

ЗМІСТ

Вступ.....	2
1. Постановка задачі	3
2. Машинне навчання	4
2.1 Класичне машинне навчання	5
2.1.1 Навчання з вчителем.....	6
2.1.2 Навчання без вчителя	7
2.1.3 Ансамблі класифікаторів.....	8
2.1.4. Глибоке навчання.....	10
3. Штучні нейронні мережі	11
3.1 Функція активації ReLU.....	12
3.2 Глибокі навчальні програми та бібліотеки	13
3.3 Нейромережеві методи.....	13
4. Згорткова нейронна мережа	14
4.1 Модельні архітектури.....	21
4.1.1. Fully Convolutional Network	21
4.1.2. SegNet.....	22
4.1.1. U-net.....	22
5. Програмна реалізація	24
5.1 Огляд набору даних.....	24
5.2 Інструменти та технології.....	25
5.3 Побудова архітектури згорткової нейронної мережі	25
5.4 Навчання мережі.....	26
5.5 Результати	26
ВИСНОВКИ.....	29
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	30

Вступ

Машинне навчання – широка і динамічна область досліджень, яка використовує велику кількість теоретичних і практичних методів. Є три основні причини, що підтверджують необхідність машинного навчання, коли недостатньо просто запрограмувати комп'ютер. По-перше, розробники не можуть передбачити усі ситуації, в яких опиниться машина. Наприклад, робот, розроблений для проходження лабіринтів, мусить дізнаватися план кожного нового лабіринту, який йому трапляється. По-друге, розробники не можуть передбачити усі зміни з часом: програма для прогнозування цін на фондовій біржі мусить навчатися для того, щоб могли адаптуватися, коли умови на ринку змінюються. По-третє, іноді люди не мають жодного уявлення про те, як запрограмувати розв'язок. Наприклад, більшість людей добре розпізнають обличчя своїх знайомих, але навіть найкращі програмісти не можуть створити програму для виконання цієї задачі, не використовуючи алгоритми навчання.

Історія машинного навчання розпочалася в 1950 роки, коли інформатикам вдалося навчити комп'ютер грати в шашки. З розвитком комп'ютерної техніки зростала і складність обчислювальної потужності, а, отже, складність закономірностей і передбачень, які здатний розпізнати комп'ютер.

Машинне навчання в наш час переживає нове народження і поступово входить в усі галузі людської діяльності, починаючи з рекомендованої реклами, закінчуючи управлінням транспорту.

Сегментація зображення - це широко використовуваний метод цифрової обробки і аналізу зображень для поділу зображення на кілька частин або областей, часто заснований на характеристиках пікселів в зображенні. Сегментація зображення може включати відділення переднього плану від фону або кластеризацію областей пікселів на основі подібності кольору або форми. Люди можуть виконувати сегментацію зображення, не знаючи, що таке об'єкти. Виконання сегментації без знання 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.

1. Постановка задачі

Розробити систему сегментації зображення, побудова вимагатиме вирішення наступних задач :

- 1) підготувати набір даних;
- 2) розробити нейронну мережу;
- 3) натренувати нейронну мережу на підготованому наборі даних;
- 4) проаналізувати отримані результати.

2. Машинне навчання

Мета машинного навчання - передбачити результат за вхідними даними. Чим різноманітніші вхідні дані, тим простіше машині знайти закономірності і тим точніший результат.

Отже, якщо ми хочемо навчити машину, нам потрібні три речі:

- Велика кількість даних

Щоб виявляти спам, необхідно зібрати якомога більше прикладів спам-листів; щоб передбачати курси акцій - потрібна історія цін; дізнатися інтереси користувача - потрібні його лайки або пости. Даних потрібно якомога більше.

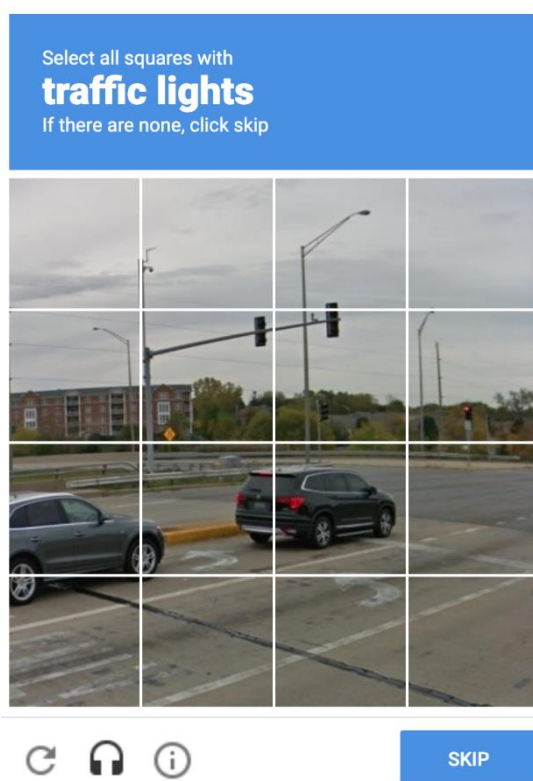


Рис.2.1

Дані збирають, як тільки можуть. Хтось збирає вручну. Хоч процес триваліший і даних менше, однак без помилок. Хтось автоматично - просто зливає машині все, що знайшлося. Великі компанії (Google, Facebook) використовують своїх же користувачів для безкоштовної розмітки (ReCaptcha, яка просить користувача знайти на фотографії всі дорожні знаки (рис.2.1)).

За хорошими наборами даних (датасетами) йде велике полювання.

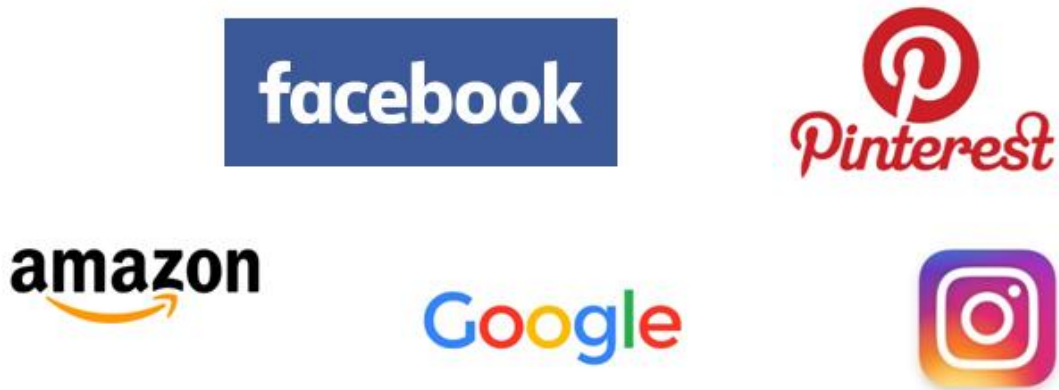


Рис.2.2

Великі компанії інколи розкривають свої алгоритми, але датасети - вкрай рідко.

- Ознаки

Ми називаємо їх властивостями, характеристиками - ними можуть бути пробіг автомобіля, стать користувача, ціна акцій, навіть лічильник частоти появи слова в тексті. Машині потрібно знати, на що їй конкретно дивитися.

- Алгоритм

Зазвичай, одну й ту ж задачу майже завжди можна розв'язати різними методами-способами. Від вибору методу залежить точність, швидкість роботи і розмір готової моделі. Але є один нюанс: якщо дані - «сміття», то навіть найкращий алгоритм не допоможе. Тому DataScience-спеціалісти радять не зациклюватися на відсотках, а краще збирати побільше різноманітних даних.

2.1 Класичне машинне навчання

Перші алгоритми були винайдені в 50-х роках минулого століття з чистої статистики. Вони шукали закономірності в числах, оцінювали близькість точок в просторі і вираховували напрямки.

Класичне навчання можна поділити на дві групи: навчання з вчителем і без вчителя. Якщо у першому випадку у машини є «хтось», хто повідомляє, як правильно вважати, тобто «вчитель» вже заздалегідь знає всі властивості даних, і машина навчається на конкретних прикладах, то у випадку навчання без вчителя машині «згодовують» велику кількість нерозмічених даних, і вона намагається сама знайти будь-які закономірності.

2.1.1 Навчання з вчителем

Навчання з вчителем (Supervised Learning) ділиться на два типи: класифікація (передбачення категорії об'єкта), і регресія (передбачення місця на числовій прямій).

Сьогодні класифікацію використовують для:

- Спам-фільтрів;
- Визначення мови;
- Пошук схожих документів;
- Аналіз тональності;
- Розпізнавання рукописних букв і цифр;
- Визначення підозрілих транзакцій;

Популярні алгоритми:

- Наївний Баєс
- Дерева Рішень,
- Логістична Регресія,
- Метод К-найближчих сусідів,
- Метод Опорних Векторів.

Регресію використовують для:

- Прогноз вартості цінних паперів
- Аналіз попиту, обсягу продажів
- Медичні діагнози
- Будь-які залежності числа від часу

Популярні алгоритми:

- Лінійна регресія
- Логістична регресія
- Поліноміальна регресія
- Ridge регресія (гребенева регресія)
- Регресія LASSO
- ElasticNet регресія (регресія еластичної мережі)

2.1.2 Навчання без вчителя

Навчання без вчителя (Unsupervised Learning) було винайдено пізніше, аж у 90-ті, і на практиці використовується рідше. Але бувають задачі, де просто немає вибору.

Хоча розмічені дані – це дорога рідкість, однак навчання без вчителя частіше використовують, як метод аналізу даних, а не як основний алгоритм.

Цікавий тип навчання без вчителя – кластеризація. Цей тип розділяє об'єкти за невідомою ознакою. Машина сама вирішує, як краще. Кластеризація - це класифікація, але без заздалегідь відомих класів. Вона сама шукає схожі об'єкти та об'єднує їх в кластери. Кількість кластерів можна задати заздалегідь або довірити це машині. Кластеризацію використовують для:

- Сегментації ринку (типів покупців, лояльності);
- Об'єднання близьких точок на карті;
- Стиснення зображень;
- Аналіз і розмітки нових даних;
- Детектори аномальної поведінки.

Популярні алгоритми:

- 1) Метод К-середніх;
- 2) Mean-Shift;
- 3) DBSCAN

Також виділяють тип - Зменшення Розмірності (Узагальнення). Такий тип збирає конкретні ознаки в абстракції вищого рівня. Його використовують для:

- Рекомендаційних систем;
- Красивих візуалізацій;
- Визначення тематики та пошуку схожих документів;
- Аналіз фейкових зображень;
- Ризик-менеджмент.

Алгоритми цього типу:

- 1) Метод головних компонент (PCA);
- 2) Сингулярне розкладання (SVD);
- 3) Латентне розміщення Діріхле (LDA);
- 4) Латентно-семантичний аналіз (LSA, pLSA, GLSA), t-SNE (для візуалізації)

Сьогодні також використовують такий тип навчання без вчителя, як пошук правил (асоціація). Його використовують для:

- Аналізу товарів, які купують разом;
- Прогнозу акцій і розпродажів;
- Розташування товарів на полицях;
- Аналізу патернів поведінки на веб-сайтах.

Сюди входять всі методи аналізу продуктових кошиків, стратегій маркетингу інших послідовностей. Алгоритми цього типу:

- 1) Apriori;
- 2) Euclat;
- 3) FP-growth

Більш складніший тип навчання без вчителя – навчання з підкріпленням. Його застосовують там, де задача полягає не в аналізі даних, а у виживанні в реальному середовищі. Це використовується для:

- Автопілотів автомобілів
- Роботів-порохотягів
- Ігор
- Автоматизованої торгівлі
- Управління ресурсами підприємств

Популярні алгоритми:

- 1) Q-Learning;
- 2) SARSA;
- 3) DQN;
- 4) A3C,
- 5) Генетичний Алгоритм

2.1.3 Ансамблі класифікаторів

Ансамбль класифікаторів – це множина класифікаторів, рішення яких комбінуються деяким чином для отримання остаточної класифікації спостереження. Ансамблі класифікаторів використовують для:

- Всього, де можна скористатися класичними алгоритмами (але працюють точніше);

- Пошукові системи;
- Розпізнавання об'єктів.

Є три перевірених способи створення ансамблів.

- Стекінг – навчаємо кілька різних алгоритмів і передаємо їх результати на вхід останньому, який приймає остаточне рішення.
- Бустинг – метод, що полягає у зважуванні спостережень навчальної вибірки. Ідея бустингу полягає в тому, що класифікатор ансамблю будується послідовно і на кожній ітерації відбувається корекція (переважування) спостережень навчальної вибірки (на першій ітерації вага всіх спостережень є рівною). Корекція здійснюється таким чином, щоб існуючий класифікатор робив менше помилок на тих спостереженнях, на яких часто робили помилки класифікатори, що побудовані на попередніх ітераціях алгоритму. Крім цього, кожному класифікатору присвоюється певна вага, яка визначається з кількості помилок при класифікації.
- Бенінг (Bootstrap AGGREGatING) – метод, що полягає в навчанні одного алгоритму багато разів на випадкових вибірках з вихідних даних. В кінці усереднюємо відповіді.

Найпопулярніший приклад бегінга - алгоритм Random Forest. Він застосовується в камері на телефоні (окреслення обличч людей в кадрі жовтими прямокутниками). Нейромережа буде занадто повільна в реальному часі, а бегінг ідеальний, адже він може обчислювати свої дерева паралельно.



Рис.2.3

2.1.4. Глибоке навчання

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

Для реалізації алгоритмів глибокого навчання потрібна дуже велика кількість даних і апаратних засобів з високою обчислювальною потужністю. 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). В даний час глибоке навчання має багато областей застосування, в основному це автоматичне розпізнавання мови, розпізнавання зображень і обробка природної мови.

Існує багато різних типів архітектур глибокого навчання.

Основні архітектури глибокого навчання:

- Глибока нейронна мережа;
- Глибинна мережа переконань;
- Згорткова нейронна мережа;
- Глибока ядрова мережа;
- Машина Больцмана;
- Автокодувальник;
- Глибока мережа кодування.

3. Штучні нейронні мережі

Нервова система людини є дуже складною, і науковці ще й досі намагаються досягнути її природу. Відомо, що нервова система побудована з нейронів (рис.3.1), де кожен нейрон володіє такими унікальними здатностями, як прийом, обробка і передача електрохімічних сигналів по нервових шляхах, які утворюють комунікаційну систему мозку.

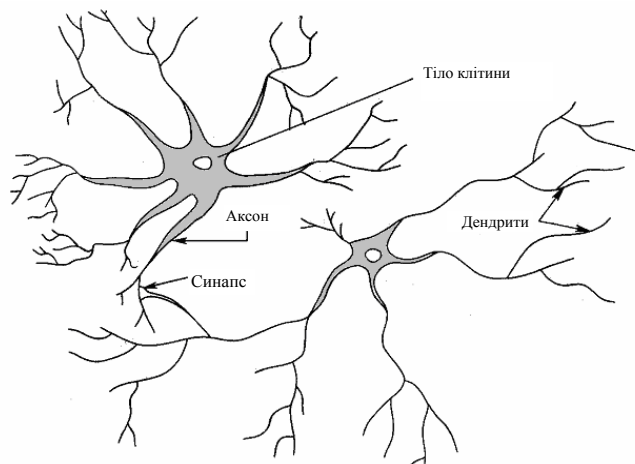


Рис. 3.1 – Біологічний нейрон

ШНМ були розроблені у процесу вивчення мозку людини. Коли нейрони біологічної нервової системи з'єднуються один з одним, структури, визначені як штучні нейрони в системах ШНМ, пов'язуються один з одним. Нейрон можна назвати базовою обчислювальною одиницею в ШНМ. Структура штучного нейрона показана на рис.3.2.

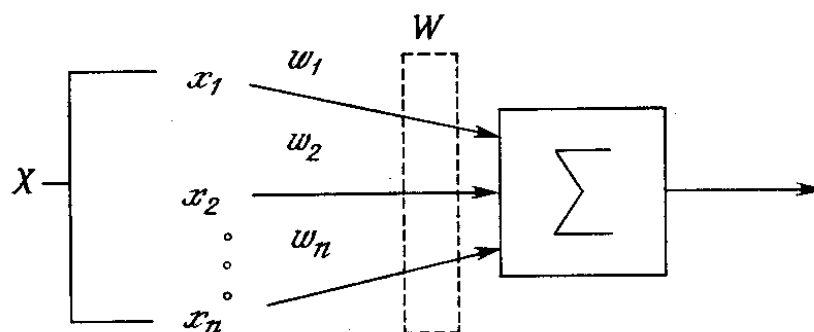


Рис 3.2 – Структура штучного нейрона

Будь-яка нейромережа - це набір нейронів і зв'язків між ними. Нейрон найкраще уявляти собі просто, як функцію з кількома входами і одним виходом. Завдання нейрона - взяти значення зі своїх входів, виконати над ними функцію і віддати

результат на вихід. Простий приклад корисного нейрона: знайти суму всіх цифр зі входів, і якщо їх сума більше N - видати на вихід одиницю, інакше - нуль.

Зв'язки - це канали, через які нейрони шлють один одному цифри. Кожен зв'язок має свою вагу - її єдиний параметр, який можна умовно уявити як міцність зв'язку. Коли через зв'язок з вагою 0,5 проходить число 10, воно перетворюється в 5. Сам нейрон не розбирається, що до нього прийшло і сумує все підряд - ось ваги й потрібні, щоб керувати, на які входи нейрон повинен реагувати, а на які - ні.

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 в інший нейрон. Крім того, 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.

$$y = f \left(\sum_{i=1}^n W_i x_i + b \right)$$

3.1 Функція активації ReLU

Існує безліч функцій активації, таких як лінійна, ступінчаста, сигмоподібна, гіперболічна дотична (tanh), ректифікована лінійна одиниця (ReLU – рис. 3.3) і порогові функції.

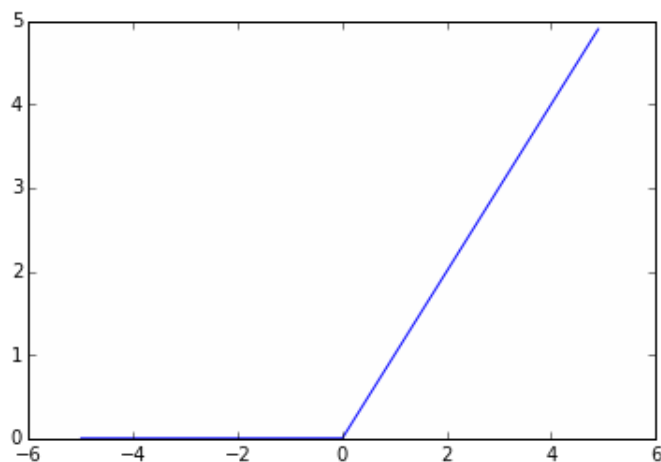


Рис.3.3

Функція активації ReLU $f(x) = \max(0, x)$, генерує висновок з порогом 0 для кожного з вхідних значень. Останнім часом ця функція є дуже популярною в ШНМ, саме її було використано при реалізації задачі.

3.2 Глибокі навчальні програми та бібліотеки

Робочий процес глибокого навчання є багатостадійним, ітеративним процесом. Для успішного глибокого навчання можуть знадобитися зібрані і попередньо оброблені, якщо це необхідно, великомасштабні набори даних. Завдяки Інтернету та великим джерелам, даних масиви даних швидко зростають. Моделі мережі, створені після етапу навчання, перевіряють на правильність роботи. 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. Для виконання цих обчислювальних процесів центральних процесорів у комп'ютері недостатньо. Оскільки процесори з певною потужністю обробки та архітектурою не можуть виконувати багато операцій одночасно, навчальні та тестові фази моделі займають багато часу. Через це віддають перевагу графічним процесорам, які дозволяють паралельну обробку даних.

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.

3.3 Нейромережеві методи

Глибоке навчання значно спростило конвеєр для виконання семантичної сегментації і дає результати високої якості.

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'ютерних систем (зокрема поява потужних GPU: навчання моделей займає адекватний час, і їх застосування можливо в реальному часі навіть на мобільних пристроях).

4. Згорткова нейронна мережа

Згорткова нейронна мережа, введена 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.

Всі нейрони в шарі в прямих штучних нейронних мережах 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, щоб 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. 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 згорткового і об'єднувального шарів передається в один або більше повністю з'єднаних шарів. Потім, прогнозування виходу отримують шляхом перенесення на шар класифікатора. Проста архітектура згорткової нейронної мережі являє собою комбінацію згорткових, об'єднуючих і повністю пов'язаних шарів (рис.4.1)

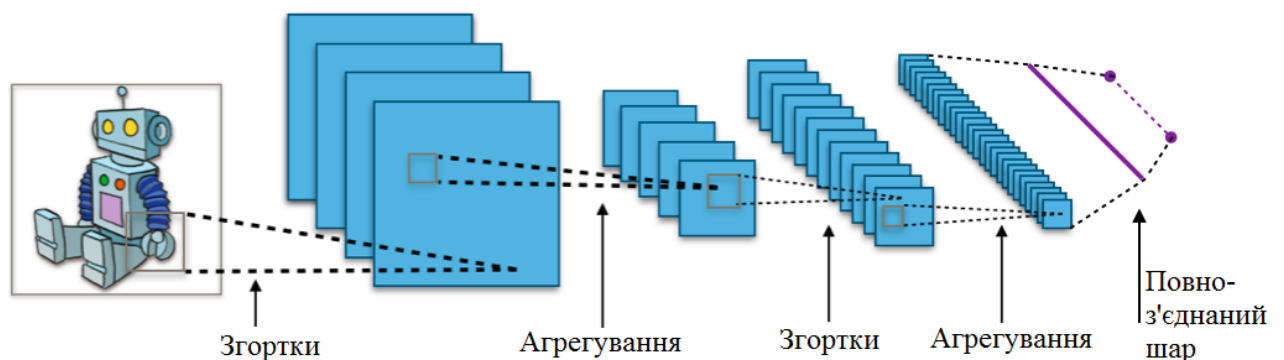


Рисунок 4.1 – Приклад простої архітектури згорткової нейронної мережі

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 (рис. 4.2).

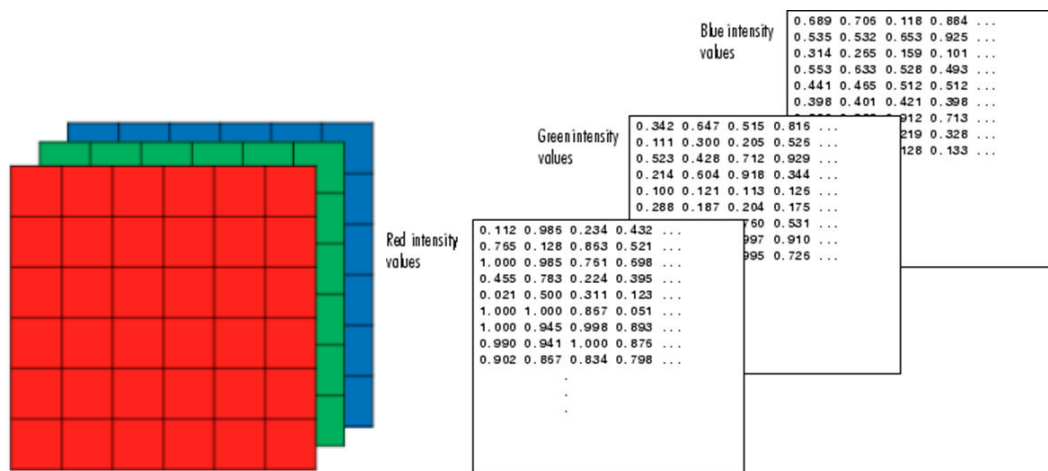


Рис.4.2

Кожен шар згорткової мережі складається з набору просторових фільтрів розміром $d \times h \times w$, які являють собою просторові розміри h і w , які мають об'єм нейронів і кількість ядер (або фільтрів) каналів функції d . Кожен з цих фільтрів піддається згортці з відповідним обсягом вхідного зображення, і ковзає по всьому зображенню (розміром $D_1 \times H_1 \times W_1$, де H_1 , W_1 - просторові розміри і D_1 - номер каналу) по всіх його просторовим вимірам H_1 , W_1 . Згортка відноситься до суми елемента множення елементів нейронів у кожному фільтрі з відповідними значеннями на вході. Таким чином, можна припустити, що першим шаром в згорткової 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 певним чином для отримання виходу згорткового шару.

Один нейрон у фільтрі певного шару може бути відображений у зв'язані нейрони у всіх попередніх шарах, слідує 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 сегмент вхідного зображення. Таким чином, ієрархія (карта, мапа) ознак генерується від локального до глобального.

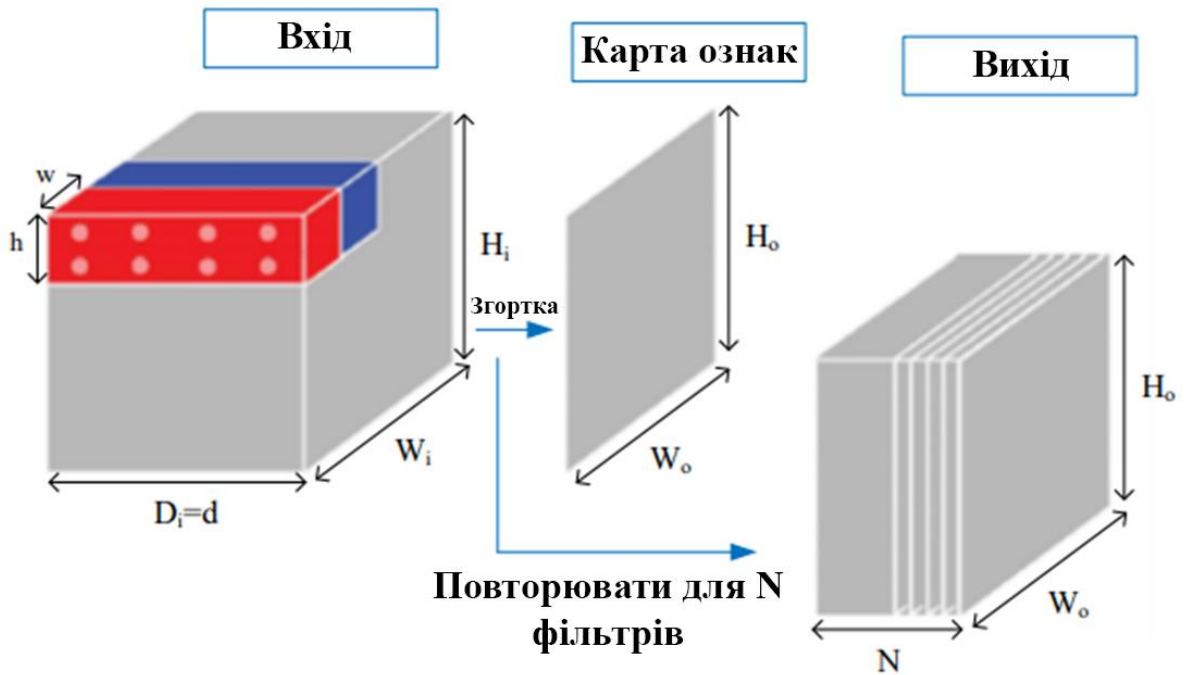


Рисунок 4.3 – Ілюстрація одного згорткового шару

Червоні і сині області на рисунку 4.3. представляють дві позиції однакових фільтрів розміром $d \times h \times w$, 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 $D_1 = d = 3$.

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).

$$D_0 = N \quad (1)$$

$$H_0 = \frac{H_i - f + 2p}{s} + 1 \quad (2)$$

$$W_0 = \frac{W_i - f + 2p}{s} + 1 \quad (3)$$

На рисунок 4.4 показано фільтр 3×3 , який ковзає по матриці зображення 5×5 , що представляє бінарне зображення і кінцевий стан карти об'єкта. 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.

Фільтр зображення

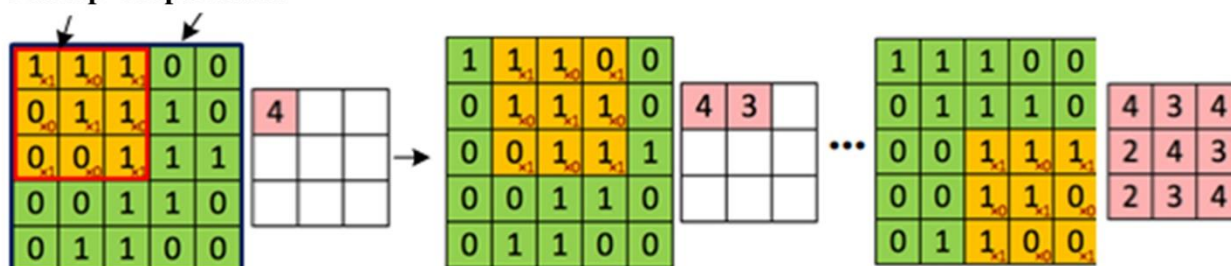


Рис.4.4

Згортка зображення з різними фільтрами може виконувати такі операції, як виявлення країв, розмиття і різкість із застосуванням фільтрів. У наведеному нижче прикладі показані різні зображення згортки після застосування різних типів фільтрів (ядер). Нижче на рис. 4.5 наведені приклади різних фільтрів.



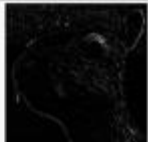
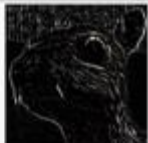



Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Рис.4.5

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. Просторове об'єднання може бути різних типів: Макс, Середнє, Сума і т.д. 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, роблять 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 рис.4.6 (приклад підвибірки) видно, що вхідний об'єм $64 \times 224 \times 224$ підсумовується до обсягу $64 \times 112 \times 112$ 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.

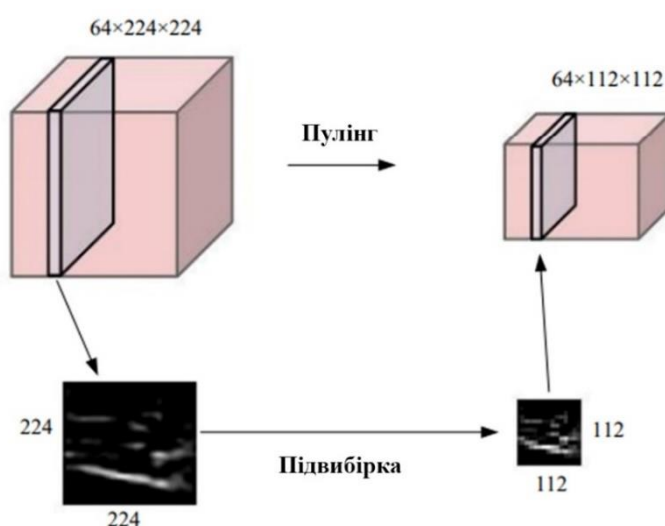


Рис.4.6

На рис.4.7 показана операція об'єднання, що виконується з максимальною функцією об'єднання, шляхом переміщення фільтра 2×2 , шару 2.

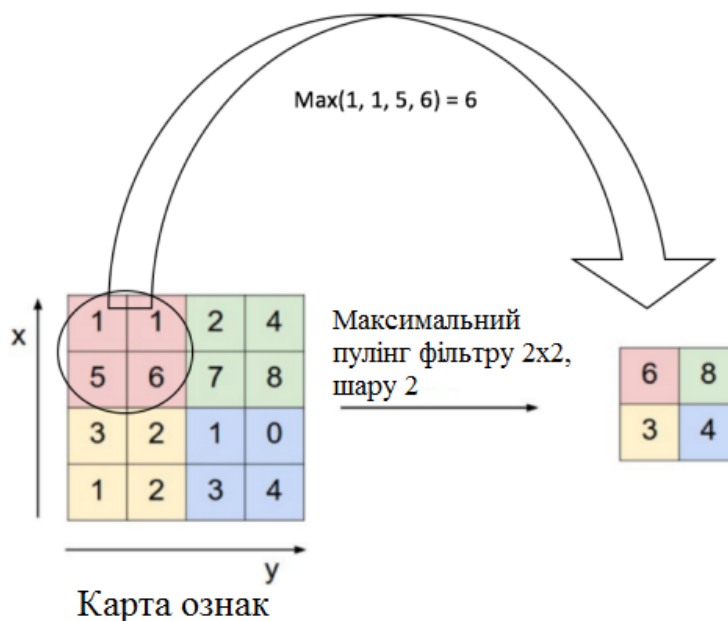


Рисунок 4.7 – Операція об'єднання з фільтром 2×2 , крок 2

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 найбільш ефективна у згорткових мережах.

ReLU негативні входи стають нулями, позитивні входи, як описано в (7), передаються без змін.

$$f(x) = \begin{cases} x, & x \geq 0 \\ 0, & \text{інші} \end{cases} \quad (7)$$

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

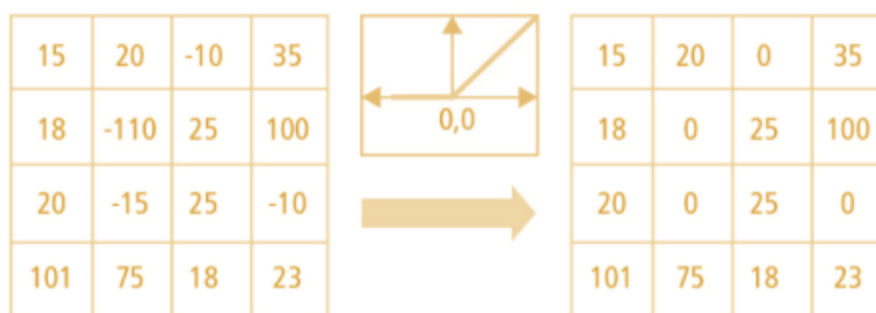


Рис.4.7

У шарі ReLU операція виконується окремо для кожного значення пікселя. Наприклад, вихід ReLU є таким, як показано на рисунку 4.7.

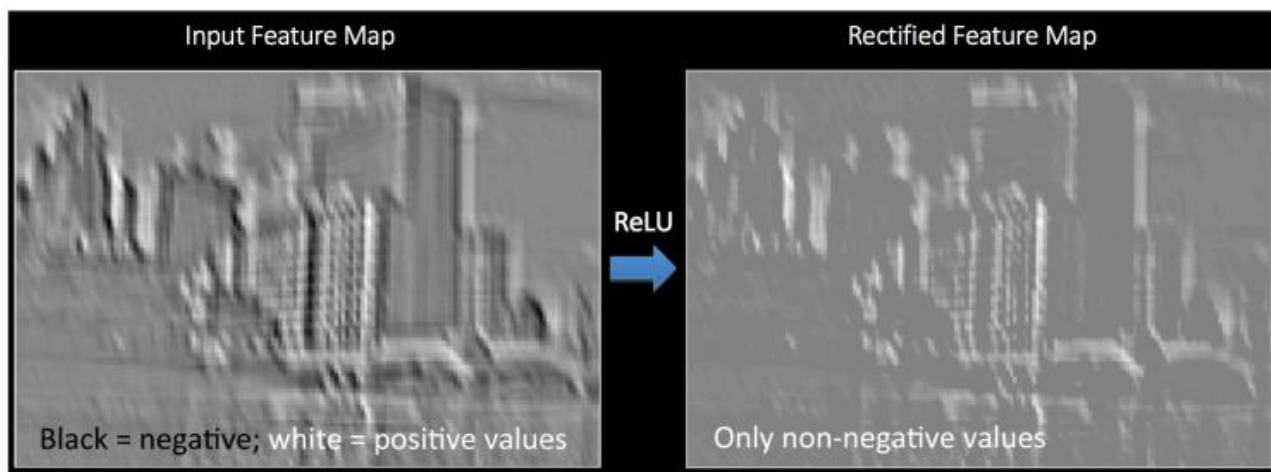


Рисунок 4.8 – Приклад операції ReLU

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, 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. Повністю пов'язаний шар можна розглядати як шар, який подає класифікатор.

4.1 Модельні архітектури

4.1.1. Fully Convolutional Network

Однією з найпростіших і найпопулярніших архітектур, що використовуються для семантичного сегментації, є повністю згортова 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 білінійну інтерполяцію, або через серію транспонованих згорток. Цей набір транспонованих згорток зазвичай називається декодером.

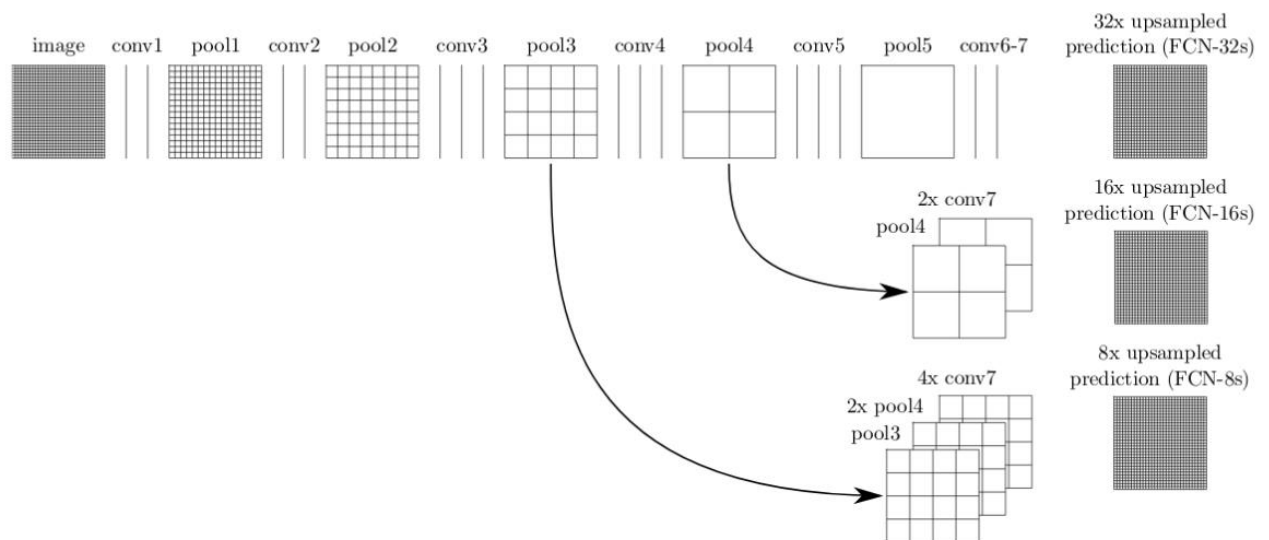


Рисунок 4.9 – FCN архітектура.

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).

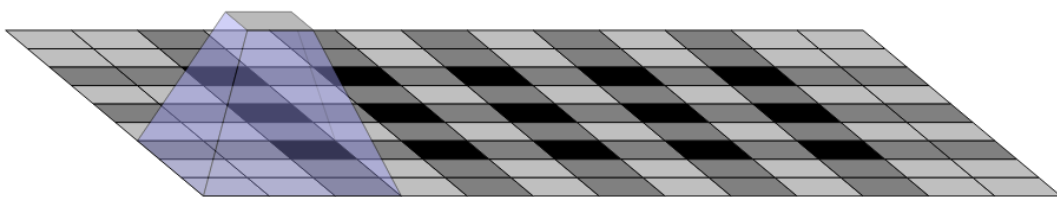


Рис. 4.10 – Формування артефактів шахової дошки.

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

4.1.2 SegNet

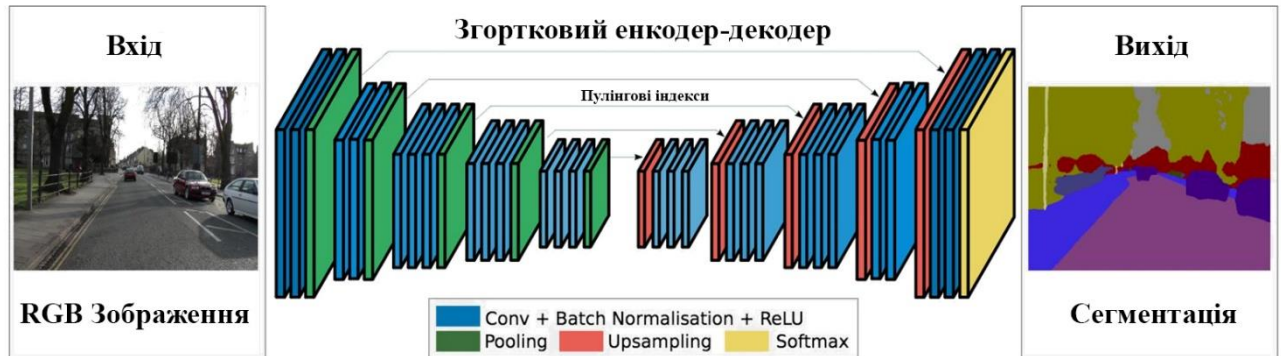


Рисунок 4.11 – Архітектура SegNet

Модель SegNet є типовим автокодером, заснованим на згортковій нейронній мережі. Схема цієї моделі представлена на рис.2.11. Мережа складається з блоків, в кожному блоці присутні згортки і пулінг (субдискретизуючі шари) або шари, що підвищують дискретизацію (апсемплінг шари), а також активаційні шари Relu і шари нормалізації батчнормалізації (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).

4.1.1 U-net

U-net - архітектура є одним з яскравих представників сегментаційних енкодер - декодер моделей. Вперше заявлена, як мережа для сегментації біологічних знімків, вона добре зарекомендувала себе, як основа при вирішенні практичних завдань сегментації. Саме цю архітектуру використано для реалізації задачі.

Переваги використання U-Net:

- Обчислювально ефективна

- Тренування з невеликим набором даних
- Навчається наскрізь

Архітектура мережі показана на рис.4.12 і містить дві частини, де одна звужується (енкодер), а інша - розширюється (декодер). Перша являє собою типову архітектуру згорткової класифікаційної мережі. Складається з повторюваних застосувань двох згорток 3x3, за якими слідує 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), 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 шарів.

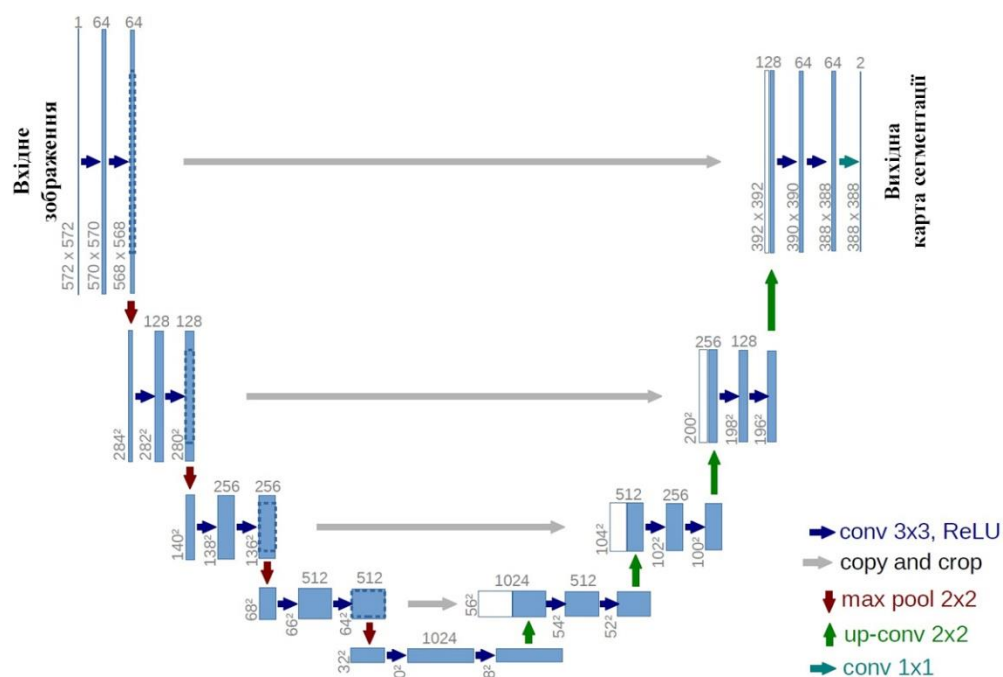


Рисунок 2.12 – Архітектура U-Net (приклад для 32x32 пікселів у низькому рівні)

5. ПРОГРАМНА РЕАЛІЗАЦІЯ

5.1 Огляд набору даних

Набір даних який надає зображення даних і позначені семантичними сегментам зображення, зафіксовані за допомогою автосимулятора CARLA. 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.



Рис. 5.1-4 – Приклади зображення з Lyft Udacity Challenge

5.2 Інструменти та технології

Для реалізації задачі використано мову програмування Python, оскільки ця мова програмування є значно простішою ніж C# чи C++, легко обробляє велику кількість даних і представлена багатьма корисними бібліотеками, особливо для аналітики, та згорткову нейронну мережу U-Net.

Для побудови 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 - 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 використовувати TensorFlow або Theano у фоновому режимі. Keras є дуже корисним, оскільки спрощує інтерфейс бібліотек TensorFlow і Theano. Keras має дуже поширене використання в додатках для обробки зображень.

5.3 Побудова архітектури згорткової нейронної мережі

Нейронна мережа побудована за принципом згорткової U-net. Спершу нейронна мережа проходить шлях скорочення - енкодер, де застосовуються регулярні згортки і макс пулінгу.

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

$$600 \times 800 \times 3 \rightarrow 300 \times 300 \times 8 \rightarrow 150 \times 200 \times 16 \rightarrow 75 \times 100 \times 32 \rightarrow 75 \times 100 \times 64$$

Це означає, що мережа дізнається інформацію "WHAT" на зображенні, однак вона втратила інформацію "WHERE". Наступний крок - шлях розширення декодер, де застосовуються транспоновані згортки разом з регулярними згортками.

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

застосовуючи вибірку. 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).

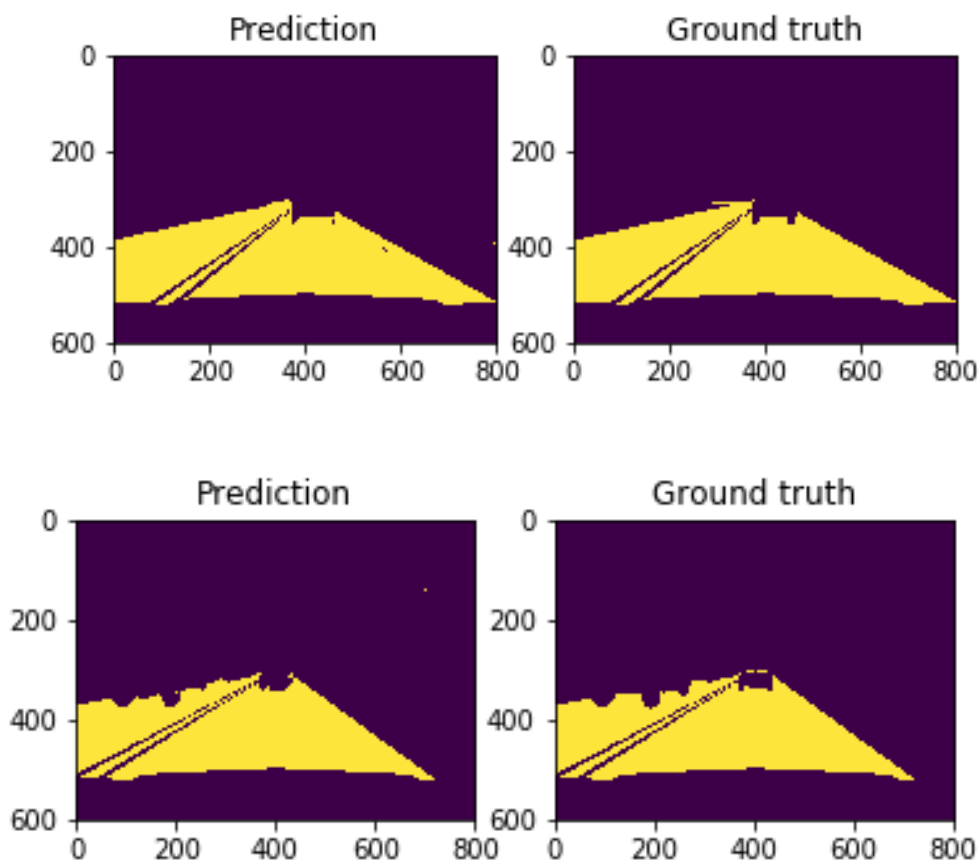
5.4 Навчання мережі

Модель компілюється з оптимізатором Adam і використовує функцію бінарної крос-ентропії. Модель навчається на графічному процесорі, а навчання займає менше години.

5.5 Результати

Виконано сегментацію дороги. Похибка алгоритму сегментації становить приблизно 0.0152, а це означає, що сегментація має високу точність.

Отримано такі результати:



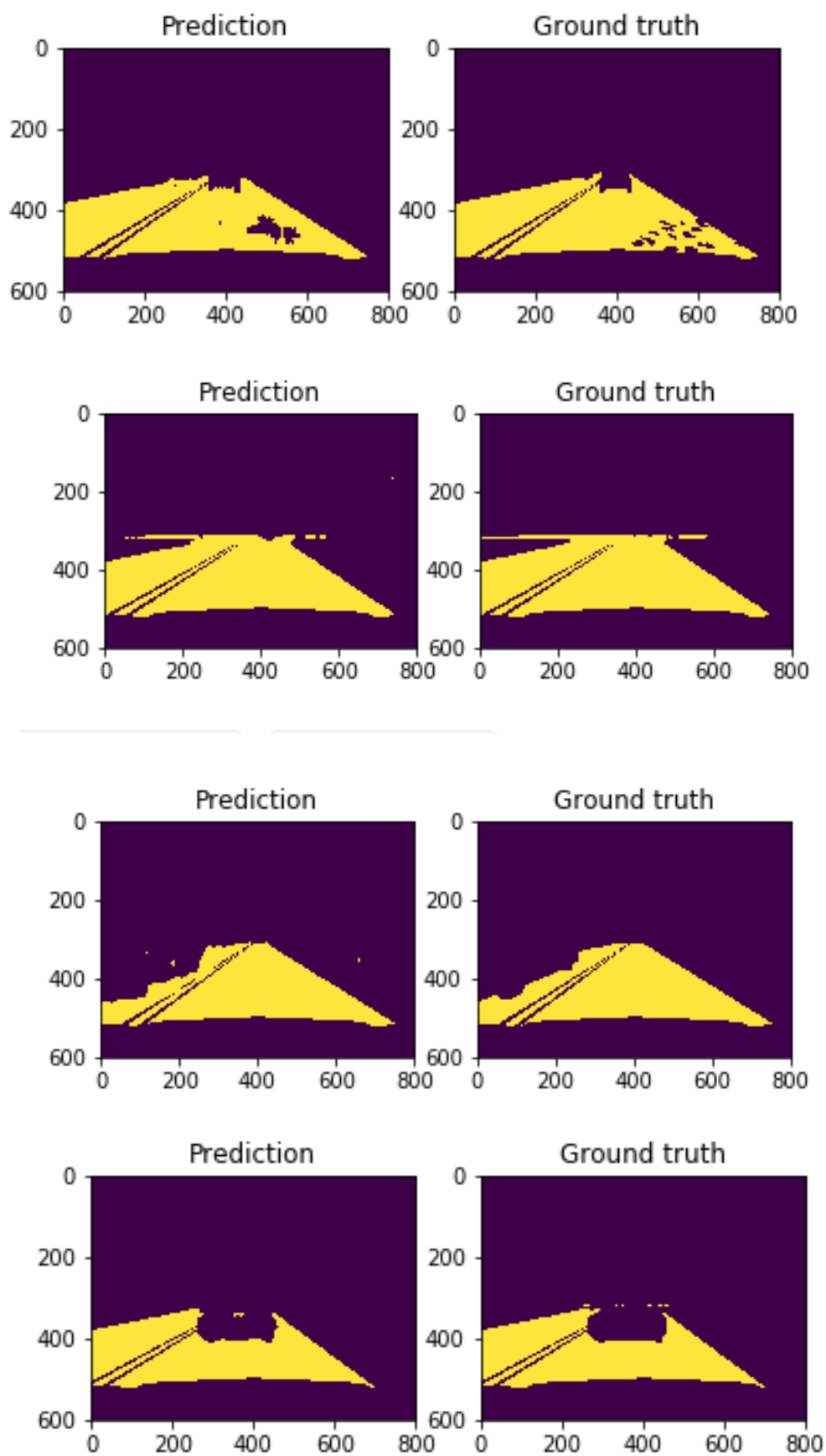


Рис.5.5-10

Сегментація машин:

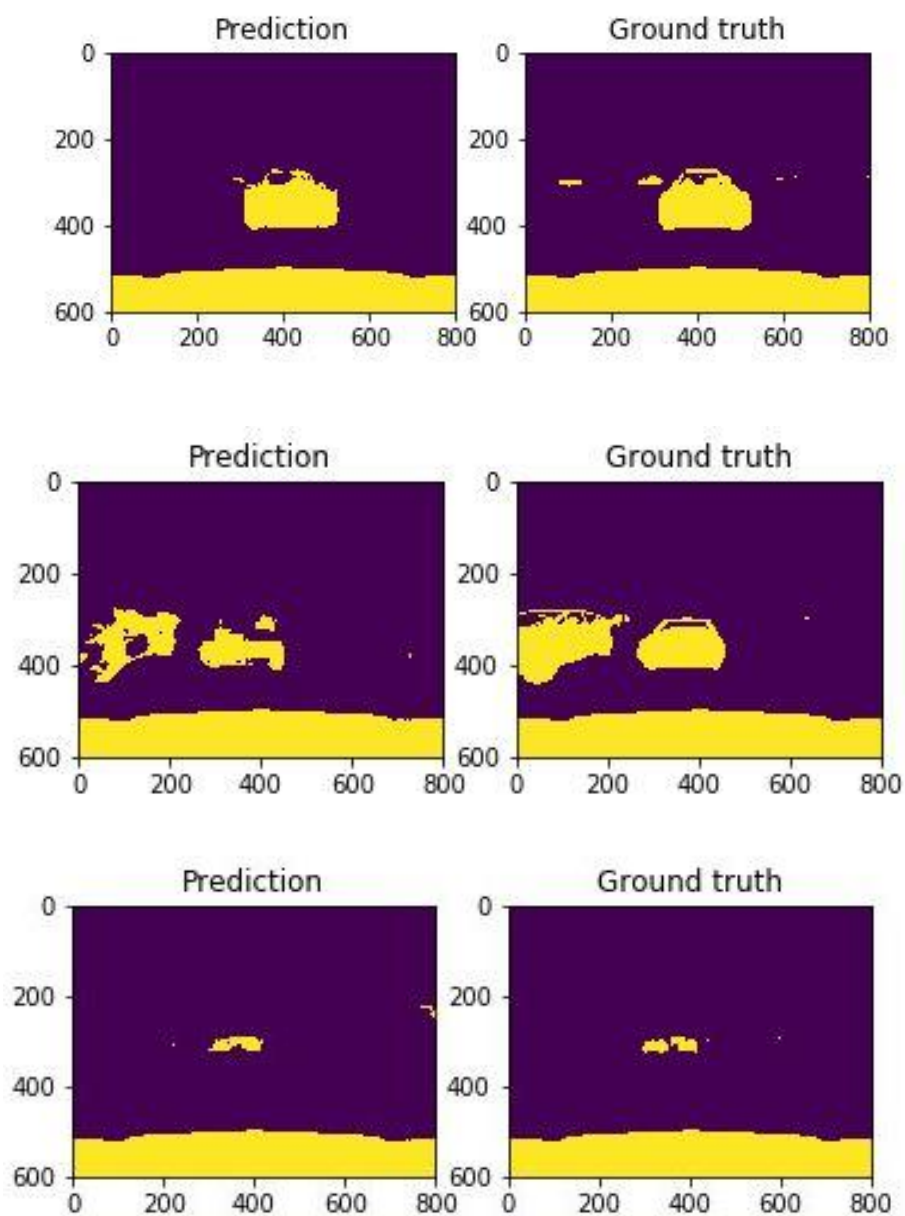


Рис.5.13

ВИСНОВКИ

Під час виконання дипломної роботи було досліджено наукові роботи і конференції по нейронним мережам, було досліджено фреймворки Tensorflow і Keras, різні архітектури згорткових нейронних мереж, зокрема U-Net архітектуру, яка були необхідна для реалізації практичної частини.

Було створено згорткову нейронну мережу, завданням якої є розпізнавання типових об'єктів у місті.

Створена згорткова нейронна мережа для семантичної сегментації може застосовуватись, як вирішення проблеми «автопілот» для машин. Чим краще натренована мережа, тим простіше і значно безпечніше у навколишньому середовищі.

Під час створення нейронної мережі було вирішено наступні задачі:

1. підготовка набору даних CARLA Lyft Udacity Challenge
2. розробка U-Net архітектури нейронної мережі
3. тренування нейронної мережі
4. аналіз результатів

З очевидних переваг семантичної сегментації слід виокремити наступні:

- сегментація дає глибоке розуміння як працюють візуальні системи, та які в них обмеження;
- алгоритм є більш ширшим, аніж виявлення та розпізнавання об'єктів.

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

- Кількість даних необхідних для навчання алгоритму.
- Точність результатів сегментації. Досягнення більш високої точності є корисним, але важливо розуміти наслідки неправильних сегментацій, особливо якщо це стосується автономного водіння.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Jonathan Long, Evan Shelhamer, Trevor Darrell - Fully Convolutional Networks for Semantic Segmentation [Електронний ресурс]. – Режим доступу: <https://arxiv.org/pdf/1411.4038.pdf>
2. Olaf Ronneberger, Philipp Fischer, and Thomas Brox - U-Net: Convolutional Networks for Biomedical Image Segmentation [Електронний ресурс]. – Режим доступу: <https://arxiv.org/pdf/1505.04597.pdf>
3. Vijay Badrinarayanan, Alex Kendall, Roberto Cipolla SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation [Електронний ресурс]. – Режим доступу: <https://arxiv.org/pdf/1511.00561.pdf>
4. CARLA open source simulator for autonomous driving research [Електронний ресурс]. – Режим доступу: <http://carla.org/>
5. Guo Y., Liu Y., Georgiou T. review of semantic segmentation using deep neural networks [Електронний ресурс]. – Режим доступу: <https://link.springer.com/article/10.1007/s13735-017-0141-z>
6. Kaymak C. Brief Survey and an Application of Semantic Segmentation for Autonomous Driving [Електронний ресурс]. – Режим доступу: <https://arxiv.org/ftp/arxiv/papers/1808/1808.08413.pdf>
7. 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.
8. Liaw A., Wiener M. et al. Classification and regression by randomForest // R news. – 2002. – Vol. 2, no. 3. – P. 18–22.
9. Raj B. A simple guide to semantic segmentation [Електронний ресурс]. – Режим доступу: <https://medium.com/beyondminds/a-simple-guide-to-semantic-segmentation-effcf83e7e54>
10. Satellite Image Segmentation: a Workflow with U-Net [Електронний ресурс]. – Режим доступу: <https://medium.com/vooban-ai/satellite-image-segmentation-a-workflow-with-u-net-7ff992b2a56e>
11. PULKIT SHARMA Computer Vision Tutorial: A Step-by-Step Introduction to Image Segmentation Techniques [Електронний ресурс]. – Режим доступу: <https://www.analyticsvidhya.com/blog/2019/04/introduction-image-segmentation-techniques-python/>
12. Understanding of Convolutional Neural Network (CNN) — Deep Learning [Електронний ресурс]. – Режим доступу:

- <https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148>
13. An Intuitive Explanation of Convolutional Neural Networks [Электронный ресурс]. – Режим доступа: <https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/>
 14. Машинне навчання простими словами [Електронний ресурс]. – Режим доступу: http://www.mmf.lnu.edu.ua/ar/1739?fbclid=IwAR3lY-irfgMy9_bl8jT6VTx7n3BVYsLkm1HJQqu_i_4dWnpTGRtk9vkVzY8
 15. Convolutional neural network [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Convolutional_neural_network
 16. Cross entropy [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Cross_entropy
 17. GitHub repo – Режим доступу: <https://github.com/SofiaGorlata/ImageSegmentation>