Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет __Комп'ютерної інженерії та управління_____
(повна назва)

Кафедра Автоматизації проектування обчислювальної техніки_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти
____другий (магістерський)_____
(рівень вищої освіти)

Модель бінаризації растрового зображення за допомогою нейронних мереж
(тема)

Виконав: студентка 2 курсу, групи СКСм-21-1

_____Карпенко К.О._____
(прізвище, ініціали)

Спеціальність *123 – Комп'ютерна інженерія*_____
(код і повна назва спеціальності)

Тип програми ____освітньо-професійна __
(освітньо-професійна або освітньо-наукова)

Освітня програма *Спеціалізовані*_____

*комп'ютерні системи*_____
(повна назва освітньої програми)

Керівник __Хаханова Г.В_____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____
____ Чумаченко С.В
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет_____Комп'ютерної інженерії та управління_____

Кафедра_____Автоматизації проектування обчислювальної техніки

Рівень вищої освіти____другий (магістерський)_____

Спеціальність_____123_ Комп'ютерна інженерія _____
(шифр і назва)

Тип програми _____Освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

Освітня програма _____Спеціалізовані комп'ютерні системи_____
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____20___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові____Карпенко Катерині Олександрівні_____
(прізвище, ім'я, по батькові)

1. Тема роботи Модель бінаризації растрового зображення за допомогою нейронних мереж_____
_____
затверджена наказом по університету від 14 _листопада__2022 р. № 1478 Ст_____
2. Термін подання студентом роботи до екзаменаційної комісії 17 грудня 2022 р.
3. Вихідні дані до роботи
_____1. Растрове зображення_____
_____2. Згорткова нейронна мережа_____
_____3. Перелік алгоритмів бінарізації_____
_____4. Перелік алгоритмів бінарізації_____
_____
_____
_____
_____
_____
4. Перелік питань, що потрібно опрацювати в роботі _._
____1. Дослідження алгоритмів бінаризації зображень_____
____2. Дослідження згорткових нейронних мереж_____
____3. Дослідження бібліотек для навчання нейронних мереж_____
____4. Тестування розробленої моделі_____
_____
_____

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) _____
слайди презентації_____
_____
_____

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

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

7. Дата видачі завдання 01 вересня 2022_____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання	03.09.2022 - 10.09.2022	
2	Аналіз предметної області	10.09.2022 - 30.09.2022	
3	Аналіз джерел з проблемної галузі	30.09.2022 - 15.10.2020	
4	Постановка задачі	15.10.2022- 15.11.2022	
5	Розробка алгоритмів бінаризації на JavaScript. Дослідження згоркових нейронних мерж	15.11.2022 - 05.12.2022	
6	Розробка моделі навчання та її тестування	05.12.2022 - 08.12.2022	
7	Оформлення пояснювальної записки. Оформлення графічного матеріалу	08.12.2022 - 14.12.2022	
8	Перевірка виконаного проекту керівником	10.12.2022 - 15.12.2022	
9	Захист проекту	16.12.2022 - 23.12.2022	

Студент _____ _____
(підпис)

Керівник роботи _____ _____ _Хаханова Г.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка містить 78 сторінок, 30 рисунків та 21 джерело за переліком посилань.

НЕЙРОННІ МЕРЕЖІ, НЕЙРОН, ДОДАТОК, БІНАРИЗАЦІЯ, ПОРОГ, ЗОБРАЖЕННЯ, НАВЧАННЯ, ЗГОРТКА, TENSORFLOW, JAVASCRIPT, ВТРАТА, ІНТЕГРАЛЬНІ ЗОБРАЖЕННЯ, ПІКСЕЛЬ

Метою роботи є створення моделі бінаризації растрового зображення за допомогою нейронних мереж.

У ході виконання були розглянуті принципи та готові рішення для автоматизації створення та навчання нейронної мережі. Розглянуто методи бінаризації растрового зображення. Досліджено структуру та роботу згорткових нейронних мереж. Було виділено основні переваги і недоліки кожного з методів. Реалізовано та протестована найпопулярніші методи бінаризації на мові програмування JavaScript. Створена модель із власним шаром бінаризації для навчання нейронної мережі.

ABSTRACT

The report contains: 78 pages, 30 figures, 21 sources according to the list of links.

NEURAL NETWORKS, NEURON, APPENDIX, BINARIZATION, THRESHOLD, IMAGE, LEARNING, CONVULSION, TENSORFLOW, JAVASCRIPT, LOSS, INTEGRAL IMAGES, PIXEL

The purpose of the work is to create a binarization model of a raster image using neural networks.

During the implementation, the principles and ready-made solutions for automating the creation and training of a neural network were considered. The methods of binarization of the raster image have been expanded. The structure and operation of convolutional neural networks have been studied. The main advantages and disadvantages of each of the methods were highlighted. The most popular methods of binarization in the JavaScript programming language have been implemented and tested. A model with its own binarization layer was created for learning a neural network.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ І СКОРОЧЕНЬ

НР – нейронні мережі;

SARSA – state action reward state action;

ConvNet/CNN – convolutional neural network;

RGB – red green blue;

HSV – hue saturation value;

CMYK – cyan magenta yellow black;

8K – роздільна здатність 7680x4320;

T – порогове значення;

PDE – partial differential equation;

iOS – iPhone operational system:

TPU – tensor processing unit;

REST – representational state transfer;

gRPC – google remote procedure calls;

API – application programming interface;

WebGL – web graphics library;

GPU – graphics processing unit;

СКП – середня квадратична помилка;

JSON – javaScript object notation;

ВСТУП

Прикладні завдання обробки кольорових растрових зображень, потрібні на сучасному етапі розвитку інформаційного суспільства, пов'язані з вилученням інформації з растрових документів у всіх сферах людської діяльності – наукової, освітньої тощо.

Під сегментацією зображення розуміється процес його розбивки на складові, що мають змістовний зміст: фрагменти, їх межі або інші інформативні характеристики, характерні геометричні особливості та ін. сегментація полегшує завдання достовірної оцінки параметрів фрагментів – кандидатів на наступну класифікацію.

Процес розпізнавання фрагментів кольорових растрових зображень може бути реалізований шляхом виконання наступних етапів [1]:

– перед обробка зображення, що включає шумозаглушення, корекцію контрастності/яскравості та колірного балансу;

– оконтурювання областей та виділення на зображенні фрагментів кандидатів, які можуть містити шуканий фрагмент. Це можливо як з застосуванням сегментації на однорідні області, так і на основі перевірки всіх можливих фрагментів зображення, що вимагає складнішого підходу до розпізнавання. Критичним є випадок визначення параметрів у накладених чи спотворених фрагментах;

– класифікація фрагментів з урахуванням параметрів;

– розпізнавання фрагмента зображення на підставі ідентифікації по деякому набору ознак певного класу.

Бінаризація – класичне завдання обробки зображень. Часто бінаризація використовується для спрощення даних та прискорення подальшої обробки, що в наш час уже не видається важливим. Але при аналізі пористих матеріалів бінаризація важлива, оскільки модель даних тут не передбачає проміжного стану між порожнім часом і непроникною матрицею. А

алгоритму, який чудово працює «з коробки», як завжди, немає. Є алгоритми з настроювальними параметрами, чудові нейромережеві архітектури. Що ж робити, якщо в нашому завданні отримання еталонних відповідей є дуже трудомістким? На допомогу прийдуть нейронні мережі.

Мозок і цифровий комп'ютер виконують різні завдання і мають різні властивості. У типовому мозку людини є у 1000 разів більше нейронів, ніж логічних елементів у процесорі типового комп'ютера високого класу. Відповідно до закону Мура і з урахуванням того, що за деякими розрахунками, кількість нейронів у мозку має подвоюватися приблизно через кожні 2-4 мільйони років, може бути зроблений прогноз, що кількість логічних елементів у процесорі стане рівною кількості нейронів у мозку. Безперечно, ці прогнози мало про що говорять; крім того, ця відмінність щодо кількості елементів є незначною порівняно з різницею у швидкості перемикання та ступеня розпаралелювання. Мікросхеми комп'ютера здатні виконати окрему команду менше ніж за наносекунду, тоді як нейрони діють у мільйони разів повільніше. Але мозок сторицею заповнює цей свій недолік, оскільки всі його нейрони діють одночасно, тоді як більшість сучасних комп'ютерів мають лише один процесор (але з кількома ядрами) або невелику кількість процесорів. Таким чином, навіть незважаючи на те, що комп'ютер має перевагу більш ніж у мільйон разів у фізичній швидкості перемикання, виявляється, що мозок у порівнянні з ним виконує всі свої дії приблизно у 100 000 разів швидше.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Характеристика предметної області

В першу чергу у даному курсовому проекті необхідно розглянути існуючі алгоритми для бінаризації растрових зображень. Визначити їх переваги і недоліки.

По-друге, необхідно розробити нейрону мережу для подальшого навчання.

На першому етапі розробки потрібно навчити нейрону мережу виконувати найпростіші дії, а саме:

– розробити методи бінаризації на мові програмування JavaScript;
– розробити модель для навчання нейронної мережі;
– протестувати розроблені методи і модель навчання.

1.2 Актуальність роботи

Нейронні мережі та нейрокомп'ютери - галузь знань, дуже популярна нині. Це проявляється, зокрема, у великій кількості публікацій, конференцій та різноманітних застосувань. Одна з підстав такої популярності – їх чудові здібності до навчання за спостережуваними прикладів та формування прийнятних висновків на базі неповної, зашумленої та неточної вхідної інформації. Роботи з нейронних мереж спочатку були розпочаті біологами. За допомогою нейромереж дослідники прагнули вивчити властивості та особливості роботи головного мозку.

В даний час все більший інтерес до штучних нейронних мереж виявляють різні галузі промисловості та непромислової сфери. Штучні нейронні мережі ефективно використовуються для розпізнавання

відеозображень, письмового тексту та мовлення, розв'язання різноманітних задач прогнозування та у багатьох інших областях.

В даний час відома велика кількість комерційних програмних систем моделювання, що дозволяють досліджувати та розробляти штучні нейронні мережі для різних додатків, а також розроблено значну кількість нейрокомп'ютерних систем.

Потенціал у нейронних технологій величезний, але їх ефективне використання потребує певного рівня знань та розуміння принципів їх дії. Нейронні мережі, на відміну статистичних методів багатовимірного класифікаційного аналізу, базуються на паралельній обробці інформації та мають здатність до самонавчання, тобто отримувати обґрунтований результат на підставі даних, що не зустрічалися у процесі навчання. Ці властивості дозволяють нейронним мережам вирішувати складні (масштабні) завдання, які на сьогоднішній день вважаються важкорозв'язними. Основними перевагами нейронних мереж перед традиційними обчислювальними методами є:

– процес створення нейронної мережі більше відноситься до процесу навчання, ніж до програмування;

– нейрокомп'ютери особливо ефективні там, де потрібна подоба людської інтуїції, зокрема до таких завдань належать прийняття рішень у процесі оцінки фінансового стану деякого економічного об'єкта;

– гнучкість структури нейронних мереж дозволяє різними способами комбінувати прості складові нейрокомп'ютерів - нейрони та зв'язки між ними. За рахунок цього один нейрокомп'ютер можна застосовувати для вирішення різних завдань, часто ніяк не пов'язаних між собою;

– нейронні мережі дозволяють створити ефективне програмне забезпечення для високопаралельних комп'ютерів. Створюючи математичне забезпечення на основі нейронних мереж, можна для широкого класу задач вирішити проблему ефективності одночасного розв'язання паралельних

завдань. Крім того, паралельна обробка інформації забезпечує високу швидкість розв'язання задач;

– розв'язання задач в умовах невизначеності – здатність навчання нейронної мережі дозволяє вирішувати завдання з невідомими закономірностями та залежностями між вхідними та вихідними даними, що дозволяє працювати з неповними даними. Крім того, взаємини між величинами заздалегідь не встановлюються, оскільки метод передбачає вивчення існуючих взаємозв'язків на готові моделі;

– стійкість до шумів у вхідних параметрах – нейронна мережа може самостійно визначати неінформативні для аналізу параметри та виробляти їх відсівання, у зв'язку з чим зникає необхідність додатковий аналіз інформаційного вкладу вхідних даних;

– адаптування до змін навколишнього середовища – нейронні мережі можуть бути перевчені в нових умовах довкілля, що описуються незначними коливаннями параметрів цього середовища. Тобто можна проводити перенавчання нейронних мереж на основі незначних коливань параметрів середовища;

– якщо завдання вирішується за умов нестаціонарного середовища (де статистика змінюється з часом), то можуть бути створені нейронні мережі, що переучуються у реальному часі. Чим вище адаптивні;

– можливості системи, тим паче стійкою буде її робота у нестаціонарному середовищі. Потенційна відмовостійкість нейронних мереж обґрунтована незначним зниженням їх продуктивності за несприятливих умов. Ця особливість пояснюється розподіленим характером зберігання інформації в нейронній мережі, завдяки чому можна стверджувати, що тільки серйозні ушкодження структури нейронної мережі суттєво вплинуть на її працездатність;

– є певна схожість у використанні нейронних мереж та методів багатовимірного статистичного аналізу в оцінці фінансово-економічного стану підприємства.

Бінаризація зображання приходить у нагоді, коли якість картинку у фотоапараті, камері відеонагляду, смартфоні дуже низька, чи коли необхідно передати чітке зображення з мінімальним розміром файлу. Іноді при обробці доводиться мати справу із зображеннями, що зберігаються як напівтонові, але за своїм змістом мало відрізняються від бінарних. До них належать текст, штрихові малюнки, креслення, зображення відбитка пальця. Заміна вихідного напівтонового зображення бінарним препаратом вирішує дві основні завдання.

По-перше, досягається більша наочність при візуальному сприйнятті, ніж у вихідного зображення.

По-друге, відчутно скорочується обсяг пристрою для зберігання зображення, оскільки бінарний препарат для запису кожної точки бінарного зображення вимагає лише 1 біт пам'яті, в той час як напівтонове зображення для вирішення того ж завдання при найчастіше застосовуваному форматі представлення - 8 біт.

2 МЕТОДИ БІНАРИЗАЦІЇ РАСТРОВИХ ЗОБРАЖЕНЬ

2.1 Глобальні методи бінаризації

2.1.1 Порогові методи бінаризації

У глобальних методах бінаризації відбувається робота з усім зображенням одночасно. В ході роботи знаходиться поріг бінаризації t, за допомогою якого відбувається розподіл на чорне та біле, причому величина порога t залишається незмінною протягом усього процесу бінаризації. До пороговим методам бінаризації відносяться:

– бінаризація з нижнім порогом;
– бінаризації з верхнім порогом;
– бінаризація з подвійним обмеженням;
– неповна порогова обробка;
– багаторівневе граничне перетворення .

Одним з найпростіших методів перетворення зображення є бінаризація з нижнім порогом, у якому розглядається лише одне значення порога:

$$F'(m,n) = \begin{cases} 0, F(m,n) \geq t \\ 1, F(m,n) < t \end{cases} \quad (1)$$

Якщо у наведеній вище формулі для точки зображення виконується перша умова, то така точка є точкою об'єкта, якщо ж виконується друга умова, то точка буде точка фону.

У деяких випадках можна використовувати варіант методу бінаризації з нижнім порогом, внаслідок якого виходить негатив вихідного зображення. Такий метод називається бінаризацією з верхнім порогом і представляється формулою:

$$F'(m,n) = \begin{cases} 0, F(m,n) \geq t \\ 1, F(m,n) < t \end{cases} \quad (2)$$

Якщо необхідно виділити певні області, значення яскравості пікселів у яких можуть змінюватись у певному діапазоні, то застосовується метод бінаризації з подвійним обмеженням. Такий метод називається бінаризацією з верхнім порогом і представляється формулою:

$$F'(m,n) = \begin{cases} 0, F(m,n) \geq t_1 \\ 1, t_1 < F(m,n) \leq t_2 \\ 0. F(m,n) > t_2 \end{cases} \quad (3)$$

Найпростіше у реалізації це верхній (нижній) пороги, якщо амплітуда пікселя нижче (вище) порога перетворюється на чорний, решта білий.

Наведемо приклад лише нижньої межі, так як . насправді верхня кордон є інверсією. Друге зображення є перетворене вихідного зображення методом нижньої бінаризації відображено на рисунку 2.1.

Складність у цій картинці полягає в знаходженні країв, так як перепад дуже низький між краями.

Рисунок 2.1 – Зображення деталі, перетворене методом нижньої бінаризації

Якщо необхідно отримати найпростіше для подальшого аналізу зображення, то варто застосувати алгоритм неповної порогової обробки, під час якого зображення позбавляється фону з усіма його деталями, що були на вихідному зображенні. Формула неповної порогової бінаризації представлена нижче:

$$F'(m,n) = \begin{cases} F(m,n), F(m,n) > t, \\ 0, F(m,n) \leq t \end{cases} \quad (4)$$

У разі, якщо необхідно отримати зображення, яке містить у собі сегменти, які мають різну яскравість, можна застосувати метод багаторівневого порогового перетворення. Однак, при цьому отримане в ході перетворення зображення вже не буде бінарним. Формула цього перетворення представлена нижче:

$$F'(m,n) = \begin{cases} 1, F(m,n) \in D_1, \\ 2, F(m,n) \in D_2, \\ \dots \\ n. F(m,n) \in D_n, \\ 0, \text{в інших випадках} \end{cases} \quad (5)$$

Розглянемо те ж саме зображення, перетворений методом багаторівневого порогового перетворення на рисунку 2.2.

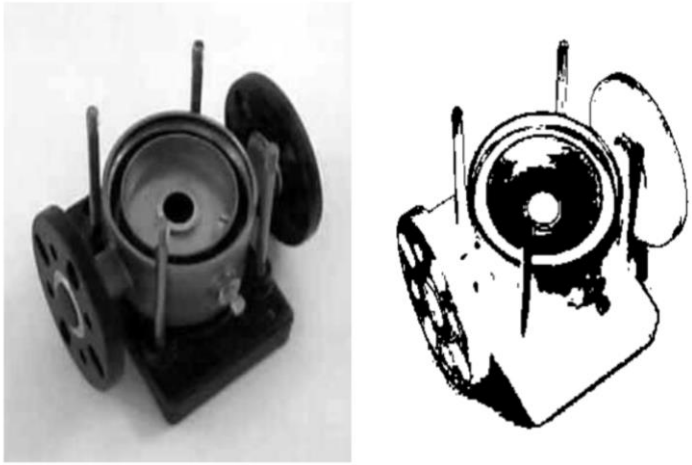

Рисунок 2.2 – Зображення деталі методом багаторівневого порогового перетворення

Якщо порівняти два перетворення, то краще зображення, зокрема краї деталі на фото, було отримано саме за допомогою метода багаторівневого порогового перетворення.

2.1.2 Метод Отцу

Популярним методом глобальної бінаризації зображень є метод Отцу. За допомогою даного методу обчислюється поріг t, який мінімізує середню помилку сегментації, тобто. середню помилку від ухвалення рішення про належність пікселів зображення об'єкта або фону. Значення яскравостей пікселів зображення можна розглядати як випадкові величини, які гістограму

– як оцінку щільності розподілу імовірностей. Якщо густини розподілу ймовірностей відомі, то можна визначити оптимальний (у сенсі мінімуму помилки) поріг для сегментації зображення на два класи c0 та c1 (об'єкти та фон).

Гістограма будується за значеннями $p_i = \frac{n_i}{N}$. У цій формулі N – загальна кількість пікселів зображення з рівнем яскравості i. Поріг t є цілим значенням від 0 до L=max. За допомогою гістограми всі пікселі поділяються на «корисні» (об'єктні) і фонові. Кожному виду відповідають відносні частоти W0 та W1:

$$W_0(t) = \sum_{i=1}^{t} p_i \tag{6}$$

$$W_1(t) = \sum_{i=t+1}^{L} 1 - W_0(t) \tag{7}$$

Далі обчислюються середні рівні кожного виду зображення за формулами:

$$\mu_0(t) = \sum_{i=1}^{t} 1 - \frac{ip_i}{W_0} \tag{8}$$

$$\mu_1(t) = \sum_{i=t+1}^{L} t + 1\frac{ip_i}{W_1} \tag{9}$$

Далі шукається поріг, який зменшує дисперсію усередині виду пікселів, що визначається наступною формулою:

$$\delta_W^2(t) = W_1(t)\delta_1^2(t) + W_2(t)\delta_2^2(t) \quad (10)$$

Наступним кроком визначається міжкласова дисперсія за формулою, представленою нижче:

$$\delta_{кл}^2(t) = W_0(t)W_1\ (t) * (\mu_1(t) - \mu_0(t))^2 \quad (11)$$

Потім обчислюється максимальне значення для оцінки якості поділу зображення на дві частини, що відповідає шуканому порогу:

$$\eta(t) = \max\left[\frac{\sigma_{кл}^2(t)}{\sigma_W^2(t)}\right] \quad (12)$$

Даний метод найкращий спосіб знайти глобальний поріг зображення, і він самоочевидний, і він підходить для більшості випадків, коли потрібний глобальний поріг зображення.

Переваги методу Отцу є:

– простота реалізації;

– адаптація до різноманітних зображень, за допомогою вибору оптимального порога;

– швидкий час виконання.

Недоліками методу є:

– чутливий до шуму зображення;

– його можна сегментувати лише з однієї мети;

– коли співвідношення розмірів мети і фону велике, а функція дисперсії між класами може показувати подвійні чи множинні піки, нині ефект дуже хороший.

На рисунку 2.3 зображено перетворення вихідного зображення методом Отцу.

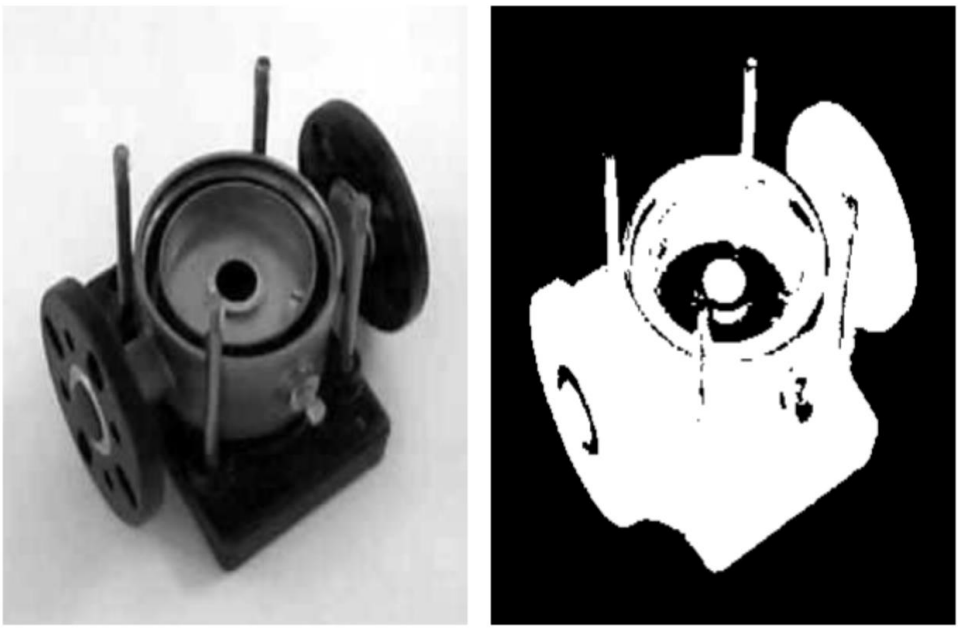

Рисунок 2.3 – Зображення деталі методом Отцу

2.2 Локальні (адаптивні) методи бінаризації

Локальні або адаптивні методи бінаризації розбивають зображення на декілька областей, для кожної з яких необхідно обчислити поріг, ґрунтуючись на інформації про інтенсивність пікселів.

Алгоритми даного класу припускають розбиття зображення на блоки певного розміру, при цьому розмір блоку має бути мінімальним, але достатнім для збереження вихідних особливостей та деталей зображення. Однак при цьому блоки мають бути настільки великими, щоб шуми впливали на результат мінімально. Функція згладжування результуючого растру при адаптивній бінаризації дозволяє отримати задовільний результат без використання додаткових фільтрів.

Розглянемо найпопулярніші адаптивні методи бінаризації зображень:

– метод Ніблека;
– метод Бернсена;
– метод Ейквеля;

– метод Саувола;
– метод Яновица и Брукштейна;
– метод Крістіана.

2.2.1 Метод Ніблека

У методі Ніблек для кожного пікселя зображення необхідно отримати своє значення порога. Його величина визначається на основі обчислення локального середнього та локального середньоквадратичного відхилення. Значення порогу для точки з координатами (m, n) вважається за такою формулою:

$$t(m, n) = \mu(m, n) + k * \sigma(m, n) \quad (13)$$

У цій формулі є середнє, σ (m, n) – середньоквадратичне відхилення в локальній околиці розглянутої точки зображення, а значення k визначає, яку саме частину межі об'єкта необхідно взяти як об'єкт.

Метод Ніблека за рахунок своєї простоти дозволяє досягти найвищої швидкості бінаризації зображень. Метод використовується на практиці для швидкої фільтрації контрастних зображень, на яких практично відсутні сильно зашумлені ділянки з плавними переходами яскравості.

На рисунку 2.4 показано застосування методу Ніблека для бінаризації зображення.

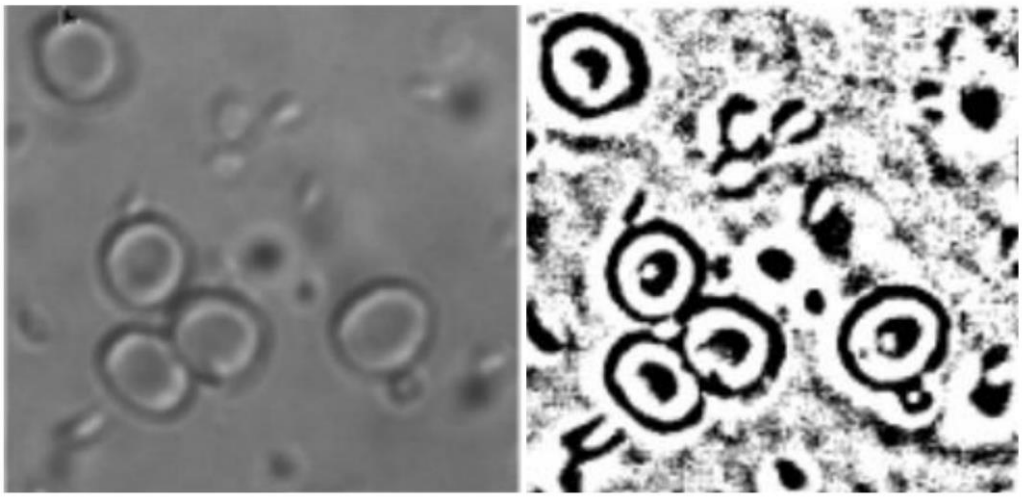

Рисунок 2.4 – Застосування методу Ніблека для бінаризації зображення

2.2.2 Метод Бернсена

Метод Бернсена передбачає розподіл всього зображення на квадрати розміром r×r, мають центр у точці (m, n). Кожен піксель має поріг у межах квадрата, обчислюється за такою формулою:

$$t(m,n) = \frac{Jhight + Jlow}{2} \qquad (14)$$

У цій формулі Jhight і Jlow є найбільший і найменший рівні яскравість квадрата. Якщо поточний піксель $t(m,n) < \varepsilon$, де ε є константою, спочатку задану користувачем, то піксель буде відноситься лише до одного з видів: чорному чи білому.

Якщо середнє відхилення менше порога контрасту - то піксель, що розглядається стає того кольору, який ставився параметром «колір сумнівного пікселя».

Цей метод має ряд недоліків: в отриманому зображенні при обробці однорідних областей формуються досить сильні перешкоди, що у ряді випадків призводить до появи хибних чорних плям. Ці недоліки можуть

компенсуватись за рахунок додаткової обробки – постпроцессингу, і навіть разом із ним є найшвидшим. Метод може бути застосований для схематичних та картографічних зображень [4].

Результати застосування методу Бернсена для бінаризації зображено рисунку 2.5.

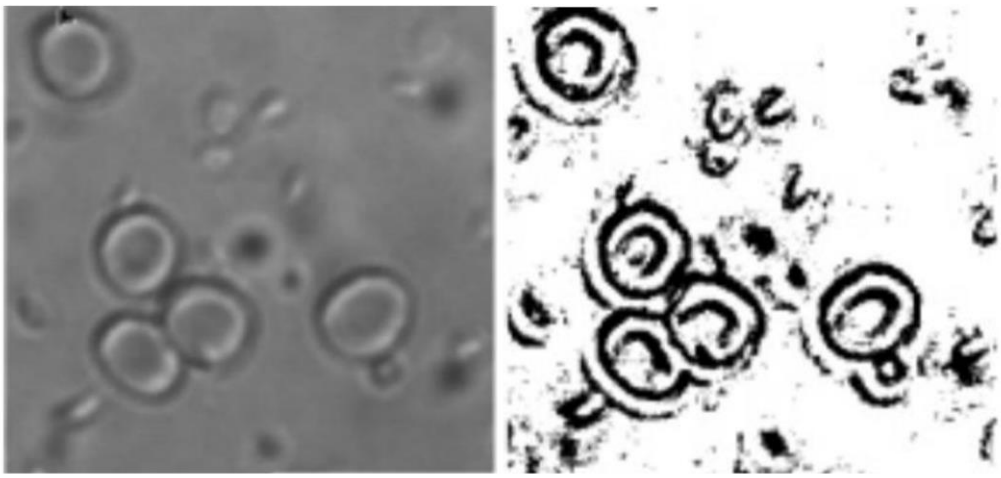

Рисунок 2.5 – Застосування методу Бернсена для бінаризації зображення

2.2.3 Метод Ейквеля

Найбільш продуктивним методом є метод Ейквеля, що часто застосовується для обробки чітких чи контрастних зображень. Суть даного методу полягає в тому, що зображення обробляється за допомогою двох концентричних вікон: маленького – S та великого L. Вікна мають квадратну форму. Дані вікна послідовно накладаються на зліва направо, зверху вниз, використовуючи крок у довжину сторони маленького вікна S.

Для великого вікна L розраховується поріг B таким чином, щоб пікселі можна було поділити на два кластери.

Якщо математичні очікування рівня яскравості у двох кластерах мають різницю, що перевищує деякий заданий користувачем рівень /μ1-μ2/≥l, то всі пікселі всередині вікна S бінаризуються відповідно до порогу t. Інакше

яскравість пікселів з вікна S замінюється значенням, близьким до вихідного.

2.2.4 Метод Саувола

До методів локальної адаптивної бінаризації належить метод Саувола. Визначення локального порога бінаризації здійснюється за допомогою проходження всього зображення вікном w x w. У методі бінаризації Саувола поріг t(x,y) визначається такою формулою, використовуючи для обчислення середнє значення m(x,y) та середньоквадратичне відхилення s(x, y) інтенсивності пікселя у вікні w x w навколо пікселя (x, y):

$$t(x,y) = m(x,y)\left[1 + k\left(\frac{S(x,y)}{R} - 1\right)\right] \quad (15)$$

У даній формулі R являє максимальне відхилення (R = 128 для зображення відтінках сірого), а k є параметром, який набуває значення в діапазоні [0.2, 0.5].

У методі Сауволи є поняття інтегрального зображення. Інтегральне зображення є зображення, у якого значення пікселів визначає суму всіх значень пікселів вище та лівіше позиції у вихідному зображенні. Одного разу порахувавши інтегральне зображення, можна знайти середнє значення у вікні, виконавши всього три арифметичні операції замість підсумовування всіх пікселів у вікні за формулою:

$$m(x,y) = (I)\left(x + \frac{w}{2}, y + \frac{w}{2}\right) + I\left(x - \frac{w}{2}, y - \frac{w}{2}\right) - \\ -I\left(x + \frac{w}{2}, y - \frac{w}{2}\right) - -I(x - \frac{w}{2}, y - \frac{w}{2}))/w^2 \quad (16)$$

Далі обчислюється дисперсія, за такою формулою:

$$s^2(x,y) = \frac{1}{w^2} \sum_{i=x-\frac{w}{2}}^{x+\frac{w}{2}} \sum_{i=y-\frac{w}{2}}^{y+\frac{w}{2}} g^2(i,j) - m^2(x,y), \qquad (17)$$

У цій формулі g(i,j) є значенням пікселя в точці (i,j). Метод Саувола широко застосовується до зображень, у яких яскравість зображення розподіляється нерівномірно. Однак алгоритм Саувола менш стійкий до зашумленості вхідного зображення, ніж, наприклад, алгоритм Отцу. Також у методу є труднощі з зображеннями, у яких мало освітлення, особливо у випадках, коли значення пікселів об'єкти знаходяться близько один до одного. При обробці тонких ліній, що перетинаються, можуть виникати розриви, тому метод хороший для товстих ліній та великих об'єктів.

Результати застосування методу Саувола для бінаризації зображення представлені на рисунках 2.6 – 2.7.

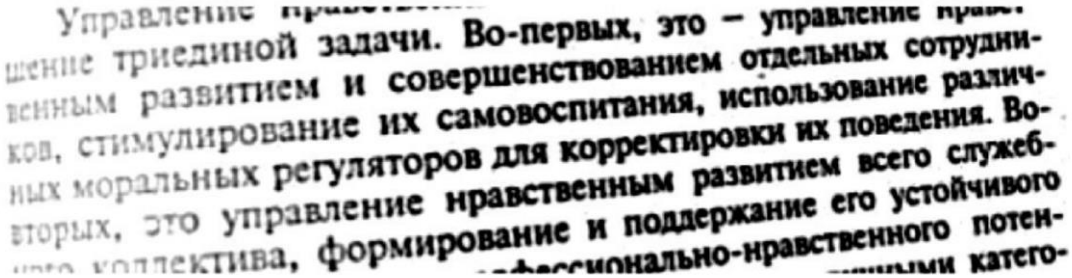
Управление
шение триединой задачи. Во-первых, это — управление
венным развитием и совершенствованием отдельных сотрудни-
ков, стимулирование их самовоспитания, использование различ-
ных моральных регуляторов для корректировки их поведения. Во-
вторых, это управление нравственным развитием всего служеб-
ного коллектива, формирование и поддержание его устойчивого
профессионально-нравственного потен-
ными катего-

Рисунок 2.6 – Вхідне зображення

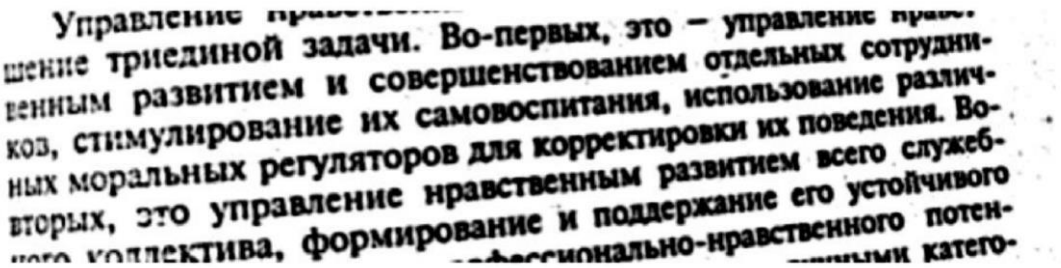
Управление
шение триединой задачи. Во-первых, это — управление
венным развитием и совершенствованием отдельных сотрудни-
ков, стимулирование их самовоспитания, использование различ-
ных моральных регуляторов для корректировки их поведения. Во-
вторых, это управление нравственным развитием всего служеб-
ного коллектива, формирование и поддержание его устойчивого
фессионально-нравственного потен-
ными катего-

Рисунок 2.7 – Застосування методу Саувола для бінаризації зображення

Загалом адаптивну бінаризацію можна рекомендувати у разі, якщо необхідно обробити для обробки напівтонових зображень невисокої якості (сканований знімок), на яких через нерівномірність фону звичайна бінаризація дає погані результати.

Невдачі в процесі бінаризації можуть призвести до спотворень, таких, як розриви в лініях, втрата значущих деталей, порушення цілісності об'єктів, поява шуму та непередбачуване спотворення символів через неоднорідність фону.

Різні методи бінаризації мають свої слабкі місця: так, наприклад, метод Отцу може призводити до втрати дрібних деталей та «злипання» довколишніх символів, а метод Ніблека грішить появою хибних об'єктів у разі неоднорідностей фону з низькою контрастністю.

З цього можна зробити висновок, що кожен із розглянутих методів повинен бути застосований у певній області, а також для кожної області можуть бути розроблені найбільш підходящі методи.

3 ВИДИ НЕЙРОННИХ МЕРЕЖ ТА МЕТОДИ ЇХ НАВЧАННЯ

3.1 Основні складові нейронної мережі

У наші дні зростає необхідність у системах, які здатні не тільки виконувати одного разу запрограмовану послідовність дій над заздалегідь визначеними даними, але й здатні самі аналізувати інформацію, що надходить, знаходити в ній закономірності, проводити прогнозування і так далі. У цій галузі додатків найкраще зарекомендували себе так звані нейронні мережі – системи, що самонавчаються, що імітують діяльність людського мозку. Розглянемо докладніше структуру штучних нейронних мереж (НР) та їх застосування у конкретних завданнях.

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

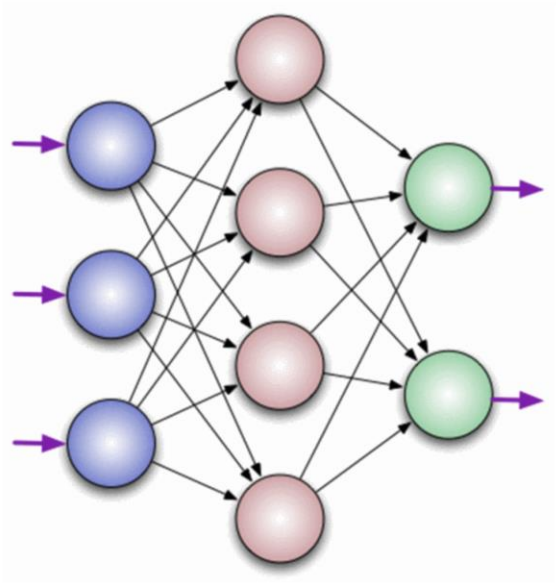

Рисунок 3.1 – Модель нейрона

Також є нейрон зміщення та контекстний нейрон.

Нейрон зміщення або bias нейрон - це третій вид нейронів, що використовується в більшості нейромереж. Особливість цього нейронів полягає в тому, що його вхід і вихід завжди дорівнюють 1 і вони ніколи не мають вхідних синапсів. Нейрони зміщення можуть бути присутніми в нейронній мережі по одному на шарі, або повністю відсутніми, 50/50 бути не може (червоним на схемі позначені ваги і нейрони які розміщувати не можна). З'єднання у нейронів зміщення такі ж, як у звичайних нейронів - з усіма нейронами наступного рівня, за винятком того, що синапс між двома bias нейронами бути не може. Отже, їх можна розміщувати на вхідному шарі і всіх прихованих шарах, але ніяк не на вихідному шарі, тому що їм просто не буде з чим формувати зв'язок.

Даний нейрон потрібен для того, щоб мати можливість отримувати вихідний результат шляхом зсуву графіка функції активації вправо або вліво. Зі шкільного курсу математики, ми знаємо, що якщо взяти функцію y = ax+b і змінювати в неї значення “a”, то змінюватиметься нахил функції (колір ліній на графіку зліва), а якщо змінювати “b”, то ми зміщуватимемо функцію

вправо або вліво (кольори ліній на графіку праворуч). Тобто, коли в ході навчання ми регулюємо ваги прихованих та вихідних нейронів, ми змінюємо нахил функції активації. Однак, регулювання ваги нейронів усунення може дати нам можливість зрушити функцію активації осі X і захопити нові ділянки.

Також нейрони зміщення допомагають у тому випадку, коли всі вхідні нейрони отримують на вхід 0 і незалежно від того які у них ваги, вони передадуть на наступний шар 0, але не у разі присутності нейрона зміщення. Однією із найважливіших компонентів нейронної мережі є синопсис.

Синапс - це зв'язок між двома нейронами. У синапсів є один параметр - вага. Завдяки йому вхідна інформація змінюється, коли передається від одного нейрона до іншого. Припустимо, є 3 нейрони, які передають інформацію наступному. Тоді у нас є 3 ваги, які відповідають кожному з цих нейронів. У того нейрона, у якого вага буде більшою, та інформація і буде домінуючою в наступному нейроні (приклад - змішання кольорів). Насправді, сукупність терезів нейронної мережі або матриця терезів - це своєрідний мозок всієї системи. Саме завдяки цим вагам, вхідна інформація обробляється та перетворюється на результат.

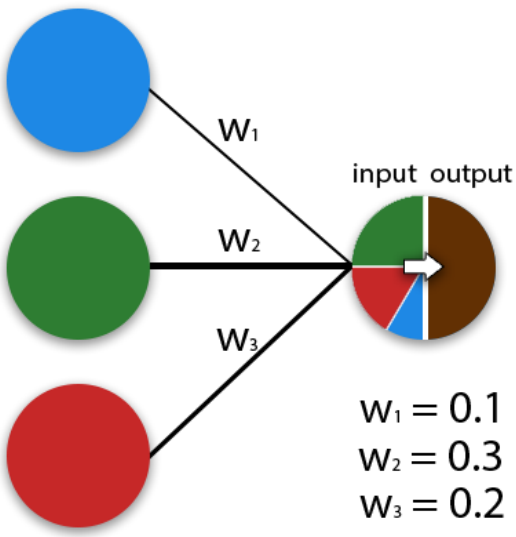

Рисунок 3.2 – Синапсис нейронної мережі з вагами

Розглянемо з чого складається базова нейронна система.

Базова нейронна мережа на основі персептрону є концептуально простою. Персептрон може складатися не більше ніж з двох вхідних вузлів і одного вихідного вузла, з'єднаного зваженими з'єднаннями.

Розмірність вхідних даних має відповідати розмірності вхідного шару. Термін «розмірність» може бути трохи заплутаним, тому що більшість людей не можуть візуалізувати щось більш ніж у трьох вимірах. Все це дійсно означає, що вхідні дані – наприклад, шаблон, який необхідно класифікувати – це вектор із заданою довжиною, і вхідний шар повинен мати вузол для кожного елемента у векторі. Тому, потрібно класифікувати шаблон, представлений серією з 20 точок даних, тоді потрібно 20 вхідних вузлів.

Вихідний вузол генерує дані, які становлять інтерес для розробника. Кількість вихідних вузлів залежить від програми.

Дані, які переміщуються від одного вузла до іншого, множаться на ваги.

3.2 Функція активації

В той чай коли ми вже є вхідні дані, можна отримати вихідні дані, підставивши вхідне значення у функцію активації. Функція активації - це спосіб нормалізації вхідних даних. Тобто, якщо на вході буде велика кількість, пропустивши його через функцію активації, можна отримати вихід у потрібному діапазоні. Розглянемо основі функції активації: лінійна, сігмоїд (логістична) та гіперболічний тангенс. Головні їхні відмінності – це діапазон значень.

3.2.1 Лінійна функція активації

$$f(x) = x$$

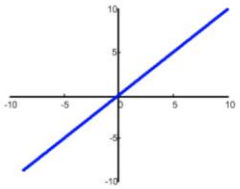

Рисунок 3.3 – Графік лінійної функції

Ця функція майже ніколи не використовується, за винятком випадків, коли потрібно протестувати нейронну мережу або передати значення без перетворень.

3.2.2 Сігмоїд або логістична функція активації

$$f(x) = \frac{1}{1+e^{-x}}$$

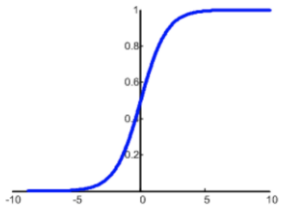

Рисунок 3.4 – Графік логістичної функції

Це найпоширеніша функція активації, її діапазон значень [0,1]. Саме на

ній показано більшість прикладів у мережі, також її іноді називають логістичною функцією. Відповідно, якщо є негативні значення, то вам знадобиться функція яка захоплює і негативні значення.

3.2.3 Гіперболічний тангенс

$$f(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$

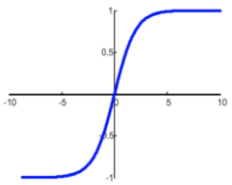

Рисунок 3.4 – Графік гіперболічного тангенс

Має сенс використовувати гіперболічний тангенс тільки тоді, коли ваші значення можуть бути і негативними, і позитивними, так як діапазон функції [-1,1]. Використовувати цю функцію лише з позитивними значеннями недоцільно, оскільки це значно погіршить результати вашої нейромережі.

3.3 Методи навчання нейронних мереж

Під навчанням нейронної мережі розуміють процес, знаходження таких параметрів (ваг) нейронної мережі, при яких мережа наближала б потрібну функцію з максимально можливою точністю.

Модель нейронних мереж, запропонована Мак-Каллоком та Піттсом не мала здатність навчатися, параметри (ваги) нейронної мережі для входів у нейронів необхідно було задавати заздалегідь.

Дональд Хебб ще 1949 року вперше виступив з ідеєю навчання

нейронної мережі. У своїй книзі The Organization of Behaviour (Організація поведінки) Хебб виклав ключові ідеї, які лягли в основу навчання нейронної мережі.

На сьогоднішній день існують три парадигми навчання нейронних мереж: навчання з учителем, навчання без вчителя та навчання з підкріпленням. Розглянемо кожен із них.

3.3.1 Навчання нейронної системи з учителем

Навчання з учителем - це напрямок машинного навчання, що поєднує алгоритми та методи побудови моделей на основі безлічі прикладів, що містять пари «відомий вхід - відомий вихід».

Іншими словами, щоб алгоритм ставився до навчання з учителем, він повинен працювати з прикладами, які містять не лише вектор незалежних змінних (атрибутів, ознак), а й значення, яке має видавати модель після навчання (таке значення називається цільовим). Різниця між цільовим і фактичним виходами моделі називається помилкою навчання (нев'язкою, залишками), яка мінімізується в процесі навчання і виступає як «вчитель». Значення вихідної помилки використовується для обчислення корекцій параметрів моделі на кожній ітерації навчання.

В аналізі даних машинне навчання використовується у задачах класифікації та регресії. У першому випадку як цільова змінна використовується мітка класу, а в другому – числова змінна цілого або речового типу.

В даний час розроблено велику кількість алгоритмів навчання з учителем, кожен з яких має свої сильні та слабкі сторони. Немає єдиного алгоритму , який найкраще підходить всім завдань аналізу.

До алгоритмів навчання з учителем для вирішення завдань класифікації відносяться:

– дерева рішень;

– машини опорних векторів;
– байєсівський класифікатор;
– лінійний дискримінантний аналіз;
– метод k-найближчих сусідів.

3.3.2 Навчання нейронної системи без вчителя

У навчанні без вчителя модель має набір даних, і немає явних вказівок, що з ним робити. Нейронна мережа намагається самостійно знайти кореляції даних, витягуючи корисні ознаки та аналізуючи їх.

Залежно від завдання, модель систематизує дані по-різному:

– кластеризація. Навіть без спеціальних знань експерта-орнітолога можна подивитися на колекцію фотографій та розділити їх на групи за видами птахів, спираючись на колір пера, розмір або форму дзьоба. Саме в цьому полягає кластеризація – найпоширеніша задача для навчання без учителя. Алгоритм підбирає схожі дані, знаходячи загальні ознаки і групують їх разом;

– виявлення аномалій. Банки можуть виявити шахрайські операції, виявляючи незвичайні дії у купівельній поведінці клієнтів. Наприклад, підозріло, якщо одна кредитна картка використовується в Каліфорнії та Данії в той самий день. Схожим чином, навчання без вчителя використовують для знаходження викидів даних;

– асоціації. Виберете в онлайн-магазині підгузки, яблучне пюре та дитячий кухоль-непроливайку та сайт порекомендує вам додати нагрудник та радіоняню до замовлення. Це приклад асоціацій: деякі характеристики об'єкта корелюють з іншими ознаками. Розглядаючи кілька ключових ознак об'єкта, модель може передбачити інші, із якими існує зв'язок;

– автоенкодери. Автоенкодери приймають вхідні дані, кодують їх, а потім намагаються відтворити початкові дані з отриманого коду. Не так багато реальних ситуацій, коли використовують простий автоенкодер. Але

варто додати шари та можливості розширяться: використовуючи зашумлені та вихідні версії зображень для навчання, автоенкодери можуть видаляти шум із відеоданих, зображень або медичних сканів, щоб підвищити якість даних;

У навчанні без вчителя складно обчислити точність алгоритму, оскільки даних відсутні «правильні відповіді» чи мітки. Але ці дані часто ненадійні або їх занадто дорого отримати. У таких випадках, надаючи моделі свободу дій для пошуку залежностей, можна отримати добрі результати.

3.3.3 Навчання нейронної системі з підкріпленням

У навчанні з підкріпленням використовується спосіб позитивної нагороди за правильну дію та негативну за неправильну. Таким чином, метод надає позитивні значення бажаним діям, щоб спонукати систему, і негативні значення – небажаним. Це програмує її на пошук довгострокової та максимальної загальної винагороди для досягнення оптимального рішення. Ці довгострокові цілі не дають системі можливості зупинятися на досягнутому. Згодом вона вчиться уникати негативних дій і робить лише позитивні.

Область навчання з підкріпленням складається з кількох алгоритмів, які використовують різні підходи. Відмінності переважно пов'язані зі своїми стратегіями взаємодії з довкіллям.

State-Action-Reward-State-Action (SARSA). Цей алгоритм навчання із підкріпленням починається з надання системі такого коефіцієнта, як політика (on-policy). У разі політика – це ймовірність, з допомогою якої алгоритм оцінює шанси певних дій, які призводять до винагородам чи позитивним станам.

Q-Learning. У цьому підході Reinforcement Learning використовується протилежний підхід. Система не отримує політики (on-policy), відповідно, його дослідження довкілля є самостійним. У Q-learning ми не маємо

обмежень на вибір дії (action) для алгоритму. Він вважає, що всі наступні вибори actions будуть оптимальними за умовчанням, тому алгоритм робить операцію вибору виходячи з максимізації оцінки Q.

Deep Q-Networks (Глибокі Q-мережі). Цей алгоритм використовує нейронні мережі на додаток до методів навчання з підкріпленням (reinforcement learning). Нейромережі здійснюють самостійне дослідження (research) середовища навчання з підкріпленням вибору найбільш оптимального значення. Те, як алгоритм поводитиметься і підбиратиме значення, засноване на вибірці минулих позитивних дій, отриманих нейронною мережею.

Основними недоліками такого виду навчання є те, що він може бути важким для розгортання і, на жаль, залишається обмеженим у застосуванні. Одна з перешкод для розгортання цієї галузі машинного навчання – це залежність від дослідження навколишнього середовища.

Час, необхідний для правильного навчання за допомогою reinforcement learning, може обмежити його корисність і вимагати значних обчислювальних ресурсів. У міру того як середовище навчання стає більш складним, зростають і вимоги до часу та обчислювальних ресурсів. Саме ці проблеми фахівцям у галузі навчання з підкріпленням доведеться вирішити у найближчому майбутньому.

4 ДОСЛІДЖЕННЯ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ

Штучний інтелект був свідком монументального зростання в подоланні розриву між можливостями людей і машин. Дослідники та ентузіасти працюють над багатьма аспектами галузі, щоб створювати дивовижні речі. Однією з багатьох таких сфер є комп'ютерне бачення.

Порядок денний для цієї сфери полягає в тому, щоб дозволити машинам дивитися на світ так само, як люди, сприймати його подібним чином і навіть використовувати знання для багатьох завдань, таких як розпізнавання зображень і відео, аналіз і класифікація зображень, відтворення медіа, рекомендації. Системи, обробка природної мови тощо. Удосконалення комп'ютерного зору з глибоким навчанням були створені та вдосконалені з часом, головним чином за допомогою одного конкретного алгоритму – загорткової нейронної мережі.

Згорткова нейронна мережа (ConvNet/CNN) – це алгоритм глибокого навчання, який може сприймати вхідне зображення, призначати важливість (ваги та зміщення, які можна вивчити) різним аспектам/об'єктам зображення та мати можливість відрізнити один від іншого. Попередня обробка, необхідна в ConvNet, набагато нижча порівняно з іншими алгоритмами класифікації. У той час як у примітивних методах фільтри розробляються вручну, після достатнього навчання, ConvNets мають можливість вивчати ці фільтри/характеристики.

Архітектура такої мережі аналогічна структурі підключення нейронів у людському мозку та була натхненна організацією зорової кори. Окремі нейрони реагують на стимули лише в обмеженій області поля зору, відомої як рецептивне поле. Набір таких полів перекривається, щоб охопити всю візуальну область.

Згорткові нейронні мережі забезпечують часткову стійкість до змін масштабу, зсувів, поворотів, зміни ракурсу та інших спотворень. Згорткові

нейронні мережі поєднують три архітектурні ідеї, для забезпечення інваріантності до зміни масштабу, повороту зсуву та просторових спотворень:

– локальні рецепторні поля (забезпечують локальну двовимірну зв'язність нейронів);

– загальні синаптичні коефіцієнти (забезпечують детектування деяких рис в будь-якому місці зображення та зменшують загальну кількість вагових коефіцієнтів);

– ієрархічна організація із просторовими підвибірками.

На даний момент загорткова нейронна мережа та її модифікації вважаються найкращими за точністю та швидкістю алгоритмами знаходження об'єктів на сцені. Починаючи з 2012 року, нейромережі займають перші місця на відомому міжнародному конкурсі з розпізнавання образів ImageNet.

4.1 Переваги згорткових нейронних мереж над мережами прямого зв’язку

Зображення – це не що інше, як матриця значень пікселів, вірно? Тож чому б просто не звести зображення (наприклад, матрицю зображення 3x3 у вектор 9x1) і не передати його на багаторівневий персептрон для цілей класифікації?

У випадках надзвичайно простих бінарних зображень метод може показувати середню оцінку точності під час виконання передбачення класів, але матиме незначну точність або взагалі не матиме точності, коли мова йде про складні зображення, які повсюдно залежать від пікселів.

ConvNet може успішно фіксувати просторові та часові залежності в зображенні за допомогою застосування відповідних фільтрів. Архітектура забезпечує кращу адаптацію до набору даних зображення завдяки зменшенню кількості залучених параметрів і можливості повторного

використання вагових коефіцієнтів. Іншими словами, мережу можна навчити краще розуміти складність зображення.

4.2 Топологія загорткових нейронних мереж

Дана мережа складається з різних видів шарів: згорткові (convolutional) шари, субдискретизуючі (subsampling, підвибірка) шари та шари «звичайної» нейронної мережі – персептрону, відповідно до рисунка 4.1.

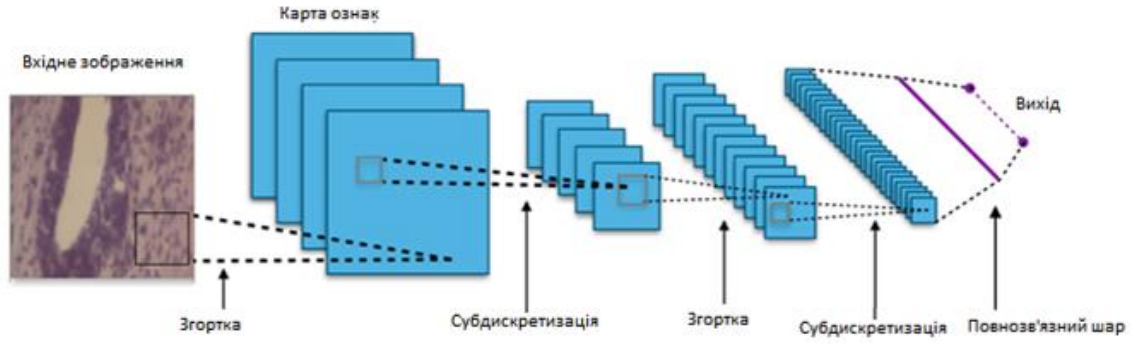

Рисунок 4.1 – Шари згорткової мережі

Перші два типи шарів (convolutional, subsampling), чергуючись між собою, формують вхідний вектор ознак для багатошарового персептрона.

4.2.1 Вхідний шар

На рисунку 4.2 ми маємо зображення RGB, яке розділене трьома кольоровими площинами – червоною, зеленою та синьою. Існує кілька таких колірних просторів, у яких існують зображення — відтінки сірого, RGB, HSV, CMYK тощо.

Ви можете уявити, наскільки обчислювально інтенсивними будуть речі, коли зображення досягнуть розмірів, скажімо, 8K (7680 × 4320). Роль ConvNet полягає в зменшенні зображень до форми, яку легше обробляти, без втрати функцій, які є критично важливими для отримання хорошого

прогнозу. Це важливо, коли ми хочемо розробити архітектуру, яка не тільки добре вивчає функції, але й масштабується до масивних наборів даних.

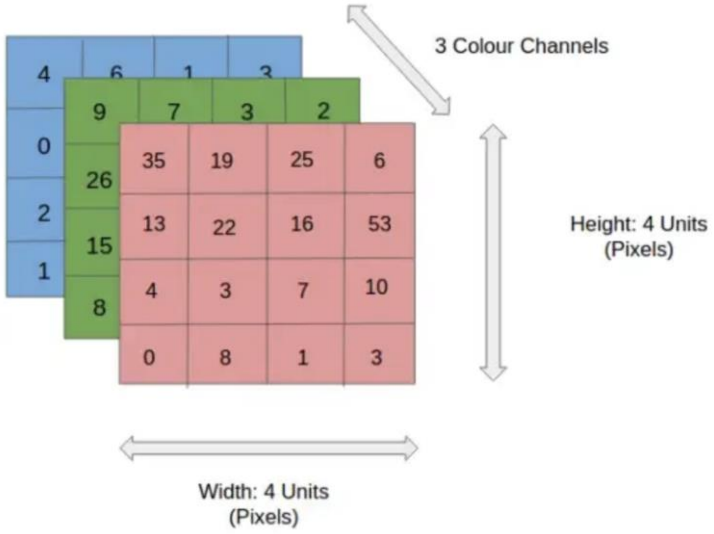

Рисунок 4.2 – RGB зображення

4.2.2 Шар розгортки

Згортковий шар являє собою набір карт, у кожної карти є синаптичне ядро.

Кількість карток визначається вимогами до завдання, якщо взяти велику кількість карток, то підвищиться якість розпізнавання, але збільшиться обчислювальна складність. Виходячи з аналізу наукових статей, у більшості випадків пропонується брати співвідношення один до двох, тобто кожна карта попереднього шару (наприклад, у першого згорткового шару, попереднім є вхідний) пов'язана з двома картами згорткового шару.

Розмір у всіх карт згорткового шару – однакові та обчислюються за формулою 22:

$$(\omega, h) = (mW - kW + 1, mH - kH + 1), \quad (22)$$

де (ω, h) – розмір обгорткової карти, що обчислюється, mW – ширина попередньої карти, mH – висота попередньої карти, kW – ширина ядра, kH – висота ядра.

Ядро - це фільтром, яке ковзає по всій області попередньої карти і знаходить певні ознаки об'єктів. Наприклад, якщо мережу навчали на безлічі осіб, то одна з ядер могла б у процесі навчання видавати найбільший сигнал у сфері ока, рота, брови чи носа, інше ядро могло б виявляти інші ознаки. Розмір ядра зазвичай беруть у межах від 3x3 до 7x7. Якщо розмір ядра маленький, воно не зможе виділити будь-які ознаки, якщо занадто велике, то збільшується кількість зв'язків між нейронами. Також розмір ядра вибирається таким, щоб розмір карт згорткового шару був парним, це дозволяє не втрачати інформацію при зменшенні розмірності в шарі підвибіркового.

Ядро являє собою систему ваг або синапсів, що розділяються, це одна з головних особливостей згорткової нейромережі. У звичайній багатошаровій мережі дуже багато зв'язків між нейронами, тобто синапсами, що дуже сповільнює процес детектування. У згортковій мережі – навпаки, загальні ваги дозволяє скоротити кількість зв'язків і дозволити знаходити ту саму ознаку по всій області зображення.

Спочатку значення кожної карти згорткового шару дорівнюють 0. Значення ваги ядер задаються випадковим чином в області від -0.5 до 0.5. Ядро ковзає по попередній карті і здійснює операцію згортка, яка часто використовується для обробки зображень. Це можна виразити у формулі 23:

$$(f * g)[m, n] = \sum_{k,l} f[m - k, n - l] * g[k, l], \qquad (23)$$

де f – початкова матриця зображення, g – ядро згортки.

Загалом цю операцію можна описати так — вікном розміром ядра g проходимо із заданим кроком (зазвичай 1) все зображення f, кожному кроці

поэлементно множимо вміст вікна на ядро g, результат підсумовується і записується в матрицю результату, як на рисунку 4.3.

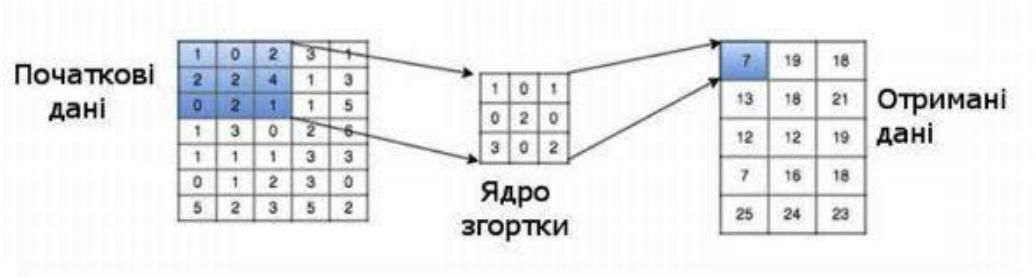

Рисунок 4.3 – Робота ядра згортки

При цьому в залежності від методу обробки країв вихідної матриці результат може бути меншим від вихідного зображення (valid), такого ж розміру (same) або більшого розміру (full), відповідно до рисунку 4.4.

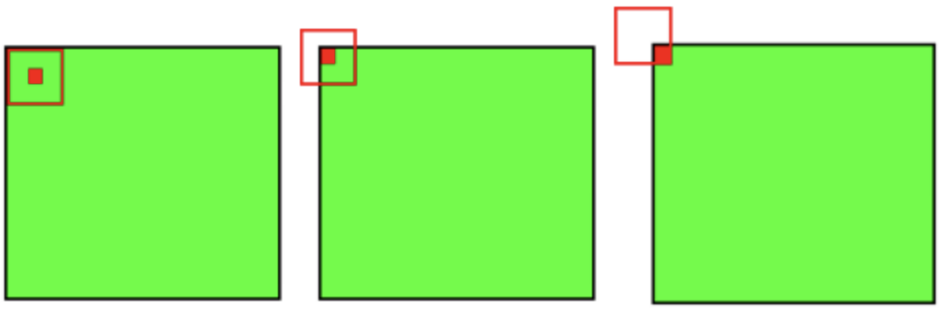

Рисунок 4.4 – Варіанти вихідної матриці

У спрощеному вигляді цей шар можна описати наступною формулою:

$$x^l = f(x^{l-1} * k^l + b^l), \quad (24)$$

де x^l – вихід шару l, f() – функція активації, b^l – коефіцієнт зсуву шара l, $*$ – операція згортки входу x із ядром k.

При цьому за рахунок крайових ефектів розмір вихідних матриць

зменшується. Це можна відтворити у наступній формулі:

$$x_j^i = f\left(\sum_i x_i^{l-1} * k_j^l + b_j^l\right) \quad (25),$$

де x_j^i – карта ознак j (вихід шару l), f() – функція активації, b_j^l – коефіцієнт зсуву шару l для карти ознак j, k_j^l – ядро згортки j карти, шару l,

$*$ – операція згортки входу x з ядром k.

4.2.3 Підвибірковий шар

Підвибірковий шар також, як і згортковий має карти, але їх кількість збігається з попереднім (згортковим) шаром. Мета шару - зменшення розмірності карт попереднього шару. Якщо на попередній операції згортки вже були виявлені деякі ознаки, то для подальшої обробки настільки докладне зображення вже не потрібне і воно ущільняється до менш докладного. До того ж, фільтрація вже непотрібних деталей допомагає не перевчитися.

У процесі сканування ядром підвибіркового шару (фільтром) карти попереднього шару, скануюче ядро не перетинається на відміну від згорткового шару. Зазвичай кожна карта має ядро розміром 2x2, що дозволяє зменшити попередні карти згорткового шару в 2 рази. Вся карта ознак поділяється на комірки 2x2 елементи, у тому числі вибираються максимальні за значенням.

Зазвичай, у підвибірковому шарі застосовується функція активації RelU. Операція підвиборки (або MaxPooling – вибір максимального) відповідно до рисунку 4.5.

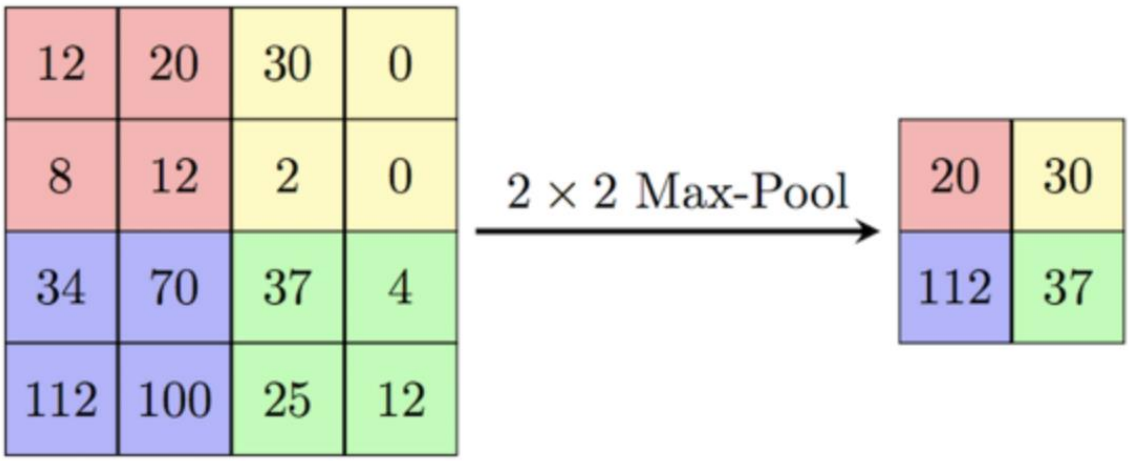

Рисунок 4.5 – Операція підвиборки

Формально шар може бути описаний наступною формулою:

$$x^l = f(a^l * \text{subsample}(a^{l-1}) + b^l), \quad (26)$$

де x^l – вихід шару l, f() – функція активації, a_j^l b_j^l – коефіцієнти зсуву шару l , sabample() – subsample операція виборку локальних максимальних значень.

4.2.4 Повнозв'язковий шар

Останній з типів шарів – це шар звичайного багатошарового персептрона. Мета шару – класифікація, що моделює складну нелінійну функцію, оптимізуючи яку, покращується якість розпізнавання.

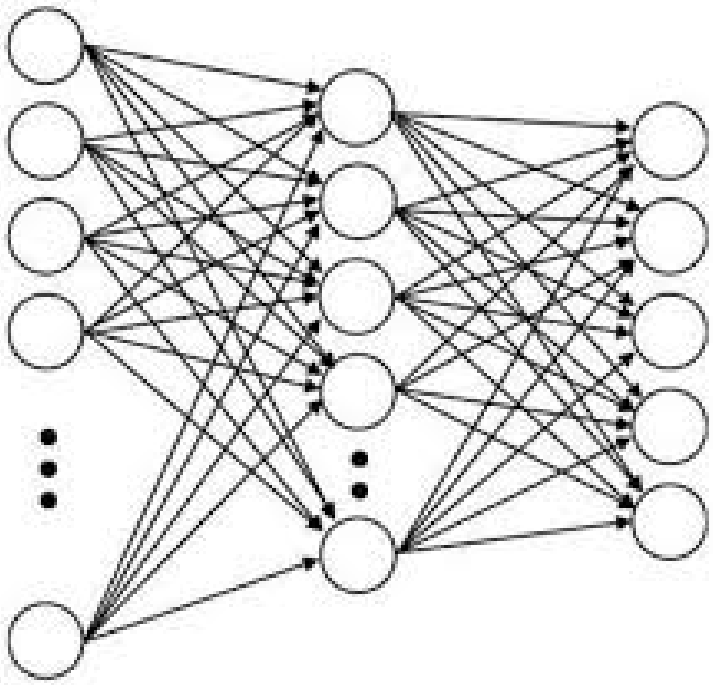

Рисунок 4.6 – Робота повнозв'язкового шару

Нейрони кожної карти попереднього підвиборчого шару пов'язані з одним нейроном прихованого шару. Таким чином число нейронів прихованого шару дорівнює числу карт підвибіркового шару, але зв'язки можуть бути не обов'язково такими, наприклад, тільки частина нейронів будь-якої з карт підвибіркового шару бути пов'язана з першим нейроном прихованого шару, а частина, що залишилася з другим, або всі нейрони першої карти пов'язані з нейронами 1 та 2 прихованого шару. Обчислення значень нейрона можна описати формулою:

$$x_j^l = f(\sum_i x_i^{l-1} * \omega_{i,j}^{l-1} + b_j^{l-1}), \quad (27)$$

де x_j^l – карта ознак j , f() – функція активації, b_j^l – коефіцієнти зсуву шару l , $\omega_{i,j}^{l-1}$ – матриця вагових коефіцієнтів шару l.

4.2.5 Вихідний шар

Вихідний шар пов'язаний із усіма нейронами попереднього шару. Кількість нейронів відповідає кількості класів, що розпізнаються. Але для зменшення кількості зв'язків та обчислень для бінарного випадку можна використовувати один нейрон і при використанні як функцію активації гіперболічний тангенс.

4.3 Вибір функції активації

Одним із етапів розробки нейронної мережі є вибір функції активації нейронів. Вигляд функції активації багато в чому визначає функціональні можливості нейронної мережі та метод навчання цієї мережі. Класичний алгоритм зворотнього поширення помилки добре працює на двошарових та тришарових нейронних мережах, але при подальшому збільшенні глибини починають з’являтися проблеми. Однією з причин є так зване згасання градієнтів. У міру поширення помилки від вихідного шару до вхідного на кожному шарі відбувається збільшення поточного результату на похідну функції активації. Похідна у традиційної сигмоїдної функції активації менше одиниці по всій області визначення, тому після кількох шарів помилка стане близькою до нуля. Якщо ж, навпаки, функція активації має необмежену похідну (як, наприклад, гіперболічний тангенс), може статися вибухове збільшення помилки у міру поширення, що призведе до нестійкості процедури навчання.

5 РЕАЛІЗАЦІЯ ОБРАНИХ АЛГОРИТМІВ БІНАРИЗАЦІЇ НА МОВІ ПРОГРАМУВАННЯ JAVASCRIPT

5.1 Метод Отцу

Порогове значення – це метод, який використовується для видалення наміченого об’єкта або цільового об’єкта з його фонового зображення шляхом визначення значення інтенсивності T (порогове значення) для кожного пікселя. Наприклад, кожен піксель класифікується як точка фону або точка об’єкту. У різних програмах обробки зображень рівні сірого пікселів, для фону та об’єкту, істотно відрізняються. Тоді порогове значення стає простим, але ефективним інструментом для відділення об’єктів від фону. Прикладами порогових програм є аналіз зображень документів, метою якого є виділення друкованих символів, логотипів, графічного вмісту або музичних партитур, тощо.

Зображення можна описати як двовимірну функцію і:е f(x, y), де x і y є просторовими (площинними) координатами, а амплітуда «f» у будь-якій парі координат (x, y) називається інтенсивністю або рівнем сірого зображення в цій точці. Кожен елемент масиву називається пікселем. Пікселі зберігаються як цілі числа. Цілі числа можуть бути 8-бітними, 24-бітними або 32-бітними залежно від типу зображення. Зображення у градаціях сірого є 8-бітними зображеннями, тоді як 32-бітні зображення мають додатковий канал прозорості. Зображення в градаціях сірого складається з середнього значення RGB для кожного пікселя.

Вирівнювання гістограми – це техніка регулювання інтенсивності зображення для підвищення контрастності. Завдяки такому налаштуванню інтенсивність може бути краще розподілена на гістограмі. Це дозволяє областям із нижчим локальним контрастом отримувати вищий контраст. Вирівнювання гістограми досягається шляхом продуктивного розширення

найбільш постійних значень інтенсивності. Цей метод є конструктивним для зображень із яскравим або темним фоном і переднім планом. Зокрема, метод може призвести до кращого перегляду структури кістки на рентгенівських зображеннях.

Метод порогового значення Отцу передбачає ітерацію всіх можливих порогових значень і обчислення міри поширення для рівнів пікселів по кожній стороні порогового значення, тобто пікселів, які потрапляють на передній або фоновий план. Мета полягає в тому, щоб знайти порогове значення, при якому сума розповсюджень переднього плану та фону досягає свого мінімуму. Під час обробки зображень метод порогового значення Отцу використовується для автоматичного визначення рівня бінаризації на основі форми гістограми. Алгоритм передбачає, що зображення складається з двох основних класів: переднього плану та фону. Потім він обчислює оптимальне порогове значення, яке мінімізує зважені дисперсії в межах класу цих двох класів.

Основною проблемою цього методу є залежність від рівню інтенсивності, оскільки жодне інше співвідношення пікселів не розглядається. Також проблему створюють пікселі на межі, а іноді й тіні об’єкта. Шум також впливає на поріг.

Нижче представлений код реалізації метода Отцу на мові JavaScript.

Лістинг 1 - Метод Отцу на мові JavaScript

```
function otsu=(pixelNumber)=>{
   const sum =0
   let sumB =0
   let wB =0
   let wF= 0
   let mB=0;
   let mF=0;
   let max=0;
   let between =0;
   let threshold = 0
   for (let i = 0; i < 256; ++i) {
       wB += histogram[i];
```

```
        if (wB === 0)
            continue;
        wF = pixelNumber - wB;
        if (wF === 0)
            break;
        sumB += i * histogram[i];
        mB = sumB / wB;
        mF = (sum - sumB) / wF;
        between = wB * wF * Math.pow(mB - mF, 2);
        if (between > max) {
            max = between;
            threshold = i;
        }
    }
    return threshold;
}
```

Для отримання гістограми було створена наступна функція.

Лістинг 2 - Отримання гістограми зображення

```
const getHistogram=(imageData)=>{
    for (let i = 0; i < 256; ++i)
        histogram[i] = 0
    for (let i = 0; i < imageData.length; i += 4) {
        const red = imageData[i]
        const blue = imageData[i + 1]
        const green = imageData[i + 2]
        const gray = red * .2126 + green * .7152 + blue * .0722
        histogram[Math.round(gray)] += 1
    }
}
```

5.2 Метод Бредлі

Само по собі порівняння двох величин – порога і яскравості пікселя, – це елементарна задача. Надійність полягає в знаходжені порогового значення, яке повинно максимально надійно відокремити символи не тільки від фону, але і від шуму, тіней і всього інформаційного шуму. Часто для цього звертаються до методів математической статистики. В методі Бредлі ми заходимо з іншої сторони – зі сторони інтегральних зображень.

Інтегральне зображення (також відоме як таблиця сумарної площі) – це

інструмент, який можна використовувати, коли ми маємо функцію від пікселів до дійсних чисел f(x,y) (наприклад, інтенсивність пікселів), і ми хочемо обчислити суму цієї функції по прямокутній області зображення.

Без інтегрального зображення суму можна обчислити в лінійному часі для прямокутника шляхом обчислення значення функції для кожного пікселя окремо. Однак, якщо нам потрібно обчислити суму для кількох прямокутних вікон, що перекриваються, ми можемо використати цілісне зображення та досягти постійної кількості операцій на прямокутник лише з лінійною кількістю попередньої обробки. Щоб обчислити інтегральне зображення, ми зберігаємо в кожному місці S(x,y) суму всіх членів I(x,y) ліворуч і над пікселем (x,y). Це досягається в лінійному часі за допомогою наступної формули для кожного пікселя (з урахуванням граничних випадків):

$$S(x,y) = I(x,y) + S(x-1,y) + S(x,y-1) - S(x-1,y-1), \quad (28)$$

де S – результат попередніх ітерацій цієї позиції пікселя, I – яскравість пікселя вихідного зображення. Якщо координати виходять за межі зображення, вони вважаються нульовими. Сутність цього методу представлена на рисунку 5.1.

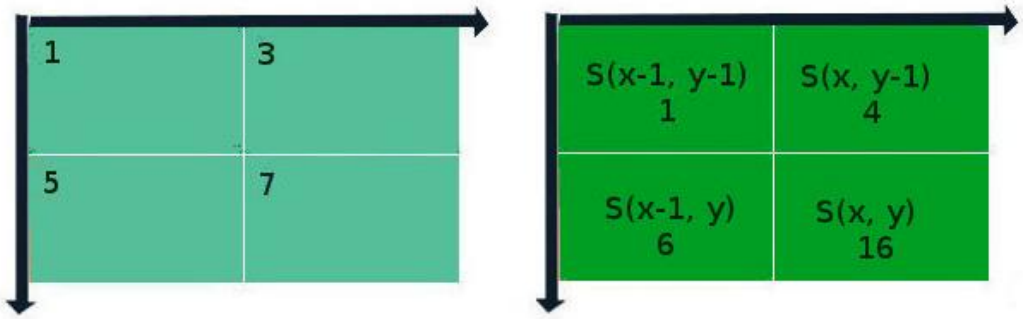

Рисунок 5.1 – Метод інтегральних зображень

Як тільки ми маємо інтегральне зображення, сума функції для будь-

якого прямокутника з верхнім лівим кутом (x1,y1) і нижнім правим кутом (x2,y2) можна обчислити за постійний час за допомогою наступного рівняння.

$$I(x,y) = S(x_2,y_2) - S(x_2,y_1-1) - S(x_1-1,y_2) + S(x_1-1,y_1-1) \quad (19)$$

Найкращій момент цього алгоритму полягає в тому, що один раз склавши інтегральну матрицю зображення, можна швидко обчислювати суму значень пікселів будь-якої прямокутної області в межах цього зображення. Нехай ABCD – цікава для нас прямокутна область, що зображена на рисунку 5.2.

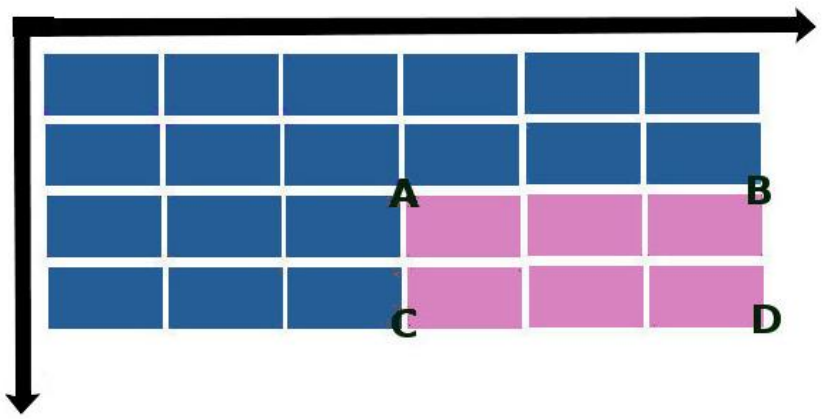

Рисунок 5.2 – Розглядаєма область

Тоді сумарна яскравість S у цій галузі обчислюється за такою формулою:

$$S(x,y) = S(A) + S(D) - S(B) - S(C), \quad (29)$$

де S(A), S(B), S(C) та S(D) – значення елементів інтегральної матриці у напрямку на „північний захід“ від перетинів сторін прямокутника. На рисунку 5.3 зображено необхідні елементи інтегральної матриці.

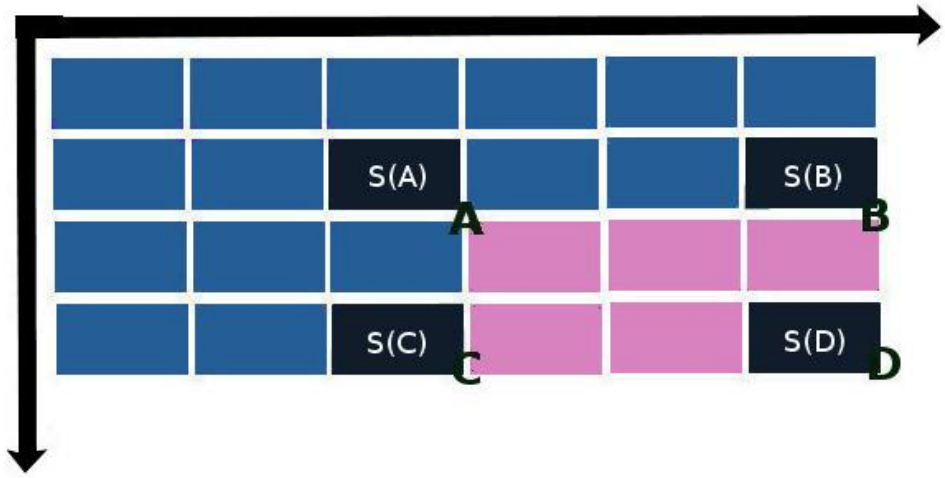

Рисунок 5.3 – Елементи інтегральної матриці

Нижче представлений код реалізації метода Бредлі на мові JavaScript.

Лістинг 2 - Алгоритм Бредлі на мові JavaScript

```
function computeAdaptiveThreshold(sourceImageData, ratio, callback) {
    var integral = buildIntegral_Gray(sourceImageData);

    var width = sourceImageData.width;
    var height = sourceImageData.height;
    var s = width >> 4;
    var sourceData = sourceImageData.data;
    var result = createImageData(width, height);
    var resultData = result.data;
    var resultData32 = new Uint32Array(resultData.buffer);

    var x = 0,
        y = 0,
        lineIndex = 0;

    for (y = 0; y < height; y++, lineIndex += width) {
        for (x = 0; x < width; x++) {

            var value = sourceData[(lineIndex + x) << 2];
            var x1 = Math.max(x - s, 0);
            var y1 = Math.max(y - s, 0);
            var x2 = Math.min(x + s, width - 1);
            var y2 = Math.min(y + s, height - 1);
            var area = (x2 - x1 + 1) * (y2 - y1 + 1);
```

```
            var localIntegral = getIntegralAt(integral, width,
x1, y1, x2, y2);
            if (value * area > localIntegral * ratio) {
                resultData32[lineIndex + x] = 0xFFFFFFFF;
            } else {
                resultData32[lineIndex + x] = 0xFF000000;
            }
        }
    }
    return result;
}
```

Імплементація функції для отримання інтегрального зображення представлене у лістингу 3.

Лістинг 3 - Отримання інтегрального зображення

```
function buildIntegral_Gray(sourceImageData) {
    var sourceData = sourceImageData.data;
    var width = sourceImageData.width;
    var height = sourceImageData.height;
    var integral = new Int32Array(width * height)
    var x = 0,
        y = 0,
        lineIndex = 0,
        sum = 0;
    for (x = 0; x < width; x++) {
        sum += sourceData[x << 2];
        integral[x] = sum;
    }

    for (y = 1, lineIndex = width; y < height; y++, lineIndex +=
width) {
        sum = 0;
        for (x = 0; x < width; x++) {
            sum += sourceData[(lineIndex + x) << 2];
            integral[lineIndex + x] = integral[lineIndex - width
+ x] + sum;
        }
    }
    return integral;
}
```

Функція для отримання інтегралу окремого пікселя зображення виглядає наступним чином.

Лістинг 4 - Отримання інтегральну окремого пікселя

```
function getIntegralAt(integral, width, x1, y1, x2, y2) {
    var result = integral[x2 + y2 * width];
    if (y1 > 0) {
        result -= integral[x2 + (y1 - 1) * width];
        if (x1 > 0) {
            result += integral[(x1 - 1) + (y1 - 1) * width];
        }
    }
    if (x1 > 0) {
        result -= integral[(x1 - 1) + (y2) * width];
    }
    return result;}
```

Перевагами методу Бредлі-Рота є простота реалізації та висока швидкість виконання. Метод добре працює з неоднорідним тлом.
З останньої гідності логічно випливає недолік методу Бредлі, а саме, погана чутливість до низькоконтрастних деталей зображення.

5.3 Метод Сауволи

Метод Сауволи приймає зображення в градаціях сірого як вхідні дані. Оскільки більшість зображень документів є кольоровими, потрібне перетворення кольорових зображень у градації сірого. Для цього ми вирішили використовувати класичну формулу яскравості, засновану на сприйнятті ока:

$$\text{Яскравість} = 0{,}299 \times \mathrm{R} + 0{,}587 \times \mathrm{G} + 0{,}114 \times \mathrm{B}. \quad (20)$$

На основі зображення в градаціях сірого Саувола запропонував обчислити поріг для кожного пікселя, використовуючи наступну формулу:

$$t(x,y) = m(x,y)\left[1 + k\left(\frac{s(x,y)}{R} - 1\right)\right], \quad (21)$$

де k є параметром, визначеним користувачем, m і s відповідно є середнім значенням і місцевим стандартним відхиленням, обчисленим у вікно розміром w з центром на поточному пікселі, а R — динамічний діапазон стандартного відхилення (R = 128 для 8-бітових зображень рівня сірого). Розмір вікна, що використовується для обчислення m і s, залишається визначеним користувачем в оригінальному документі. Данна формула ґрунтується на припущенні, що пікселі тексту мають значення, близькі до чорного (відповідно пікселі фону мають значення, близькі до білого).

У поєднанні з оптимізацією, як інтегральні зображення, однією з головних переваг методу Сауволи є його обчислювальна ефективність. Він може працювати менш ніж за 60 мс з документами формату A4 300 dpi на сучасному комп'ютері. Ще одна перевага полягає в тому, що він відносно добре працює на шумних і розмитих документах. Завдяки формулі бінаризації користувач повинен надати два параметри (w, k). Для їх оцінки були запропоновані деякі методики. Бадекас і Папамаркос [5] стверджують, що w = 14 і k = 0,34 є найкращим компромісом для наскрізного видалення та якості пошуку об'єктів у класичних документах. Рангоні та ін. [8] базувався на дослідженні параметрів якості результату оптичного розпізнавання символів (OCR) і знайшов w = 60 і k = 0,4. Сезгін і Санкур [1] і Саувола і Пієтікайнен [4] використовували w = 15 і k = 0,5. Налаштування цих параметрів зазвичай вимагає попереднього знання набору документів, щоб отримати найкращі результати. Тому в дослідницькому співтоваристві немає консенсусу щодо значень цих параметрів.

Нижче представлений код реалізації метода Саувола на мові JavaScript.

Лістинг 5 - Реалізація методу Саувола

```
function sauvolaMethod=(w,h,window)=>{
```

```
let m;
let s;
for (int i; i<= w; i++){
   for (int j; j<= h; j++){
const size =window/2
  if(j>(h-size) and i>(w-size)){
     m   =   (b[j,i]+b[j-size,i-size]-b[j,i-size]-b[j-size,i])   /
(window * window)
     s  =  (c[j,i]+c[j-size,i-size)]-c[j,i-size)]-c[j-size,i])  /
(window * window)
        }
else if (i > (w - size) and j < (h - size)){
     m  =  (b[j  +size,i]+  b[j-size,i-size]-b[j+size,i-size]-b[j-
size,i]) / (window * window)
     s   =   (c[j+size,i]+c[j-size,i-size]-c[j+size,i-size]-c[j  -
size, i]) / (window * window)
}
  else if (j>(h-size) and i<(w-size)){
 m  =  (  b[j,i+size]  +  b[j-size,  i-size  -  b[j,  i-size  -b[j-size
+size] ) /(window*window)
   s = (  c[j, i+size] + c[j-size, i-size] - c[j ,i-size] - c[j-
size, i+size])/(window*window)
}

else if(j<(h-size ) and i<(w-size)){
           m=(b[j+size,i+size]+b[j-size,i-size]-b[j+size,i-
size)]-b[j-size,i+size])/(window*window)
            s=(c[j+size,i+size]+c[j-size,i-size]-c[j+size,i-
size]-c[j-size,i+size])/(window*window)
}
        const  var = ((s)- (pow((m), 2)))/(window*window)
        const std = sqrt(abs(var))
        const threshold = m * (1 + (k * ((std / r) - 1)))
      return threshold
}
}
}
```

6 МОДЕЛЬ НАВЧАННЯ НЕЙРОННОЇ МЕРЕЖІ

6.1 Опис обраної технології

Машинне навчання – це складна дисципліна, але впровадження моделей машинного навчання набагато менш складне, ніж це було раніше, завдяки структурам машинного навчання, таким як Google TensorFlow, які спрощують процес отримання даних, навчання моделей, надання прогнозів і уточнення майбутніх результатів.

TensorFlow, створена командою Google Brain і вперше опублікована в 2015 році, є бібліотекою з відкритим кодом для чисельних обчислень і великомасштабного машинного навчання. Технологія об’єднує низку моделей і алгоритмів машинного та глибокого навчання (так звані нейронні мережі) і робить їх корисними за допомогою загальних програмних метафор. Він використовує Python або JavaScript, щоб забезпечити зручний інтерфейсний API для створення додатків, одночасно виконуючи ці додатки на високопродуктивній C++.

Бібліотека, яка конкурує з такими фреймворками, як PyTorch і Apache MXNet, може навчати та запускати глибокі нейронні мережі для класифікації рукописних цифр, розпізнавання зображень, вбудовування слів, рекурентних нейронних мереж, моделей послідовності для машинного перекладу, обробки природної мови та моделювання на основі PDE (рівняння в частинних похідних). Найкраще те, що TensorFlow підтримує масштабне прогнозування виробництва з тими самими моделями, що використовуються для навчання. Фреймворк також має широку бібліотеку попередньо навчених моделей, які можна використовувати у власних проектах. Ви також можете використовувати код із саду моделей TensorFlow як приклади найкращих практик для навчання власних моделей.

TensorFlow дозволяє розробникам створювати графіки потоків даних

— структури, які описують, як дані переміщуються через графік або ряд вузлів обробки. Кожен вузол на графі представляє математичну операцію, а кожне з'єднання або ребро між вузлами є багатовимірним масивом даних або тензором.

Програми TensorFlow можна запускати майже на будь-якій зручній цілі: локальній машині, кластері в хмарі, пристроях iOS і Android, процесорах або графічних процесорах. Якщо ви використовуєте власну хмару Google, ви можете запустити TensorFlow на спеціальному модулі обробки TensorFlow (TPU) Google для подальшого прискорення. Однак отримані моделі, створені TensorFlow, можна розгорнути на будь-якому пристрої, де вони використовуватимуться для надання прогнозів.

TensorFlow 2.0, випущений у жовтні 2019 року, багато в чому оновив фреймворк на основі відгуків користувачів, щоб зробити його легшим для роботи (як приклад, за допомогою відносно простого Keras API для навчання моделей) і більш продуктивним. Розподілене навчання легше проводити завдяки новому API, а підтримка TensorFlow Lite дає змогу розгортати моделі на більшій різноманітності платформ.

Навчену модель можна використовувати для доставки прогнозів як служби через контейнер Docker за допомогою REST або gRPC API. Для більш складних сценаріїв обслуговування ви можете використовувати Kubernetes.

6.1.1 Використання TensorFlow з JavaScript

Python є найпопулярнішою мовою для роботи з TensorFlow і машинного навчання в цілому. Але JavaScript тепер також є першокласною мовою для TensorFlow, і одна з величезних переваг JavaScript полягає в тому, що він працює будь-де, де є веб-браузер.

TensorFlow.js, як називається бібліотека JavaScript TensorFlow, використовує API WebGL для прискорення обчислень за допомогою будь-

яких графічних процесорів, доступних у системі. Для виконання також можна використовувати бек-енд WebAssembly, і він швидше, ніж звичайний бек-енд JavaScript, якщо ви працюєте лише на центральному процесорі, хоча найкраще використовувати графічні процесори, коли це можливо. Попередньо створені моделі дають змогу почати роботу з простими проектами, щоб дати вам уявлення про те, як все працює.

6.1.2 Навіщо використовувати TensorFlow

Найбільша перевага TensorFlow для розробки машинного навчання – це абстракція. Замість того, щоб мати справу з дрібними деталями реалізації алгоритмів або з'ясуванням належних способів зв'язати вихідні дані однієї функції з вхідними даними іншої, розробник може зосередитися на загальній логіці програми. TensorFlow подбає про деталі за лаштунками.

TensorFlow пропонує додаткові зручності для розробників, яким потрібно налагодити додатки і отримати самоаналіз. Кожну операцію графа можна оцінювати та змінювати окремо та прозоро, замість того, щоб будувати весь графік як один непрозорий об'єкт і оцінювати його весь одразу. Цей так званий «режим активного виконання», наданий як опція в старіших версіях бібліотеки, тепер є стандартним.

Даний фреймворк також отримує багато переваг завдяки підтримці комерційної організації. Google сприяв швидкому розвитку проекту та створив багато важливих пропозицій, які полегшують розгортання та використання TensorFlow.

6.1.3 Навчання детермінованої моделі за допомогою TensorFlow

Кілька деталей реалізації бібліотеки ускладнюють отримання повністю детермінованих результатів навчання моделі для деяких навчальних завдань. Іноді модель, навчена на одній системі, дещо відрізнятиметься від моделі,

навченої на іншій, навіть якщо вони отримують однакові дані. Одна з причин полягає в тому, як і куди вводяться випадкові числа; інша пов’язана з певною недетермінованою поведінкою під час використання GPU. Гілка TensorFlow 2.0 має можливість увімкнути детермінізм у всьому робочому процесі за допомогою кількох рядків коду. Однак ця функція має низьку продуктивність, і її слід використовувати лише під час налагодження робочого процесу.

6.2 Опис розробленої моделі

Було розроблено модель навчання простої згорткової нейронної мережі (CNN) для бінаризації зображень за допомогою TensorFlow бібліотеки.

Було використано три згорткові шари:

– перший шар матиме фільтри 32-3 x 3;

– другий шар матиме фільтри 64-3 x 3;

– третій шар матиме фільтри 128-3 x 3.

Крім того, існує три шари максимального об’єднання, кожен розміром 2 x 2.

Необхідно почати з визначення навчальних ітерацій epochs і розміру пакета batchSize. Всі ці параметри є гіперпараметрами, і вони не мають фіксованих значень, оскільки вони відрізняються для кожної моделі.

Тим не менш, ось що зазвичай можна очікувати:

– ітерації навчання вказуються кількість разів, скільки необхідно разів навчити мережу;

– розмір пакету означає, що тренувальні зображення будуть розділені на фіксований пакет, і для кожного пакету буде взято фіксовану кількість зображень для їх навчання. Рекомендовано використовувати розмір партії в ступені 2. Оскільки число фізичного процесора часто є ступенем 2, використання кількох віртуальних процесорів, відмінних від ступеня 2, призводить до низької продуктивності. Крім того, використання дуже

великого розміру пакета може призвести до помилок пам’яті, тому необхідно переконатися, що машина, на якій буде запускатися код, має достатньо оперативної пам’яті для обробки вказаного розміру пакета.

Лістинг 6 - Визначення гіперпараметрів

```
const batchSize = 100
const epochsValue = 300
```

У лістингу 7 коду визначено базу згортку за допомогою загального шаблону: стек шарів Conv2D і MaxPooling2D.

Як вхідні дані CNN приймає тензори форми (image_height, image_width, color_channels), ігноруючи розмір партії. Параметр color_channels відноситься до (R,G,B). У цьому прикладі ви налаштуєте свій CNN на обробку вхідних даних форми (200, 200, 3). Ви можете зробити це, передавши аргумент input_shape своєму першому шару.

Лістинг 7 - Визначення моделі навчання

```
const buildModel = function (trainData) {
   const model = tf.sequential();
   // add the model layers
   model.add(tf.layers.conv2d({
       inputShape: [200,200,3],
       filters: 8,
       kernelSize: 1,
       padding: 'same',
       activation: 'relu'
   }));
   model.add(tf.layers.maxPooling2d({
       poolSize: 2,
       strides: 2
   }));
   model.add(tf.layers.conv2d({
       filters: 16,
       kernelSize: 5,
       padding: 'same',
       activation: 'relu'
   }));
```

```
    model.add(tf.layers.maxPooling2d({
        poolSize: 3,
        strides: 3
    }));

    model.add(tf.layers.flatten());
    model.add(tf.layers.dense({
        units: 1
    }));
//Custom layer for binarization
    model.add(new     TresholdingOtsuLayer({pixelsNumber:     1.5,
inputShape:[200,200,3]}));

    // compile the model
    model.compile({
        optimizer: 'sgd',
        loss: 'meanSquaredError',
        metrics: ['accuracy']
    });

    return model;
};
```

Розглянемо детальніше кожен із шарів.

6.2.1 Шар згортки Conv2D

Цей рівень створює ядро згортки, яке згортається разом із вхідними даними шару для створення тензора виходів. Якщо use_bias має значення True, вектор зміщення створюється та додається до виходів. Нарешті, якщо активація не None, вона також застосовується до виходів.

У розробленій моделі було використано функцію “relu” в якості функції активації.

Лістинг 8 - Визначення першого шару ядра згортки

```
 model.add(tf.layers.conv2d({
        inputShape: [200,200,3],
        filters: 8,
        kernelSize: 1,
        padding: 'same',
        activation: 'relu'
```

```
}));
```

Параметр filters визначає розмірність вихідного простору, тобто кількість вихідних фільтрів у згортці. В нашому випадку було обрано 8 фільтрів. Параметр padding відповідає за доповнення у вхідні данні. Значення "same" призводить до доповнення нулями рівномірно ліворуч/праворуч або вгору/вниз від введення.

Лістинг 9 - Визначення другого шару ядра згортки

```
model.add(tf.layers.conv2d({
       filters: 16,
       padding: 'same',
       activation: 'relu'
}));
```

Для другого шару ядра згортки незмінними залишилися майже всі параметри окрім filters. Даний параметр було збільшено у 2 рази.

6.2.2 Підвибірковий шар MaxPooling2D

Даний шар зменшує дискретизацію вхідного сигналу вздовж його просторових розмірів (висота та ширина), беручи максимальне значення у вікні введення (розміру, визначеному параметром poolSize) для кожного каналу вхідного сигналу. Вікно зсувається кроками вздовж кожного виміру. В нашому випадку, було обрано для першого шару poolSize рівне 2, а для другого 3 відповідно. У лістингах 10 та 11 представлена реалізація підвибіркових шарів.

Лістинг 10 - Перший підвибірковий шар

```
model.add(tf.layers.maxPooling2d({
       poolSize: 2,
       strides: 2
```

```
   }));
```

Лістинг 11 - Другий підвибірковий шар

```
  model.add(tf.layers.maxPooling2d({
       poolSize: 3,
       strides: 3
   }));
```

6.2.3 Повнозв'язковий шар

Лістинг 12 - Другий підвибірковий шар

```
  model.add(tf.layers.dense({
   units: 1
}));
```

У лістингу 12 представлено повнозв’язний шар. Він використовується для створення повністю пов’язаних шарів, у яких кожен вихід залежить від кожного входу. Параметр units визначає розмірність вихідного простору.

6.2.4 Компіляція створеної моделі

Компіляція моделі виконується за допомогою методу compile. Дана функція налаштовує та створює модель для процесу навчання та оцінювання. Викликаючи метод compile ми готуємо модель з оптимізатором, втратами та метриками.

Лістинг 13 - Компіляція моделі із параметрами

```
model.compile({
       optimizer: 'sgd',
       loss: 'meanSquaredError',
       metrics: ['accuracy']
   });
```

Якщо надати деякі навчальні дані, наша ненавчена мережа, швидше за все, не дасть правильної відповіді. Функція втрат вимірює ступінь відмінності отриманого результату від цільового значення, і саме функцію втрат ми хочемо мінімізувати під час навчання. Щоб обчислити втрати, ми робимо прогноз, використовуючи вхідні дані нашої даної вибірки даних, і порівнюємо їх із справжнім значенням мітки даних. В нашому випадку функція втрати — середня квадратична помилка (СКП). Щоб обчислити СКП, необхідно взяти різницю між прогнозами моделі та основною правдою, звести її та усереднити для всього набору даних. Значення цієї функції ніколи не буде негативним, оскільки вона завжди зводить помилки у квадрат.

У якості функції оптимізації було обрано стохастичний градієнтний спуск - sgd. Взагалі оптимізація – це процес коригування параметрів моделі для зменшення помилок моделі на кожному кроці навчання. Алгоритми оптимізації визначають, як виконується цей процес. Вся логіка оптимізації інкапсульована в об’єкт оптимізатора.

6.2.5 Власний шар бінаризації зображення

Було створено власний шар бінаризації зображення. Для реалізації цього необхідно було успадкуватися від основного класу бібліотеки, а саме від tf.layers.Layer і перевизначити головні методи даного класу: build, call, getConfig.

У лістингу 14 представлено конструктор класу власного шару

Лістинг 14 - Конструктор класу TresholdingLayer

```
constructor(config) {
   super(config);
   this.pixelsNumber = config.pixelsNumber;
}
```

Аргумент pixelsNumber представляє собою кількість пікселів у

зображені, що обробляється.

Реалізація методу build представлена у лістингу 15. Цей метод викликається, коли власний об’єкт шару підключено до вперше. Тут створюються ваги.

Лістинг 15 - Реалізація build методу

```
build(inputShape) {
 this.x    =    this.addWeight('x',    inputShape,    'float32',
tf.initializers.ones());
}
```

Функція getConfig генерує об’єкт JSON, який використовується під час збереження та завантаження об’єкта спеціального шару.

Лістинг 16 - Реалізація getConfig методу

```
getConfig() {
   const config = super.getConfig();
   Object.assign(config, {pixelsNumber: this.pixelsNumber});
   return config;
}
```

Найважливіший метод розробленого класу — call. Саме він містить фактичне числове обчислення шару. По своїй суті, це фунція яка виконує бінарізацію зображення. Можна використати будь який метод із реалізованих у розділі 4. У лістингу 17 преведено приклад реалізації метода Отцу.

Лістинг 17 - Реалізація call методу

```
call(input) {
   getHistogram(input)
   const otsu=()=>{
       const sum =0
       let sumB =0
       let wB =0
       let wF= 0
       let mB=0;
```

```
    let mF=0;
    let max=0;
    let between =0;
    let threshold = 0
    for (let i = 0; i < 256; ++i) {
        wB += histogram[i];
        if (wB === 0)
            continue;
        wF = this.pixelsNumber - wB;
        if (wF === 0)
            break;
        sumB += i * histogram[i];
        mB = sumB / wB;
        mF = (sum - sumB) / wF;
        between = wB * wF * Math.pow(mB - mF, 2);
        if (between > max) {
            max = between;
            threshold = i;
        }
    }
    return threshold;
}
```

7 ТЕСТУВАННЯ РОЗРОБЛЕНОЇ МОДЕЛІ

7.1 Тестування розроблених алгоритмів

7.1.1 Тестування метода Отцу

Було протестовано код, який був представлений у розділі 5.1. Було отримано значення для порогу 108.

На рисунку номер показано зображення, до бінарізаціїї після застосування метода Отцу відповідно.

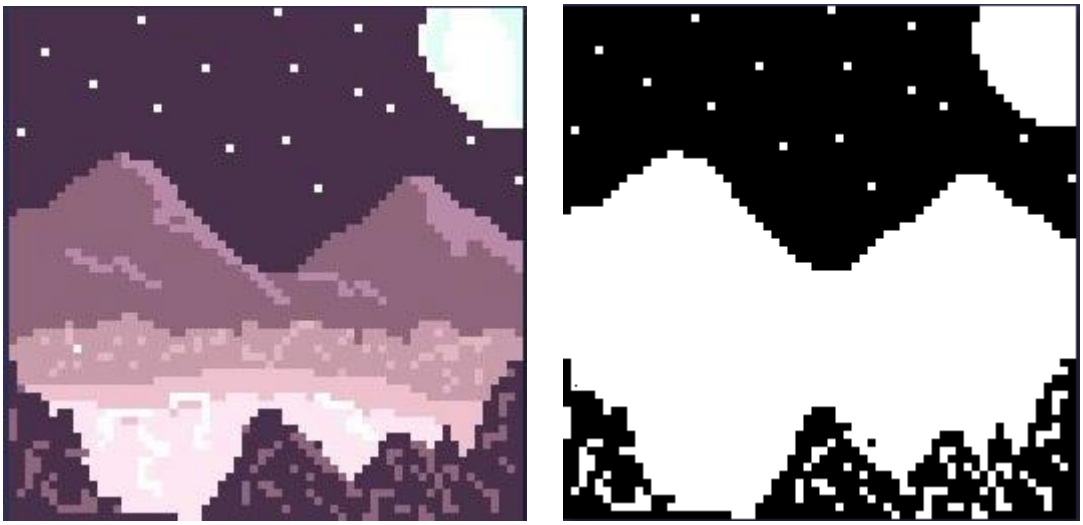

Рисунок 7.1 – Результат роботи метода Отцу

Як зазначалось раніше, даний метод залежить від рівню інтенсивності. У даному випадку шуми і тіні зображення не відобразились у повній мірі.

7.1.2 Тестування метода Бредлі

Метод Бредлі показав кращий результат ніж метод Отцу. Завдяки даному алгоритму було отримано поріг із значенням 123. Результат

бінаризації представлено на рисунку 7.2.

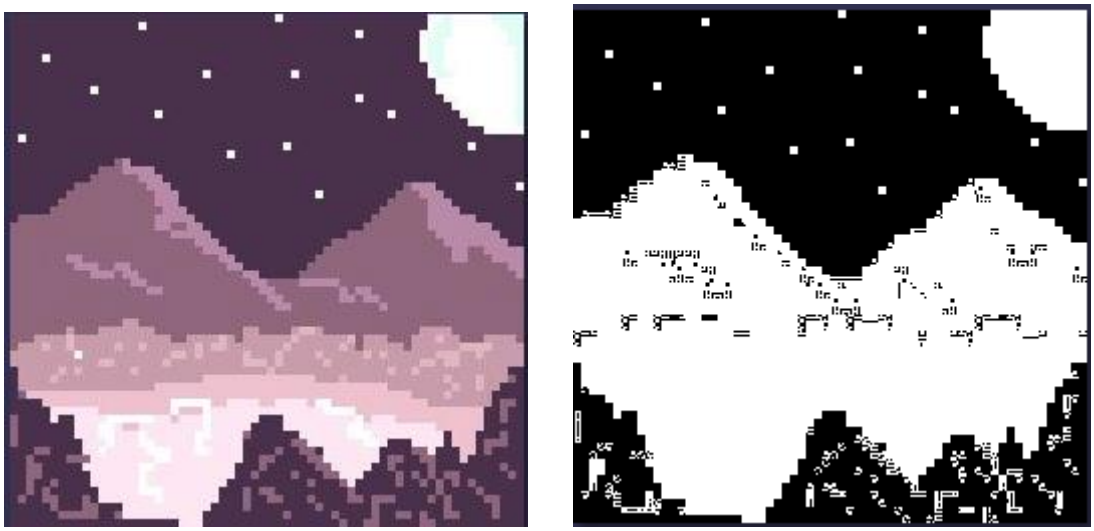

Рисунок 7.2 – Результат роботи метода Бредлі

Неозброєнім оком видно, що даний метод набагато краще впорався із тінями та шумами на зображені, ніж метод Отцу. Але тіні світлих тонів не зміг розпізнати.

7.1.3 Тестування метода Сауволи

Алгоритм Сауволи показав середній результат, було отримано поріг, який дорівнює 117.

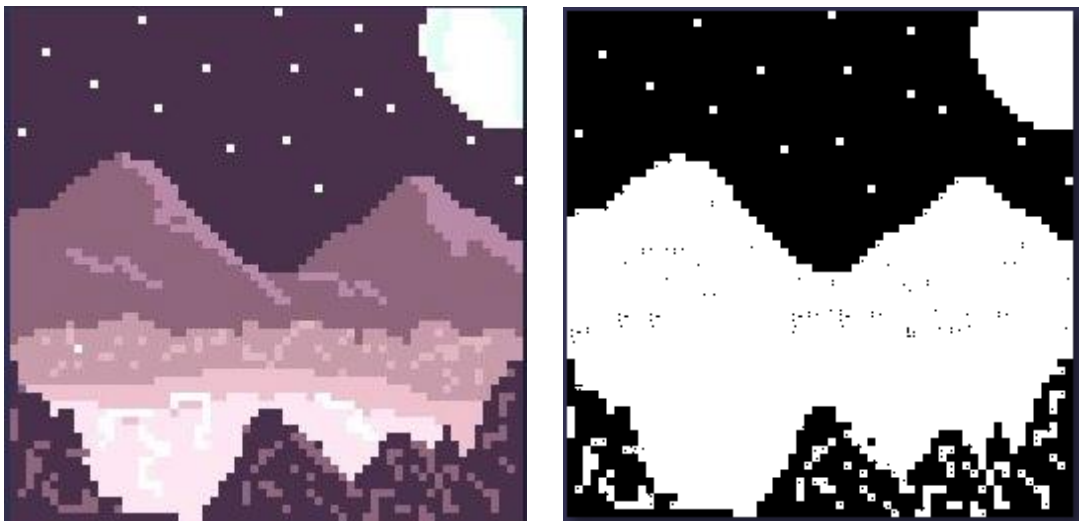

Рисунок 7.3 – Результат роботи метода Сауволи

Даний метод набагато краще впорався із тінями на відміну від метода Отцу. На результуючому зображені можна розгледіти тіні темних тонів, але набагато гірше ніж у методі Бредлі. Із тінями світлих відтінків алгоритм Сауволи також не впорався.

7. 2 Тестування моделі навчання

На рисунку 7.4 показана детальна інформація про розроблену модель.

```
_________________________________________________________________________________________
Layer (type)                Input Shape               Output shape              Param #
=========================================================================================
conv2d_Conv2D1 (Conv2D)     [[null,200,200,3]]        [null,200,200,8]          32
_________________________________________________________________________________________
max_pooling2d_MaxPooling2D1 [[null,200,200,8]]        [null,100,100,8]          0
_________________________________________________________________________________________
conv2d_Conv2D2 (Conv2D)     [[null,100,100,8]]        [null,100,100,16]         3216
_________________________________________________________________________________________
max_pooling2d_MaxPooling2D2 [[null,100,100,16]]       [null,33,33,16]           0
_________________________________________________________________________________________
flatten_Flatten1 (Flatten)  [[null,33,33,16]]         [null,17424]              0
_________________________________________________________________________________________
dense_Dense1 (Dense)        [[null,17424]]            [null,1]                  17425
_________________________________________________________________________________________
tresholding_otsu_layer_Tres [[null,1]]                [null,1]                  1
=========================================================================================
Total params: 20674
Trainable params: 20674
Non-trainable params: 0
```

Рисунок 7.4 – Інформація про шари розробленої моделі

На рисунках 7.5-7.7 представлено результат навчання нейронної мережі із втратами та точністю визначення пікселей.

```
Epoch 1/300
5/5 [==============================] - 0s 52ms/step - loss: 456.8831 - accuracy: 0.2746 - v
al_loss: 470.7413 - val_accuracy: 0.2500
Epoch 2/300
5/5 [==============================] - 0s 8ms/step - loss: 432.0455 - accuracy: 0.2746 - va
l_loss: 444.4698 - val_accuracy: 0.2500
Epoch 3/300
5/5 [==============================] - 0s 8ms/step - loss: 407.1481 - accuracy: 0.2746 - va
l_loss: 418.3409 - val_accuracy: 0.2500
Epoch 4/300
5/5 [==============================] - 0s 8ms/step - loss: 382.6049 - accuracy: 0.2746 - va
l_loss: 392.2401 - val_accuracy: 0.2500
Epoch 5/300
5/5 [==============================] - 0s 8ms/step - loss: 358.6322 - accuracy: 0.2746 - va
l_loss: 366.3926 - val_accuracy: 0.2500
Epoch 6/300
5/5 [==============================] - 0s 8ms/step - loss: 334.6218 - accuracy: 0.2746 - va
l_loss: 341.2526 - val_accuracy: 0.2500
Epoch 7/300
5/5 [==============================] - 0s 8ms/step - loss: 311.6069 - accuracy: 0.2746 - va
l_loss: 317.1813 - val_accuracy: 0.2500
Epoch 8/300
5/5 [==============================] - 0s 8ms/step - loss: 289.4217 - accuracy: 0.2746 - va
l_loss: 295.3412 - val_accuracy: 0.2500
Epoch 9/300
5/5 [==============================] - 0s 8ms/step - loss: 269.0434 - accuracy: 0.2746 - va
l_loss: 274.5162 - val_accuracy: 0.2500
Epoch 10/300
5/5 [==============================] - 0s 8ms/step - loss: 249.1496 - accuracy: 0.2746 - va
l_loss: 254.2753 - val_accuracy: 0.2500
Epoch 11/300
5/5 [==============================] - 0s 10ms/step - loss: 230.1510 - accuracy: 0.2746 - v
al_loss: 234.0888 - val_accuracy: 0.2500
```

Рисунок 7.5 – Інформація про втрати та точність навчання з 1 до 11 епохи

```
Epoch 157/300
5/5 [==============================] - 0s 8ms/step - loss: 1.4621 - accuracy: 0.6197 - val_
loss: 1.3945 - val_accuracy: 0.6111
Epoch 158/300
5/5 [==============================] - 0s 8ms/step - loss: 1.4777 - accuracy: 0.6690 - val_
loss: 1.3915 - val_accuracy: 0.6667
Epoch 159/300
5/5 [==============================] - 0s 7ms/step - loss: 1.4174 - accuracy: 0.6690 - val_
loss: 1.4426 - val_accuracy: 0.6389
Epoch 160/300
5/5 [==============================] - 0s 8ms/step - loss: 1.4466 - accuracy: 0.6197 - val_
loss: 1.4397 - val_accuracy: 0.6667
Epoch 161/300
5/5 [==============================] - 0s 8ms/step - loss: 1.4621 - accuracy: 0.6690 - val_
loss: 1.4194 - val_accuracy: 0.6389
Epoch 162/300
5/5 [==============================] - 0s 7ms/step - loss: 1.4661 - accuracy: 0.6901 - val_
loss: 1.5945 - val_accuracy: 0.6389
Epoch 163/300
5/5 [==============================] - 0s 9ms/step - loss: 1.4330 - accuracy: 0.6620 - val_
loss: 1.4470 - val_accuracy: 0.6389
Epoch 164/300
5/5 [==============================] - 0s 8ms/step - loss: 1.4068 - accuracy: 0.6972 - val_
loss: 1.3772 - val_accuracy: 0.6389
Epoch 165/300
5/5 [==============================] - 0s 8ms/step - loss: 1.4028 - accuracy: 0.6479 - val_
loss: 1.3690 - val_accuracy: 0.6111
Epoch 166/300
5/5 [==============================] - 0s 8ms/step - loss: 1.4010 - accuracy: 0.6338 - val_
loss: 1.3519 - val_accuracy: 0.6667
Epoch 167/300
5/5 [==============================] - 0s 8ms/step - loss: 1.3845 - accuracy: 0.6831 - val_
loss: 1.3665 - val_accuracy: 0.6389
```

Рисунок 7.6 – Інформація про втрати та точність навчання з 157 до 167 епохи

```
Epoch 291/300
5/5 [==============================] - 0s 8ms/step - loss: 0.9536 - accuracy: 0.7817 - val_
loss: 1.0380 - val_accuracy: 0.7222
Epoch 292/300
5/5 [==============================] - 0s 8ms/step - loss: 1.0057 - accuracy: 0.7042 - val_
loss: 0.9648 - val_accuracy: 0.6944
Epoch 293/300
5/5 [==============================] - 0s 8ms/step - loss: 0.9700 - accuracy: 0.7746 - val_
loss: 0.9456 - val_accuracy: 0.6667
Epoch 294/300
5/5 [==============================] - 0s 7ms/step - loss: 0.9427 - accuracy: 0.7887 - val_
loss: 0.9296 - val_accuracy: 0.7500
Epoch 295/300
5/5 [==============================] - 0s 7ms/step - loss: 0.9524 - accuracy: 0.7394 - val_
loss: 0.9034 - val_accuracy: 0.6667
Epoch 296/300
5/5 [==============================] - 0s 9ms/step - loss: 0.9580 - accuracy: 0.8099 - val_
loss: 0.9081 - val_accuracy: 0.6944
Epoch 297/300
5/5 [==============================] - 0s 7ms/step - loss: 0.9444 - accuracy: 0.7535 - val_
loss: 0.9835 - val_accuracy: 0.7500
Epoch 298/300
5/5 [==============================] - 0s 7ms/step - loss: 0.9432 - accuracy: 0.7465 - val_
loss: 0.9114 - val_accuracy: 0.6944
Epoch 299/300
5/5 [==============================] - 0s 8ms/step - loss: 0.9345 - accuracy: 0.7746 - val_
loss: 0.9765 - val_accuracy: 0.6944
Epoch 300/300
5/5 [==============================] - 0s 7ms/step - loss: 0.9263 - accuracy: 0.7887 - val_
loss: 0.9635 - val_accuracy: 0.6944
```

Рисунок 7.7 – Інформація про втрати та точність навчання з 291 до 300 епохи

Ці дані демонструють, що навчання нейронної мережі проходить успішно. Розмір втрат зменшується, а точність у порівнянні із першою епохою зросла у 3 рази.

На рисунку 7.8 представлені детальні графіки зміни значення втрат і точності на відрізку 300 епох.

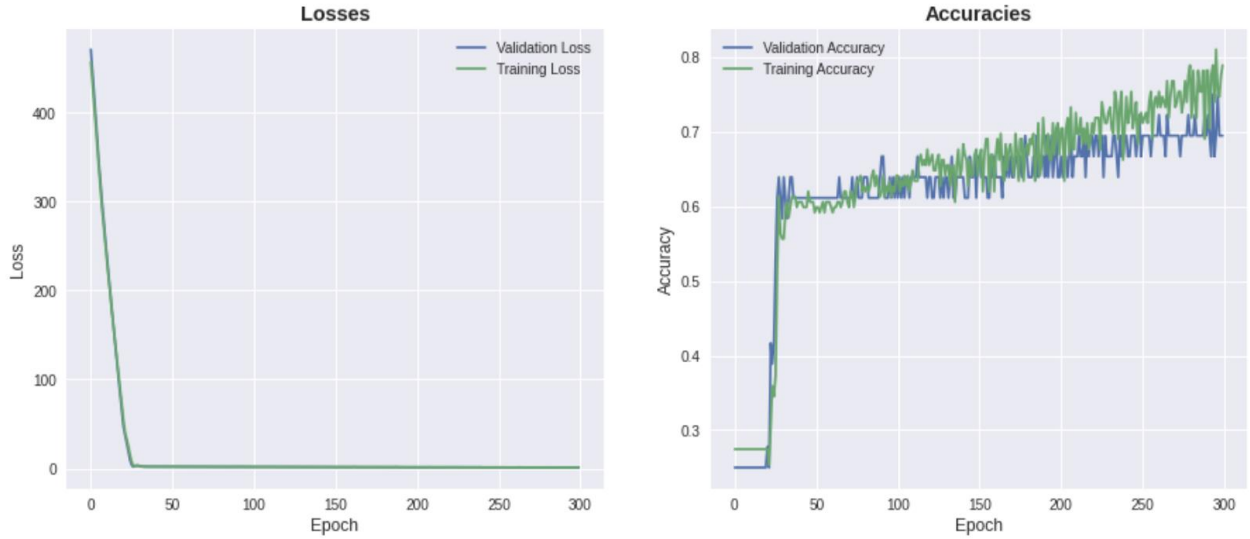

Рисунок 7.8 – Графік зміни значення параметрів втрат і точності протягом 300 епох навчання

ВИСНОВКИ

В результаті виконання роботи були розглянуті методи бінаризації зображень. Представлені їх переваги і недоліки. Найпопулярніші алгоритми були реалізовані на мові програмування JavaScript та протестовані на практиці.

Невдачі в процесі бінаризації можуть призвести до спотворень, таких, як розриви в лініях, втрата значущих деталей, порушення цілісності об'єктів, поява шуму та непередбачуване спотворення символів через неоднорідність фону.

Різні методи бінаризації мають свої слабкі місця: так, наприклад, метод Отцу може призводити до втрати дрібних деталей та «злипання» довколишніх символів, а метод Ніблека грішить появою хибних об'єктів у разі неоднорідностей фону з низькою контрастністю.

З цього можна зробити висновок, що кожен із розглянутих методів повинен бути застосований у певній області, а також для кожної області можуть бути розроблені найбільш підходящі методи.

Також було розглянуто структуру нейронної мережі і методи її навчання. Визначено основні недоліки кожного із методу. Досліджено згорткову нейронну мережу, детально розглянуто основні її шари.

Реалізовано на практиці модель для навчання згорткової нейронної мережі із власним шаром для бінаризації зображання.

Було протестовано розроблену модель навчання на практиці. Представлені графіки змін таких параметрів як втрати і точність із плином епох.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Порівняльне дослідження методів адаптивної бінаризації в задачі автоматизованого аналізу зображень клітин імуноцитохімії, Кравцова Т.О. ,Молодіжний науково-технічний вісник, 2015.
2. Бінарізація чорно-білих зображень: стан та перспективи розвитку, Федоров А. [Електронний ресурс]. Режим доступу: http://itclaim.ru/Library/Books/ITS/wwwbook/ist4b/its4 /fyodorov.htm/.
3. Хаустов П.А. Алгоритми розпізнавання рукописних символів на основі побудови структурних моделей // КО. [Електронний ресурс], 2017. №1. Режим доступу: http://cyberleninka.ru/article/n/algoritmy-raspoznavaniya-rukopisnyh-simvolov-na-osnove-postroeniyastrukturnyh-modeley/.
4. Нейронні мережі: повний курс, Саймон Хайкін, Вильямс, 2008.
5. Нейронные сети для начинающих. Часть 2 [Електронний ресурс], 2017. №1. Режим доступу: https://habr.com/ru/post/313216/.
6. Improvement of Image Binarization Methods Using Image Preprocessing with Local Entropy Filtering for Alphanumerical Character Recognition Purposes, Hubert Michalak and Krzysztof Okarma , 4 June 2019
7. Bradley, D.; Roth, G. Adaptive thresholding using the integral image. J. Graph. Tools 2007, 12, 13–21.
8. BERNSEN, J. 1986. Dynamic thresholding of gray-level images. In Int. Conf. Pattern Recognition, vol. 2, 1251– 1255.
9. Kapur, J.; Sahoo, P.; Wong, A. A new method for gray-level picture thresholding using the entropy of the histogram. Comput. Vis. Graph. Image Process. 1985, 29, 273–285.
10. Lech, P.; Okarma, K.; Wojnar, D. Binarization of document images using the modified local-global Otsu and Kapur algorithms. Przegl ˛ad Elektrotechniczny 2015, 91, 71–74.

11. Gatos, B.; Pratikakis, I.; Perantonis, S. Adaptive degraded document image binarization. Pattern Recognit. 2006, 39, 317–327.

12. Michalak, H.; Okarma, K. Fast adaptive image binarization using the region based approach. In Artificial Intelligence and Algorithms in Intelligent Systems; Silhavy, R., Ed.; Springer International Publishing: Cham, Switzerland, 2019; Volume 764, AISC, pp. 79–90.

13. Bataineh, B.; Abdullah, S.N.H.S.; Omar, K. An adaptive local binarization method for document images based on a novel thresholding method and dynamic windows. Pattern Recognit. Lett. 2011, 32, 1805–1813.

14. Khurshid, K.; Siddiqi, I.; Faure, C.; Vincent, N. Comparison of Niblack inspired binarization methods for ancient documents. In Proceedings of the Document Recognition and Retrieval XVI, San Jose, CA, USA, 18–22 January 2009; Volume 7247, pp. 7247:1–7247:9.

15. Kulyukin, V.; Kutiyanawala, A.; Zaman, T. Eyes-free barcode detection on smartphones with Niblack's binarization and Support Vector Machines. In Proceedings of the 16th International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV'2012), Las Vegas, NV, USA, 16–19 July 2012; Volume 1, pp. 284–290.

16. Shrivastava, A.; Srivastava, D.K. A review on pixel-based binarization of gray images. In ICICT 2015; Springer: Singapore, 2016; Volume 439, pp. 357–364.

17. Stathis, P.; Kavallieratou, E.; Papamarkos, N. An evaluation technique for binarization algorithms. J. UCS 2008, 14, 3011–3030. [CrossRef]

18. TensorFlow [Електронний ресурс] Режим доступу: https://js.tensorflow.org/api/latest/?_gl=1*1205gj2*_ga*NjkzODI4ODYxLjE2Njk0ODYyNTI.*_ga_W0YLR4190T*MTY3MDY3Mjg4NS4yNS4xLjE2NzA2NzU1MjEuMC4wLjA.

19. Lecun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11):2278 - 2324

20. Abdel-Hamid O, Mohamed A R, Jiang H, et al. Applying Convolutional

Neural Networks concepts to hybrid NN-HMM model for speech recognition[C]// Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on. IEEE, 2012:4277 - 4280

21. Sun Y, Wang X, Tang X. Deep Convolutional Network Cascade for Facial Point Detection[C]/ / Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on. IEEE, 2013:3476-3483.