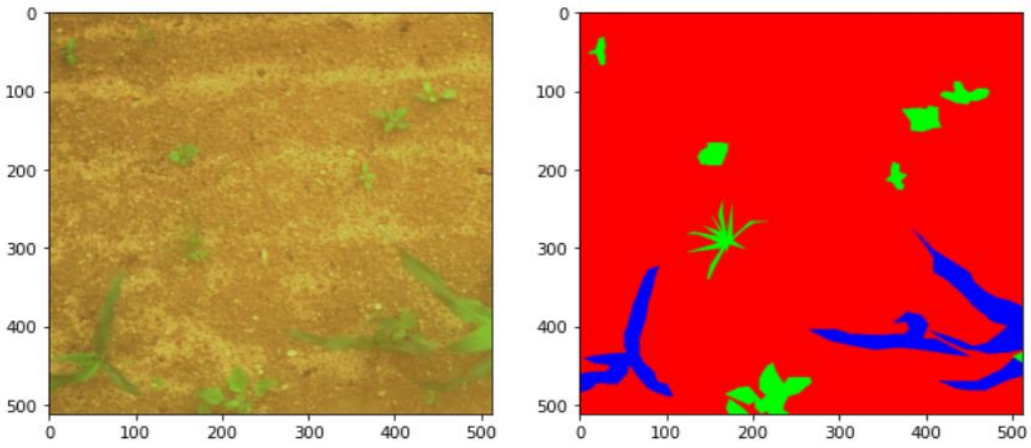

Рисунок 3.6 – Коректна сегментація

На рисунку 3.7 показана некоректна робота, система виявила лише одну росилину замість двух.

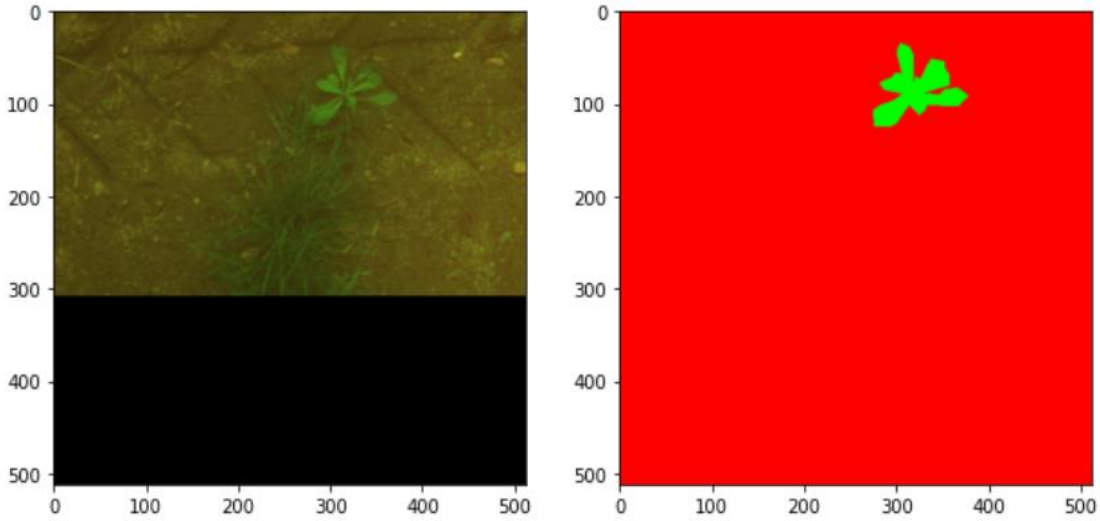

Рисунок 3.7 – Некоректна робота нейронної мережі

На рисунку 3.8 показана робота нейронної мережі де бур’ян та культура проросли разом та були переплетені, но мережа змогла розпізнати що є що.

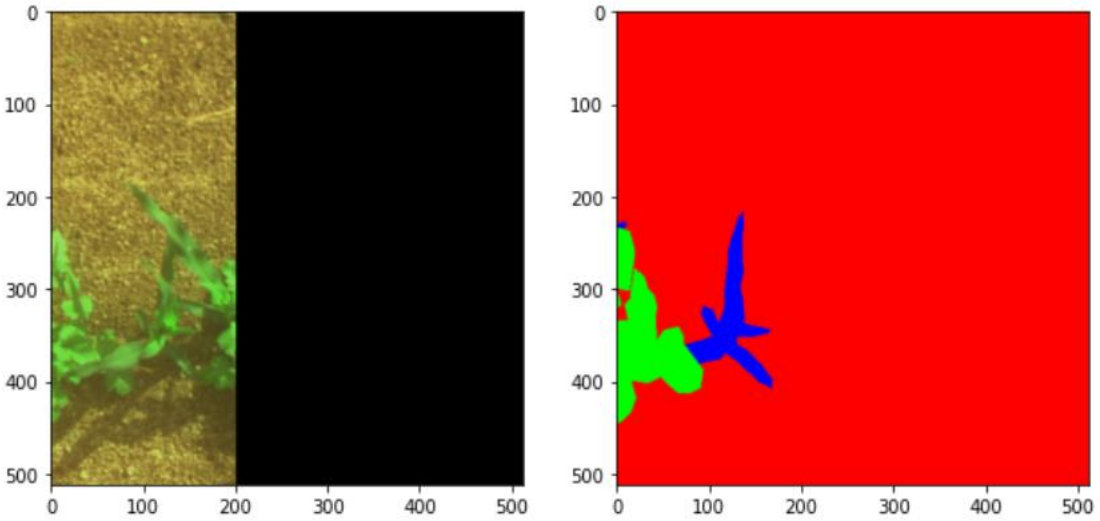

Рисунок 3.8 – Виявлення при сплутанні рослин

На мою думку якщо надати нейронної мережі набагато більше навчального матеріалу то результат буде набагато краще. Можна сказати що фото для навчання дуже легко робити тому що дрони на сьогодні мають дуже якісні камери які можуть робити фото у високій якості на довільно високій висоті щоб потоком повітря не порушувати структуру рослин.

ВИСНОВКИ

У рамках кваліфікаційної роботи був розроблений і реалізований метод сегментації зображень за допомогою U-Net та Keras.

Робота присвячена вирішенню проблеми машинного зору для агропромисловості на зображеннях здійснених на полях України. Цей метод та реалізація можуть допомогти великим господарствам у вирішенні проблеми з бур’яном, а саме получати статистику на окремих ділянках та приймати рішення про необхідність та методи вирішення.

Проведено аналіз підходів та методів розв’язання задачі сегментації, обгрунтовано використання згорткових нейронних мереж.

На основі проведених досліджень визначено архітектуру, склад та компоненти згорткової нейронної мережі для задачі сегментації.

За умови недостатності навчальних прикладів у відкритому наборі даних «Rose reval 2019» реалізовано аугменцію зображень.

Експериментально доведено ефективність програмного рішення для вирішення задачі по сегментації бур’янів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Детекторы и дескрипторы особых точек FAST, BRIEF, ORB URL: https://habr.com/ru/post/414459/ (дата звернення: 24.03.2022)

2. Обзор алгоритмов сегментации URL: https://habr.com/ru/company/intel/blog/266347/ (дата звернення: 24.03.2022)

3. Сегментация изображения с использованием wavelet-декомпозиции и watershed-преобразования URL: https://cyberleninka.ru/article/n/segmentatsiya-izobrazheniya-s-ispolzovaniem-wavelet-dekompozitsii-i-watershed-preobrazovaniya/viewer (дата звернення: 24.03.2022)

4. Robert M.Haracklinda G.Shapiro; Image segmentation techniques URL: https://www.sciencedirect.com/science/article/pii/S0734189X85901537 (дата звернення: 02.04.2022)

5. Dilpreet Kaur, Yadwinder Kaur; Various Image Segmentation.Techniques: A Review//IJCSMC, vol. 3, issue. 5, May 2014, pg 809-814. ISSN 2320-088X

6. Su Hnin Hlaig, Aung Soe Khaing; Weed and crop segmentation and classification using area thresholding URL: https://www.cse.unr.edu/~bebis/CS479/PaperPresentations/PlantClassificationWACV2014.pdf (дата звернення: 03.04.2022)

7. Plant segmentation by supervised machine learning methods URL: https://acsess.onlinelibrary.wiley.com/doi/10.1002/ppj2.20001 (дата звернення: 05.04.2022)

8. Image based leaf segmentation and counting in rosette plants URL: https://www.sciencedirect.com/science/article/pii/S2214317318301562 (дата звернення: 10.04.2022)

9. Comparing Semantick and Instance Segmentatio for Weed and Crop Detection URL: https://omdena.com/blog/image-segmentation-techniques-for-weed-or-crop-detection/ (дата звернення: 10.04.2022)

10. Plant Images Segmentation with Deep Learning URL: https://medium.com/zaka-ai/plant-images-segmentation-with-deep-learning-ff1ed67e80e6 (дата звернення: 17.04.2022)

11. Сверточные нейронные сети с нуля URL: https://medium.com/p/4d5a1f0f87ec/ (дата звернення: 17.04.2022)

12. Комплексное руководство по сверточным нейронным сетям URL: https://datastart.ru/blog/read/kompleksnoe-rukovodstvo-po-svertochnym-neyronnym-setyam-dlya-chaynikov (дата звернення: 18.04.2022)

13. Обзор CNN для сегментации изображения URL: https://russianblogs.com/article/3677224873/ (дата звернення: 18.04.2022)

14. How U-net works ? URL: https://developers.arcgis.com/python/guide/how-unet-works/ (дата звернення: 22.04.2022)

15. Deep Residual Learning for Image Recognitiion URL: https://arxiv.org/pdf/1512.03385.pdf (дата звернення: 22.04.2022)

16. Dropout метод решения проблемы переобучения в нейронных сетях URL: https://habr.com/ru/company/wunderfund/blog/330814/ (дата звернення: 23.04.2022)

17. Нейронные сети для начинающих. Часть 1 URL: https://habr.com/ru/post/312450/ (дата звернення: 23.04.2022)

18. Fig Plant Segmention from Aerial Images Using a Deep Convolution Encoder-Decoder Network URL: https://www.mdpi.com/2072-4292/11/10/1157 (дата звернення: 23.04.2022)

19. Starter: rose eval 2019 7c364575-1 URL: https://www.kaggle.com/code/kerneler/starter-rose-eval-2019-7c364575-1/data (дата звернення: 24.04.2022)

20. Разница между U-Net и FCN URL: https://blog.csdn.net/weixin_40519315/article/details/104408388 (дата звернення: 24.04.2022)

21. Моделювання у процесі проектування інтелектуальної системи діагностування / О.Я. Кузіомін, О.О. Василенко //Радіоелектроіка та інформатіка. 2019. №2. С. 61-66.

22. Розробка багатоагентних структур для вирішення проблем медичної системи діагностування / О.Я. Кузьомін, О.О. Василенко, Свістунов І.О. // Радіоелектроніка та інформатіка. 2020. №2. С. 47-54.

23. Розробка структур медичних агентів для вирішення проблем медичної системи діагностування / О.Я. Кузьомін, О.О. Василенко, Горшколєпов А. В.// Радіоелектроніка та інформатика. 2020. №2. С. 55-65.

24. Research of the intellectual system of knowledge search in Databases / Oleksii Vasilenko, Oleksandr Kuzomin, Bohdan Maliar // International Journal " Inforamation Models and Analyses" Volume 7, Number 4. PP.2019. 327-338.

25. Research of medical diagnostic data search methods / Oleksii Vasilenko, Oleksandr Kuzomin, Oleksandr Shapoval // International Journal " Information Models and Analyses&qout; Volume 7, Number 4. 2019. PP. 339-349.

26. Intellectual Models and Means of the Biometrick System Dynamics of Rinosinusite / Oleksii Vasilenko, Oleksandr Kuzomin, Tatyana Khripushina // International Journal" Inforamation Models and Analyses" Volume 7, Number 4. 2019. PP. 350-361.

27. Forming Medical Database and knowledge for Diagnostic Disease / Oleksii Vasilenko, Oleksandr Kuzomin, Vladislav Shvets. // International Journal " Information Models and Analyses" Volume 7, Number 4. 2019. PP. 362-372.

28. Automated Tests for Errors in Computer System ‘Environment’ / Jleksii Vasilenko, Olksandr Kuzomin. // International Journal" Information Models and Analyses" Volume 7, Number 4. 2019. PP. 373-384.

29. Developing methods bases on text mining technology to Improve the quality and speed of automatic clustering of Documents / Oleksii Vasilenko, Oleksandr Kuzomin, Artem Mertsalov // International Journal " Information Models and Analyses" Volume 7, Number 4. 2019. PP. 385-397.

30. The patient organism modeling for diagnosis with the usage of a multi agent representation / Kuzomin O., Dudka O., Vasylenko O., Lyashenko V. // International Journal of Emerging Trends in Engineering Research, 2020, 8(9), pp. 5733-5739.

31. Mobile expert system for diagnostic human state in emergency situations. / Kuzomin O., Dudka O., Vasylenko O., Shylo R., Lyashenko V. // International Journal of Advanced Trends in Computer Science and Engineering, 2020, 9(4), pp. 6484-6489, 334.

32. Using of ontologies for building databases and knowledge bases for consequences management of emergency/Kuzomin O., Dudka O, Vasylenko O., Darchenko V., Lyashenko V. // International Journal of emergency / Kuzomin O., Dudka O., Vasylenko O., Shylo R., Lyashenko V. // International Journal of Advaced Trends in Computer Trends in Computer Science and Engineering, 2020, 9(4), pp. 5040-4045.

33. Inelligent geoinformatic expert system for providing emergency help during extreme situations. / Kuzomin O., Stukin M., Bozhkov D., // International Multidisciplinary Scientific GeoConference Surveying Geology and Mining Ecology Management, GEM, 2019, 18(2.2), pp. 269-276.