НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ О.В.Коваль
(підпис) (ініціали, прізвище)
“___”_____2018 р.

Дипломна робота
на здобуття ступеня бакалавра

з напряму підготовки
6.050101 “Комп’ютерні науки’’

на тему: Сегментація зображень на основі згорткових нейронних мереж

Виконав: студент 4 курсу, групи ТР-52

Софієнко Антон Юрійович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник доцент, к.т.н. Шаповалова Світлана Ігорівна
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний
Кафедра автоматизації проектування енергетичних процесів і систем
Рівень вищої освіти перший рівень
Напрям підготовки 6.050101 “Комп’ютерні науки’’

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2018 р.

ЗАВДАННЯ
на дипломну роботу студенту
Софієнку Антону Юрійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи “Сегментація зображень на основі згорткових нейронних мереж”

керівник роботи _____доцент, к.т.н. Шаповалова Світлана Ігорівна_____
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”___”_____ 201__ р. № _____

2. Строк подання студентом роботи ___ _____ 201__ р.
3. Вихідні дані до роботи набір даних (DataSet) - Sugar Beet 2016___
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) проаналізувати існуючі програмні рішення та можливі засоби реалізації системи сегментування бур’янів на зображеннях, обґрунтувати обрані програмні застосунки та шляхи розробки програмних додатків, розробити програмне забезпечення, розробити користувацький інтерфейс, зробити висновки за результатами роботи_____
5. Перелік ілюстраційного матеріалу (з точним зазначенням обов’язкових креслень)
1. Задача сегментації. 2.Задача розпізнавання бур’яну 3. Характеристика DataSet Sugar Beets 2016 . 4.Схема навчання. 5.Аугментація зображень. 6. Згорткові нейронні мережі. 7.Підхід Encoder Decoder. 8.Архітектура U-net. 9.Схема навчання. 10. Схема роботи. 11. Приклади сегментації.

6. Дата видачі завдання ”___”_____201___ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи		
2.	Вивчення та аналіз задачі		
3.	Розробка архітектури та загальної структури системи		
4.	Розробка структур окремих підсистем		
5.	Програмна реалізація системи		
6.	Оформлення пояснювальної записки		
7.	1 Захист програмного продукту		
8.	2 Передзахист		
9.	Захист		

Студент _____ Софієнко А.Ю.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Шаповалова С.І.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Метою роботи було дослідження застосування згорткових нейронних мереж для розв'язання задачі сегментації на прикладі сегментації бур'янів.

Було розглянуто алгоритмічні підходи та архітектури згорткових нейронних мереж призначені для сегментації зображень. Для створення системи було обрано архітектуру U-net. Програмну модель архітектури для сегментації було реалізовано за допомогою мови програмування Python та бібліотеки Keras. Було проведено дослідження поведінки зображення бур'яну у внутрішніх шарах мережі.

Результатом роботи стала мережа для сегментації бур'янів на зображеннях з відеокамери наземного робота.

Записка містить 70 сторінок, 19 рисунків, 2 таблиці, 5 додатків і 26 посилань.

Ключові слова: СЕГМЕНТАЦІЯ, КОМП'ЮТЕРНИЙ ЗІР, СІЛЬХОЗРОБОТИ , U-NET ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ.

ABSTRACT

The aim of the work was to investigate the use of convolutional neural networks to solve the segmentation problem by using a weed segmentation example. Algorithmic approaches and architectures of convolutional neural networks were considered for image segmentation. The U-net architecture was chosen to create the system. The software architecture for segmentation was implemented using the Python programming language and the Keras library. We conducted a survey of the behavior of the image of the weed in the inner layers of the network.
The result of the work was the network for segmentation of weeds in the images from the video camera of the ground robot.

The note contains 70 pages, 19 figures, 2 tables, 5 annexes and 26 references.

Keywords: SEGMENTATION, COMPUTER VISION, AGRICULTURE ROBOTS, U-NET, CONVOLUTIONAL NEURAL NETWORK.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

CNN — Convolutional Neural Network.

FCN — Fully Convolutional Network.

IoU — Intersection over Union.

API — Application Programming Interface.

GUI — Graphical User Interface.

TCP — Transmission Control Protocol.

Вступ

Стрімкий розвиток інформаційних технологій супроводжується збільшенням ролі розумних пристроїв у повсякденному житті людини. Для роботизованих систем необхідно володіти достатньою кількістю відомостей про навколишнє середовище для прийняття вірного 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 відеопотоку бортової камери. На основі отриманої інформації робот обирає модель поведінки. Важливо точно розпізнати інформацію із зображення, щоб система керування могла вірно сформувати доцільну поведінку робота, для вирішення поставленої задачі. Отже, задача сегментації інформації на зображеннях з бортових систем 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 бур'яна.

В даній роботі було використано архітектуру U-net згорткових нейронних мереж для для сегментації бур'янів на зображеннях з відеокамери наземного роботу.

У першому розділі записки сформульовано постановку задачі сегментації, розглянуто алгоритмічні підходи, розглянуто проблеми та дослідження по сегментації рослин, сформульовано постановку задачі; у другому розділі розглянуту згорткові нейронні мережі, розвиток та застосування ЗНМ для задач сегментації, розглянуто архітектуру U-net; третій розділ містить опис програмної реалізації; в п'ятому розділі наведено експеременти по роботі з моделлю; у п'ятому розділі показано приклади сегментації.

1 ПІДХОДИ І МЕТОДИ РОЗВ'ЯЗАННЯ ЗАДАЧІ СЕГМЕНТАЦІЇ

Існує безліч алгоритмів та основних методів придатних до сегментації зображення, проте цей розділ комп'ютерного зору все ще потребує інноваційних підходів для покращення якості та швидкості розпізнавання.

У сільському господарстві машинний зір може бути застосований для інтелектуального контролю та боротьби з бур'янами. В даному дослідженні розв'язується задача сегментації бур'янів на зображеннях фрагменту поля зроблених з рухомої платформи.

1.1 Задача сегментації

Сегментація — це процес присвоєння пікселям зображення міток так, що області, які мають спільні властивості мали однакові мітки. Також сегментацією називають пошук проекцій об'єкта на зображення. Мета сегментації полягає у спрощенні і/або зміні представлення зображення для полегшення його аналізу. Результатом сегментації зображення є множина сегментів, які разом покривають все зображення, або множина контурів, виділених з зображення. Всі пікселі в сегменті схожі за деякою характеристикою або за визначеною властивістю, наприклад, колір, яскравість або текстура [1]. Сусідні сегменти істотно відрізняються за цими характеристиками.

Ця задача є більш складною ніж задача класифікації зображень оскільки вимагає не тільки визначення класів об'єктів, але й визначити їх положення та попіксельного маркування об'єктів на зображенні.

Це фундаментальна задача комп'ютерного зору, яка вирішується різними способами, кожен з яких володіє тими чи іншими недоліками.

1.2 Загальні підходи

Процес сегментації зображень відбувається по інформації наявній в самому зображенні. Це може бути інформації про колір пікселя, перепади кольорів та відтінків чи текстура зображення.

Техніки сегментації можна поділити на дві групи:

— методи на основі подібності;

— методи на основі границь (порогів).

Кожний підхід має різні варіанти реалізації. На рисунку 1.1 представлено основні методи сегментації.

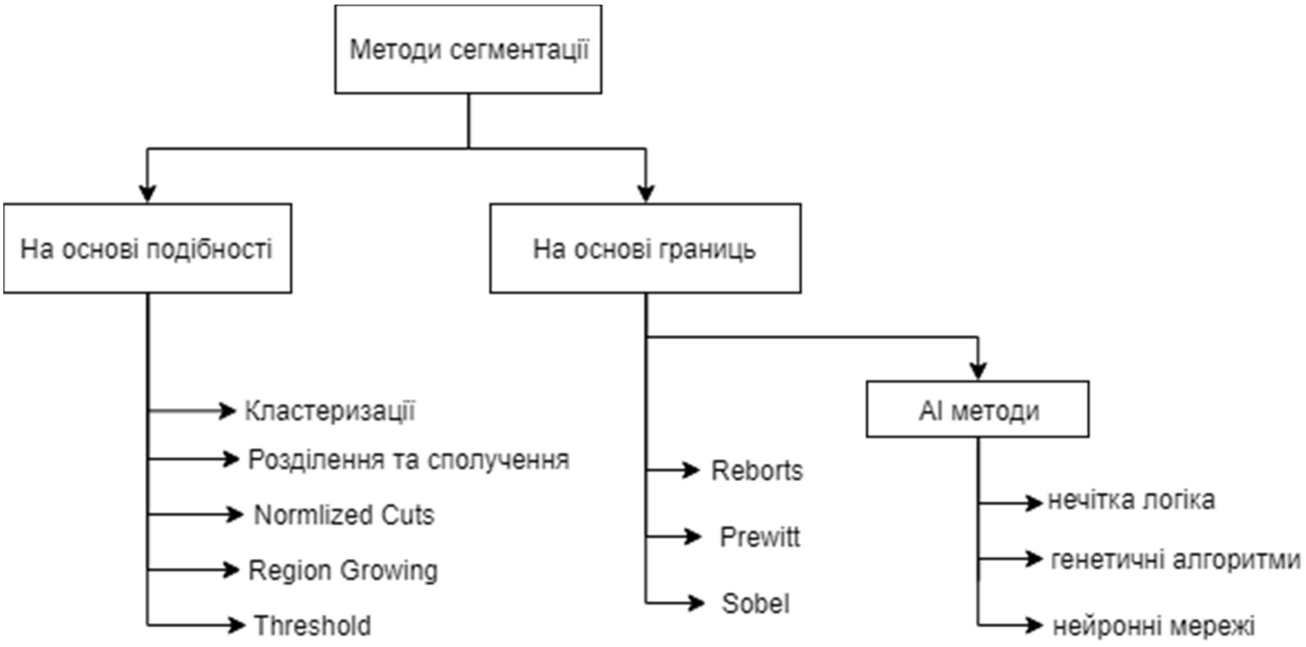


Рисунок — 1.1 Класифікація алгоритмів сегментації

Підхід на основі границь або порогова сегментація – це підхід, в якому зображення сегментується на регіони на основі певного розриву в характеристиках. Завдяки чому виділяються краї сегментів. На основі виявлення країв сегмент потрапляє в одну з категорій, які формуються за рахунок розриву інтенсивності між регіонами. Ця техніка є однією із

найпоширеніших серед методів тому що є однією з найпростіших. Ключовим моментом у використанні даного методу є вибір значення порогу. Для його визначення на практиці використовують метод максимальної ентропії, збалансований поріг гістограми, гібридне порогове значення, метод Otsy (максимальної дисперсії).

Підхід на основі виявлення подібності (кластеризації) – це підхід, в якому зображення сегментується на регіони на основі подібності. Методи, розділяють зображення на регіони, що мають подібний набір пікселів. Подібно до кластеризації (Методи кластеризації також використовують цю методологію. Вони поділяють зображення на множину кластерів, що мають подібні особливості, засновані на деяких заздалегідь визначених критеріях). Його суть полягає в тому, щоб автоматично визначити оптимальний поріг за певним критерієм. Використовуючи це, виконується кластеризація за рівнем сірого. Основна ідея регіонального алгоритму полягає в тому, щоб об'єднати пікселі за подібними властивостями. Тобто для кожної області зайти один піксель, який буде точкою зростання, насіниною, а потім об'єднуються навколишні пікселі з подібними властивостями. Далі виконується визначення країв на основі розриву в кольорах [2].

1.3 Алгоритм сегментації WaterShed

Алгоритм сегментації по водорозділах (WaterShed) – це один із інтуїтивно зрозумілих методів сегментації на основі границь. Він працює із зображенням, як із функцією з двома змінними, де змінні – це координати пікселя. Значеннями функції може бути інтенсивність пікселя. В місцях перепаду інтенсивності утворюється хребти, а в однорідних регіонах рівнини. Після знаходження мінімумів функції, йде процес заповнення "водою", який розпочинається з глобального мінімуму. Після досягнення одного з локальних мінімумів, він теж

починає заповнюватися. Як тільки два регіони починають зливатися, будується бар'єр, щоб запобігти злиттю двох регіонів. Процес заповнення продовжується доки всі регіони не будуть розділені бар'єрами. Ілюстрацію роботи алгоритму WaterShed наведено зі статті [3] на рисунку 1.2.

Цей алгоритм може бути використаний у випадках коли на зображенні мала кількість регіонів, тобто локальних мінімумів. В інакшому випадку виникають зайві розбиття на регіони.

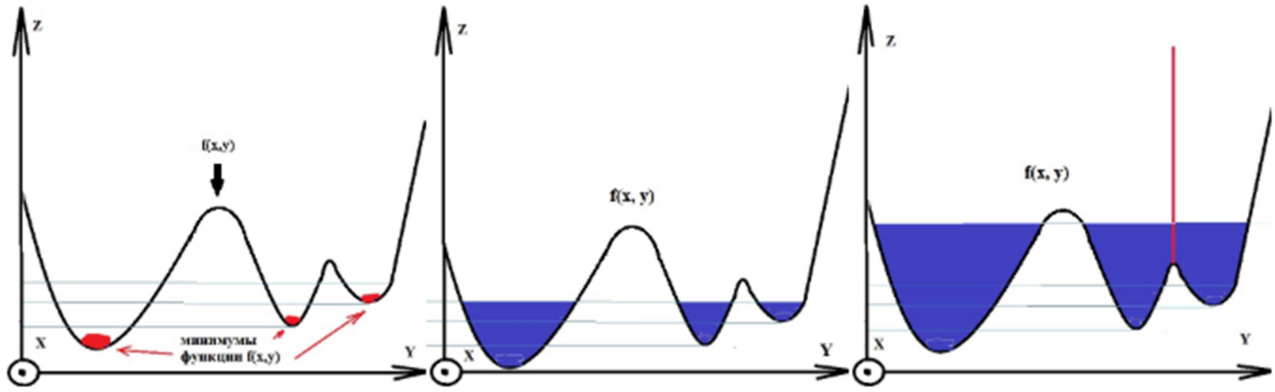


Рисунок — 1.2 Ілюстрація роботи алгоритму WaterShed [3]

Задачі сегментація зображення можна поділити на два основні типи по відношенню до області зображення: локальна сегментація (пов'язана з певною частиною або областю зображення) і глобальна сегментація (пов'язана з сегментація всього зображення, що складається з великої кількості пікселів).

1.4 Сегментації рослин

Одна із головних задач людства є виробництво достатньої кількості продовольства, кормів та палива для постійно зростаючого населення світу одночасно зменшуючи екологічний вплив сільськогосподарського виробництва.

Використання агрохімікатів потрібно скоротити, щоб зменшити забруднення навколишнього середовища та зупинити зниження біорізноманіття.

Цю задачу можна вирішити виконуючи постійний моніторинг стану рослин з використанням автономних роботів з автономними системами прийняття рішення.

При звичайній боротьбі з бур'янами системи обробляють все поле рівномірно з однаковою дозою хімічних засобів. Використовуючи роботизовані системи можна проводити вибіркове оброблення поля або, навіть, механічне видалення бур'янів. Це вимагає системи класифікації рослин, що здатна аналізувати зображення та маркувати індивідуальні зображення, як культуру або бур'ян.

Сегментації рослин на зображення полях зазвичай відбувається в такій послідовності:

— Подання на вхід системи кольорове зображення і, за наявності, інфрачервоне зображення подаються на вхід системи;
— визначення рослинності на зображенні та відділення його від фону. При зйомці з багатоспектральних камер доцільно використовувати вегетаційний індекс;
— класифікація рослини;
— маркування ділянки зі знайденою рослиною на зображенні;

Задачі по сегментації рослин вирішували дослідники з різних університетів.

Науковці технічного університету Mandalay зробили опис [4] розробленої системи для сегментації рослинності на фоні ґрунту. Ними було розроблено алгоритм для вилучення рослин на основі інтенсивності зеленого та сірого кольорів та подальшої їх класифікації.

Haug [5] запропонував спосіб розрізнення рослин моркви та бур'янів зі знімків RGB камери та інфрачервоних зображень (NIR). Розроблена система

може розрізняти рослини моркви та рослин бур’янів, що ростуть на полях. Система здатна обробляти перекриття між рослинами.

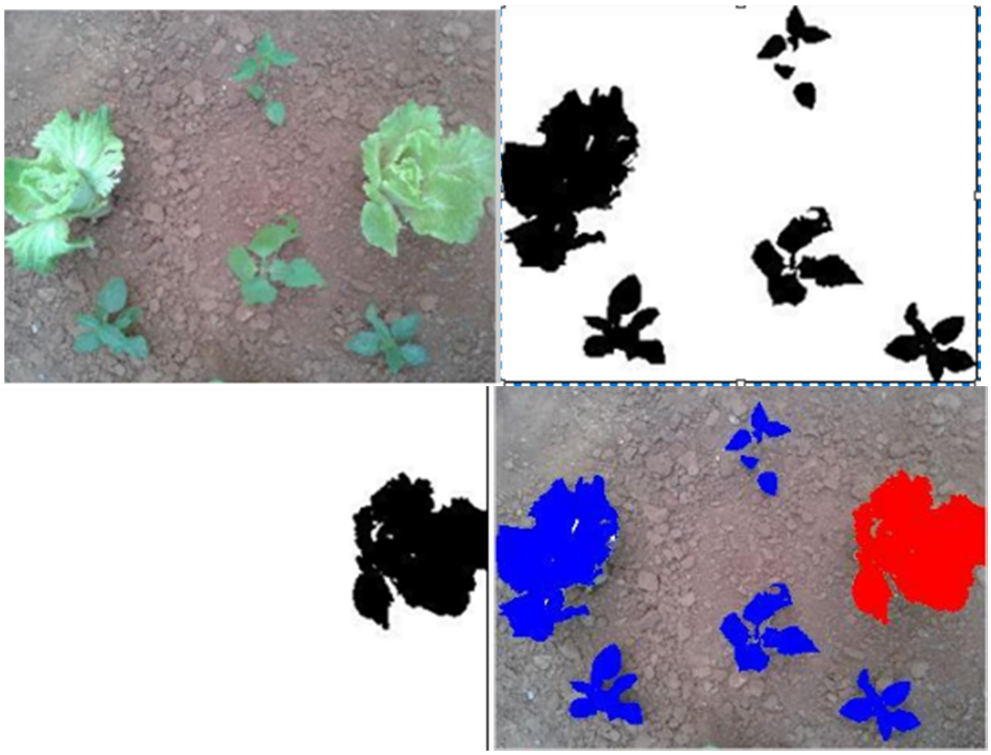


Рисунок — 1.3 Покроковий процес вилучення класифікації та сегментації рослин. [4]

Замість сегментації зображення на окремі листя або рослини, дослідники використовуємо класифікатор Random Forest для визначення належності пікселя до бур’яну чи моркви у базуючись на сусідні пікселі. Отримані результати згладжуються шляхом інтерполяції. Робота дає середню точність класифікації 93,8%.

В роботі Lottes [6] використав інформацію про положення рослини культури відносно поля. Оскільки рослини висаджуються рядам (рисунок 1.4), тобто одна за одною, а бур’яни в хаотичному порядку, ця особливість була використана в роботі, як додатковий параметр для класифікації. Інформація про геометричне положення поєднується з візуальною інформацією, для отримання більш точних прогнозів. Також, в цій роботі було вирішені проблеми сегментації рослин з різними кольорами ґрунту та різним освітленням. При переході від

сегментації в лабораторії до польової сегментації можна зіткнутися з проблемою накладання однієї рослин на іншу.

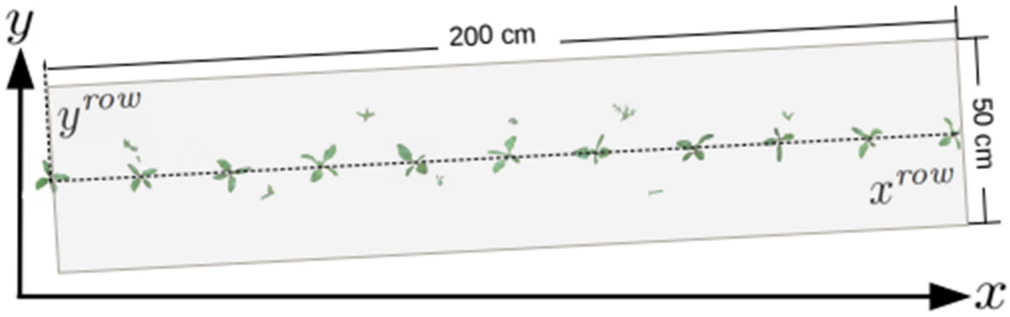


Рисунок — 1.4 Геометричне положення рослин культур на ділянці поля [6]

В статті “Сегментація листя з накладанням на основі Chan–Vese моделі та оператора Собеля” [7] Wang та колеги розв’язують проблему сегментації з накладанням листя огірка один на одного (рисунок 1.5).

Дослідники використовують Chan–Vese (C-V) модель для виявлення для виявлення границь на зображення листя з поєднанням локальної інформації з глобальною.

Запропоновано ефективний метод сегментації зображень для перекриття листя врожаю, що поєднує в собі модель C-V і Собель оператора. Спосіб включає чотири основних модуля, а саме видалення фону, виявлення краю Собеля, контур C-V видобуток і сегментація результату синтезу. Фон модуль видалення використовується для видалення тих пікселів, які не є вважається частиною листкових областей. Контур цілі лист витягується з використанням моделі C-V. У третьому модулі, вісім спрямованих операторів Собеля реалізовано для виявлення краї аркуша. Нарешті, в модулі злиття результатів, результати, отримані моделем C-V і оператором Собеля об'єднують і витягують цільовий лист.

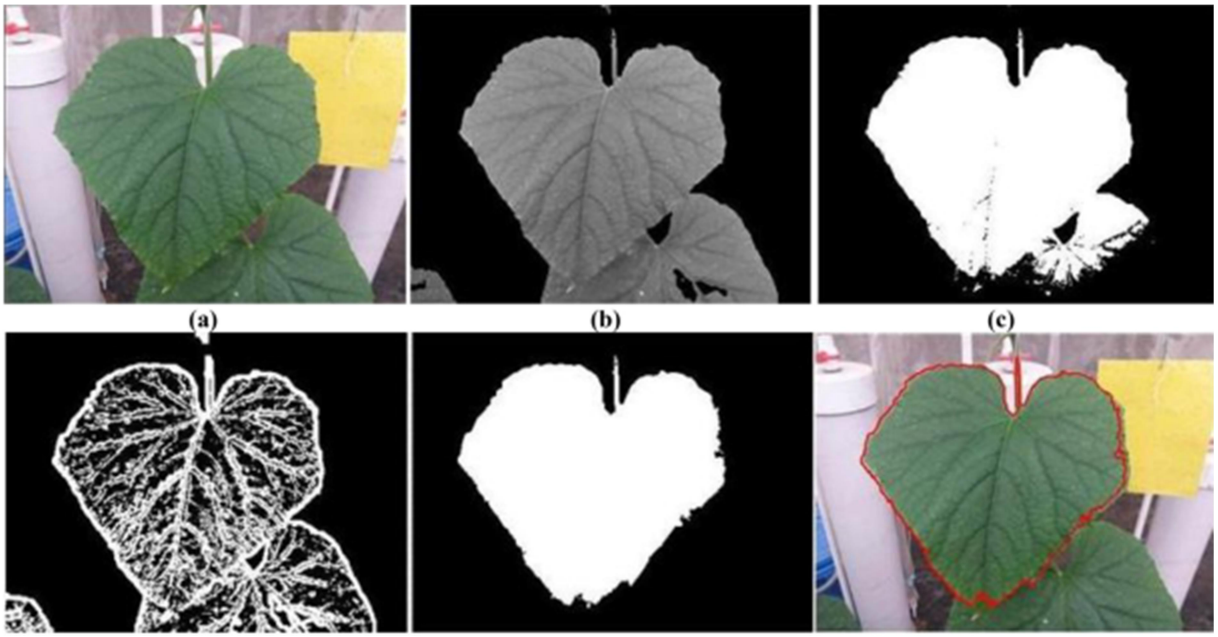


Рисунок — 1.5 Результат сегментації листя. (a) оригінальне зображення; (b) результат з видаленим фоном; (c) результат сегментації C-V моделі; (d) результат після виділення кутів; (e) після злиття; (f) кінцевий результат [7]

Використовуючи наявний набір даних розробити інформаційну систему для сегментації рослин перію на зображеннях ділянки поля цукрового буряка створених з висоти 30 см за допомогою рухомої роботизованої платформи засобами згорткових нейронних мереж.

Метою роботи є створення програмної системи на основі нейронних мереж для вирішення задачі сегментації бур'янів на зображеннях з камери робота.

Дослідники університету Бонна створили робот для прополювання бур'яну. Робот виконував 4-канальну мультиспектральну фотозйомку та сенсор RGB-D, щоб захопити детальну інформацію про плантацію. Кілька датчиків lidar і GPS, а також колісні кодери забезпечували вимірювання, що стосуються локалізації, навігації та відображення. Таким чином, дослідниками було отримали близько 5 ТБ даних.

Науковці частково виклали на відкритий ресурс [8] датасет зі знімками поля цукрового буряка отриманий на території кампусу Klein Altendorf in Bonn,

Німеччини. Датасет складається із зображень розмірами 966 на 1296 пікселів фрагменту поля з камери, розташованої на висоті 30 см від ґрунту. Набір даних для навчання було поділено на тренувальну вибірку 239 зображень, валідаційну з 44 та 30 зображень було використано для тестування. Дані наведено в таблиці 1.1.

Таблиця 1.1 Вибір та кількість зображень

Назва вибірки	Кількість зображень
Тренувальна	239
Валідаційна	44
Тестувальна	30

В зв'язку з малою кількістю доступних навчальних прикладів було прийнято рішення збільшити вибірку шляхом аугментації наявних зображень.

В розроблену систему на вхід подається кольорове зображення ділянки поля з висоти 30 см. Після обробки вихідними даними є бінарна маска цього зображення, з маркованими пікселями, що належать бур'яну. Для наочності маска була накладена на вхідне зображення, таким чином візуально виділивши бур'ян.

Протягом десятків років було розроблено велика кількість різних алгоритмів по сегментації, проте для сегментації рослинності їх застосовують мало. Семантична сегмнетція рослин — це складна задача. На це є декілька причин:

1. Візуальна схожість. Більшість із рослин зеленого кольору, різниця полягає лише у відтінках.
2. Велика кількість видів. Рослинний світ одного регіону може істотно відрізнятися від іншого, це унеможливлює створення універсального класифікатора.
3. Ріст рослин. Рослини з часом розвивається, змінює форму листя та

структуру. На різних стадіях росту зовнішній вигляд змінюється.

4. Накладання рослин. На одній ділянці часто присутня рослинність різних видів. В процесі росту рослини можуть накладатися одна на одну, переплітатися.
5. Деформації та мутації рослин. Ріст і розвиток рослини залежить від багатьох факторів, таких вид та якість ґрунту, вологість та опади, кількість сонячного світла та пошкодження шкідниками. Тож дві рослини одного й того ж виду можуть кардинально різнитися у зовнішньому вигляді.
6. Погодні умови та освітленість при зйомці. Для сегментації потрібно отримати знімок з поля. Час доби та погодні умови в процесі зйомки спотворюють зображення, що в свою чергу додає труднощів для обробки зображення. Недостатня освітленість рослин, наявність крапель роси, деформація рослин від вітру можуть призвести до неякісної або хибної сегментації.

Нейронні мережі мають потенціал до вирішення більшості перелічених проблем. Оскільки здатні навчатися та покращувати якісь своє роботи зі збільшенням кількості наявних прикладів. Тобто, щоб покращити якість сегментування зображень з поганим освітленням, достатньо надати мережі більше навчальних прикладів з поганим освітленням. Нейронні мережі можна адаптувати під конкретний природній регіон та набір рослин.

Недоліком використання мереж для сегментація рослин є їхня недетермінованість. Тобто, після навчання мережі логіка прийняття рішення про належність пікселя до того чи іншого класу рослин є незрозумілою для експерта.

2 СЕГМЕНТАЦІЯ ЗГОРТКОВИМИ НЕЙРОННИМИ МЕРЕЖАМИ

Сегментація зображення також може бути досягнута за допомогою нейронних мереж.

Нейронна мережа – це математична модель пов'язаних між собою штучних нейронів, які організовані в єдину структуру та впливають на роботу один одного. Ця концептуальна модель була розроблена на основі роботи клітин головного мозку. Завданням нейромережі є перетворення вхідного вектора у вихідний. У випадку задачі сегментації вхідним вектором є зображення, вихідним вектором є його маска.

Теоретично, повнозв'язна нейронна мережа прямого поширення може бути використана для задач комп'ютерного зору, проте використання цієї архітектури до зображень є непрактичним, оскільки необхідно набагато більше число параметрів та нейронів так як кожен піксель зображення має бути параметром мережі. До того ж, пошук необхідних ознак буде відбуватися тільки в тих локальних частинах зображення, де вони були під час тренування.

2.1 Згорткові нейронні мережі

Згорткові нейронні мережі (ЗНМ, англ. convolutional neural network, CNN) [9] – це клас глибоких нейронних мереж прямого поширення, що містять шари згортки. Основна ідея полягає в повторному використанні однієї і тієї ж частини нейронної мережі з різними локальними ділянками входу невеликого розміру. Це зменшує кількість вільних параметрів дозволяючи мережі бути глибшою при цьому зберегти малу кількість вхідних параметрів. Це допомагає вирішити проблему затухання або збільшення градієнтів режимі тренування.

Цей тип мереж зазвичай використовується для обробки зображень, хоча також застосовуються для обробки аудіо та природної мови.

Свою назву отримали від математичної операції згортки (рисунок 2.1). Вона виконує лінійне перетворення вхідних даних.

Особливістю та перевагою цього типу мереж є те, що вони здатні розпізнати об’єкт незалежно від його положення на зображенні.

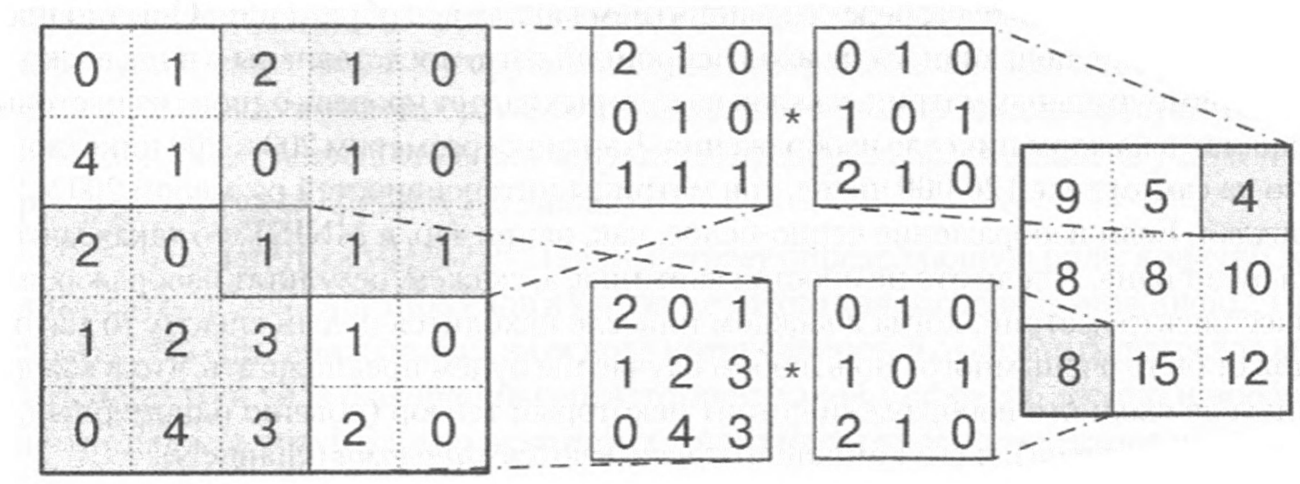


Рисунок — 2.1 Приклад обрахунку операції згортки [17].

Однією із головних особливостей згорткових нейромереж є те, що вони не потребують попередньої обробки даних. У випадки обробки зображень згортковими мережами це означає те, мережа сама створює фільтр, які, у випадку традиційних підходів, мали розробляти експерти. Перевагою є те, що такі мережі є незалежними від апріорних знань по предметній області.

Згорткова нейронна мережа між входом та виходом має приховані шари. Вони зазвичай складаються зі згорткових шарій, агригаційний шарів (анг. pooling layers), шарів нормалізації та повнозв’язних шарів.

В ідея роботи згорткових нейронних мереж було використано дані досліджень роботи зорової кори головного мозку. Згорткові шари імітують реакцію нейроній головного мозку на візуальних сигнал. Один згортковий нейрон може обробляти інформацію лише в межах свого рецептивного поля.

Рецептивні поля всіх нейронів перекриваються таким чином, що вони перекривають все зображення.

2.2 Розвиток CNN архітектур для сегментації

Історично першими для семантичної сегментації використовувалися підходи машинного навчання, такі як класифікатори TextonForest і Random Forest. З появою згорткових нейронних мереж CNN вирішення задачі комп'ютерного зору перейшло на новий етап розвитку. Архітектури, в яких за декількома шарами згорток йшли повнозв'язні шари нейронів були успішно застосовують для класифікації зображення. Після успішного використання CNN для класифікації заданих об'єктів на зображеннях з'явилися дослідження по використання CNN для сегментації зображень.

Першою спробою сегментації, яка базується на CNN, була латкова класифікація [10]. Метод класифікував кожен піксель зображення на основі обробки згортковими шарами частини зображення (латки), що містила його, а потім рішення про належність до того чи іншого класу приймалася повноз'язними шарами. В зв'язку зі зростанням часу роботи та вимог до обчислювальних машин метод не набув популярності.

Спроби адаптувати класифікаційні CNN для задач сегментації не були успішними. Їхнім недоліком є використання повнозв'язних шарів та пулінг шарів (анг. pooling layers). Використання повнозв'язних шарів накладає обмеження на розмір вхідного зображення – сегментувати можна тільки зображення одного розміру. Використання пудлінгових шарів дозволяє взяти більше ознаки з більшої частини зображення та об'єднати їх. Але при цьому втрачається інформація про положення ознак на зображення. Що є неприпустимим для сегментації.

Вперше ці обмеження усунуто Jonathan Long в роботі "Повнозгорткові нейронні мережі для семантичної сегментації". Повнозгорткова мережа (FCN) (Fully Convolutional Network) [11] – це архітектура CNN мережі, в якій повнозв'язні шари, що виконують функцію класифікації ознак, замінені на шар вибірки (анг. upsampling), який локалізує ознаки (рисунок 2.2.1).

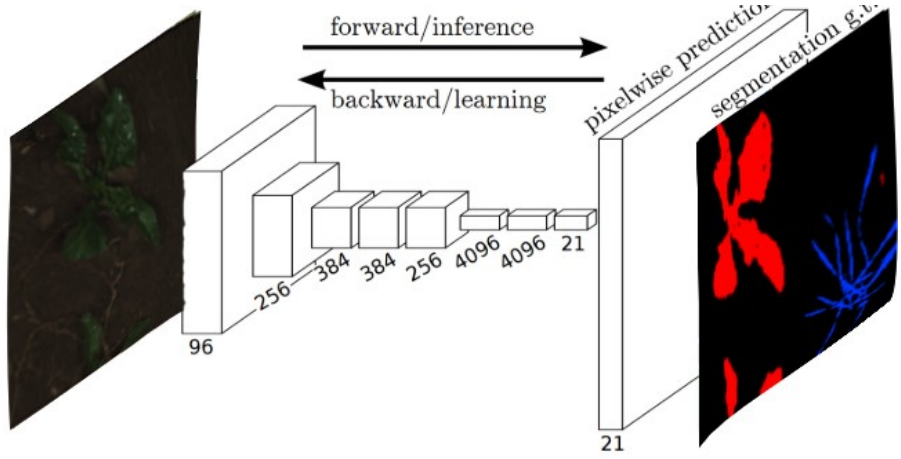


Рисунок — 2.2 Мережа для сегментації FCN.

Це дозволяє приймати на вхід зображення будь-якого розміру та генерувати карти сегментації, при цьому цей підхід працює швидше, оскільки не потрібно проводити класифікацію. Розроблена архітектура FCN-8 стала основою для подальших розробок архітектор, що виконують сегментацію зображень.

Базуючись на роботу Лонга, більшість мереж для сегментації будуються, як поєднання двох мереж, одна з яких виконує роль кодувальника, та може бути попередньо натренованою класифікаційною мережею (VGG/ResNet чи іншою), а інша - декодувальника, яка розгортає вихідний вектор кодувальника до двовимірної маски вхідного зображення. Таким чином на першому етапі кодувальник визначає що саме знаходиться на зображенні, а декодувальник – де саме знаходиться об'єкт. Різні архітектури мереж виконують процедуру декодування по різному. Серед мереж призначених для сегментації можна

виділити FCN, SegNet, U-Net, FC-Densenet E-Net & Link-Net, RefineNet, PSPNet, Mask-RCNN.

2.3 Архітектура U-net

Для виконання сегментації було використано згорткову нейронну мережу побудовано на основі архітектури U-net [12].

Це спеціалізована архітектура повнозгорткової нейронної мережі розроблена для сегментації біомедичних зображень, де кількість навчальних зображень є обмеженою. Цей тип мережі добре себе зарекомендував в численних змаганнях на платформі Kaggle завдяки можливості навчатися на малій кількості навчальних матеріалів та швидкості роботи.

Мережа побудована за принципом кодувальник-декодувальник. Ліва частина мережі виконує кодування ознак, а права – розкодування. Схема архітектури схожа на букву “U” латинського алфавіту, що було відображено в назві. На рисунку 2.4 наведено діаграму архітектури з положенням блоків та зв’язками.

Запропонована мережа складається з 4 блоків кодування, середнього блоку та 4 блоків розкодування.

Блок кодування складається з двох згорткових шарів (на зображенні позначені фіолетовою стрілкою) з вікном згортки 3 пікселі на 3 пікселі та функцію ReLu для активації, за якими слідує пулінговий шар (позначений червоною стрілкою) з вікном 2 пікселі на 2 пікселі завдяки чому розмірність зображення зменшується в два рази.

Після проходу всих 4 блоків кодування зображення зменшується до розмірності 32 пікселі на 32 пікселі з 512 каналами ознак. Після проходу середнього блоку розпоченається процес розкодування.

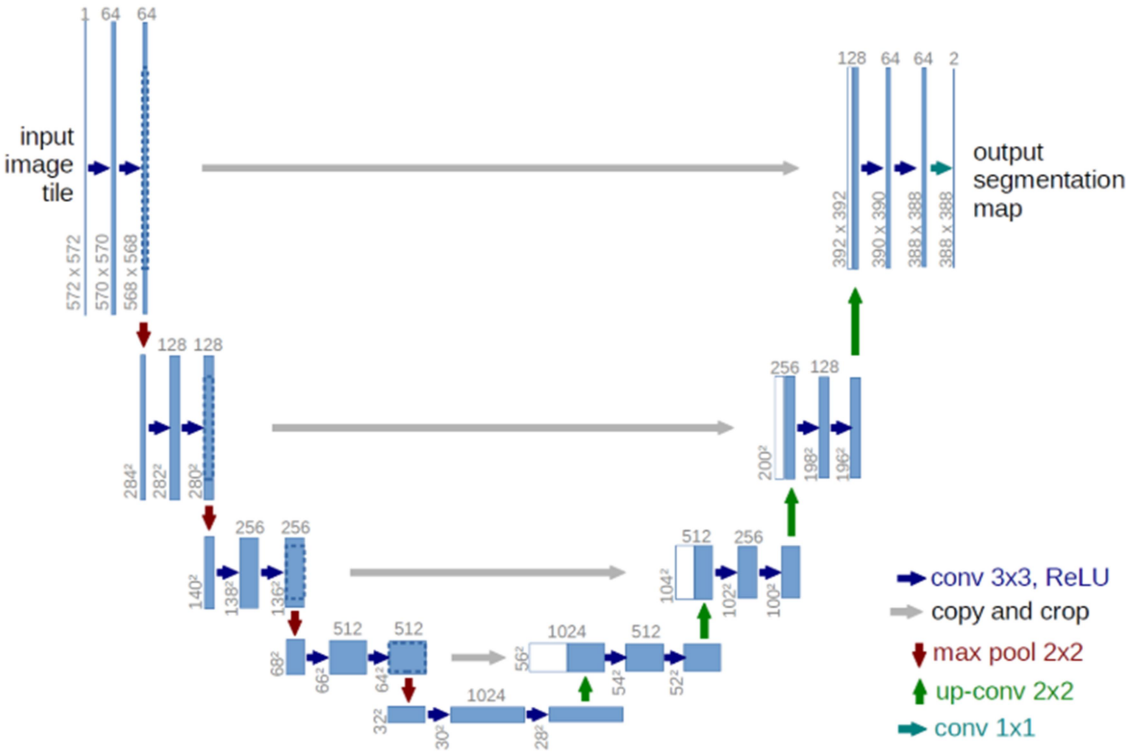


Рисунок — 2.3 Архітектура U-net [12]

Блок розкодування також складається з двох послідовних шарів згортки, проте за ними йде шар розгортки (позначена зеленим кольором), який збільшує розмірність зображення в два рази. Після проходження останнього блоку розкодування вихідне зображення набуває такої ж самої розмірності, як і на вході. Останнім є шар згортки з вікном 1 піксель на 1 піксель, з функцією активації Sigmoid, який створює вихідну маску зображення. Всього в архітектурі, яку запропонував Olaf Ronneberger було використано 23 згорткові шари.

Для відновлення дрібнозернистої інформації використовуються з’єднання між симетричними блоками кодувальника та декодувальника (на зобраежнні позначені сірою стрілкою).

2.4 Модифікація U-net Residual блоками

Для покращення результатів навчання потрібно збільшувати кількість шарів. Але це в свою чергу приводить до погіршення результатів. Коли більш глибока мережа починає згортатися, виникає проблема: зі збільшенням глибини мережі її точність спершу збільшується, а потім швидко погіршується.

Причиною є те, що на етапі оновлення вагів мережі помилка затухає.

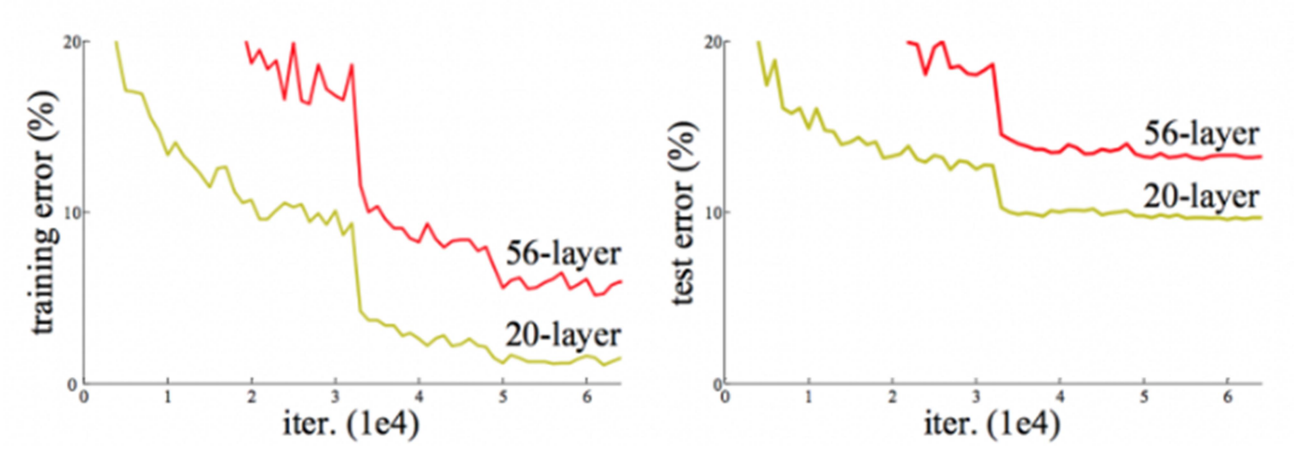


Рисунок — 2.4 Помилка на етапі навчання (ліворуч) та тесту (праворуч) на датасеті на CIFAR-10 з 20-шаровими та 56-шаровими “простими” згортковими мережами. [13]

Для вирішення цієї проблеми, команда з компанії Майкрософт розробила підхід глибокого залишкового навчання (анг. deep residual learning) , що лягло в основу мереді ResNet [14], а також подальших мереж. В базовій структурі нової моделі немає нічого нового: це шари, що йдуть послідовно один за одним в деякому випадку з нормалізацією по міні-бачам. Різниця в тому, що в кінцевому блоці шар з нейронів можна обійти, тобто є спеціальний зв’язок між виходом попереднього шару $x^{(k)}$ та наступним $x^{(k+1)}$, що йде напряму, не проходячи через обчислювальний шар (рисунок 2.6)

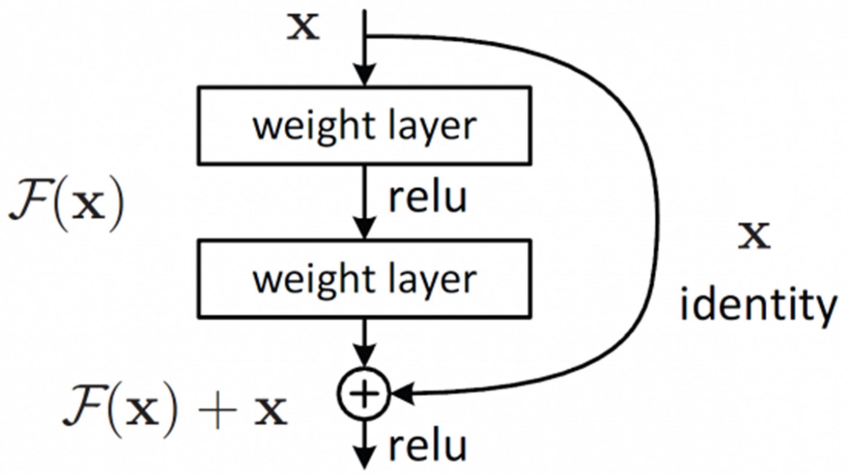


Рисунок — 2.5 Залишковий блок ResNet [14]

Два вихідні вектори, один з яких обхідний, об'єднуються шляхом додавання.

Залишковий блок виражається функцією 2.1:

$$y^{(k)} = F(x^{(k)}) + x^{(k)} \tag{2.1}$$

Де $x^{(k)}$ – вхідний вектор шару k , $F(x)$ – функція, яку обчислює шар нейронів, $y^{(k)}$ – вихід залишкового блоку, який потім стане входом наступного шару $x^{(k+1)}$.

Обхідне з'єднання часто називають з'єднанням швидкого доступу (анг. shortcut connections), воно може пропускати один або декілька шарів.

2.5 Функції помилки та регуляризації

Більшість задача машинного навчання зводиться до задач оптимізації. Функції активації нейронної мережі є важливим компонентом глибокої нейронної мережі. Функції активації визначають вихід моделі навчання, її

точність, а також обчислювальну ефективність навчання моделі. Від правильного вибору функції активації модель може значно прокращити свою ефективіність, при невдалому - перестати навчатися взагалі. Функції активації також мають великий вплив на здатність нейронної мережі до зближення і швидкість збіжності, або в деяких випадках, функції активації можуть перешкоджати зближенню нейронних мереж [15].

Функції активації є математичними рівняннями, які визначають вихід нейронної мережі. 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 три типи функцій активацій:

1. бінарна функція
2. лінійна функція активації
3. нелінійна функція активації.

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.

Регуляризація в нерйонних мережах важлива, оскільки мережа, в якої велика кількість параметрів добре “запам’ятовує” значення з тренувальної вибірки і в результаті недостатньо добре робить передбачення потрібних значень в нових точках. Сучасні нейронні мережі — це моделі з великою

кількістю параметрів, навіть не сама складна архітектура може мати мільйони вагів.

Існує декілька ідей регуляризації. Спробуємо від кожної змінної, яку ми оптимізовуємо приймати не занадто великі значення. Це можна зробити додавши до цільових функції регуляризатори в будь-якому зручному вигляді. Зазвичай використовують два види регуляризаторів:

L_2 регуляризатор, сума квадратів вагів : $\lambda \sum_{\omega} \omega^2$

L_1 регуляризатор, сума модулів вагів : $\lambda \sum_{\omega} |\omega|$

В теорії нейронних мереж це називається скороченням вагів тому, що дійсно приводить до зменшення їх абсолютних значень.

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

— kernal_regularizer – на матрицю вагів;

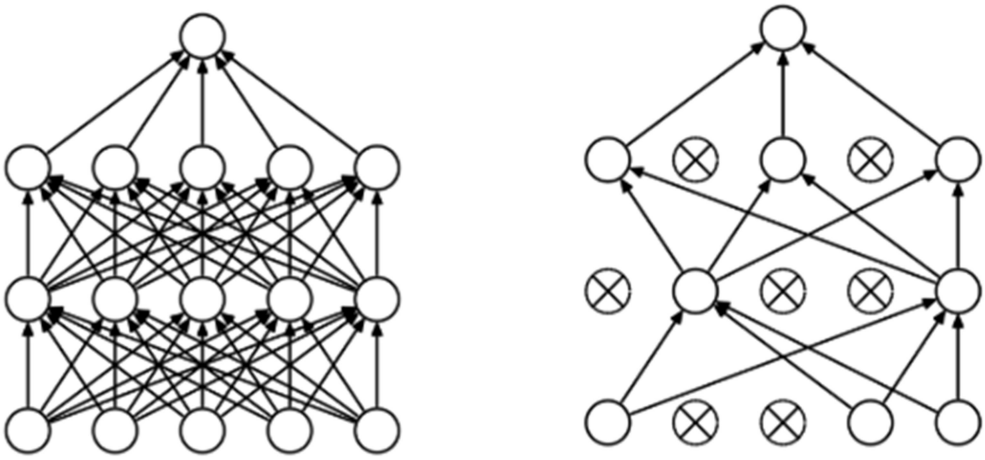
— bias_regularizer – на вектор вільних членів;

— activity_regularization – на вектор виходів.

Інша ідея регуляризації полягає в створенні валідаційного набору даних, і будемо навчати мережу на тренувальній множині, а обчислювати похибку на валідаційній. Це спирає ться на те, що помилка на валідаційній множині буде добре оцінювати помилку і на нових точках (тестовому датасеті), оскільки вона взята з даних тієї ж природи, але не тих, на яких мережа навчалася. Також, це дає можливість зупинити процес навчання не тоді, коли мережа дійде до локального оптимума для тренувальної вибірки, а тоді, коли розпочне погіршуватися помилка на валідаційній множині. В теорії нерйонних мереж це називаєтсья ранньою запинкуою (анг, early stopping).

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

буде нейрон використовуватися. Якщо ні, то вихід нерйона стає рівним 0. Це приводить до того, що нейрон фактично випадає з графу обчислення і ні пряме обчислення, ні зворотнє поширення градієнту не відбувається. На рисунку 2.7.а зображена звачайна нейронна мережа з двома прихованими гарами. На рисунку 2.7.б та ж сама мережа з двома прихованими шарами та застосованою технікою dropout. Закреслені нейроні були вимкнуті з мережі.



а б

Рисунок — 2.6 Dropout нейронної мережі з двома прихованими шарами [16]

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, що є корисним для еволюції. Важливо не зібрати хорошу комбінацію генів, важливо зібрати стабільну комбінацію генів. А цього простіше досягти, якщо примушувати працювати гени наодинці, не розраховуючи на сусідні нейрони.

2.6 Глибокі мережі

Тоерема Хорника [18] стверджує, що будь-яку неперервну функцію можна скільки завгодно наблизити нейронною мережею з один прихованим шаром. Проте глибокі архітектури часто можуть виконати теж саме, наблизити ті ж функції більш ефективно ніж неглибокі. Інша сторона глибоких мереж, це те, що глибока мереже створює також розподілене представлення.

Може виникнути питання: чому революція нейронних мереж відбулася після 2000-х років? На це питання є дві відповіді. Перша – математична. Справа в тому, що глибокі мережі можна навчати тим же методом градієнтного спуску, але в базовому варіанті, без додаткових хитрощів працювати це не буде. Уявімо, що ви почали навчати глибоку мережу алгоритмом зворотнього поширення помилки. Останній, найближчий до виходів шар нейронів навчатиметься доволі швидко. Але потім виявиться, що більшість нейрнонів останнього рівня на всіх тестових прикладах дають вихід рівний 0 або ж 1. Якщо в них класична сигмоїдальна функція активації, то це значить, що похідна від цієї функції рівна 0 з кожної сторони. Але на цю похідну ми маємо перемножити всі градієнти в алгоритмі зворотнього поширення помилки.

Таким чином, виходить, що навчений шар нейронів блокує поширення градієнту далі по графу обчилсення, і більш попередні шари нейронної мережі в результаті навчаються дуже повільно, фактично не навчаються взагалі. Ця проблема називаєься проблемою затухаючого градієнту (анг. vanishing gradients). А в рекурентних мережах присутні інша проблема — градієнти можуть “вибухати”.

На початку 2000-х років з’явилися перші прототипи глибоких мереж. Проблема затухаючого градієнту вирішувалася до навчанням нейронної мережі рівень за рівнем за допомогою особливого випадку ненаправленої графічної модель, так званої обмеженої машини Больцмана (restricted Boltzmann machine, RBM). Градієнтний спуск добре знаходить локальний мінімум функції, а різні

оптимізовані методи здатні знайти кращий локальний мінімум. Але все рівно градієнтний спуск по своїй суті “локальний”, він робить лиш невеликі кроки від початкової, і результат сильно залежить від початкової ініціалізації. Тому попереднє навчання досить ефективне, оскільки моделі вже трішки розуміють в структуру пропонованих даних, початкові значення вагів вже не випадкові, а достатньо розумні [17].

Також, з 2000-чних років покращилися підходи до навчання глибоких мереж. З’явилися важливі інструменти регуляризації та оптимізації нейронних мереж.

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 природньо, 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 ініціалізації вагів є до навчання без вчителя (анг. unsupervised pretraining). 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 (анг. transfer learning). Підхід полягає у використанні попередньо навченої 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 донавчати на існуючих даних. В результаті до навчання отримуємо модель, яку потім можна до навчати на розмічених даних. Модель, навчена таким чином в кінцевому рахунку стабільно сходиться до найкращих існуючих рішеннях ніж при випадковій ініціалізації.

До навчання — це складна та дорога операція. Фактично виходить, що потрібно навчити робити дві різні процедури навчання, налаштовувати їх, причому до навчання швидше за все буде більш складною процедурою навчання.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

В роботі було створену систему для сегментації рослин перію на зображеннях. Для створення системи була використана мова програмування Python, оскільки для неї присутня велика кількість бібліотек для роботи з нейронними мережа та зображеннями.

3.1 Розроблена мережа

Для сегментації було реалізовану мережу U-net. Перелік шарів мережі, розмірність виводу та кількість параметрів наведено в таблиця 3.1. Всього мережа містить 31 032 837 параметрів. Входом мережі є зображення у вигляді тривимірного масиву. Виходом є двохвимірна маска на зображення. Кожний елемент містить імовірність належності пікселя зображення до шуканого класу.

В ході дослідження при реалізації мережі згорткові шари було замінено на Residul блоки. Завдяки цьому глибина мережі збільшилася, що покращило точність сегментації. Всього в мережі було використано 175 шарів.

Таблиця 3.1. Розроблена мережа U-net.

Шар	Вихідна розмірність	Кількість параметрів
input_3 (InputLayer)	(None, 256, 256, 3)	0
conv2d_71 (Conv2D)	(None, 256, 256, 64)	1792
conv2d_72 (Conv2D)	(None, 256, 256, 64)	36928
max_pooling2d_9 (MaxPooling2D)	(None, 128, 128, 64)	0
conv2d_73 (Conv2D)	(None, 128, 128, 128)	73856
conv2d_74 (Conv2D)	(None, 128, 128, 128)	147584
max_pooling2d_10 (MaxPooling2D)	(None, 64, 64, 128)	0
conv2d_75 (Conv2D)	(None, 64, 64, 256)	195168
conv2d_76 (Conv2D)	(None, 64, 64, 256)	590080

Продовження таблиці 3.1

max_pooling2d_11 (MaxPooling2D)	(None, 32, 32, 256)	0
conv2d_77 (Conv2D)	(None, 32, 32, 512)	1180160
conv2d_78 (Conv2D)	(None, 32, 32, 512)	2359808
dropout_11 (Dropout)	(None, 32, 32, 512)	0
max_pooling2d_12 (MaxPooling2D)	(None, 16, 16, 512)	0
conv2d_79 (Conv2D)	(None, 16, 16, 1024)	4719616
conv2d_80 (Conv2D)	(None, 16, 16, 1024)	9438208
dropout_12 (Dropout)	(None, 16, 16, 1024)	0
up_sampling2d_5 (UpSampling2D)	(None, 32, 32, 1024)	0
conv2d_81 (Conv2D)	(None, 32, 32, 512)	2097664
concatenate_9 (Concatenate)	(None, 32, 32, 1024)	0
conv2d_82 (Conv2D)	(None, 32, 32, 512)	4719104
conv2d_83 (Conv2D)	(None, 32, 32, 512)	2359808
up_sampling2d_6 (UpSampling2D)	(None, 64, 64, 512)	0
conv2d_84 (Conv2D)	(None, 64, 64, 256)	524544
concatenate_10 (Concatenate)	(None, 64, 64, 512)	0
conv2d_85 (Conv2D)	(None, 64, 64, 256)	1179904
conv2d_86 (Conv2D)	(None, 64, 64, 256)	590080
up_sampling2d_7 (UpSampling2D)	(None, 128, 128, 256)	0
conv2d_87 (Conv2D)	(None, 128, 128, 128)	131200
concatenate_11 (Concatenate)	(None, 128, 128, 256)	0
conv2d_88 (Conv2D)	(None, 128, 128, 128)	295040
conv2d_89 (Conv2D)	(None, 128, 128, 128)	147584
up_sampling2d_8 (UpSampling2D)	(None, 256, 256, 128)	0
conv2d_90 (Conv2D)	(None, 256, 256, 64)	32832
concatenate_12 (Concatenate)	(None, 256, 256, 128)	0
conv2d_91 (Conv2D)	(None, 256, 256, 64)	73792
conv2d_92 (Conv2D)	(None, 256, 256, 64)	36928
conv2d_93 (Conv2D)	(None, 256, 256, 2)	1154
conv2d_94 (Conv2D)	(None, 256, 256, 1)	3

3.2 Структура системи

В ході роботи було реалізовано дві системи: для навчання мережі (рисунок 3.1) та для її використання (рисунок 3.2).

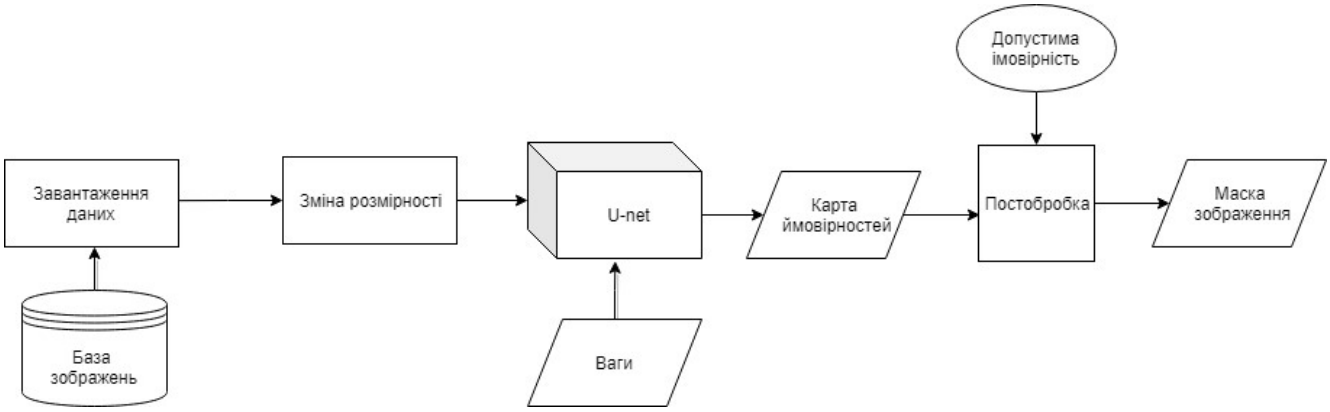


Рисунок — 3.1 Схема навчання моделі

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 модифікація 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. Згенеровна маска порівнюється з наявною маскою, проводиться обчислення похибки на основі якої мережа проводить корекцію вагів.

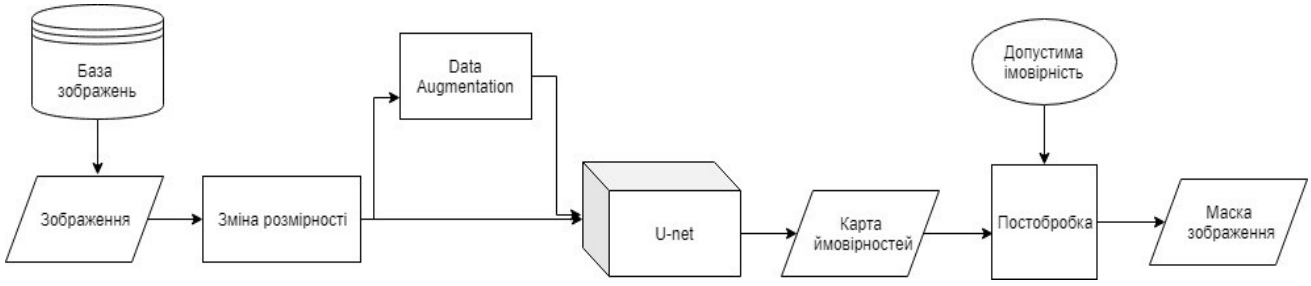


Рисунок — 3.2 Схема роботи системи в режимі передбачень

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.

3.3 Аугментація зображення

Для отримання хороших результатів тренування для нейронної мережі необхідна велика кількість навчальних матеріалів. Проте кількості наявних даних було недостатньо для навчання мережі. Для вирішення проблемита та водночас покращення результатів було штучно збільшено кількість даних, шляхом модифікації наявних зображень (рисунок 2.3). Цей процес прийнято називати аугментацію даних. Для зображення було застосовано комбінацію методів обробки зображення:

— обертання;

— зсув;

— обтинання;

— масштабування;

— відображення відносно осей.

Комбінація методів та їх ступінь інтенсивність в процесі обробки обирається 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. Приклад аугментації зображено на рисунку 3.3.

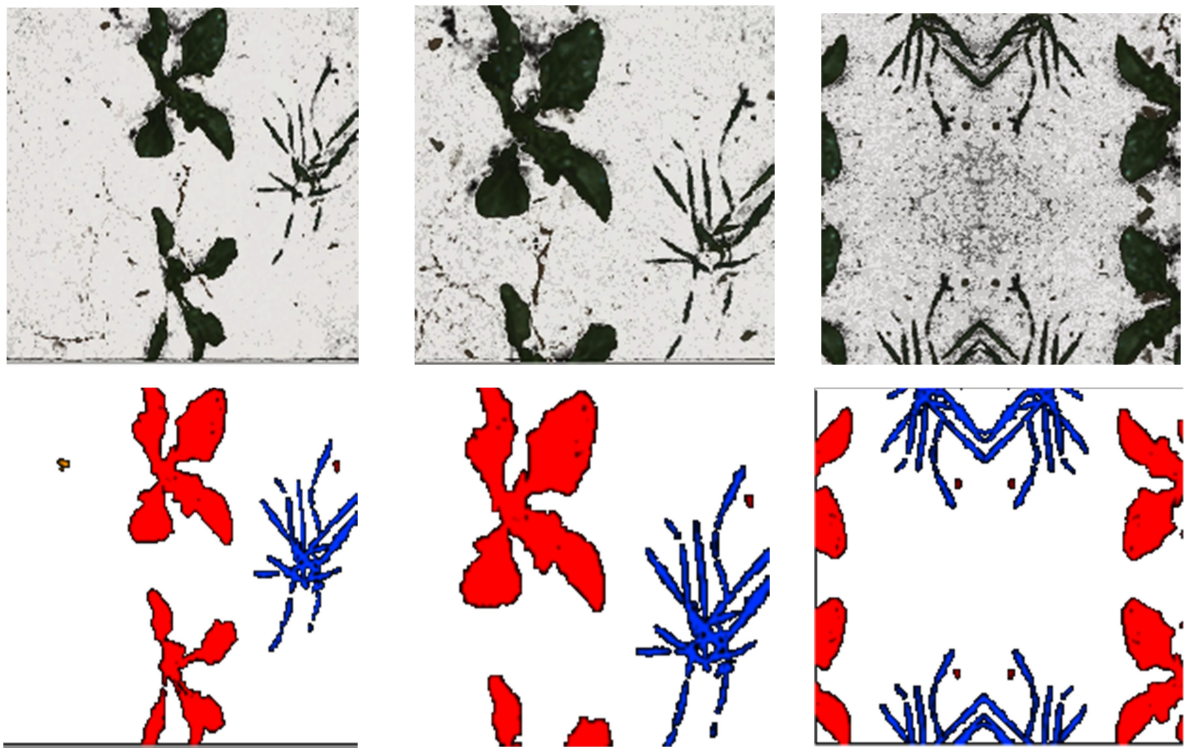


Рисунок — 3.3 Оригінальне зображення та його маска (зліва), аугментовані шляхом масштабування (по середині), аугментовані дані шляхом відображення відносно двох осей (справа)

Для виконання аугментації було використано пакети skimage.transform та cv2, які підтримують всі необхідні операції.

3.4 Зміна розмірності

Для коректної роботи на вхід моделі повинні передаватися зображення одного 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 256 на 256 пікселів. Для зміни розмірів використовується метод transform.resize пакету skimage.

3.5 Обгортка мережі

Процес розробки, навчання та тестування проводився в середовищі Jupyter Notebook, проте для зручного користування мережею його функціональності недостатньо. Було прийнято ріщення створити клієнт-серверну систему для роботи з нейронною мережею (рисунок 3.4).

Було розроблено один HTTP API метод за допомогою бібіліотеки Flask. При його виклику із зображенням, як параметр, відбувається запуск системи по сегментуванню. Зображення перетворюється в масив та передається в мережу. Після створення мережеою бінарної маски зі знайденими бур'янами, вона відправляється у відповідь на запит користувачу.

Рисунок — 3.4 Клієнт серверна система по сегментації

При першому зверненні до API відбувається завантаження моделі в пам'ять та запуск системи по сегментації.

Для роботи з API було розроблено клієнтську частину на мові програмування C# та платформі WPF. Його призначенням є вибір зображення, відправка на сервер запиту на сегментуання та відображення зображення

результату.

3.6 Засоби розробки

В процесі навчання глибоких нейронних мереж відбувається велика кількість матричних обчислень, що вимагає комп'ютера з потужною відео картою. Тому для виконання даного дослідження було прийнято рішення використовувати платформу Google Colaboratory [22].

Google Colaboratory — це хмарний сервіс для досліджень в сфері машинного навчання та глибокого навчання. Colaboratory надає Jupyter Notebook запущений на віддаленому комп'ютері з підключеною відеокартою. На платформі вже встановлені всі необхідні бібліотеки та фреймворки для навчання нейронних мереж.

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

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

Keras – це бібліотека з відкритим кодом, написана на Python [8]. Бібліоткеа надає високорівневий інтерфейс для роботи з бібліотеками TensorFlow та Theano. Бібліотека містить численні реалізації широко вживаних будівельних блоків нейронних мереж, таких як шари, цільові та передавальні функції, оптимізатори, та безліч інструментів для спрощення роботи із зображеннями та текстом.

Для зручного використання та натренованої мережі за межами Google Colaboratory було розроблено серверний веб додаток на мікрофреймворці Flask[9].

Flask називається мікрофреймворком, оскільки він не вимагає спеціальних засобів чи бібліотек. У ньому відсутній рівень абстракції для роботи з базою

даних, перевірки форм або інші компоненти, які надають широковживані функції за допомогою сторонніх бібліотек. Однак, Flask має підтримку розширень, які забезпечують додаткові властивості таким чином, наче вони були доступні у Flask із самого початку. Існують розширення для встановлення об'єктно-реляційних зв'язків, перевірки форм, контролю процесу завантаження, підтримки різноманітних відкритих технологій аутентифікації та декількох поширених засобів для фреймворку. Розширення оновлюються частіше аніж базовий код

4 ОБЧИСЛЮВАЛЬНІ ЕКСПЕРЕМЕНТИ

Для побудови нейронної мережі використовуються значна кількість компонентів. Від правильного їх вибору залежить якість та швидкість роботи мережі. Нейронні мережі часто називають “чорним ящиком” оскільки важко передбачити якість роботи мережі при використанні певних компонентів, їх можна визначити тільки емпіричним шляхом.

4.1 Дослідження обробки зображення в глибоких шарах мережі

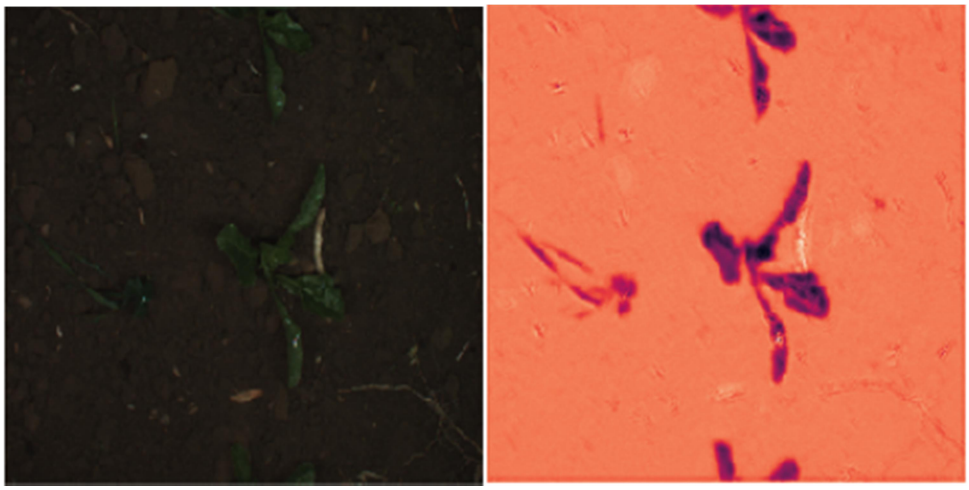
В процесі навчання може здатися, корисно розуміти, як зображення обробляється проходячи через певні фільтри загорткової нейронної мережі. Для цього було розроблено функцію, що приймає на вхід порядковий номер шару та номер фільтру та повертає модель, останнім шаром якої є вказаний в якості аргументу шар.

Зробивши передбачення на основі отриманої моделі можна отримати уявлення про обрубку, що відбувається всередині мережі. На зображеннях показано вхідне зображення та зображення першого шару результуючих тензорів після 16, 33, 76, 140 та останнього 175-го шару. Можна побачити, як з вхідного зображення (рисунок 4.1.а) на етапі роботи кодувальника (рисунок 4.1 б-г) поступово вилучалися ознаки бур'яну при цьому ширина та висота зображення поступово зменшувалися при наближенні до найглибших шарів.

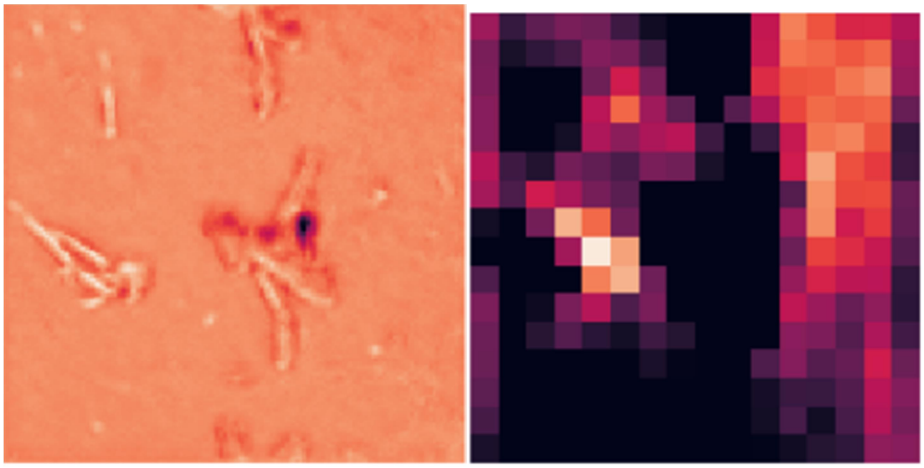
На етапі роботи декодувальника (рисунок 4.2 а-б) ділянки з бур'яном виділялися та зображеннях повертається до вхідного розміру. В процесі роботи перегляду зображень можна зрозуміти, як саме мережа сегментує зображення.

Спершу виділяється рослинність на зображення, оскільки її колір відрізняється від кольору фону, далі активуються ділянки зображення, що належать саме бур’яну.

Можна припусти, що активація пікселів бур’яну відбувається за рахунок специфічної форми рослини та кольору.

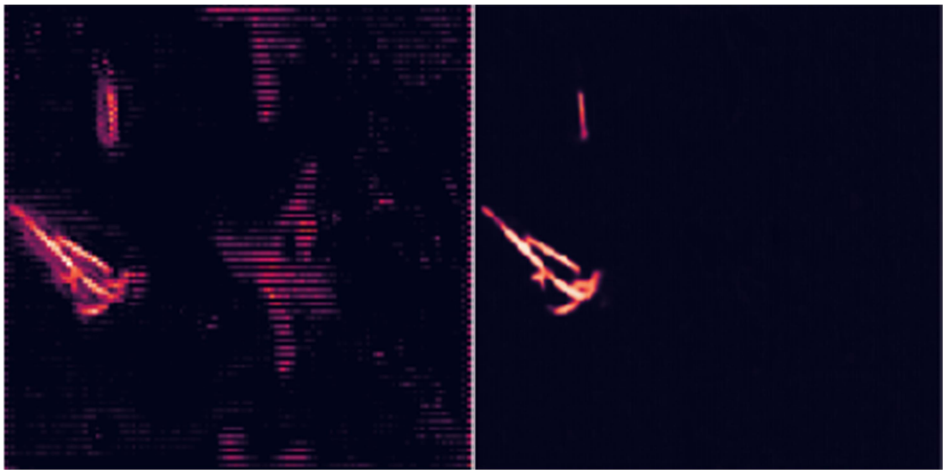


а б



в г

Рисунок — 4.1 Етапи обробки зображення на етапі кодувальника U-net



а б

Рисунок — 4.2 Етапи обробки зображення на етапі декодувальника U-net

4.2 Вибір функції похибки

Оцінюючи стандартну модель машинного навчання, ми зазвичай класифікуємо наші прогнози на чотири категорії: істинні позитиви, помилкові спрацьовування, істинні негативи та помилкові негативи. Однак для задачі прогнозування сегментації зображення не відразу зрозуміло, що вважається "справжнім позитивним" і, загалом, як ми можемо оцінити наші прогнози. Тож для навчання мережі по сегмнетації зображень потрібно обрати функцію похибки. Загально прийнято використовувати функцію похибки накладання (анг. Intersection over Union , IoU), яка визначається, як функція 4.1:

$$IoU = \frac{target \cap prediction}{target \cup prediction} \tag{4.1}$$

Метрика IoU вимірює загальну кількість пікселів між передбаченою маскою та вірною маскою, поділеними на загальну кількість пікселів, наявних у обох масках.

Використання цієї функції втрати не дало очікуваного результату. Було прийнято рішення використовувати стандартну функцію втрати categorical_accuracy, яка зазвичай використовується для класифікаційних мереж. На рисунку 4.3 показано графік навчання мережі.

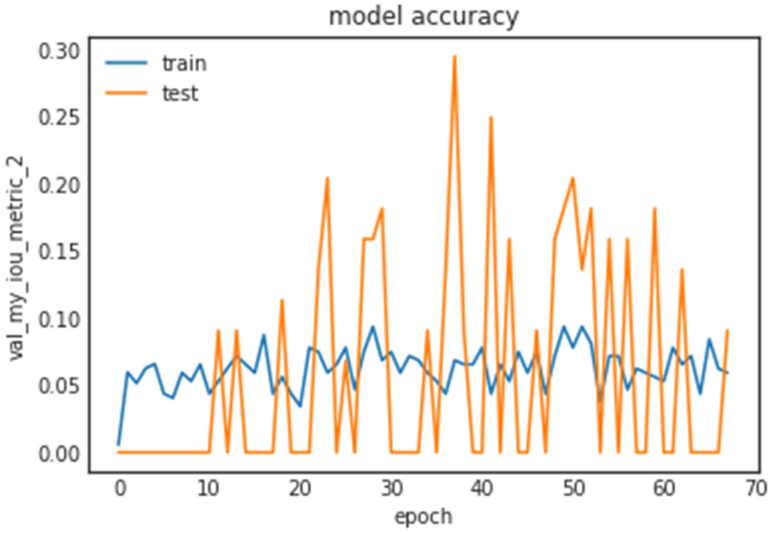


Рисунок — 4.3 Графік навчання мережі

Функція categorical_accuracy чудово спрацювала, та дала точність передбачення 97.3% на тестувальній вибірці.

4.3 Навчання нейронної мережі з використанням графічних процесорів

Центральний процесор (ЦП), є центром будь-якого обчислювального пристрою, що виконує інструкції програми, обробляє ввід та вивід даних. Перший процесор, блок 4004, був розроблений компанією Intel лише 50 років тому в 1970-х роках. Більшість ЦП 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.

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

Графічний процесор (GPU), з іншого боку, має менші розміри, але багато інших логічних ядер (арифметичні логічні одиниці або ALU, керуючі блоки і кеш-пам'ять), основна конструкція яких полягає в обробці набору більш простих та ідентичних обчислень в паралельному режимі. Наприклад, обробку трьохвимірної графіки, візуалізацій.

Можна сказати, що процесори найкраще підходять для обробки поодиноких, більш складних обчислень послідовно, тоді як графічні процесори краще обробляти декілька, але простіших обчислень паралельно. Використання GPU дозволяє навчати нейронні мережі в кілька разів швидше, ніж на CPU. Також варто зазначити, що бібліотека є цілком безкоштовною та її можна завантажити на сайті розробника. Наприклад, для мережі, яка розпізнає об'єкти з набором даних CIFAR-10, застосування GPU дозволило прискорити навчання більше ніж у 7 разів, порівняно з CPU.

Для навчання мережі U-net спершу був використаний CPU Intel Core i5 6-го покоління. Навчання мережі проводилося успішно, проте після п’ятої години навчання було прийнято рішення перервати процес навчання, не чекаючи закінчення, оскільки час навчання перевершив очікуваний. Для продовження навчання було використано хмарне середовище для навчання нейронних мереж — Google Collaborator, на якому в безкоштовному режимі було отримано доступ до GPU GeForce RTX 2080 Ti. З вигортанням нового апаратного забезпечення навчання мережі пройшло за 2 години 43 хвилини.

5 РОБОТА КОРИСТУВАЧА З СИСТЕМОЮ

5.1 Системні вимоги

Для забезпечення коректної та безвідмовної роботи системи по сегментації бур'янів на зображеннях персональний комп'ютер повинен мати процесор не гірше, ніж Intel ® Pentium ® або з тактовою частотою не менше 1,8 Грц для користувачів процесорів від фірми AMD. Також комп'ютеру користувача повинно бути доступно не менше 4 Gb оперативної пам'яті та графічне ядро не гріше, ніж Nvidia 210 GPU. На комп'ютері повинна бути встановлена система Windows 10, Windows 7. На комп'ютері має бути встановлений Python 3.6, Також на жорсткому диску повинно бути не менше, ніж 1 Gb вільного місця для завантаження та запуску системи.

Якщо користувач хоче запустити навчання мережі, то чим більш потужний комп'ютер, тим швидше пройде навчання мережі. Варто зазначити, що в процесі навчання можливо доведеться зменшити кількість епох та кількість одночасно оброблюваних файлів, оскільки це залежить від об'єму відео пам'яті.

5.2 Послідовність запуску системи

При першому запуску необхідно встановити всі Python залежності виконавши CMD команду "pip install -r requirements.txt" . Після чого виконати команду "python server.py" завдяки чому буде запущений локальний сервер.

Після цього можна запустити клієнтську програму та сегментувати зображення.

5.3 Тестові задачі сегментації

Графічний інтерфейс користувача складається з двох кнопок “Обрати збраження” та “Відправити”. Першого кнопкою користувач обирає необроблене зображення ділянки поля. Зображення з’являється у вікні. Після чого він може натиснути наступну кнопку, яка відправить обране зображення на локальний сервер, де фото буде опрацьоване. В результаті буде створено маску для зображення на якій будуть виділені пікселі зі знайденеми бур’янами. Результуюча маска комбінується із вхідними зображенням та відправляється на клієнтську частину де відображається користувачу.

Для демонтрації роботи програми було обрано 3 зображення з наявного датасету. На рисунку 5.1 показано приклад сегментації ділянкти поля.

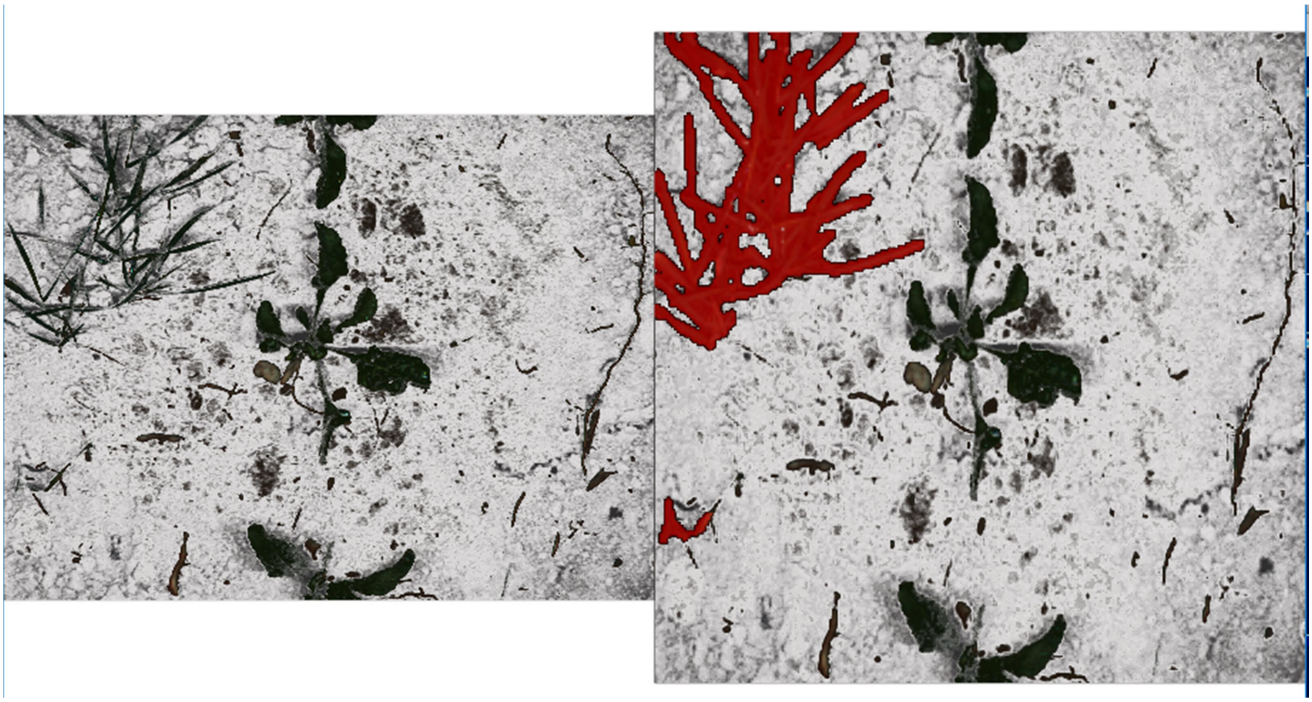


Рисунок — 5.1 Результат сегментації

На рисунку 5.2 показана проблемна ділянка з накладанням рослини цукрового буряку на рослину бур’яну. Нейронна мереже успішно просегментувала зображення.

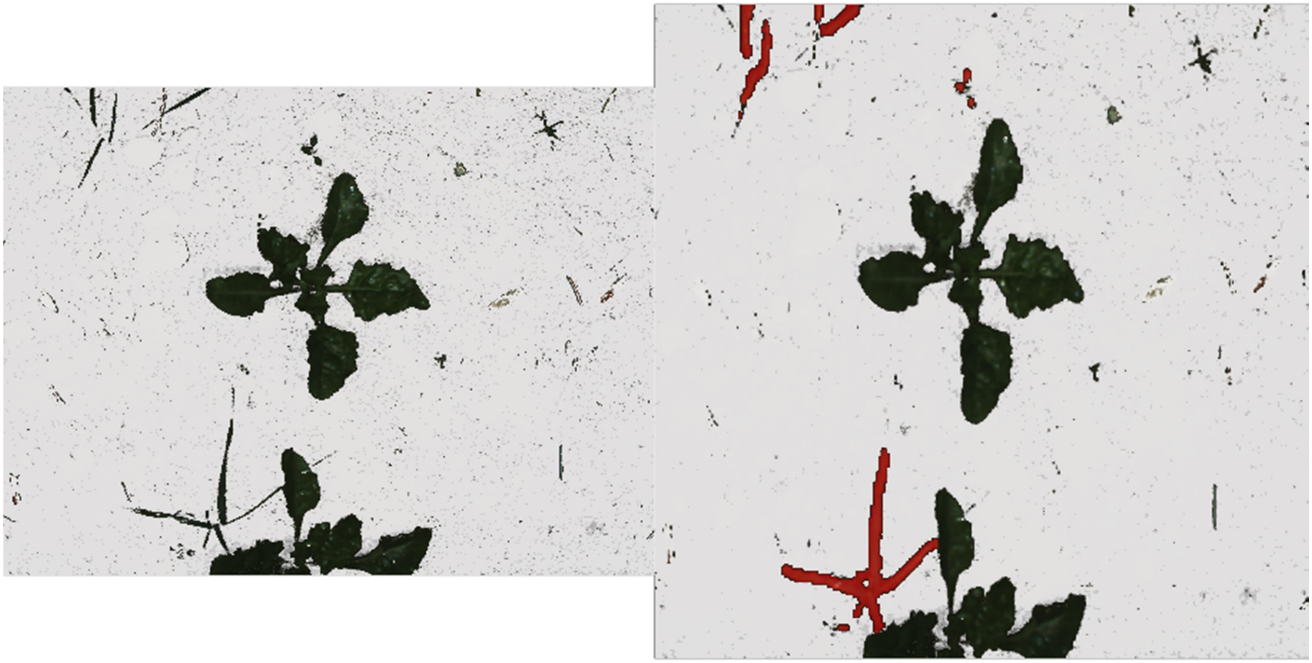


Рисунок — 5.2 Результат сегментації

На рисунку 5.3 показано приклад невдалої сегментації. У правій верхній частини рослина бур’яну була не повністю відсегментована.

Рисунок — 5.3 Результат сегментації

ВИСНОВКИ

1. Робота присвячена вирішенню проблеми машинного зору сільхозроботів. В роботі розв’язується задача сегментації бур’янів на зображеннях з бортової камери.
2. Проведено аналіз підходів та методів розв’язання задачі сегментації. Обгрунтовано використання згорткових нейронних мереж.
3. На основі проведених досліджень визначено архітектуру, склад та компоненти мережі згорткової нейронної мережі для задачі сегментації.
4. За умови недостатності навчальних прикладів у відкритому наборі даних “Sugar Beets 2016” реалізовано аугментацію зображень.
5. На основі проведених досліджень обрано засоби програмної реалізації та розроблено структуру системи.
6. Експерементально доведено ефективність програмного рішення для вирішення задачі по сегментації бур’янів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Robert M.HaralickLinda G.Shapiro; Image segmentation techniques [Електронний ресурс] — Режим доступу: https://www.sciencedirect.com/science/ article/ pii/S0734189X85901537

2. Dilpreet Kaur, Yadwinder Kaur; Various Image Segmentation.Techniques: A Review// IJCSMC, Vol. 3, Issue. 5, May 2014, pg.809 – 814. ISSN 2320–088X

3. Обзор алгоритмов сегментации. [Електронний ресурс] — Режим доступу: https://habr.com/ru/company/intel/blog/266347/

4. Su Hnin Hlaing, Aung Soe Khaing; Weed and crop segmentation and classification using area thresholding //International Journal of Research in Engineering and Technology eISSN: 2319-1163 | pISSN: 2321-7308

5. Sebastian Haug, Andreas Michaels; Plant classification system for crop /weed discrimination without segmentation// Published in IEEE Winter Conference 2014. DOI:10.1109/WACV.2014.6835733

6. Philipp Lottes, Cyrill Stachniss; Semi-Supervised Online Visual Crop and Weed Classification in Precision Farming Exploiting Plant Arrangement. [Електронний ресурс] — Режим доступу: http://flourish-project.eu/fileadmin/user_upload/publications/lottes17iros.pdf

7. Yao Wang ; Yisong Chen ; Peng Lu ; Heng Wang Sobel Heuristic Kernel for Aerial Semantic Segmentation// 2018 25th IEEE International Conference on Image Processing (ICIP), Electronic ISSN: 2381-8549

8. The 2016 Sugar Beets Dataset Recorded at Campus Klein Altendorf in Bonn,Germany [Електронний ресурс] — Режим доступу: http://www.ipb.unibonn.de/data /sugabeets2016/

9. Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel: Backpropagation Applied to Handwritten Zip Code Recognition, Neural Computation, 1(4):541-551, Winter 1989.

10. Le Hou ; Dimitris Samaras ; Tahsin M. Kurc ; Patch-Based Convolutional Neural Network for Whole Slide Tissue Image Classification //2016 IEEE Conference on Computer Vision and Pattern Recognition, ISSN: 1063-6919

11. Fully Convolutional Networks for Semantic Segmentation [Електронний ресурс] — Режим доступу: https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf

12. U-Net: Convolutional Networks for Biomedical Image Segmentation [Електронний ресурс] — Режим доступу: https://lmb.informatik.uni-freiburg.de/people/ ronneber/u-net/

13. Improve Deep Learning Models performance & deep network tuning [Електронний ресурс] — Режим доступу: https://medium.com/@jonathan_hui/improve-deep-learning-models-performance-network-tuning-part-6-29bf90df6d2d

14. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun; Deep Residual Learning for Image Recognition [Електронний ресурс] — Режим доступу: https://arxiv.org/abs/1512.03385

15. Types of Neural Network Activation Functions: How to Choose. [Електронний ресурс] — Режим доступу: https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/

16. Is the Braess Paradox related to Dropout in Neural Nets? [Електронний ресурс] — Режим доступу: https://hackernoon.com/is-the-braess-paradox-related-to-dropout-in-neural-nets-270ecb97cdeb

17. С. Николенко, А. Кадурин, Е. Архангельская;Глубокое обучение погружение в мир нейронных сетей. ББК 32.973.236

18. Aitkenhead-Peterson et al. 2012 [Електронний ресурс] — Режим доступу: https://www.academia.edu/1537285/Aitkenhead-Peterson_et_al._2012

19. Homik K.y Stinchcombe M., White H. Multilayer Feedforward Networks are Universal Approximators / / Neural Networks, 1989, vol. 2, no. 5. — P. 359-366.

20. Plant Leaf Recognition using Gabor Filter [Електронний ресурс] — Режим доступу: https://research.ijcaonline.org/volume56/number10/pxc3883000.pdf

21. Camera Sensor Arrangement for Crop/Weed Detection Accuracy in Agronomic Images [Електронний ресурс] — Режим доступу: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3673087/

22. Introducing Colaboratory [Електронний ресурс] — Режим доступу: https://colab.research.google.com

23. Keras: The Python Deep Learning library [Електронний ресурс] — Режим доступу: https://colab.research.google.com

24. Flask is a microframework for Python based on Werkzeug [Електронний ресурс] — Режим доступу: http://flask.pocoo.org/

ДОДАТОК А

Сегментація зображень на основі згорткових нейронних мереж

Специфікація

УКР.НТУУ”КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ТР5295_19Б

Аркушів 1

Київ 2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ТР5295_19Б	Записка.docx	Пояснювальна записка
Компоненти		
УКР.НТУУ”КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ТР5295_19Б 12-1	Training.ipynb	Основні компоненти навчання мережі
УКР.НТУУ”КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ТР5295_19Б 12-2	Server.py	Основні компоненти серверної часина
УКР.НТУУ”КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ТР5295_19Б 12-3	MainWindow.xaml.cs	Основні компоненти Клієнтської системи

ДОДАТОК Б

Сегментація зображень на основі згорткових нейронних мереж

Текст програми

УКР.НТУУ”КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ТР5295_19Б

Аркушів 7

Київ 2019

Код навчання мережі з файлу Training.ipynb

```
def get_img_id(img_path):
    return img_path[:len(img_path)-4]

#img_ids = list(map(get_img_id, list(train_masks_df.img.values)))

def load_image_disk(img_id, folder=TRAIN_PATH):
    img = imageio.imread(os.path.join(folder, img_id + ".png"))
    return img

def load_mask_disk(img_id, folder=TRAIN_MASKS_PATH):
    img = imageio.imread(os.path.join(folder, img_id + ".png"))
    return img

# Helper functions to plot car, mask, masked_car
def plot_image(img_id):
    img = imageio.imread(os.path.join(TRAIN_PATH, img_id + ".png"))
    imgplot = plt.imshow(img)
    plt.axis('off')
    plt.show()

def get_raw_img(img_id):
    return mpimg.imread(os.path.join(RAW_PATH, img_id + ".png"))

def plot_image_from_array(img_array):
    imgplot = plt.imshow(img_array)
    plt.axis('off')
    plt.show()

def plot_mask(img_id):
    mask = mpimg.imread(os.path.join(TRAIN_MASKS_PATH, img_id + ".png"))
    imgplot = plt.imshow(mask)
    plt.axis('off')
    plt.show()

def plot_masked_image(img_id):
    img = imageio.imread(os.path.join(TRAIN_PATH, img_id + ".png"))
    mask = imageio.imread(os.path.join(TRAIN_MASKS_PATH, img_id +
".png"))
    mask = mask[:,:,0:3]
```

```
    mask[mask == 255] = 1
    masked_img = img * mask
    imgplot = plt.imshow(masked_img)
    plt.axis('off')
    plt.show()

def get_mask(img_id, num_channels=-1):
    if num_channels!=-1:
        mask = imageio.imread(os.path.join(TRAIN_MASKS_PATH, img_id +
".png"), flatten=True)
        mask[mask > 128] = 1
        if len(mask.shape) == 2:
            mask = mask.reshape(mask.shape[0], mask.shape[1], 1)
        return mask
    else:
        mask = imageio.imread(os.path.join(TRAIN_MASKS_PATH, img_id +
".png"))
        mask = mask[:,:,0:3]
        return mask

def resize_img(img, new_s = new_shape):
    return trans.resize(img, new_s)

def load_imgs(img_id, folder=TRAIN_PATH):
    img = imageio.imread(os.path.join(folder, img_id + ".png"))
    return img

def gray2rgb(img):
    img = np.squeeze(img)
    w, h = img.shape
    ret = np.empty((w, h, 3), dtype=np.uint8)
    ret[:, :, 0] = img
    ret[:, :, 1] = img
    ret[:, :, 2] = img
    return ret



def get_img_id(img_path):
    return img_path[:len(img_path)-4]
```

```
def load_image_disk(img_id, folder=TRAIN_PATH):
    img = imageio.imread(os.path.join(folder, img_id + ".png"))
    return img

def load_mask_disk(img_id, folder=TRAIN_MASKS_PATH):
    img = imageio.imread(os.path.join(folder, img_id + ".png"))


                           shift_limit=(-0.0625, 0.0625),
                           scale_limit=(-0.1, 0.1),
                           rotate_limit=(-45, 45), aspect_limit=(0, 0),
                           borderMode=cv2.BORDER_REFLECT_101, u=0.5):
    if np.random.random() < u:
        height, width, channel = image.shape

        angle = np.random.uniform(rotate_limit[0], rotate_limit[1])  #
degree
        scale = np.random.uniform(1 + scale_limit[0], 1 +
scale_limit[1])
        aspect = np.random.uniform(1 + aspect_limit[0], 1 +
aspect_limit[1])

        sx = scale * aspect / (aspect ** 0.5)
        sy = scale / (aspect ** 0.5)

        dx = round(np.random.uniform(shift_limit[0], shift_limit[1]) *
width)
        dy = round(np.random.uniform(shift_limit[0], shift_limit[1]) *
height)
        cc = np.math.cos(angle / 180 * np.math.pi) * sx
        ss = np.math.sin(angle / 180 * np.math.pi) * sy
        rotate_matrix = np.array([[cc, -ss], [ss, cc]])

        box0 = np.array([[0, 0], [width, 0], [width, height], [0,
height], ])
        box1 = box0 - np.array([width / 2, height / 2])
        box1 = np.dot(box1, rotate_matrix.T) + np.array([width / 2 + dx,
height / 2 + dy])

        box0 = box0.astype(np.float32)
        box1 = box1.astype(np.float32)
```

```
            mat = cv2.getPerspectiveTransform(box0, box1)
            image = cv2.warpPerspective(image, mat, (width, height),
        flags=cv2.INTER_LINEAR, borderMode=borderMode,
                                        borderValue=(0, 0, 0,))
            mask = cv2.warpPerspective(mask, mat, (width, height),
        flags=cv2.INTER_LINEAR, borderMode=borderMode,
                                       borderValue=(0, 0, 0,))
            if len(mask.shape) == 2:
                mask = np.expand_dims(mask, axis=2)

        return image, mask

    input_layer = Input(new_shape)
    output_layer = build_model(input_layer, 16,0.5)

    model1 = Model(input_layer, output_layer)
    metrics=[my_iou_metric])
    model1.compile(optimizer = Adam(lr = 1e-4), loss =
    'binary_crossentropy', metrics = ['accuracy'])

    t_model1_start = time.time()
    print(f'Start time utc {datetime.datetime.now()}')
    print(f'Start time utc {t_model1_start}')

    history = model1.fit_generator(
                          train_generator,
                          validation_data = valid_generator,
                          epochs = epochs,
                          steps_per_epoch = steps_per_epoch,
                          validation_steps = validation_steps,
                          callbacks = [early_stopping, model_checkpoint,
    reduce_lr],
                          verbose = 2)

    t_model1_end = time.time()
    print(f"Run time = {(t_model1_end-t_model1_start)/3600} hours")
```

Код сервереної частини з файлу server.py

```
import keras
from keras.preprocessing.image import img_to_array
from keras.applications import ResNet50
from keras.applications import imagenet_utils
```

```
from PIL import Image
import numpy as np
import tensorflow as tf
import matplotlib
from scipy import misc
import os


# initialise Flask application and Keras model
app = flask.Flask(__name__)
predictedImageName = 'predicted.png'


@app.route('/')
def hello_world():
    return 'Hello, World!'

@app.route('/lastPredicted')
def lastPredictedGet():
    img = misc.imread(predictedImageName)
    return send_file(predictedImageName,mimetype='image/jpeg')



@app.route('/predict', methods=["POST"])
def predict():

    if os.path.exists("name.png"):
        os.remove("name.png")
    # initialise the data dictionary that will be returned
    # from the view
    print("New request")
    data = {'success': False}

    if flask.request.method == "POST":
        if flask.request.files.get('image'):
            # read the image in PIL format
            imageFormRequest = flask.request.files['image'].read()
            row_img = Image.open(io.BytesIO(imageFormRequest))

            smallImage = resize_img(np.array(row_img))
            dataForModel =  prepare_image_for_model(smallImage)

            with graph.as_default():
                preds = model.predict(dataForModel)
                image_with_mask = prediction_to_image(preds,smallImage,0.20 )
                matplotlib.image.imsave(predictedImageName, image_with_mask)

                return send_file(predictedImageName,mimetype='image/jpeg')

    return abort(401)


if __name__=="__main__":
    print(('* loading Keras model and Flask starting server'))
    global model
    model = load_model()
    global graph
    graph = tf.get_default_graph()

    app.run(host='0.0.0.0')
```

Код клієнтської частини

```
private async  void Send_OnClick(object sender, RoutedEventArgs e)
        {

            using (FileStream fsSource = new FileStream(fileName,
                FileMode.Open, FileAccess.Read))
            {
                var imageStream = await
UploadAsync("http://localhost:5000/predict", fsSource);
                if (imageStream == null)
                {
                    MessageBox.Show("Error on the server");
                    return;
                }

                using (var memStream = new MemoryStream())
                {

                    await imageStream.CopyToAsync(memStream);
                    memStream.Position = 0;

                    BitmapImage bitmap = new BitmapImage();
                    bitmap.BeginInit();
                    bitmap.CacheOption = BitmapCacheOption.OnLoad;
                    bitmap.StreamSource = memStream;
                    bitmap.EndInit();
                    bitmap.Freeze();
                    ImageViewer2.Source = bitmap;
                }
            }
        }

        private async void Send_OnClick2(object sender, RoutedEventArgs e)
        {
            using (var client = new HttpClient())
            {
                var response =
client.GetAsync("http://localhost:5000/file1").Result;
                BitmapImage bitmap = new BitmapImage();
                if (response != null)
                {
                    using (var stream = await
response.Content.ReadAsStreamAsync())
                    using (var memStream = new MemoryStream())
                    {

                        await stream.CopyToAsync(memStream);
                        memStream.Position = 0;
                        bitmap.BeginInit();
                        bitmap.CacheOption = BitmapCacheOption.OnLoad;
                        bitmap.StreamSource = memStream;
                        bitmap.EndInit();
                        bitmap.Freeze();
                        ImageViewer2.Source = bitmap;
                    }
                }

            }
        }
```

ДОДАТОК В

Сегментація зображень на основі згорткових нейронних мереж

Опис програми

УКР.НТУУ”КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ТР5295_19Б

Аркушів 8

Київ 2019

АНОТАЦІЯ

Даний додаток містить опис програмних системи розроблених для підготовки набору даних, навчання мережі для сегментації та обробку результатів.

Створені програмні системи показують процес обробки вхідних даних та обробку результатів передбачення. Вони виконують такі завдання:

— зменшення розмірності зображення та маски;

— виділення на масці необхідних рослин;

— перетворення масиву ймовірностей пікселів до зображення;

— перетворення зображення до трьовимірного масиву.

Програмні .

При розробці обох програмних систем використовувалась мова програмування Python та бібліоткеи Scikit-image. Код розроблений у наступних середовищах програмування: Microsoft Visual Studio Code та Jupiter Notebook.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

У додатку розглядається один з програмних модулів системи — модуль для навчання нейронної мережі з кодом УКР.НТУУ”КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ТР5295_19Б, що міститься у файлі Training.ipynb

Модуль реалізовано за допомогою бібліотеки Keras та мови програмування Python.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Навчання нейронної мережі відбувається на наборі даних, який попередньо потрібно підготувати. Завантажити до оперативної пам’яті, зменшити розмірність та провести аугментацію.

Також, після проведення передбачення мережею його потрібно перетворити до зображення для відображення користувачу. На виході мережі ми отримуємо маску передбачень належності пікселя зображення до того чи іншого пікселя. Модель виконує перетворення маски передбачення на зображення маски та її накладання на вхідне зображення для зручного аналізу.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Логічна структура модулю складається з набору методів, які викликаються по черзі.

— load_image_disk — метод завантаження зображення

— load_mask_disk — метод завантаження маски

— plot_image_from_array — метод відображення зображення

— resize_img — метод зміни розміру зображення

— get_img_id — метод отримання зображення

— arr_convert_array_to_binary_mask

— randomShiftScaleRotate — метод аугментації

— mask_convert256_to1 — перетворення маски передбачень до зображення

ТЕХНІЧНІ ЗАСОБИ ЩО ВИКОРИСТОВУЮТЬСЯ

Для розробки було використано мову програмування Python та її вбудовані модулі для роботи з пам’яттю. Було використано бібліотеки Scikit-image та Keras.

Scikit-image – це бібліотека з набором алгоритмів для обробки зображень. Вона доступна безкоштовно.

Keras - це бібліотека-обгортка високого рівня API, що здатна працювати поверх TensorFlow, CNTK або Theano. Вона розроблена для швидких експериментів з нейронними мережами.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Розроблена частина програми не потребує інсталяції. Достатньо відкритий файл Training.ipynb на порталі colab.google.com та покроково виконувати запуск необхідних частин коду.

Colab.google.com є хмарною версією Jupyter Notebook, який дозволяє створювати та обмінюватися документами, що містять текст та код, підтримує візуалізацію зображень та тексту.

ВХІДНІ І ВИХІДНІ ДАНІ

Вхідними даними для модуля є наявний набір даних із зображеннями рослин цукрового буряка та бур’янів. Процесі навчання зберігається набір навчених вагів, які потім можна завантажити та використовувати для передбачення. Результат роботи залежить від налаштування та порядку виклику функцій.