ЗМІСТ

[ВСТУП 4](#_Toc11141530)

[РОЗДІЛ 1. ОГЛЯД ЛІТЕРАТУРИ 7](#_Toc11141531)

[1.1. Глибоке навчання 7](#_Toc11141532)

[1.2 Штучні нейронні мережі 8](#_Toc11141533)

[1.3 Глибокі навчальні програми та бібліотеки 10](#_Toc11141534)

[РОЗДІЛ 2. ОГЛЯД МЕТОДІВ НАВЧАННЯ 12](#_Toc11141535)

[2.1. Класичні методи машинного навчання 12](#_Toc11141536)

[2.2 Нейромережеві методи 13](#_Toc11141537)

[2.3 Згорткова нейронна мережа 14](#_Toc11141538)

[2.4 Модельні архітектури 21](#_Toc11141539)

[РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ 25](#_Toc11141540)

[3.1 Огляд набору даних 25](#_Toc11141541)

[3.2 Інструменти та технології 25](#_Toc11141542)

[3.3 Пояснення побудови архітектури згорткової нейронної мережі 26](#_Toc11141543)

[3.4 Навчання мережі 30](#_Toc11141544)

[3.5 Результати 30](#_Toc11141545)

[ВИСНОВОК 33](#_Toc11141547)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 35](#_Toc11141548)

# ВСТУП

Машинне навчання в наш час переживає нове народження і поступово входить в усі галузі людської діяльності, починаючи з рекомендованої реклами, закінчуючи управлінням автоматичними заводами. У тому числі складні і важливі завдання вирішуються методами комп'ютерного зору:

* Безпілотні автомобілі використовують камери і радари для того, щоб правильно взаємодіяти один з одним.
* Системи протиповітряної оборони (ППО) виявляють стелс-літаки з допомогою високоточних камер.
* Вивіски магазинів автоматично визначають стать людини, яка проходить повз, з метою показати таргетну рекламу.
* Автоматично розпізнаються аномалії на рентгенівських знімках для подальшого дослідження.

Все це і багато іншого стало можливо в зв'язку з революцією в області високопродуктивних відеокарт і згорткових мереж.

Протягом довгої історії комп'ютерного зору одним з найважливіших завдань була семантична сегментація, яка полягає в можливості сегментувати невідомі зображення в різні частини і об'єкти (наприклад, пляж, океан, сонце, собака, плавець). Крім того, сегментація є ще глибшою, ніж розпізнавання об'єктів, оскільки розпізнавання не є необхідним для сегментації. Семантична сегментація є інформативною, оскільки класифікує кожен піксель зображення. Протягом останніх кількох років сегментацію було реалізовано повністю завдяки глибокому навчанню. Зокрема, люди можуть виконувати сегментацію зображення, не знаючи, що таке об'єкти. Виконання сегментації, не знаючи точної ідентичності всіх об'єктів на сцені, є важливою частиною нашого візуального процесу розуміння, який може дати нам потужну модель для усвідомлення світу, а також використовувати його для створення або вдосконалення існуючих методів комп'ютерного зору.

Мета практичної частини цієї дипломної робити полягає у реалізації семантичної сегментації зображень для розпізнавання доріг, використовуючи методи штучного інтелекту.

В роботі вирішуються такі задачі, необхідні для досягнення мети:

1. підготовка набору даних
2. анотація даних
3. розробка архітектури нейронної мережі
4. тренування нейронної мережі
5. аналіз результатів
6. модифікація нейронної мережі або гіперпараметрів на основі попередніх результатів

Актуальність теми цієї дипломної робити зумовлена розвитком новітніх технологій. Автономне водіння стане однією з революційних у найближчому майбутньому з точки зору впливу на життя людей, що живуть у промисловорозвинених країнах. Відомо, що зараз ми переживаємо бум розвитку безпілотного транспорту, тому багато компаній випустили свої прототипи (найвідоміші-Google, Uber, Yandex) [10]. Сучасні системи камер можуть бути розміщені в багатьох місцях, починаючи з мобільних телефонів до систем спостереження автономних транспортних засобів, для отримання дуже якісних зображень за незначних затрат. Це збільшує попит на системи, які можуть інтерпретувати та розуміти ці образи.

Багато дослідницьких спільнот зробили свій внесок у розвиток автономних систем керування, завдяки швидкому зростанню продуктивності алгоритмів бачення, таких як виявлення об'єктів, сегментація доріг і розпізнавання сигналів руху. Автономний автомобіль повинен відчувати своє оточення, знаходити дорожні знаки, пішоходів, інші машини та інші семантичні класи і діяти безпечно, щоб досягти певної мети, інакше далеко не заїдеш.

# 

# РОЗДІЛ 1. ОГЛЯД ЛІТЕРАТУРИ

## **1.1. Глибоке навчання**

Глибоке навчання - це швидкозростаючий популярний підхід до машинного навчання у сфері штучного інтелекту для створення моделі для сприйняття та розуміння великої кількості машин, таких як зображення та звук. В основному, цей підхід ґрунтується на глибоких архітектурах, які є більш структурно складними штучними нейронними мережами (ШНМ) [1]. Цей термін глибокої архітектури відноситься до штучних нейронних мереж, кількість прихованих шарів яких збільшена.

Алгоритми глибокого навчання відокремлені від існуючих алгоритмів машинного навчання. Для їхньої реалізації потрібна дуже велика кількість даних і апаратних засобів з дуже високою обчислювальною потужністю, яка може обробляти цю велику швидкість передачі даних. В останні роки кількість маркованих зображень, особливо в області комп'ютерного зору, значно зросла. Глибокий підхід до навчання привернув багато уваги завдяки великому прогресу в області паралельних обчислювальних потужностей на основі графічних процесорів(GPU). Графічні процесори з тисячами обчислювальних ядер забезпечують від 10 до 100 разів більшу продуктивність програми при обробці цих даних порівняно з центральними процесорами(CPU). В даний час глибоке навчання має багато областей застосування, в основному автоматичне розпізнавання мови, розпізнавання зображень і обробка природної мови.

Існує багато різних типів архітектури глибокого навчання. В основному, архітектури глибокого навчання можна назвати так, як на рис. 1.1.

Основні архітектури глибокого навчання:

Глибока нейронна мережа

Глибинна мережа переконань

Згорткова нейронна мережа

Оглавление

**Элементы оглавления не найдены.**

Машина Больцмана

Автокодувальник

Глибока ядрова мережа

Глибока мережа кодування

Рисунок 1.1 – Основні архітектури глибокого навчання

## **1.2 Штучні нейронні мережі**

ШНМ були розроблені у процесу вивчання мозку людини. Коли нейрони біологічної нервової системи з'єднуються один з одним, структури, визначені як штучні нейрони в системах ШНМ, моделюються так, щоб бути пов'язаними один з одним. Штучні нейронні системи можна використовувати в багатьох областях, таких як ідентифікація системи, розпізнавання зображень і мови, прогнозування та оцінка, аналіз відмов, медицина, зв'язок, трафік, управління виробництвом та багато іншого.

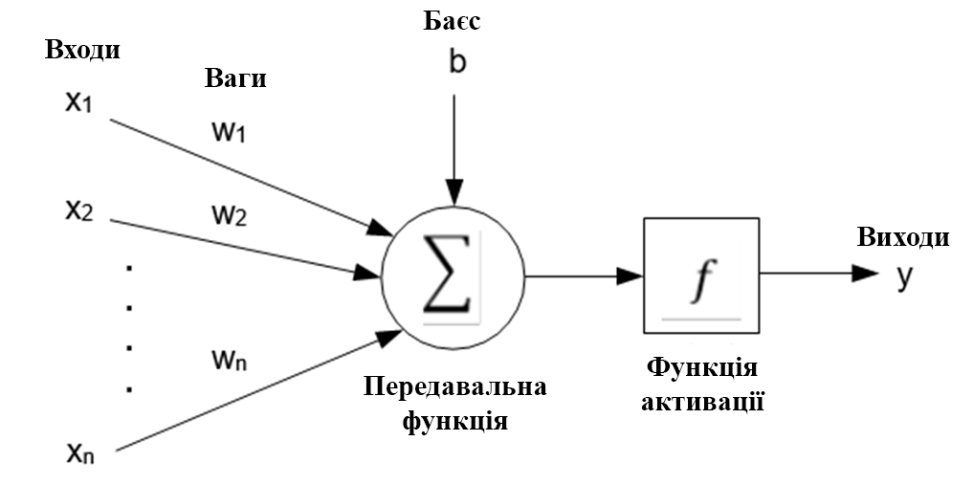
****Штучні нейронні мережі також мають штучні нейрони, оскільки біологічні нейронні мережі є нейронами. Нейрон можна назвати базовою обчислювальною одиницею в ШНМ. Нейрони також можуть називатися вузлом або блоком. Структура штучного нейрона показана на рисунку 1.2.

Рисунок 1.2 – Структура штучного нейрона

Входи є інформацією, що надходить до нейрона з зовнішнього світу. Вони визначаються зразками, для яких бажане вивчення мережі. Ваги показують важливість інформації, що надходить до нейронів, і їх вплив на нейрони. Для кожного входу є окремі ваги. Наприклад, вага на рисунку 1.2 показує вплив входу на нейрон. Той факт, що ваги великі або малі, не означає, що вони важливі або незначні. Функція передачі обчислює чистий вхід, що надходить до нейрона. Хоча для цього існує велика кількість функцій передачі, найчастіше використовується зважена сума. Кожна вхідна інформація підсумовується шляхом множення власної ваги. Функція активації визначає вихідний сигнал, який нейрон генерує у відповідь на цей вхід, обробляючи чистий вхід, що надходить до нейрона. Сформований вихід направляється у зовнішній світ або інший нейрон. Крім того, при бажанні, нейрон може також відправляти свій власний вихід як вхідний сигнал для себе.

Функція активації зазвичай є нелінійною функцією. Метою функції активації є перенесення нелінійності на вихід нейрона. Характеристикою ШНМ є нелінійність, яка обумовлена нелінійністю функцій активації.

При виборі функції активації важливо відзначити, що похідну функції легко обчислити. Це забезпечує швидке виконання розрахунків.

Існує безліч функцій активації, таких як лінійна, ступінчаста, сигмоподібна, гіперболічна дотична (tanh), ректифікована лінійна одиниця (ReLU) і порогові функції.

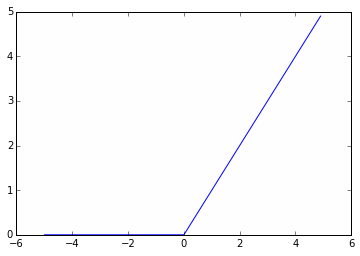
Функція активації ReLU , генерує висновок з порогом 0 для кожного з вхідних значень. Вона має характеристику, як на рисунку 1.3. Останнім часом ця функія є дуже популярною в ШНМ, саме її було використано при реалізації задачі.

Рисунок 1.3 – Характеристика функції активації ReLU

## **1.3 Глибокі навчальні програми та бібліотеки**

Робочий процес глибокого навчання є багатостадійним, ітеративним процесом. У цьому процесі дані повинні бути зібрані і попередньо оброблені, якщо це необхідно. Для успішного глибокого навчання можуть знадобитися великомасштабні набори даних. Сьогодні, завдяки Інтернету та великим джерелам даних, масиви даних швидко зростають. Мережа навчається з використанням цих наборів даних. Існуючі мережі можуть використовуватися для навчання мережі, або можуть бути розроблені нові мережі. Моделі мережі, створені після етапу навчання, повинні бути перевірені, щоб підтвердити, що вони працюють як слід. Як правило, на цьому етапі необхідно повторити певні операції для поліпшення результатів. Ці операції включають в себе повторну обробку даних, упорядкування мереж, зміну параметрів мереж або розв'язувачів і повторне тестування до отримання бажаного результату. Центральні процесори цих комп'ютерів є недостатніми для виконання цих інтенсивних обчислювальних процесів у процесі глибокого навчання. Оскільки процесори з певною потужністю обробки та архітектурою не можуть виконувати багато операцій одночасно, навчальні та тестові фази моделі займають багато часу. Через це центральні процесори віддали місце графічним процесорам, які дозволяють паралельну обробку даних. Завдяки цьому почалося швидке використання глибокого навчання в реальних додатках.

Для реалізації глибоких алгоритмів навчання існує безліч популярних програмних платформ і бібліотек, зокрема, таких як Caffe, Torch, Theano, TensorFlow, Keras і DIGITS. Більшість з них також можуть працювати на GPU.

# РОЗДІЛ 2. ОГЛЯД МЕТОДІВ НАВЧАННЯ

## **2.1. Класичні методи машинного навчання**

Перед тим, як почалася ера глибокого навчання, було використано велику кількість методів обробки зображень для сегментації зображень.

Такі методи машинного навчання, як випадковий ліс (Random Forest, RF) [7], опорна машина векторів (Support vector machine, SVM) [12], логістична регресія (Logistic Regression, LR) - є класичними методами машинного навчання з учителем, вхідна розмірність (кількість вхідних ознак) яких фіксована. Експериментально вдалося показати, що випадковий ліс краще за інших справляється із завданням семантичної попіксельної сегментації.

Принцип роботи класичних методів для попіксельної сегментації полягає в наступному:

1. зафіксувати кількість найближчих сусідів, ознаки яких будуть використовуватися для класифікації кожного пікселя. Назвемо це скануючим вікном;
2. визначити які саме ознаки пікселя будуть використовуватися, наприклад значення RGB (міра червоного, зеленого і синього кольорів);
3. скласти навчальну вибірку, де кожного пікселя (і його мітці класу), для якого потрібно передбачити клас буде відповідати вектор ознак, що складається з ознак сусідів пікселя;
4. вектора ознак для пікселів, що знаходяться на кордоні зображення, необхідно доповнити до необхідної розмірності за допомогою заповнюючого (padding) значення. Це необхідно за для обмеженням класичних методів машинного навчання (кількість вхідних ознак повинна бути фіксованою) ;
5. навчити одну або кілька моделей (в залежності від підходу).

При збільшенні скануючого вікна класифікатор буде отримувати більше інформації про сусідів пікселя, проте все менше відмінностей для нього буде між сусідніми пікселями. При цьому розмір скануючого вікна при такому підході можна варіювати від невеликого до розміру всього зображення (в такому випадку розміри всіх зображень в розподілі повинні збігатися, або повинен бути зафіксований конкретний розмір і, якщо зображення менше, то воно доповнюється заповнюючим символом, а якщо більше - обрізається)

Класичні методи показують помітно гіршу точність семантичної сегментації в порівнянні з нейромережевими підходами, а також вкрай обчислювально неефективні (через необхідність ходити по зображенню вікнами).

## **2.2 Нейромережеві методи**

Глибоке навчання значно спростило конвеєр для виконання семантичної сегментації і дає результати вражаючої якості. У цьому розділі ми розглянемо популярні архітектури моделі та функції втрат, які використовуються для навчання цих глибоких методів навчання.

В даний час методи, які використовують згорткові нейронні мережі є state of the art в багатьох задачах комп'ютерного зору і семантична сегментація не виняток.

Перевага досягається з одного боку через властивості згорткової мережі автоматично виділяти ознаки зображень (і вона робить це набагато краще, ніж фантазія фахівців з комп'ютерного зору) - за рахунок цього досягається перевага за якістю, з іншого - через зростання потужності комп`ютерної систем (зокрема поява потужних GPU), що дозволяє працювати таким нейромережевим архітектурам, навчання і застосування яких раніше вважалося неможливим. Зараз же, навчання таких моделей займає адекватний час, а застосування можливо в реальному часі навіть на мобільних пристроях.

Вперше значну перевагу нейронних мереж в задачах комп'ютерного зору було показано в роботі, присвяченій перемозі в змаганні ImageNet [6].

Дуже важливо, що нейронні мережі дозволяють відійти від крайньої обчислювально-неефективної процедури ковзного вікна на користь архітектури енкодера-декодера, працюючи з усією картинкою цілком, а не з обмеженою її областю.

## **2.3 Згорткова нейронна мережа**

Згорткова нейронна мережа, впроваджена компанією LeCun в 1989 році для додатків комп'ютерного зору, є типом багатошарового подавання штучних нейронних мереж. Сьогодні згорткові мережі стають все більш популярними серед глибоких методів навчання, оскільки вони можуть успішно вивчати моделі для багатьох комп'ютерних та візуальних програм, таких як виявлення об'єктів, розпізнавання об'єктів та сегментація семантичного зображення. Згорткові нейронні мережі можна розглядати як класифікатори, які витягують ієрархічні функції з необроблених даних. У цих мережах зображення подаються як вхідні дані в мережу, а навчання відбувається автоматично за допомогою ієрархії об'єктів, створеної без використання будь-якого методу вилучення об'єктів.

Всі нейрони в шарі в прямих штучних нейронних мереж з'єднані з усіма нейронами наступного шару. Такі пов'язані шари називаються повністю пов'язаними шарами і, крім повністю пов'язаних шарів в згорткових нейронних мережах, згортка застосовується до вхідного зображення для створення виходу. Це викликано локальним з'єднанням, що всі області у вхідному шарі пов'язані з нейронами в наступному шарі. Таким чином, вхідне зображення згортається з кожним навченим фільтром, що використовується в цьому шарі, для генерації різних карт об'єктів. Карти характеристик стають більш нечутливими до обертання і спотворень, забезпечуючи все більш складні узагальнення до вищих шарів. Крім того, карти об'єктів, отримані в згортковому шарі, піддаються шару об'єднання для того, щоб виконувати зменшення просторової розмірності і збереження важливих особливостей. Класифікатор завжди є кінцевим шаром для генерування значень ймовірності класу як вихідного.

Остаточний вихід із згоркового і об'єднувального шарів передається в один або більше повністю з'єднаних шарів. Потім, прогнозування виходу отримують шляхом перенесення на шар класифікатора. Проста архітектура згоркової нейронної мережі являє собою комбінацію згорткових, об'єднуючих і повністю пов'язаних шарів, як на рисунку 2.2.

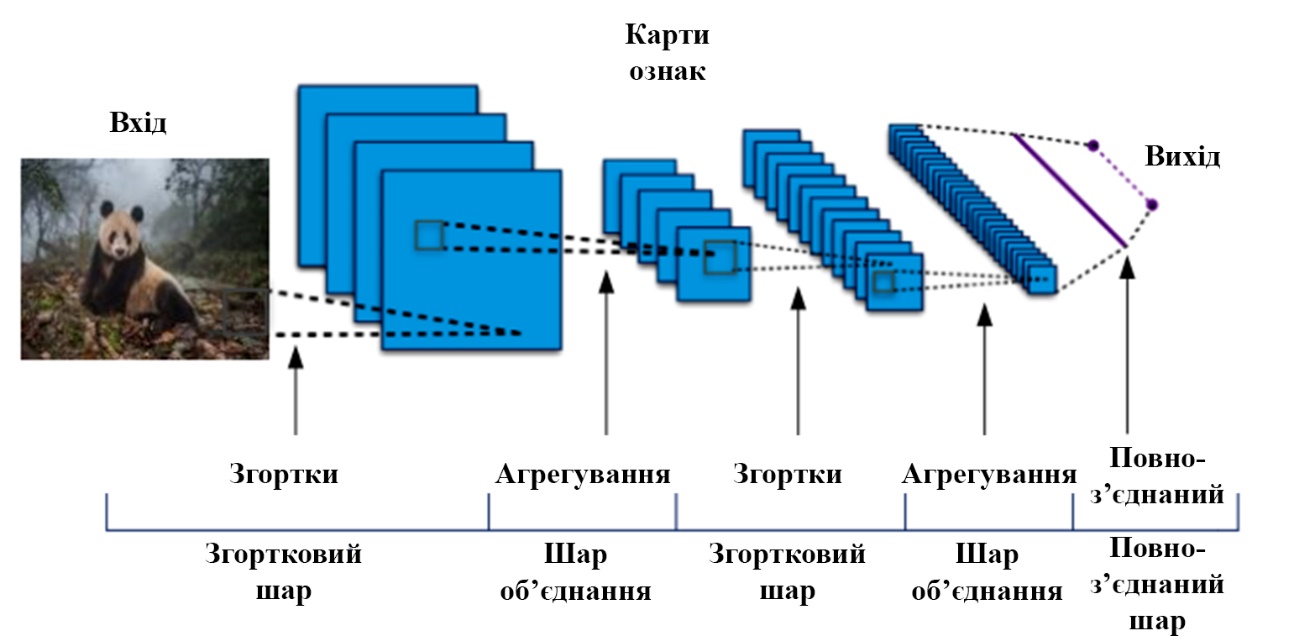


Рисунок 2.2 – Приклад простої архітектури згорткової нейронної мережі

Метою згорткового шару, який є основним шаром мережі, є згортання вхідного зображення за допомогою фільтрів, що вивчаються, і вилучення його особливостей. З кожним фільтром створюється мапа об'єктів. Згорткові нейронні мережі звертають увагу на те, що при використанні RGB-зображень (зображень, що використовуються в даній роботі) зображення є 3D-матрицею, і кожен з шарів подібним чином розташований. Це показано на рисунку 7. Кожен шар згоркової мережі складається з набору просторових фільтрів розміром d × h × w, які являють собою просторові розміри h і w, які мають об'єм нейронів і кількість ядр ( або фільтрів) каналів функції d. Кожен з цих фільтрів піддається згортці з відповідним обсягом вхідного зображення, і ковзає по всьому зображенню (розміром × × , де , - просторові розміри і - номер каналу) по всіх його просторовим вимірам , . Згортка відноситься до суми елемента множенням елементів нейронів у кожному фільтрі з відповідними значеннями на вході. Таким чином, можна припустити, що першим шаром в згоркової мережі є вхідне зображення. Виходячи з цього, згортка з одним фільтром в кожному шарі забезпечує двовимірний висновок з такими параметрами, як шари і прокладка. Це виражається у вигляді карти об'єкту або карти активації для вхідного фільтра. На кожному згортковому шарі мережі використовуються N фільтрів, кожен з яких призводить до відображення карти. Ці особливості карти складені разом в певному обсязі для отримання виходу згорткового шару.

Один нейрон у фільтрі певного шару може бути відображений у зв'язані нейрони у всіх попередніх шарах, слідуючи такій згортці. Це називається ефективним рецептивним полем нейрона. Легко бачити, що згортка призводить до дуже локальних зв'язків з нейронами в нижніх шарах (ближче до входу) з тими меншими рецептивними полями, ніж у вищому шарі. У той час як нижні шари навчаються представляти невеликі ділянки входу, більш високі шари вивчають більш специфічні значення, оскільки вони реагують на більший підрозділ вхідного зображення. Таким чином, ієрархія ознак генерується від локального до глобального.

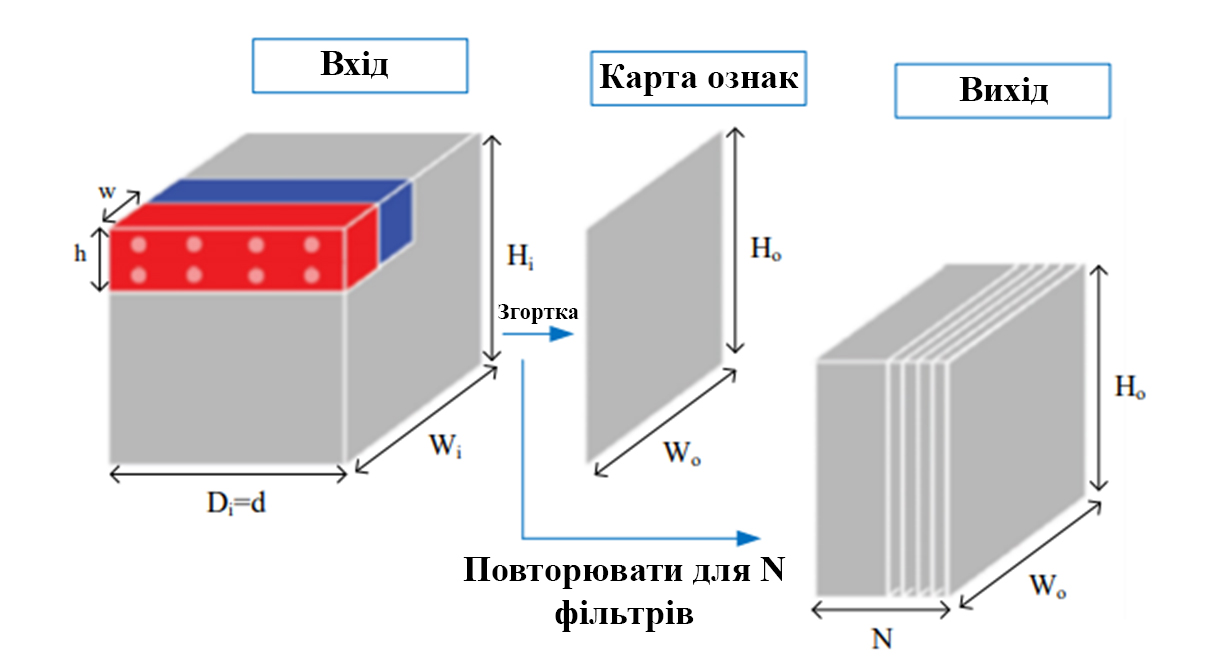


Рисунок 2.3 – Ілюстрація одного згорткового шару

Червоні і сині області на рисунку 2.3. представляють дві позиції однакових фільтрів розміром d × h × w, які піддаються згортці шляхом ковзання через вхідний об'єм. Враховуючи, що розмір фільтра становить 2 × 2, видно, що параметр кроку s дорівнює 2. Для RGB-вхідного зображення = d = 3.

Штрихи s фільтра визначаються як інтервали, коли фільтр рухається в кожному просторовому вимірі. p padding відповідає кількості пікселів, доданих до зовнішніх країв входу. Отже, крок може розглядатися як вхідний засіб підвибірки. Зазвичай використовують квадратні фільтри формату h = w = f. Вихідний обсяг такого шару обчислюється за допомогою рівнянь (1), (2) і (3).

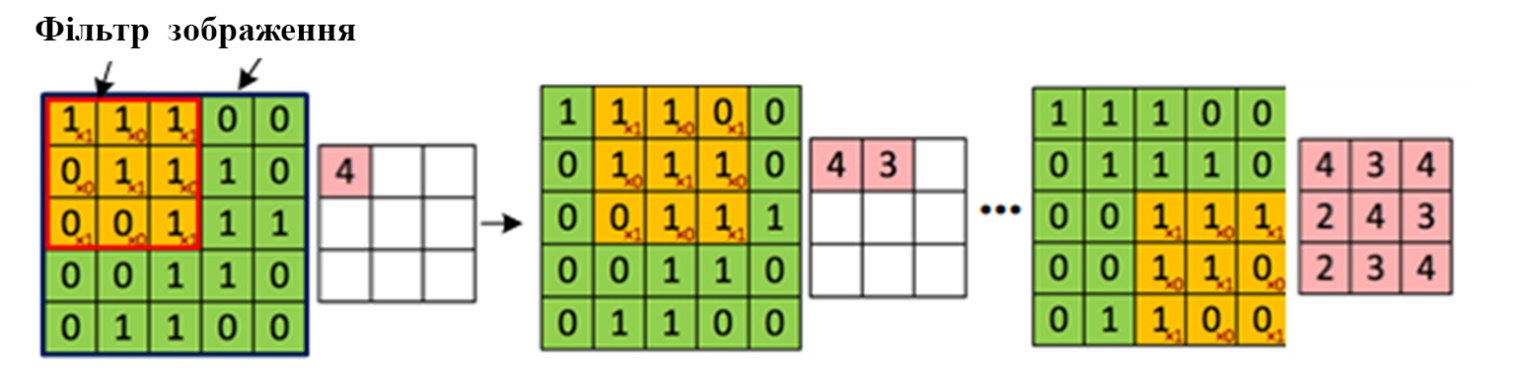
На рисунок 2.4 показано фільтр 3 × 3, який ковзає по матриці зображення 5 × 5, що представляє бінарне зображення. Ковзання фільтру відбувається зліва направо і продовжується до кінця матриці. Крок приймається як 1. За допомогою ковзання фільтрів по порядку процес завершується і отримується остаточний стан карти об'єкта, як показано нижче.

Рисунок 2.4. – Матриця зображення і кінцевий стан карти об'єкта

Часто між згортковими шарами розсіюються шари об'єднання, які допомагають просторово вбирати вхідні функції. Шари об'єднання об'єктів виконують процес відбору підсистеми вхідного зображення. Це робиться шляхом зсуву фільтру над вхідним зображенням.

Вхідне зображення (зазвичай не перекриваються) поділяється на підрегіони, і кожен підрегіон відбирається за допомогою нелінійних функцій об'єднання. Найбільш відомими з цих функцій є максимальні та середні функції об'єднання. Маючи максимальну функцію пулу, максимальне значення повертається з кожного підрегіону. Середня функція пулу повертає середнє значення підрегіону. Об'єднання забезпечує надійність мережі за рахунок зменшення кількості дисперсій трансляції зображення. Крім того, також відкидаються непотрібні та надлишкові функції, що знижують обчислювальну вартість мережі і, отже, робилять її більш ефективною.

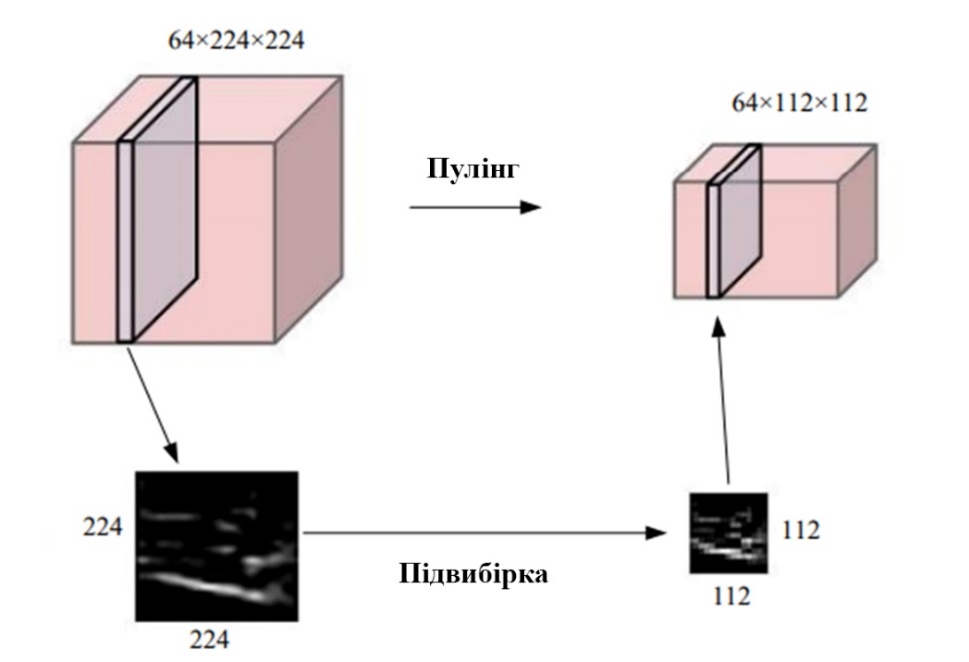
Шари об'єднання також мають параметр кроку, який забезпечує контроль над розмірами виведення. Для цього шару можуть бути використані ті ж рівняння, що використовуються для вихідного розміру згорткових шарів. На рисунку 2.5 видно, що вхідний об'єм 64 × 224 × 224 підсумовується до обсягу 64 × 112 × 112 за допомогою 2 × 2 фільтрів і 2 кроків. Операція об'єднання здійснюється окремо для кожної карти об'єкту, і розмір карти об'єкту зменшується, як показано на малюнку.

Рисунок 2.5 – Приклад підвибірки

На рисунку 2.6 показана операція об'єднання, що виконується з максимальною функцією об'єднання, шляхом переміщення фільтра 2 × 2, шару 2.

Рисунок 2.6 – Операція об'єднання з фільтром 2 × 2, крок 2

Як правило, виходи згорткового шару подаються в активаційні функції. Плани нелінійності, запропоновані для цієї мети, можуть складатися з таких функцій, як сигмоподібний, tanh і ReLU. Було виявлено, що ReLU є більш ефективним у згоркових мережах і часто є кращим.

У ReLU рівень негативних входів до наближується 0 і активує позитивні входи, як описано (7), передаючи їх без змін.

де: x - вхід ReLU, а f (x) - випрямлений вихід.

У шарі ReLU операція виконується окремо для кожного значення пікселя. Наприклад, вихід ReLU є таким, як показано на рисунку 2.7, якщо вважається, що чорні області представлені негативними пікселями, а білі області представлені позитивними пікселями у вхідній карті об'єкта.

Після вилучення функцій високого рівня з згортковими, пулу і ReLU шарами, як правило, повністю з'єднаний шар розміщується в кінці мережі. Нейрони в цьому шарі повністю залежать від усіх активацій у попередньому шарі. Найбільш важливою особливістю повністю пов'язаного шару є те, що він дозволяє нейронам у цьому шарі визначити, які ознаки відповідають яким класифікаціям. Коротше кажучи, повністю пов'язаний шар можна розглядати як шар, який подає класифікатор.

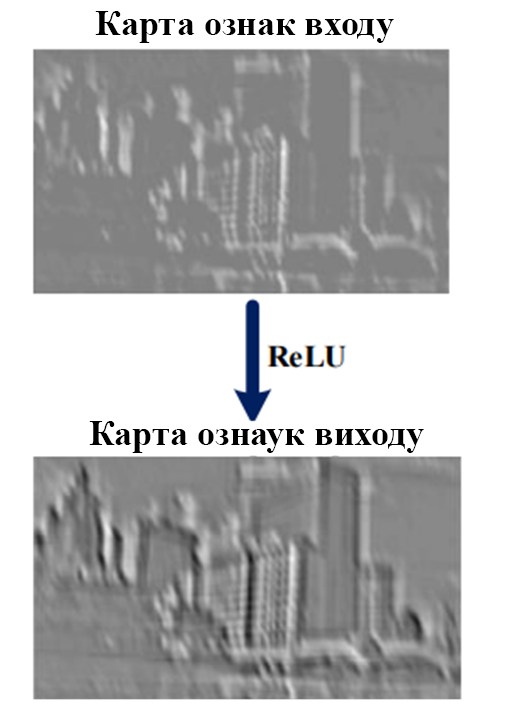


Рисунок 2.7 – Приклад операції ReLU

## **2.4 Модельні архітектури**

Однією з найпростіших і найпопулярніших архітектур, що використовуються для семантичного сегментації, є повністю конвертована мережа (FCN). У роботі FCN для семантичної сегментації автори використовують FCN для першого зменшення вхідного зображення до меншого розміру (при отриманні більшої кількості каналів) через серію згорток. Цей набір згорток називається кодером. Після цього кодований висновок підсилюється або через білінеарну інтерполяцію, або через серію транспонованих згорток. Цей набір транспонованих згорток зазвичай називається декодером [4].

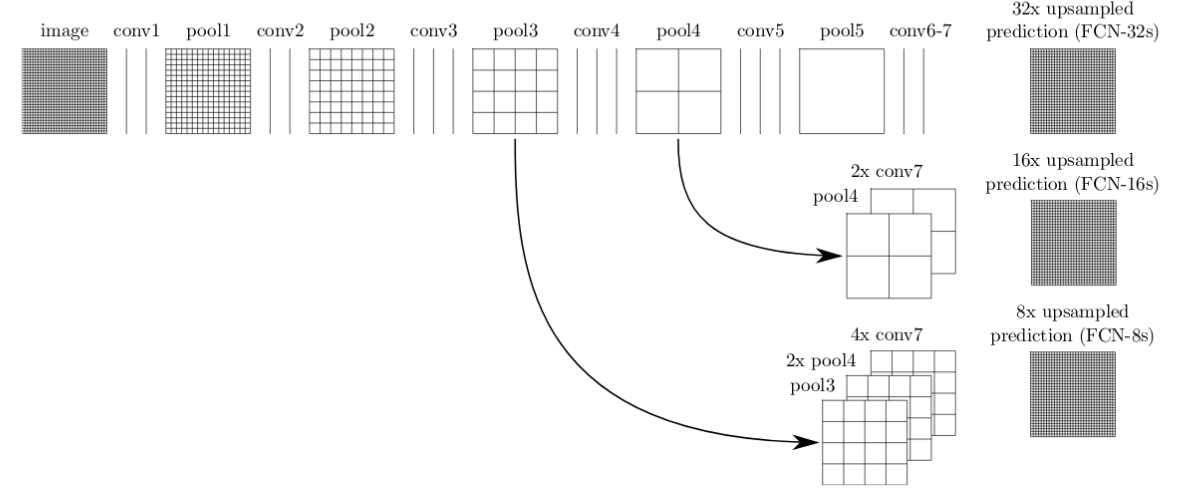
****

Рисунок 2.8 – FCN архітектура.

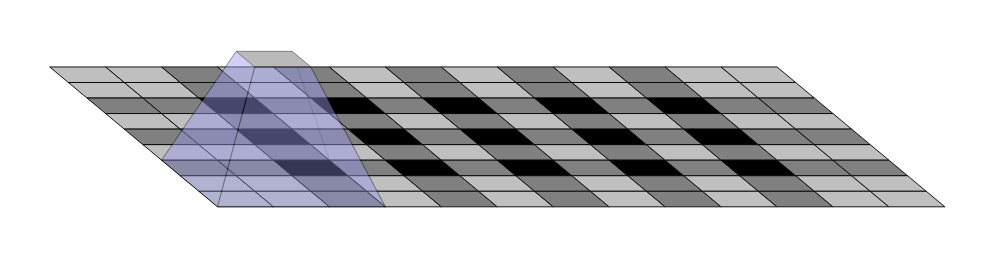
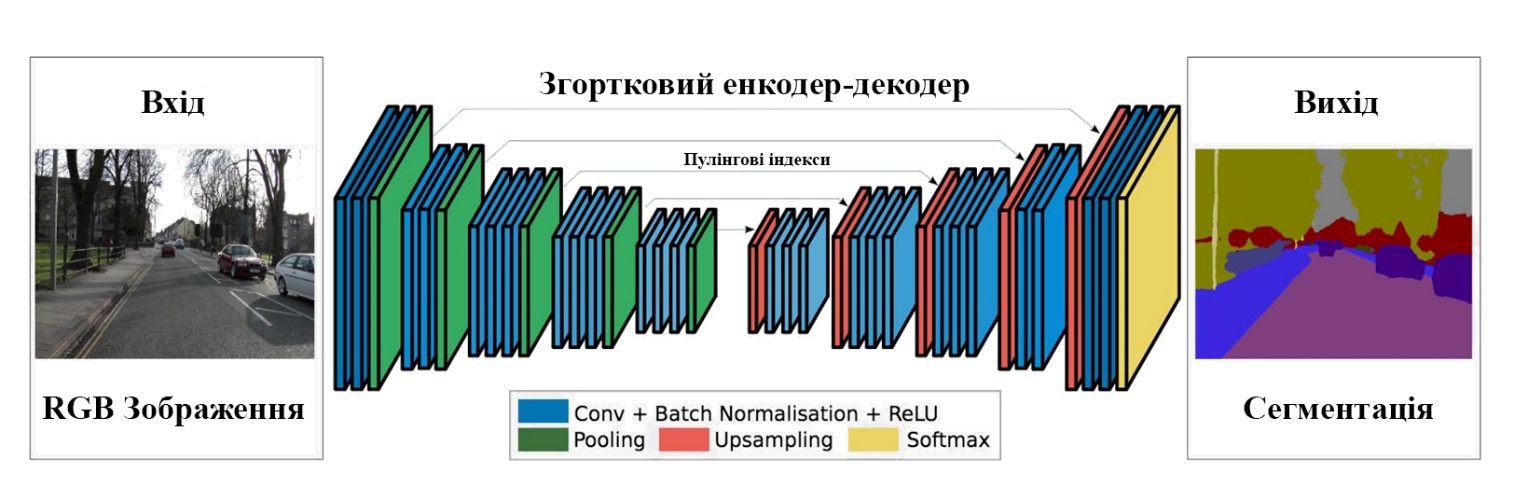
Ця основна архітектура, незважаючи на ефективність, має ряд недоліків. Одним з таких недоліків є наявність артефактів шахової дошки через нерівномірне перекриття виходу операції транспонування-згортки (або деконволюції).

Рисунок 2.9 – Формування артефактів шахової дошки.

Іншим недоліком є низька роздільна здатність на кордонах через втрату інформації від процесу кодування.

Модель SegNet [4] є типовим автокодером, заснованим на згортковій нейронної мережі. Схема цієї моделі представлена ​​на рисунку 2.10. Мережа складається з блоків, в кожному блоці присутні згортки і пулінгу (субдискретизуючі шари) або шари, що підвищують дискретизацию (апсемплінг шари), а також активаційні шари Relu і шари нормалізації батчнормалізаціі (BatchNorm). Архітектура повністю симетрична, крім шару м'якого максимуму (Softmax) на кінці декодера. Цей шар перетворює кожен піксель вихідної матриці в ціле число, що показує клас даного пікселя. Головна відмінність SegNet від звичайного згорткового автокодера в тому, що його апсемплінг шари декодера інформаційно з'єднані з відповідними пулінг шарами енкодера. Тобто, апсемплінг шари мережі не навчаються, а отримують необхідну інформацію про те, як підвищити розмірність і як відновити стислу (загублену) топологію від відповідних пулінг шарів, які зберігають індекси активованих пікселів (пікселів з найбільшим значенням у вікні).

Рисунок 2.10 – Архітектура SegNet [5]

U-net [8] є одним з яскравих представників сегментаційних енкодер - декодер моделей, незважаючи на те, що вперше він заявлений як мережа для сегментації біологічних знімків, він добре зарекомендував себе як бейзлайн при вирішенні практичних завдань сегментації. Самец ю архітектуру використано для реалізації задачі.

Переваги використання U-Net:

* Обчислювально ефективний
* Тренування з невеликим набором даних
* Навчається наскрізь

Архітектура мережі показана на рисунку 2.11 і містить дві частини, звужується (енкодер) і розширюється (декодер). Перша являє собою типову архітектуру згорткової класифікаційної мережі. Складається з повторюваних застосувань двох згорток 3x3, за якими слідують активація (Relu) і операція макс-пулінга 2x2 з кроком 2. На кожному кроці підвищуємо кількість каналів вдвічі.

Розширювана частина складається з кроків так званої зворотної згортки (деконволюції) 2x2, зменшує кількість каналів, потім конкатенація з відповідним чином обрізаними картами ознак від відповідної частини (видно на схемі) звужуються частини і дві згортки 3x3 і активація (Relu). Обрізка необхідна через втрату прикордонних пікселів.

На останньому рівні згортка 1x1 використовується для зіставлення кожному 64- компонентного вектора ознак класу.

В цілому мережа має 23 згортальних шари.

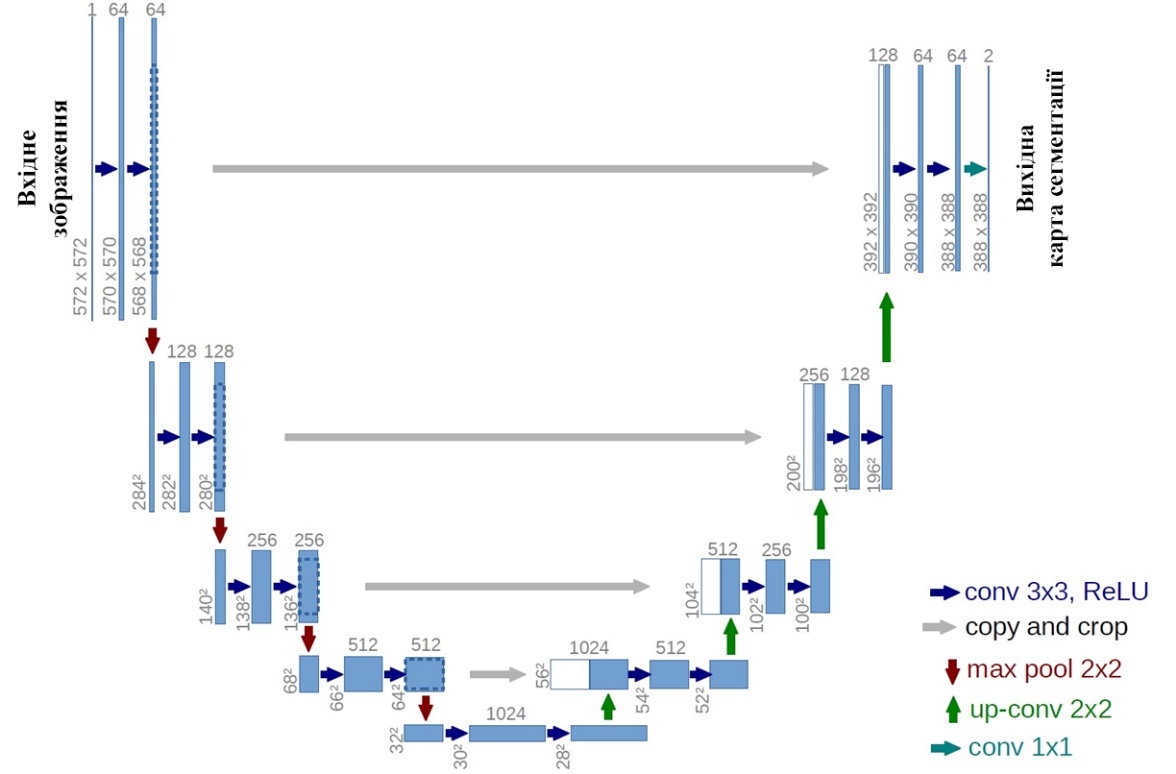


Рисунок 2.11 – Архітектура U-Net (приклад для 32x32 пікселів у низькому рівні)

# РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ

## **3.1 Огляд набору даних**

Набір даних який надає зображення даних і позначені семантичними сегментам зображення, зафіксовані за допомогою автосимулятора CARLA [5]. Дані генерувалися як частина Lyft Udacity Challenge. Цей набір даних може використовуватися для навчання алгоритмів ML (Machine Learning) для визначення семантичної сегментації автомобілів, доріг на зображенні, оскільки він має високу якість розмітки і добре опрацьовані дорожні сценарії.

Дані мають 5 наборів по 1000 зображень і відповідних міток.



Рисунок 3.1 – Приклад зображення з Lyft Udacity Challenge

## **3.2 Інструменти та технології**

Для реалізації поставленної переді мною задачі використано інтерпретовану об’єктно-орієнтовану мову програмування Python, оскільки ця мова програмування є значно простішою ніж C# чи C++, легко обробляє велику кількість данних і представлена багатьма корисними бібліотеками, особливо для аналітики, та згорткову нейронну мережу U-Net.

Для побудовий нейронної мережі використано Tensorflow - відкриту програмну бібліотеку для машинного навчання цілої низки задач, розробленої компанією Google для задоволення її потреб у систем, здатних будувати та тренувати нейронні мережі для виявлення та розшифровування образів та кореляцій, аналогічно до навчання й розуміння які застосовують люди.

Також було використано Keras. Keras - це модульна бібліотека Python, побудована на бібліотеках глибокого навчання TensorFlow і Theano. Ці дві основні бібліотеки забезпечують можливість запуску на GPU або CPU. Здійснюючи незначні зміни у файлі налаштувань Keras, можна використовуватиTensorFlow або Theano у фоновому режимі. Keras є дуже корисним, оскільки спрощує інтерфейс бібліотек TensorFlow і Theano, і більш легке застосування може бути розроблено, ніж ці дві бібліотеки. Keras має дуже поширене використання в додатках для обробки зображень.

## **3.3 Пояснення побудови архітектури згорткової нейронної мережі**

1. Оголошення змінних:

* Conv2D – це шари які означають, що застосовуються два послідовних шари згортки;
* c1, c2, c3, c4, c5, c6, c7, с8 - вихідні тензори згорткових шарів;
* p1, p2 і p3 є вихідними тензорами макс пулінг шарів;
* u5, u6, u7 і u8 є вихідними тензорами шарів вибірки (транспонованих згорткових).

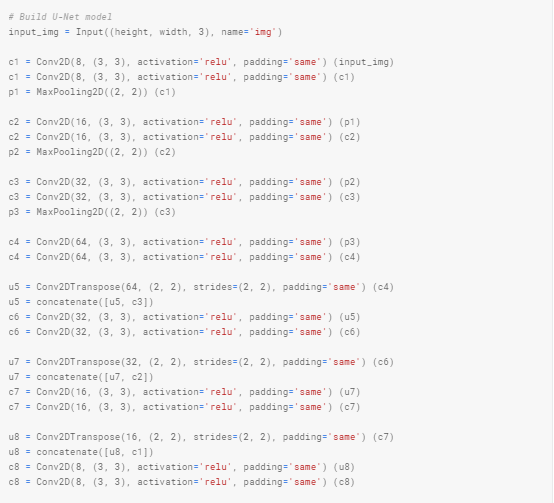


Рисунок 3.2 – код Keras для визначення вищезазначеної моделі

1. Далі нейронна мережа проходить шлях скорочення - енкодер, де застосовуються регулярні згортки і макс пул.

У енкодері розмір зображення поступово зменшується, а глибина зображення поступово збільшується. Вхідне зображення поступово перетворюється з 600×800×3 до 75×100×64. А саме:

600×800×3 –> 300×300×8 –>150×200×16 –> 75×100×32 –> 75×100×64

Це означає, що мережа дізнається інформацію "WHAT" на зображенні, однак вона втратила інформацію "WHERE". Детальніше можна розгянути зміни на русунку.

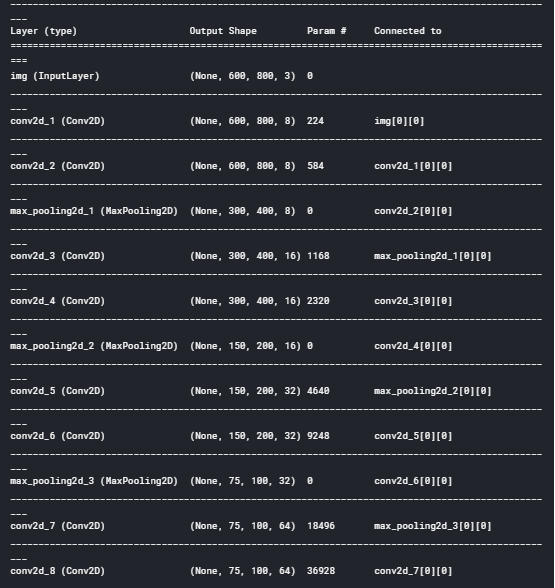


Рисунок 3.3– шлях скорочення зображення

1. Наступний крок - шлях розширення декор, де застосовуються транспоновані згортки разом з регулярними згортками.

У декодері розмір зображення поступово збільшується і глибина поступово зменшується. Починаючи з 75×100×64 до 600×800×1.Інтуїтивно, декодер відновлює інформацію "WHERE" (точна локалізація), поступово застосовуючи вибірку.

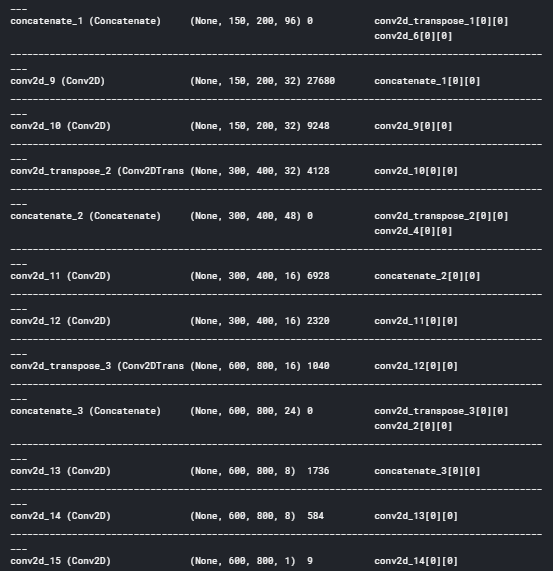


Рисунок 3.4 – Шлях розширення зображення

Щоб отримати кращі точні місця розташування, на кожному кроці декодера ми використовуємо пропущені з'єднання, конкатенуючи вихід перекладених шарів згортки з картами ознак з кодера на одному рівні. Після кожної конкатенації ми знову застосовуємо дві послідовні регулярні згортки, щоб модель могла навчитися збирати більш точний вихід. Саме це дає архітектурі симетричну U-форму, звідси і назва U-Net.

На високому рівні ми маємо наступне співвідношення:

Вхід (600x800x3) => Encoder => (75x100x64) => Декодер => Вихід (600x800x1).

## **3.4 Навчання мережі**

Модель компілюється з оптимізатором Adam і використовує функцію бінарної крос-ентропії.

Зворотний виклик Keras для реалізації:

* Зниження швидкості навчання, якщо втрата валідації не покращилася протягом 5 епох.
* Ранні зупинки, якщо втрата валідації не покращиться протягом 10 епох.
* Збереження ваги лише в разі покращення втрати валідації.

Зауважимо, що може бути багато можливостей для налаштування цих гіпер параметрів і подальшого поліпшення продуктивності моделі.

Модель навчається на графічному процесорі P4000 і займає менше 20 хвилин для навчання.

## **3.5 Результати**

Обмежившись за допомогою RAM (Random Access Memory), виконуємо лише сегментацію дороги. Похибка алгоритму сегментації становить приблизно 0.0348, а це ознає що сегментація має високу точність.

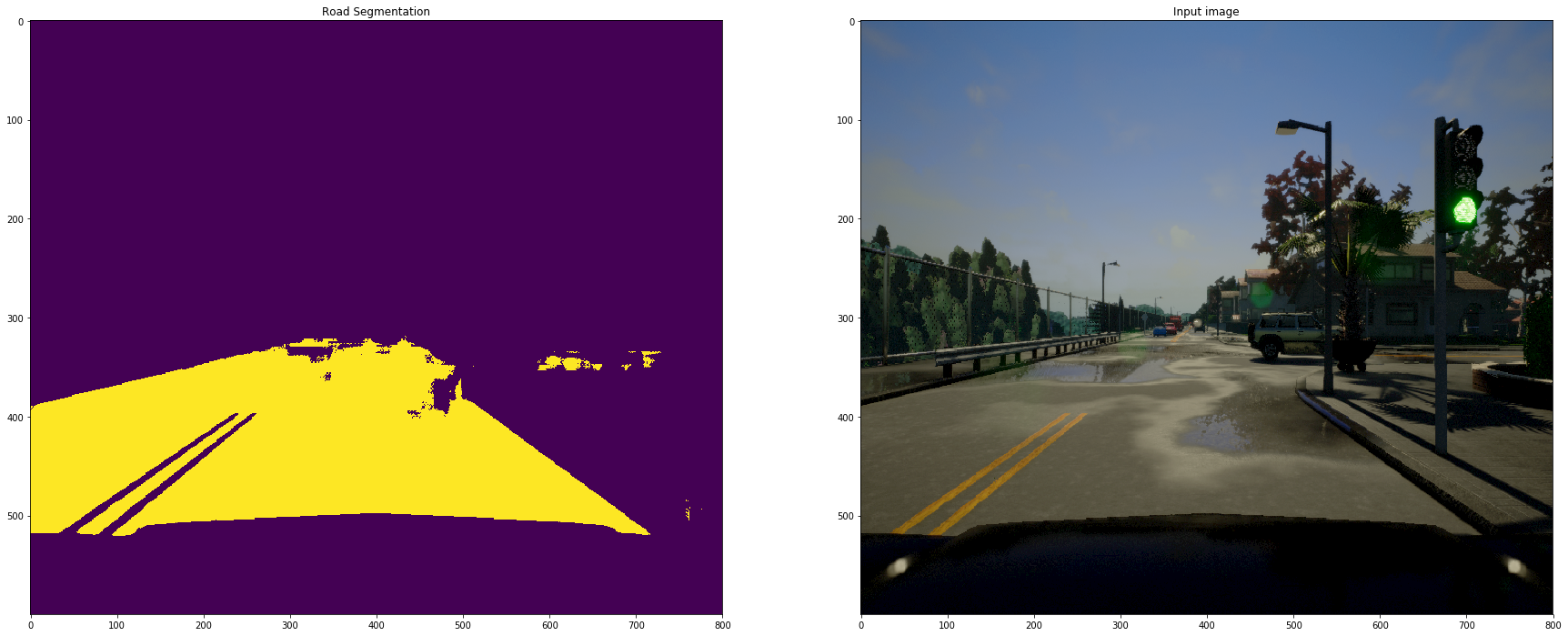


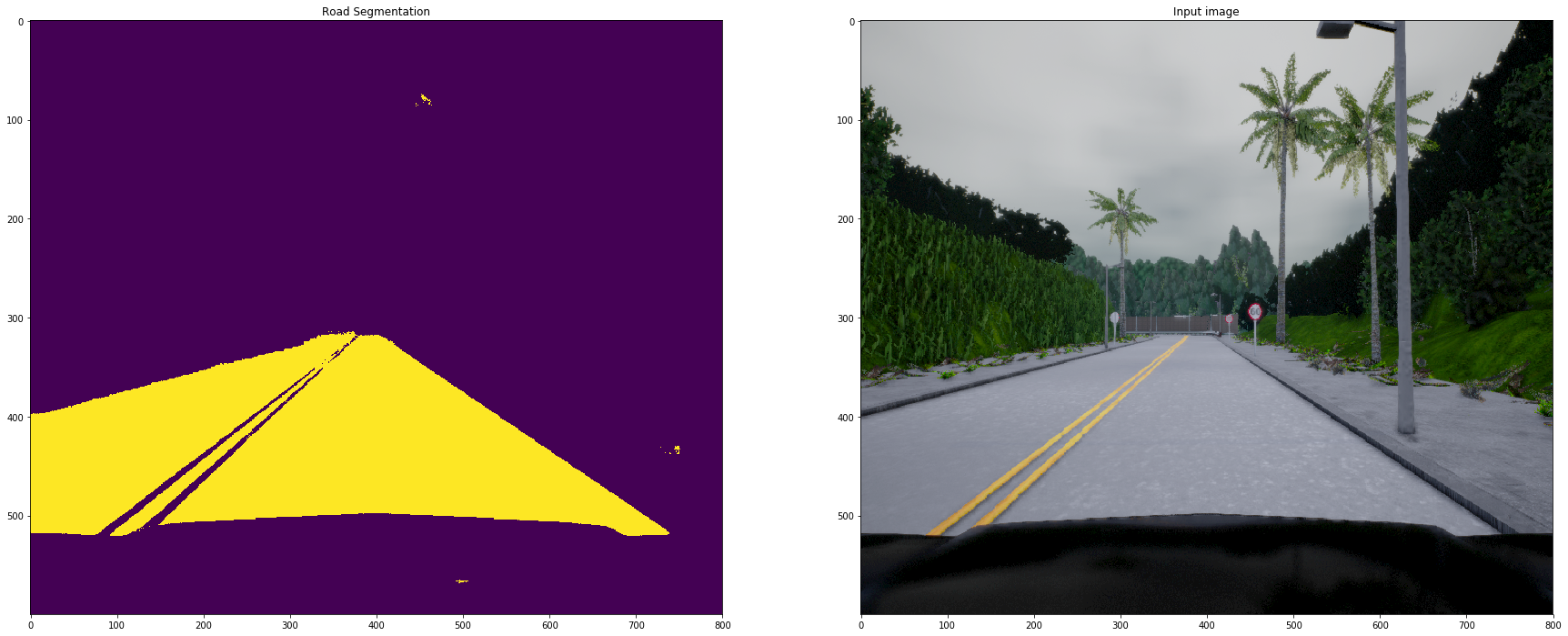
Рисунок 3.5 – Обмеження для сегментування дороги

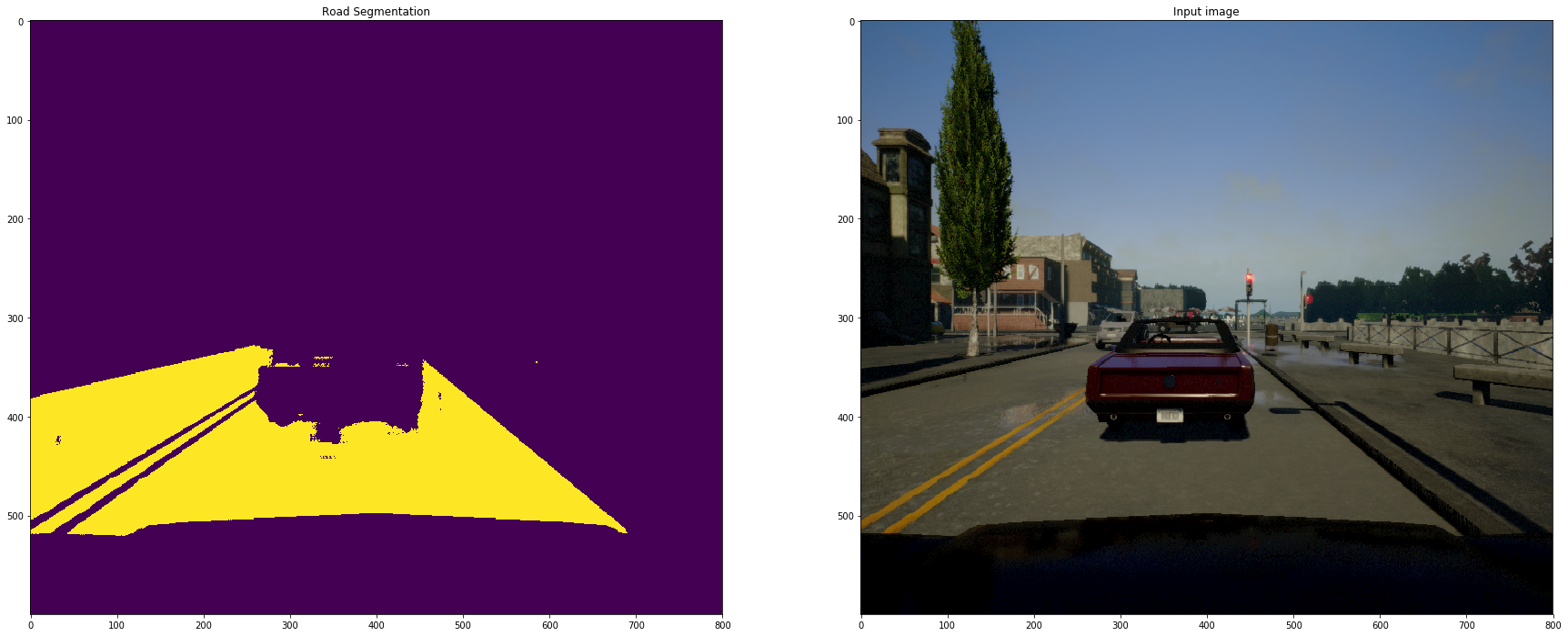
Вивід нейронної мережі виглядає таким чином:

Сегметація дороги Початкове зображення





****

****

# 

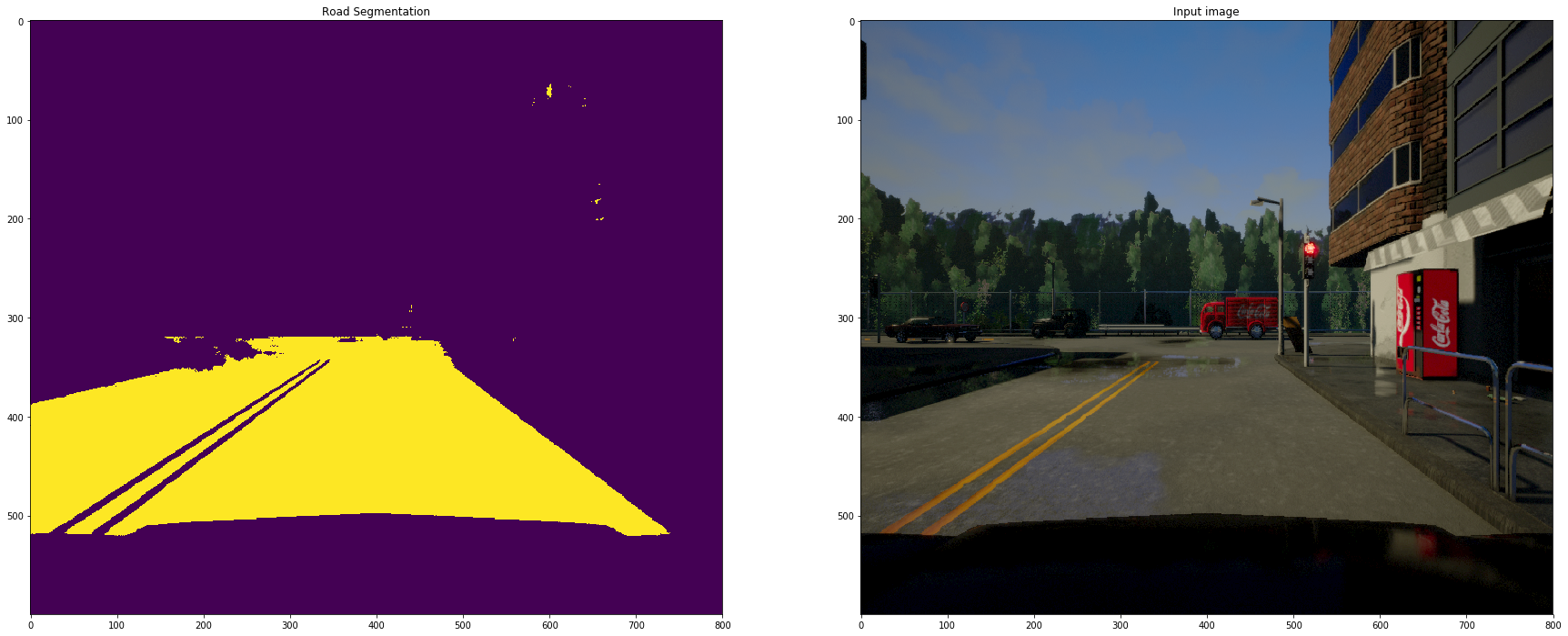


Рисунок 3.6 – Результати сегментації

# ВИСНОВОК

ПІд час виконання дипломної роботи була дослідженна спеціальна література, а саме розглянуто найкращі наукові роботи та результати конференції по нейронних мереж.

В результаті виконання дипломної роботи було досліджено фреймворки Tensorflow і Keras, різні архітектуру згорткових нейронних мереж, зокрема U-Net архітектуру які були необхідні для реалізації практичної частини.

В межах практичної частини дипломної роботи було створено згорткову нейронну мережу, завданням якої є розпізнавання дороги на фотознімках.

Створена згорткова нейронна мережа для семантичної сегментації може зробити автономне водіння простішим і значно безпечнішим, як пасажирів, так і для пішоходів.

Під час створення нейронної мережі було вирішено наступні задачі:

1. підготовка набору даних CARLA Lyft Udacity Challenge
2. анотація даних (маски зображень)
3. розробка U-Net архітектури нейронної мережі
4. тренування нейронної мережі
5. аналіз результатів
6. модифікація нейронної мережі на основі попередніх результатів

Сфера семантичної сегментації, без сумніву, одна з найцікавіших сфер комп'ютерного зору. За кілька останніх років глибоке навчання значно розширило і спростило алгоритми семантичної сегментації і проклало шлях для більшого застосування сегментації в реальних додатках. З очевидних переваг сементичної сегментації слід виокремити наступні:

* сегментація дає глибоке розуміння як працюють візуальні системи, та які в них обмеження;
* Алгоритм є більш загальним, ніж виявлення та розпізнавання об'єктів.

Проте, існують також проблеми і недоліки, які необхідно розглянути:

* Кількість даних необхідних для навчання алгоритму. Деякі з найкращих підходів вимагають величезної кількості маркованих даних. Це означає, що в деяких ситуаціях цей алгоритм буде невиправданим;
* Обчислювальні ресурси. Деякі з найкращих методів вимагають досить великого використання майже суперкомп'ютерів для фази навчання, яка може бути недоступною у всіх контекстах;
* Точність результатів сегментації. Досягнення більш високої точності є корисним, але важливо розуміти наслідки неправильних сегментацій, особливо якщо це стосується автономного водіння.

Оскільки, технології щоразу удосконалюються - це означає що результати моєї сегментації можуть бути покращені в найближчому майбутньому.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Badrinarayanan V., Kendall A., Cipolla R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation // IEEE transactions on pattern analysis and machine intelligence. – 2017. – Vol. 39, no. 12. – P. 2481–2495.
2. CARLA open source simulator for autonomius driving research [Електонний ресурс]. – Режим доступу: <http://carla.org/>
3. Guo Y., Liu Y., Georgiou T. review of semantic segmentation using deep neural networks [Електонний ресурс]. – Режим доступу:

https://link.springer.com/article/10.1007/s13735-017-0141-z

1. Kaymak C. Brief Survey and an Application of Semantic Segmentation for Autonomous Driving [Електонний ресурс]. – Режим доступу:

https://arxiv.org/ftp/arxiv/papers/1808/1808.08413.pdf

1. Krizhevsky A., Sutskever I., Hinton Geoffrey E. ImageNet Classification with Deep Convolutional Neural Networks // Advances in Neural Information Processing Systems 25 / Ed. by F. Pereira, C. J. C. Burges, L. Bottou, K. Q. Weinberger. – Curran Associates, Inc., 2012. – P. 1097–1105.
2. Liaw A., Wiener M. et al. Classification and regression by randomForest // R news. – 2002. – Vol. 2, no. 3. – P. 18–22.
3. Raj B. A simple guide to semantic segmentation [Електонний ресурс]. –

Режим доступу: https://medium.com/beyondminds/a-simple-guide-to-semantic-segmentation-effcf83e7e54

1. Ronneberger O., Fischer P., Brox T. U-Net: Convolutional Networks for Biomedical Image Segmentation // CoRR. – 2015. – Vol. abs/1505.04597. – [1505.04597.](http://arxiv.org/abs/1505.04597)
2. Satellite Image Segmentation: a Workflow with U-Net [Електонний ресурс]. – Режим доступу: <https://medium.com/vooban-ai/satellite-image-segmentation-a-workflow-with-u-net-7ff992b2a56e>
3. Shotton J., Johnson M., Cipolla R. Semantic texton forests for image categorization and segmentation // Computer vision and pattern recognition, 2008. CVPR 2008. IEEE Conference on / IEEE. – 2008. – P. 1–8.
4. Thoma M. A survey of semantic segmentation // arXiv preprint arXiv:1602.06541. – 2016.
5. Vapnik V. The Support Vector method // Artificial Neural Networks – ICANN’97 / Ed. by W. Gerstner, A. Germond, M. Hasler, J.-D. Nicoud. – Berlin, Heidelberg : Springer Berlin Heidelberg, 1997. – P. 261–271.