НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

«На правах рукопису»
УДК 004.93'1

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«06» червня 2022 р.

Магістерська дисертація

на здобуття ступеня магістра
за освітньо-науковою програмою «Інтегровані інформаційні системи»
зі спеціальності 126 «Інформаційні системи та технології»
на тему: «Інтелектуальна система розпізнавання образів
на основі згорткових нейронних мереж»

Виконав:
студент VI курсу, групи ІА-01мн
Ткаченко Максим Сергійович _____

Керівник:
к.т.н., доцент, доцент кафедри ІСТ
Сокульський Олег Євгенович _____

Рецензент:
д.т.н, професор, зав. кафедри КММПК
Лимарченко Олег Степанович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2022 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Рівень вищої освіти – другий (магістерський)
Спеціальність – 126 «Інформаційні системи та технології»
Освітньо-наукова програма «Інтегровані інформаційні системи»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«06» червня 2022 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Ткаченку Максиму Сергійовичу

1. Тема дисертації «Інтелектуальна система розпізнавання графічних образів на основі згорткових нейронних мереж», науковий керівник дисертації Сокульський Олег Євгенович, доцент, к.т.н., доц. каф. ІСТ, затверджені наказом по університету від 26.04.2022 р. № НС/88/2022

2. Термін подання студентом дисертації 08.06.2022 р.

3. Об'єкт дослідження графічний образ.

4. Предмет дослідження інтелектуальна система розпізнавання графічних образів на основі згорткової нейронної мережі.

5. Перелік завдань, які потрібно розробити 1. Аналіз методів підвищення ефективності розпізнавання та класифікації зображень. 2. Розробка системи розпізнавання графічних образів на основі згорткової нейронної мережі. 3. Навчання нейронної мережі. 4. Тестування системи на реальних зображеннях.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: структурна схема нейронної мережі, блок-схема алгоритму навчання, графіки точності розпізнавання образів системою.

7. Орієнтовний перелік публікацій 1. Ткаченко М.С., Сокульский О.Є. "Застосування R-CNN при автоматичному позиціонуванні об’єктів через нейромережевий аналіз графічних даних". 2. Ткаченко М.С., Сокульський О.Є. "Принципи організації процедури машинного аналізу на основі згорткової нейромережевої архітектури"

8. Дата видачі завдання 31.01.2022 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Пошук науковго матеріалу за обраною темою	31.01.2022 - 25.02.2022	
2.	Дослідження актуальності та новизни теми	26.02.2022 - 28.02.2022	
3.	Дослідження теми машинного навчання	01.03.2022 - 14.03.2022	
4.	Пошук матеріалу про сучасні згорткові нейронні мережі	15.03.2022 - 02.04.2022	
5.	Розробка системи розпізнавання образів	03.04.2022 - 17.04.2022	
6.	Навчання нейронної мережі	18.04.2022 - 26.04.2022	
7.	Оптимізація параметрів системи	27.04.2022 - 12.05.2022	
8.	Тестування розробленої системи	13.05.2022 - 15.05.2022	
9.	Оформлення документації	16.05.2022 - 02.06.2022	

Студент Максим ТКАЧЕНКО

Науковий керівник Олег СОКУЛЬСЬКИЙ

РЕФЕРАТ

Магістерська дисертація на здобуття ступеня «магістр» за освітньо-науковою програмою підготовки «Інтегровані інформаційні системи» на тему «Інтелектуальна система розпізнавання образів на основі згорткових нейронних мереж». Дисертація містить 102 сторінки, 54 рисунки, 3 додатки, 26 джерел.

Актуальність. Підвищення точності розпізнавання графічних образів комп'ютером є актуальною темою для побудови сучасних інформаційних систем.

Метою магістерської дисертації є підвищення ефективності систем розпізнавання графічних образів, вдосконалення технології комп'ютерного зору.

Об`єкт дослідження: графічний образ.

Предмет дослідження: інтелектуальна система розпізнавання графічних образів на основі згорткових нейронних мереж.

Наукова новизна полягає у підвищенні ефективності розпізнавання графічних образів інтелектуальними системами, а саме – у поєднанні методів попередньої обробки зображення та мінімізації помилки системи.

Публікація результатів дисертації. За результатами роботи було опубліковано наукові статті:

Ткаченко М. С., Сокульський О.Є. Застосування R-CNN при автоматичному позиціонуванні об'єктів через нейромережевий аналіз графічних даних.

Ткаченко М. С., Сокульський О.Є. Принципи організації процедури машинного аналізу на основі згорткової нейромережевої архітектури.

Ключові слова: ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, МАШИННЕ НАВЧАННЯ, КЛАСИФІКАЦІЯ, РОЗПІЗНАВАННЯ ОБРАЗІВ, ІНТЕЛЕКТУАЛЬНА СИСТЕМА.

ABSTRACT

Master's dissertation for the degree of "master" in the educational program "Integrated Information Systems" on the topic "Intelligent image recognition system based on convolutional neural networks." The dissertation contains 102 pages, 54 figures, 3 appendices, 26 sources.

Topicality. Improving the accuracy of computer image recognition is an important topic for building modern information systems.

The aim is to improve the efficiency of graphic recognition systems, and enhance computer vision technology.

The object of study — graphic image.

Purpose of the study — intelligent graphic image recognition system based on convolutional neural networks.

Scientific novelty is to increase the efficiency of graphic image recognition by intelligent systems, namely - in a combination of image pre-processing methods and minimize system error

Publication of dissertation results. Based on the results of the work, an articles were published:

Tkachenko M. Sokylskyi O. Usage of R-CNN in automatic positioning of objects through neural network analysis of graphic data.

Tkachenko M. Sokylskyi O. Principles of organization of machine analysis procedure based on convolutional neural network architecture.

Keywords: CONVOLUTIONAL NEURAL NETWORKS, MACHINE LEARNING, CLASSIFICATION, IMAGE RECOGNITION, INTELLIGENT SYSTEM.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

АП — апаратна платформа.

БД — база даних.

ПЗ — програмне забезпечення.

API — абревіатура Application Programming Interface, програмний інтерфейс додатку.

Bias — зміщення активації нейрону.

CNN — абревіатура Convolutional Neural Networks, клас згорткових нейронних мереж в машинному навчанні.

Feature map (з англ. “Карта ознак”) —матриця, що є результатом операції згортки.

IDE — абревіатура Integrated Development Environment, інтегроване середовище розробки.

ILSVRC — абревіатура ImageNet Large Scale Visual Recognition Challenge, конкурс в рамках ресурсу ImageNet, метою якого є підвищення ефективності розпізнавання графічних образів комп’ютерними системами.

Padding — метод доповнення графічного зображення пікселями по краях при обробці згортковими нейронними мережами.

ReLU — абревіатура Rectified Linear Unit, функція активації, що активно застосовується в нейронних мережах.

RGB — абревіатура Red Green Blue, адитивна кольорова модель, що використовується для кодування кольору в графічних зображеннях.

Stride — крок на згортковому шарі, з яким ядро виконує операцію згортки по зображенню.

ВСТУП

Магістерська дисертація присвячена актуальній темі розпізнавання графічних образів сучасними інформаційними системами. Застосування технології комп’ютерного зору вирішує все більше прикладних задач в житті людини. Концепція побудови таких інформаційних систем базується на машинному навчанні. Завдяки прогресу в сфері обчислювальних можливостей комп’ютерів, штучні нейронні мережі та системи на їх основі стають все більш поширеними.

Об'єкт дослідження — графічний образ. Предметом дослідження є інтелектуальна система розпізнавання образів на основі згорткових нейронних мереж.

Метою дослідження є підвищення точності розпізнавання графічних образів комп'ютером, тобто зменшення відсотку помилки при класифікації зображень.

Класичні методи розпізнавання не є ефективними для роботи із зображеннями, саме тому для вирішенням цієї задачі використано нейромережевий підход. Незважаючи на те, що вже існує велика кількість архітектур та готових рішень для створення технології комп’ютерного зору, питання підвищення точності розпізнавання образів системою все ще залишається відкритим. Саме відсоток правильно класифікованих образів є показником ефективності системи. Більшість реальних задач, де можна використати нейронні мережі для побудови систем з комп’ютерним зором є досить специфічними (розпізнавання образів автопілотом, аналіз медичних знімків тощо), через це при побудові таких систем необхідно володіти знаннями про машинне навчання і особлисвості застосування того чи іншого класу нейронних мереж.

Результатом роботи над дисертацією є комп’ютерна програма, яка здатна розпізнавати графічні образи на вході відповідно до класів

зображень, що використовувались в процесі навчання нейронної мережі в якості набору тренувальних даних. Тестування системи показало доцільність її реального використання в прикладних задачах, оскільки помилка розпізнавання є досить малою. Це означає, що описані методи в магістерській дисертації можуть бути використані для побудови аналітичної інформаційної системи, яка повинна вміти ефективно обробити та класифікувати зображення з високою точністю.

1 АНАЛІЗ ТЕХНОЛОГІЇ МАШИННОГО НАВЧАННЯ

1.1 Опис предметної області

Одним з найсучасніших напрямків в індустрії інформаційних систем є дослідження та реалізація технології комп'ютерного зору. Головною метою використання цієї технології є втілення на базі обчислювальної техніки можливостей біологічного мозку сприймати та аналізувати графічні образи. Відтворення цього процесу є надто складним процесом в рамках комп'ютерного прогрмування, оскільки ресурси машин довгий час були обмеженими, а також не інсувало ефективних концепцій щодо відповідності інформаційних систем реальним біологічним нейронним мережам. Навіть для реалізації простої на вигляд концепції знаходження та класифікації графічного образу на зображенні довелося витрати роки досліджень та розробок. Лише після багатьох десятиліть та наукового внеску багатьох людей, вдалося досягти успіху в задачі розпінавання та класифікації зображень.

В першу чергу, успішне вирішення задачі реаліації комп'ютерного зору стало можливим завдяки розвитку суміжних напрямків математики, машинного навчання, біологічних та штучних нейронних мереж. Алгоритми глибокого навчання, що лежать в основі систем комп'ютерного зору, виконують мільярди обчислень для отримання швидкого результату. Такі системи використовують також досягнення та існуючі рішення в сфері графічних процесорів, оскільки апаратна реалізація є важливим фактором для швидкодії системи.

Спроби досягти 100% точності розпізнавання продовжуються вже багато років. Вдосконалення програмної та апаратної частини комп'ютерів є рушійним фактором в сфері сучасних систем комп'ютерного зору. Стрімкий розвиток інформаційних технологій та

цифрових пристроїв став ще одним фактором масового застосування систем розпізнавання графічних образів. Приклад роботи такої системи зображено на рис. 1.1:

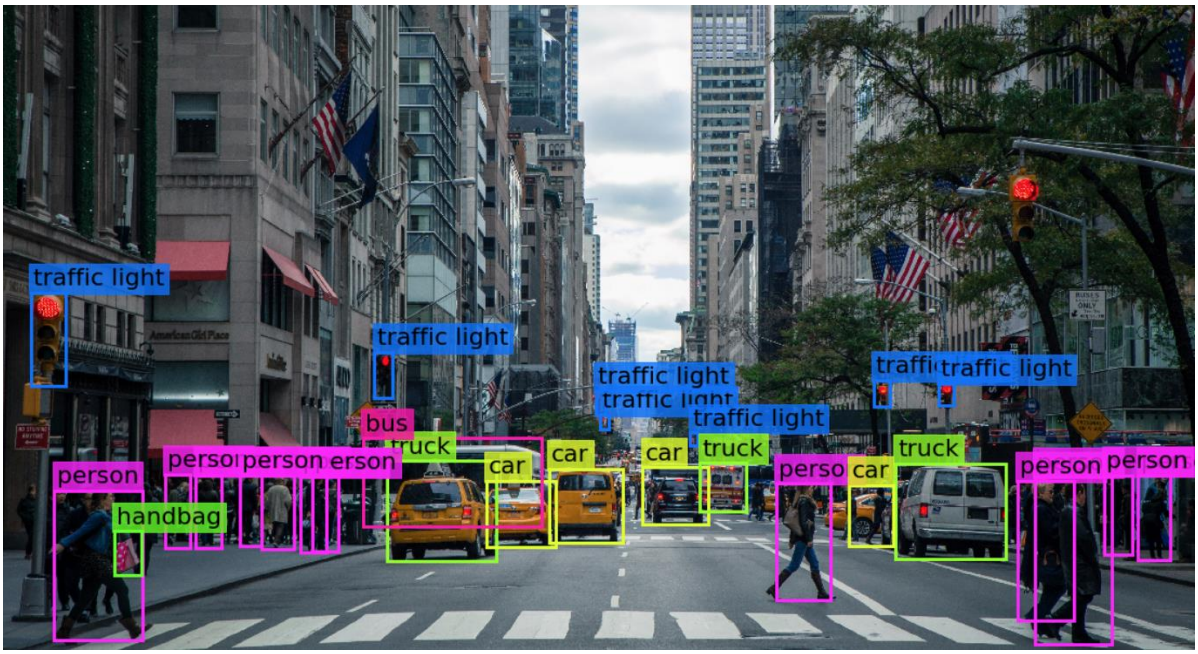


Рисунок 1.1 — Демонстація роботи комп'ютерного зору [1]

Розробка продуктів, що втілюють технологію комп'ютерного зору, пропонує великі можливості для автоматизації процесів, проектування автономних систем, що включають аналіз зображень, а також реалізацію хмарних обчислень, що використовуються системами для глибинного навчання.

В найближчі роки комп'ютерний зір буде розвиватися також за рахунок попиту на дрони, автономні або напівавтономні транспортні засоби, комунікаційні пристрої, використання цієї технології в сфері розваг, медицини тощо. Сучасні інформаційні системи дозволили розробникам використовувати високорівневі рішення для синтезу комп'ютерного зору, що робить технологію все більш поширеною і доступною в реалізації.

1.2 Об'єкт та предмет дослідження

Об'єктом дослідження магістерської дисертації є графічний образ. Завдяки розвитку обчислювальної техніки, все більше інформаційних систем використовують в тій чи іншій мірі класифікацію зображень, потокову обробку відео тощо. Саме через це питання ефективної реалізації розпізнавання графічних образів є дуже актуальним в сфері сучасних інформаційних технологій.

Предмет досідження — це інтелектуальна система розпізнавання графічних образів на основі згорткової нейронної мережі.

1.3 Постановка задачі

В ході дослідження та роботи над магістерською дисертацією необхідно вирішити наступні задачі:

1) Виконати аналіз методів підвищення ефективності розпізнавання та класифікації зображень з метою мінімізації помилки.
2) Розробити систему розпізнавання графічних образів на основі згорткової нейронної мережі з використанням мови програмування Python та бібліотек Keras, TensorFlow.
3) Виконати навчання нейронної мережі.
4) Виконати тестування системи на реальних зображеннях. Визначити точність розпізнавання та загальну помилку системи.

1.4 Концепція глибинних нейронних мереж

Ідея нейромереж була запозичена у справжнього біологічного мозку. Для розуміння структури та функціонування нейромережі в контексті вирішення задачі розпізнавання образів, варто розглянути класичну задачу комп'ютерного зору. Існує нейромережа, яка приймає на

вхід графічне зображення заданої розмірності (наприклад, 32 на 32 пікселі). Для спрощення задачі, можна розглядати нейромережу, яка працює з чорно-білими зображеннями. Таким чином, на вході нейромережі є матриця, кожен піксель якої відображає відтінок, від 0 до 1, де 0 — абсолютно чорний піксель та 1 — абсолютно білий (рис. 1.2).

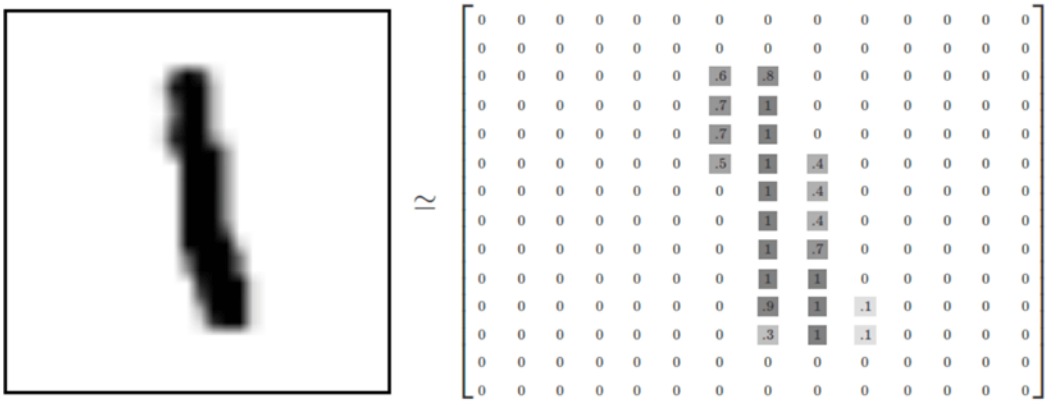


Рисунок 1.2 — Зображення у вигляді матриці пікселів

Для роботи з кольоровими зображеннями схема залишається такою ж, одна вхідна матриця може бути багатовимірною та діапазон кожного пікселі може бути різним в залежності від типу кодування зображень. Наприклад, для RGB-формату на вхід нейромережі буде подаватися 3-вимірна матриця для кодування червоного, зеленого та синього кольорів відповідно. Діапазон значень в матриці буде змінюватися від 0 до 255 (відповідно до стандартів формату RGB).

Для обраної нейромережі, що працює з чорно-білими зображеннями, можна уявити кожен нейрон як число від 0 до 1, що буде відображати відтінок. Це число можна назвати “активацією” нейрону. Вхідна матриця являє собою перший рівень (або шар) нейронної мережі. Однак для спрощення обчислень та кращого розуміння вхідна матриця подається у вигляді вектору, його розмірність буде визначатися розміром вхідного зображення. Наприклад, для обраної нейромережі, що працює із графічними зображеннями 32 на 32 пікселі, розмір вхідного вектору буде

становити $[1 \dots 32 * 32]^T = [1 \dots 1024]^T$. Як відомо, в загальному вигляді при задачі розпізнавання образів, нейронна мережа представляє собою функцію, що приймає на вхід велику кількість параметрів (в класичному випадку це кількість пікселів) і на виході, після опрацювання даних, мережа видає певні числа, за якими можна стверджувати про приналежність графічного образу до того чи іншого класу [2]. Кількість виходів нейронної мережі дорівнює кількості класів, що використовувалися для навчання мережі. Іншими словами, нейронна мережа намагається встановити відповідність між нейронами першого рівня та одним із нейронів останнього рівня. Однак, насправді структура сучасних нейронних мереж є багаторівневою, і між першим і останнім рівнем може бути велика кількість прихованих шарів (англ. "hidden layers"). В задачах класифікації та розпізнавання образів кількість рівнів нейронної мережі впливає на її точність, час опрацювання зображень та час навчання всієї мережі (відповідно, чим більше рівнів, тим більше математичних обчислень потрібно виконати для знаходження потрібних ваг кожного нейрону і, відповідно, витратити більше обчислювальних ресурсів або часу).

Суть нейронної мережі полягає у тому, що активації нейронів на будь-якому наступному рівні залежить від того, які нейрони були активовані на попередньому рівні. Таким чином, група нейронів першого рівня активує певну групу нейронів другого рівня, і цей ланцюг продовжується до останнього рівня нейронної мережі. Звісно, таке порівняння є далеким від того, як насправді діють біологічні нейрони, однак такий опис нейронної мережі дуже легко інтерпретується мовою математики і може бути запрограмований.

Задача розпізнавання будь-якого графічного образу може бути розділена на певні підзадачі. Наприклад, для розпізнавання геометричної фігури (коло, квадрат, многогранник) нейронна мережа повинна вміти

розпізнавати грані, кути, співвідношення між сторонами фігури тощо. Це означає, що геометричну фігуру (або складний графічний образ) можна представити у вигляді певних шаблонів, кожен з яких буде релевантним рівню нейронної мережі. Якщо на вході нейронної мережі буде певне зображення, то воно активує лише ті нейрони другого рівня даної мережі, які будуть співпадати з навченим шаблоном, тобто мережею будуть знайдені схожі грані та лінії в зображенні подібно тим, які використовувалися для її навчання. Нейрони другого рівня будуть активувати за аналогічною схемою нейрони третього рівня, і так до фінального рішення, за яким нейронна мережа розпізнає та класифікує оброблене зображення. Чи вдасться мережі точно класифікувати зображення — це залежить від процесу та якості її навчання. Однак, таке розуміння проміжних (схованих) шарів може бути конкретною ціллю при побудові нейронної мережі, вибору кількості рівнів та кількості нейронів на кожному з них. Окрім цього, існує велика кількість інших задач, які розбиваються на свої абстракції (наприклад, розпізнавання та класифікація звуків, розбір мовлення).

В задачі розпізнавання графічних образів основна ціль — отримати такий механізм, який зможе перетворити пікселі в грані, а потім грані в шаблони, що використовувались при навчанні, і таким чином, точно класифікувати зображення [3].

1.5 Математична модель нейронної мережі

Як було зазначено вище, активація нейронів на кожному наступному рівні залежить від того, які нейрони були активовані на попередньому рівні мережі. Можна уявити, що задача кожного нейрона на другому рівні нейронної мережі — розпізнати колір, відтінок або грань в певній частині зображення. Як математично можна описати зв’язок між

нейронами? Для кожного з’єднання між нейронами присвоюється певна вага (число). Якщо провести аналогію з біологічними нейронами, то ваги, фактично, характеризують наскільки міцно зв’язані нейрони між собою. Для обчислення активації кожного нейрону на наступному рівні, необхідно здійснити операцію сумуванню всіх з’вязків із цим нейроном, помножених на активації нейронів попереднього рівня. Таким чином, активацією нейрону зважена сума:

$$w_1 a_1 + w_2 a_2 + w_3 a_3 + w_4 a_4 + \cdots + w_n a_n \qquad (1.1)$$

де w_n — вага зв'язку з відповідним нейронном попереднього рівня;

a_n — значення активації нейрону попереднього рівня.

Однак, ця сума може бути будь-яким числом (як завгодно малим або великим). Цілком логічно буде звести отриману суму до бажаного діапазону (наприклад, від 0 до 1), тому для нормалізації значень використовуються певні математичні функції. Наприклад, функція сигмоїди, графік якої зображено нижче. Дуже негативні значення зводяться до нуля, дуже позитивні — до одиниці. Функція монотонно зростає в діапазоні, близькому до нуля (рис. 1.3).

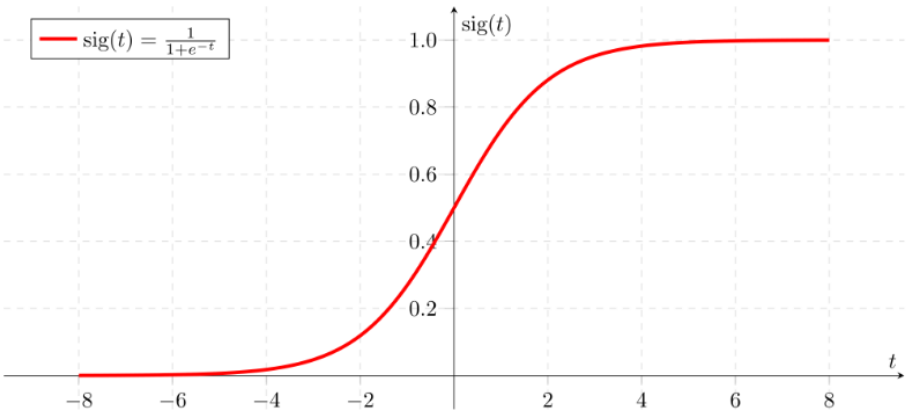


Рисунок 1.3 — Графік фукнції логістичної кривої (сигмоїда)

Це означає, що активація нейрону характеризує, наскільки позитивною є обчислена сума. Для більш гнучкого налаштування нейронної мережі, кожен нейрон, зазвичай, має значення зміщення. Зміщення дає можливість ігнорувати значення суми ваг, наприклад, коли її значення менше певного числа. Наприклад, нейрон активується лише тоді, коли сума ваг більше від 10. Зміщення додається перед тим, як сума буде опрацьована функцією сигмоїди.

Таким чином, для кожного нейрону встановлюється ваги (зв’язки з нейронами попереднього рівня мережі) та значення зміщення. Це самі ті числа, які можна змінювати та налаштовувати в процесі навчання нейронної мережі. Очевидним є те, що чим більше рівнів та нейронів на кожному з них, тим довшим буде процес навчання, однак нейронна мережа буде більш гнучкою та, ймовірно, буде точніше прогнозувати рішення. Коли йде мова про навчання нейронної мережі, то це означає, що комп’ютер повинен самостійно знайти всі ці значення ваг та здвигів так, що це вирішило поставлене завдання (рис. 1.4).

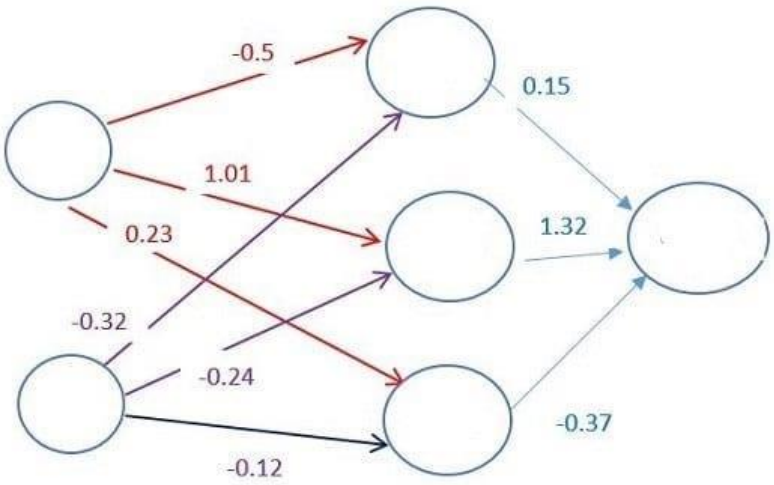


Рисунок 1.4 — Ваги в якості головного елементу зв'язку між нейронами

Для математичних обчислень, дуже зручно представляти активації кожного рівня та ваги у вигляді матриць. Це дає можливість легше запрограмувати поведінку нейронної мережі, а також використати

операції з лінійного алгебри для сумування та множення. Активація нейронів на кожному рівні представляється у вигляді вектору-стовпця, а ваги — матрицею відповідної розмірності $k\ x\ n$, де k — кількість нейронів наступного рівня, n — кількість нейронів попереднього рівня.

$$\sigma\left(\begin{bmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,n} \\ w_{1,0} & w_{1,1} & \cdots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \cdots & w_{k,n} \end{bmatrix} * \begin{bmatrix} a_0^{(0)} \\ a_1^{(0)} \\ \vdots \\ a_n^{(0)} \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix}\right), \tag{1.2}$$

де w_{ij} — вага зв'язку з нейроном попереднього рівня;

a_j — активація нейрону;

b_j — зміщення активації нейрону (bias).

У науковій літературі, більш прийнятою і поширеною є наступна форма запису:

$$\sum_{j=0}^{n} w_{ij} * a_j + b_j\ , \tag{1.3}$$

Раніше нейронні мережі використовували функцію сигмоїди для того, щоб стиснути отриману суму до певного діапазону (де верхня і нижня границі будуть аналогією активного та неактивного станів біологічного нейрону). Однак, більшість сучасних нейронних мереж не використовують функцію сигмоїди. Більш поширеною стала функція ReLU, графік якої зображено нижче. Дана функція бере максимальне значення в діапазоні від 0 до a, де a буде характеризувати ступінь активації нейрону. Іншими словами, функція буде працювати лише в тому випадку, якщо пройдено певний поріг значень (сума менше 0 не буде

активувати нейрон). Це свого роду спрощення, яке дало можливість ефективніше навчати нейронні мережі за витрат меншої кількості ресурсів. Графік функції ReLU представлений на рис. 1.5:

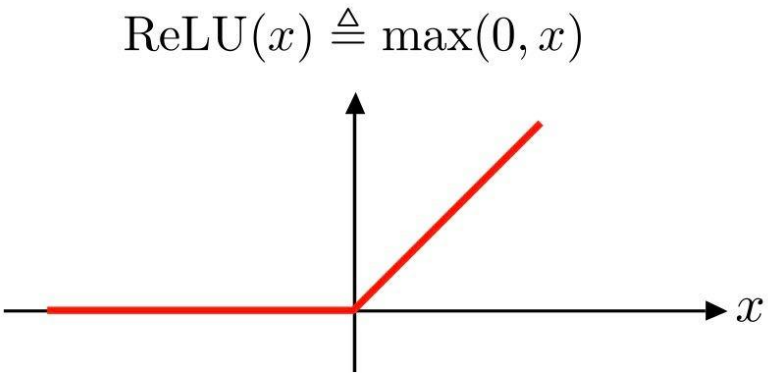


Рисунок 1.5 — Графік фукнції активації ReLU

1.6 Функція помилки

Коли нейронна мережа здатна розпізнати графічний образ, це означає що самий “яскравий” нейрон на останньому рівні буде вказувати на певний клас, до якого входять схожі графічні образи.

Як було описано в попередньому розділі, процес навчання зводиться до підбору комп’ютером всіх ваг, що з’єднують нейрони та зміщень для кожного нейрону. Для цього використовується тренувальний набір даних. Після навчання очікується, що нейронна мережа буде розпізнавати графічні образи не тільки з тренувального набору, але й довільні зображення. Щоб перевірити адекватність мережі, їй необхідно показати незнайомі дані після процесу навчання, і встановити, наскільки точно мережа розпізнає графічний образ.

На початковому етапі навчання задаються випадкові значення для всіх ваг та зміщень кожного нейрону. Очевидно, що в такому випадку нейронна мережа буде працювати з помилками, а на останньому рівні буде випадковий набір активацій. Для ефективного навчання необхідно

ввести функцію помилки, щоб на кожній ітерації мережа могла самостійно змінювати ваги та зміщення. Метою навчання є отримати нульову активацію для більшості нейронів останнього рівня, і тільки той нейрон (один або кілька) мають бути активовані, які відповідають певному класу графічних образів.

Якщо активація нейронів змінюється в заданому діапазоні (наприклад, від 0 до 1), то можна легко визначити помилку навчання через квадрат різниці між отриманим та істинним значеннями активації.

$$(a_{\text{отримане}} - a_{\text{істинне}})^2 \tag{1.4}$$

Очевидно, що значення квадрату різниці буде мінімальним лише в тому випадку, коли отримане значення активації буде максимально наближеним до істинного значення. Якщо значення функції помилки велике для всіх нейронів останнього рівня, то нейронна мережа не здатна розпізнати графічний образ і потребує подальшого навчання. Середнє значення помилки визначається в процесі обробки нейронною мережею набору тренувальних значень [4].

Обчислення значення помилки не є достатнім для навчання мережі. Потрібно математично описати, яким чином комп’ютер може покращити результат на основі тренувального набору даних. Все що комп’ютер може змінювати — це набір ваг та зміщень для кожного нейрону. Цей набір можна представити функцією, і потрібно зробити так, щоб функція помилки від параметрів нейронної мережі досягала мінімуму. Для спрощення опису математичної моделі, варто уявити функцію від одного параметру. Як знайти число на вході, щоб значення функції було мінімальним? Для простих функцій пошук мінімуму є тривіальною задачею в математиці. Однак, для складних функцій, на вході яких є безліч параметрів, задача пошуку мінімуму стає досить складною. Варто починати з існуючих параметрів, щоб зрозуміти в якому напрямку

рухатися для досягнення мінімуму. Якщо розглядати графік функції однієї змінної, то можна обчислити нахил дотичної до кривої в кожній точці на графіку. В залежності від нахилу дотичної можна сказати, зростає чи спадає функція в околі даної точки. Таким чином, можна підбирати параметри на вході функції, обчислювати динаміку зміни графіку та корегувати параметри для досягнення мінімуму.

Варто наголосити, що отриманий мінімум буде локальним. Функція може мати багато локальних мінімумів, все залежить від точки, з якої починається алгоритм пошуку мінімуму. Немає гарантії, що локальний мінімум буде найменшим значенням функції. Розмір крок ітерації варто обирати пропорціонально нахилу дотичної: чим ближче точка знаходиться до локального мінімуму, тим більше дотична буде прямувати до горизонтальної лінії (на прикладі функції $y = f(x)$ з однією змінною). Таким чином, в околі мінімуму крок стає меншим, і буде точно знайдено сам мінімум (рис. 1.6).

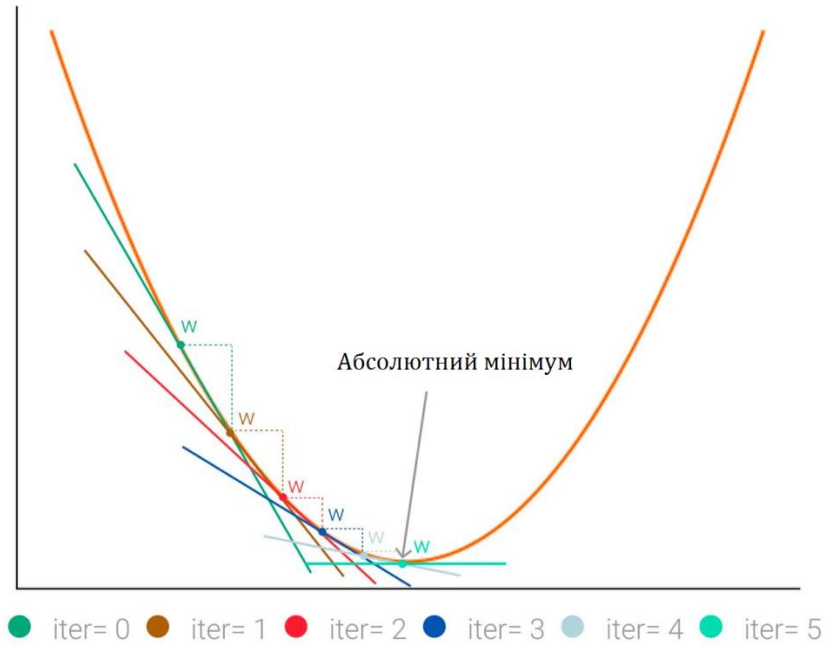


Рисунок 1.6 — Зміна кута нахилу дотичної в процесі досягнення локального або абсолютного мінімуму

1.7 Градієнтний спуск

Якщо уявити функцію від двох змінних, то область значень буде представляти собою площину. Задача залишається тією ж самою — знайти локальний мінімум функції. Замість нахилу дотичної, потрібно приймати рішення, куди саме робити крок на площині. Іншими словами, який потрібно взяти напрямок спуску, щоб зменшити функцію.

В математиці існує поняття градієнту. Це міра динаміки зміни певної фізичної величини на одиницю довжини. Градієнт — це векторна величина, яка характеризує швидкість та напрямок зміни функції, що залежить від координат. Якщо розглядати функцію двох змінних, то градієнт цієї функції буде показувати напрямок зростання, підйому. Іншими словами, в якому напрямку потрібно зробити крок, щоб функція зростала швидше.

Для скалярного поля $U(x, y, z)$ градієнт визначається формулою:

$$\nabla U = grad\ U = \frac{dU}{dx} i + \frac{dU}{dy} j + \frac{dU}{dz} k, \tag{1.5}$$

де i, j, k — орти системи відліку.

Це значення узагальнюється на простори будь-якої розмірності:

$$\nabla U = \sum_i \frac{dU}{dx} e_i \tag{1.6}$$

Таким чином, від’ємний градієнт дає напрямок, в якому функція буде зменшуватися. Більше того, довжина вектору градієнта показує, наскільки “крутим” є спуск, тобто наскільки швидше функція буде спадати у визначеному напрямку.

Градієнт можна застосовувати і до багатовимірних систем, що підходить для вирішення задачі мінімізації помилки при навчання нейронних мереж. Таким чином, обчислюючи значення градієнту на кожному кроці і роблячи відповідний крок, можна легко мінімізувати функцію шляхом зміни її параметрів. В кінці буде отримано локальний мінімум, аналогічно задачі пошуку мінімуму на кривій, де була всього лише одна змінна.

Варто представити параметри нейронної мережі (ваги та зміщення нейронів) у вигляді вектору-стовпця, а від'ємний градієнт від функції помилки — це просто вектор, певний напрямок в багатовимірному просторі, який показує зміни в параметрах нейронної мережі, що якнайшвидше призведуть до мінімізації функції помилки.

Записати процес мінімазації функції помилки можна наступним чином через векторне представлення:

$$\vec{w} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_n \end{bmatrix} \qquad -\nabla C(\vec{w}) = \begin{bmatrix} \Delta w_1 \\ \Delta w_2 \\ \Delta w_3 \\ \vdots \\ \Delta w_n \end{bmatrix}, \tag{1.7}$$

де $\vec{w}$ — вектор-стовпець параметрів функції помилки;

$\nabla C(\vec{w})$ — градієнт функції помилки;

Δw_n — необхідна зміна параметру, обчислена через градієнт.

В цьому і полягає суть навчання нейронної мережі. Зміна ваг і зміщень в процесі навчання дає можливість нейронній мережі все точніше розпізнавати шаблон (графічне зображення, звук тощо). Варто розуміти, що функція помилки включає в себе середні значення від усієї вибірки, таким чином мінімізація призведе до зменшення помилки на всіх прикладах, що використовувались при навчанні.

Алгоритм обчислення градієнту є ядром того, як навчається нейронна мережа мережа. Цей алгоритм називається методом зворотного поширення помилки. Незалежно від структури, кількості шарів та складності нейронної мережі, в усіх випадках процес її навчання — це мінімізація функції помилки. Важливий момент для мінімізації полягає у тому, що функція має бути безперервною. Саме тому штучні нейрони мають плавну функцію активації, на відміну від біологічних нейронів, які зазвичай можуть бути або активовані, або ні.

Процес зміщення значень функції на від'ємний градієнт називається "градієнтний спуск" (рис. 1.7). Це спосіб наблизитися до локального мінімуму функції помилки. Значення вектору градієнта, в залежності від знаку, показують, які саме зміни необхідно зробити з параметрами функції (додати або відняти значення). Налаштування певної ваги для нейрону може бути більш впливовим на функцію, ніж налаштування ваги для іншого нейрону. Деякі зв'язки просто є більш важливими для функціонування та навчання нейронної мережі. Геометрично це пояснюється тим, що зміна певних параметрів функції в околі точки є більш впливовим на значення функції, ніж зміна деяких інших параметрів.

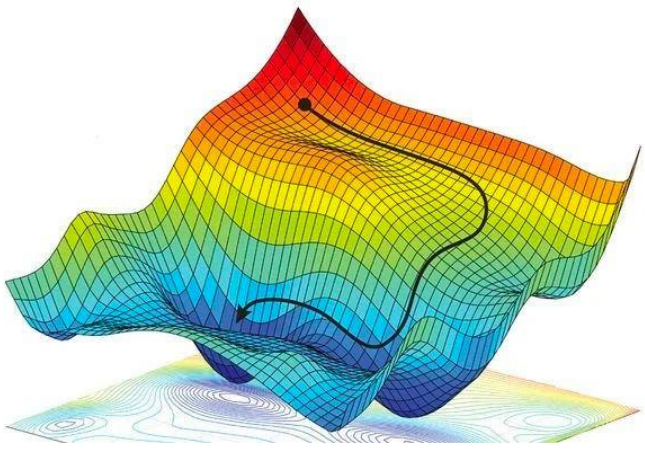


Рисунок 1.7 — Градієнтний спуск на прикладі тривимірного простору змінних [5]

Градієнт функції помилки являє собою ще один додатковий рівень складності в структурі нейронної мережі. Однак без нього навчання мережі неможливе, саме градієнт показує, зміна яких параметрів є найбільш суттєвою для мінімізації функції помилки. Після налаштування всіх ваг та зміщень нейронна мережа буде мати незначну помилку, що свідчить про завершення процесу навчання. З підібраними значенням ваг та зміщень нейронна мережа здатна працювати з новими даними, наприклад, графічними зображеннями, що не були використані в процесі її навчання. Відношення кількість правильних розпізнавань графічних образів до всієї кількості опрацьованих зображень буде показувати точність нейронної мережі.

1.8 Метод зворотного поширення помилки

Алгоритм зворотного поширення помилки є головним для навчання нейронних мереж. Для оцінки помилки використовується результат після опрацювання вхідних даних та бажаний результат на останньому рівні нейронної мережі. Помилка визначається квадратом різниці між компонентами. Повторюючи цю операцію для всього набору тренувальних даних в процесі навчання, можна отримати середнє значення помилки для всієї нейронної мережі.

Як було описано вище, мінімізація помилки зводиться до пошуку від’ємного градієнту від функції помилки. Якщо нейронна мережа містить багато шарів та велику кількість нейронів на кожному з них, то кількість параметрів для налаштувань такої мережі може становити кілька десятків тисяч. Важко уявити функцію помилки, для якої параметрами виступають ваги і зміщення, коли їх така велика кількість. При обчисленні градієнту, буде отримано вектор, розмір якого дорівнює кількості параметрів функції помилки. Іншими словами, розмір вектору градієнта дорівнює

кількості параметрів нейронної мережі. Значення компонентів цього вектору характеризують динаміку зміни функції помилки в залежності від налаштувань тих чи інших вхідних параметрів (ваг та зміщень нейронної мережі). Наприклад, значення деякого компонента вектору градієнта дорівню 2.5, а значення іншого — 0.1. Це означає, що зміна першого компоненту в 25 разів сильніше буде впливати на функцію помилки, ніж зміна другого компоненту. Якщо пояснювати це з точки зору структури нейронних мереж, то певні нейрони, ваги, зміщення є більш значущими в поведінці мережі, ніж інші.

Налаштування ваг та зміщень на кожному кроці навчання залежить від вхідних значень, що опрацьовує нейронна мереж. Для мережі, що навчається розпізнавати графічні образи, вхідними параметрами будуть пікселі зображення. Якщо проаналізувати початковий етап навчання, то активації на останньому рівні нейронної мережі будуть все ще досить випадковими, і не відповідати істинному значенню. Наприклад, якщо активація нейронів змінюється в діапазоні від 0 до 1, то значення активацій може бути 0.2, 0.5 тощо. Однак, ціль навчання полягає у тому, щоб більшість нейронів після опрацювання вхідних даних залишались не активованими, і лише один нейрон мав значення близьке до 1, яке є еталонним.

Неможливо змінювати активації в процесі навчання та роботи нейронної мережі, однак можна змінювати ваги і зміщення, що впливають на кінцевий результат. Комп’ютер повинен здійснювати аналіз того, що видає нейронна мережа на останньому рівні. Якщо задача нейронної мережі — розпізнавати графічні образи, то на кожній ітерації в процесі навчання буде зображення та зарані відомий клас, до якого відноситься графічний образ на цьому зображенні. Відповідно, активації нейронів останнього рівня будуть свідчити про приналежність зображення до того чи іншого класу. Отриманий результат, в порівнянні зі зразком на

кожному кроці, буде свідчити про те, які активації потрібно змінити для досягнення коректної роботи нейронної мережі. Значення активації для певних нейронів останнього рівня потрібно підвищити, а для деяких інших — знизити (рис. 1.8).

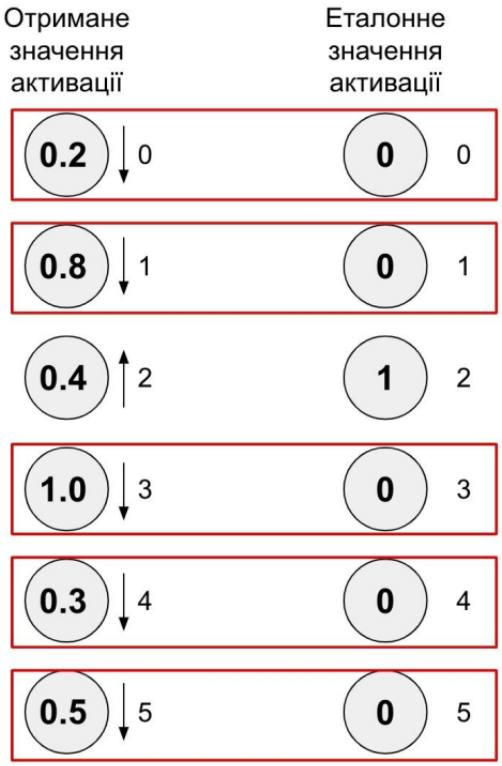


Рисунок 1.8 — Активація нейронів останнього рівня в порівнянні з еталонними значенням (стрілка біля кожного нейрону показує необхідність збільшити або знизити відповідну активацію)

Також варто зауважити, що величина зміни є прямо пропорціональною до величини відхилення від еталону. Активація потрібного нейрону може бути більш пріоритетною задачею, ніж пониження активацій інших нейронів, оскільки нейронна мережа приймає рішення, наприклад, про відношення графічного образу до певного класу саме за найбільш активованим нейроном останнього рівня.

Для кращого розуміння навчання нейронних мереж, варто зосередитися на активації останнього нейрону. Як було описано вище,

активацією є зважена сума всіх ваг на попередньому рівні, помножених на відповідні активації нейронів, плюс зміщення.

$$a^{(n)} = w_0 a_0 + w_1 a_1 + w_2 a_2 + \cdots + w_{n-1} a_{n-1} + b\,, \qquad (1.8)$$

де w — вага зв'язку між нейронами;

a — активація нейрону;

b — зміщення активації нейрону (bias).

Зважена сума підключена до певної функції згладжування (сигмоїда або ReLU). Існує три різні методи для підвищення активації потрібного нейрону на останньому рівні [6]:

1. Можна збільшити зміщення для потрібного нейрону останнього рівня.
2. Можна збільшити ваги в мережі, які, відповідно, збільшать значення зваженої суми.
3. Можна змінити активації нейронів на попередньому рівні, оскільки їх значення так само впливають на зважену суму, як і ваги.

При зміні ваг варто пам’ятати, що деякі з них більш впливають на активації нейронів наступного рівня, ніж інші. Зв’язки з найбільш “яскравими” нейронами попереднього рівня мають більший ефект, оскільки ваги будуть помножені на більші значення активації.

Це означає, що зміна ваг, що зв’язують нейрон з найбільш активованими нейронами останнього рівня буде мати більший вплив на значення функції помилки. Такий ефект можна пояснити тим, що на кожному рівні нейронної мережі нейрони намагаються знайти певний відомий шаблон (при розпізнаванні графічних образів це може бути контур, колір), і значення їх активацій буде свідчити про відповідність

вхідних даних цьому шаблону. Таким чином, кінцева активації, яка, наприклад, буде свідчити про приналежність образу до певного класу буде більше залежати від тих нейронів, які на попередніх рівнях теж активуються відповідно до навченого шаблону з цього класу зображень.

Водночас з підвищення активації потрібного нейрону, всі інші активації на останньому рівні потрібно зменшити. Як висновок, можна сказати що активації останнього рівня повністю залежать від ваг та активації на попередньому рівні. І цей алгоритм повинен працювати в напрямку від останнього до першого рівнів відповідно. Саме так виник метод зворотного поширення помилки [7].

Розглянути приклад зможе навчити нейронну мережу розпізнавати лише один графічний образ, оскільки лише для правильної активації потрібного нейрону останнього рівня будуть налаштовані ваги та зміщення. Але як навчити нейронну мережу працювати на різних даних, наприклад, вирішувати задачу класифікації графічних образів? Вищеописаний алгоритм потрібно повторити для всього набору тренувальних даних, і на кожній ітерації записувати (зберігати в базі даних або пам’яті комп’ютера) значення корегування ваг та зміщень. Після всіх ітерацій ці значення треба усереднити, і таким чином отримані ваги і зміщення будуть мінімізувати функції помилки для всіх розглянутих нейронною мережею прикладів. Усереднені значення, грубо кажучи, і будуть собою являти від’ємний градієнт функції помилки, або щось максимально наближене до нього.

На практиці комп’ютеру потрібно досить багато часу, щоб врахувати вплив кожного прикладу з набору даних на роботу мережі і значення помилки. Якщо тренувальний набір дуже великий, то його розділяють на певні підгрупи, до яких застосовуються метод зворотного поширення помилки. Можливо, цей підхід не є самим точним, однак він дає досить хорошу апроксимацію для градієнту функції помилки. При

використанні цього підходу в мережах, що навчаються розпізнавати графічні образи, весь набір тренувальних зображень перемішується і розбивається, наприклад, на групи по n зображень в кожній. Потім відбувається процес мінімізації помилки для кожної підгрупи, і отримані значення ваг та зміщень усереднюються. Це досить сильно пришвидшує процес навчання нейронних мереж на великих наборах тренувальних даних. Ця техніка називається стохастичним градієнтним спуском [8]. Приклад усерднення значень ваг за різними групами тренувальних даних зображено на рис. 1.9:

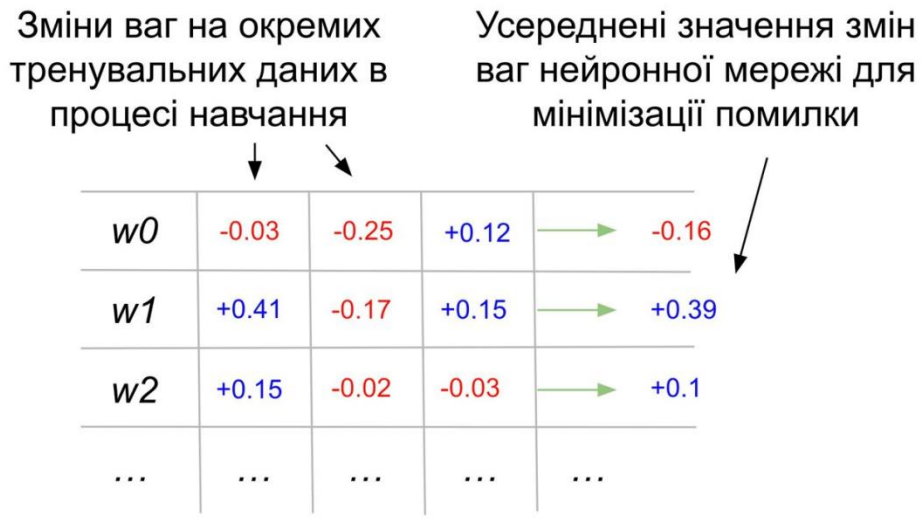


Рисунок 1.9 — Усереднення змін параметрів для мінімізації функції помилки

Варто зауважити, що важливим фактором для успішного навчання нейронної мережі завжди є кількість прикладів з набору тренувальних даних. Чим більше мережа опрацює звуків, чи графічних зображень в процесі навчання, тим краще вона зможе розпізнавати незнайомі їй вхідні дані. Для специфічних задач розпізнавання пошук і структуризація навчальних прикладів завжди є проблемою, яка має вирішуватися ще до процесу створення і навчання мережі.

1.9 Формули зворотного поширення

Тепер варто представити описаний вище метод навчання мовою математичного аналізу. Можна почати з простої нейронної мережі, кожен рівень якої має один нейрон (рис. 1.10).

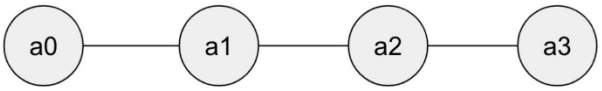


Рисунок 1.10 — Приклад простої лінійної нейронної мережі

Дана мережа містить лише 3 значення ваг та 3 зміщення, для кожного нейрону відповідно. Цей приклад дасть змогу зрозуміти, наскільки функції помилки є чуттєвої до значень цих параметрів, і відповідно, які зміни параметрів призведуть до максимального зниження функції помилки:

$$C(w_1, b_1, w_2, b_2, w_3, b_3) \tag{1.9}$$

Нехай останній нейрон мережі буде позначений як $a^{(L)}$, тоді передостанній нейрон буде з індексом $a^{(L-1)}$.

Ціль нейронної мережі буде полягати в тому, щоб дати останньому нейрону активацію зі значенням y. Для прикладу, значення y може бути 0 або 1. Тоді помилка для останнього нейрону буде виражатися формулою квадрату різниці, як це було описано вище:

$$C_0(\dots) = (a^{(L)} - y)^2 \tag{1.10}$$

Активація останнього нейрону мережі буде значення активації нейрону попереднього рівня, помножене на вагу між цими нейронами,

плюс величина зміщення. Все це потім опрацьовується функцією сигмоїди або ReLU.

$$a^{(L)} = \sigma\left(w^{(L)}a^{(L-1)} + b^{(L)}\right), \tag{1.11}$$

де $a^{(L)}$ — активація нейрону;

σ — функція активації;

$w^{(L)}$ — вага зв'язку з нейроном попереднього рівня;

$b^{(L)}$ — зміщення нейрону (bias).

Для спрощення, значення суми активації можна позначити як $z^{(L)}$:

$$z^{(L)} = w^{(L)}a^{(L-1)} + b^{(L)} \tag{1.12}$$

Тоді значення активації можна записати наступним чином:

$$a^{(L)} = \sigma\left(z^{(L)}\right) \tag{1.13}$$

Таким чином, значення ваги, активація нейрону попереднього рівня та зміщення дають змогу обчислити зважену суму, що в свою чергу дає змогу обчислити активацію та, відповідно, значення функції помилки.

Першочерговою є задача визначення впливу кожного параметру на кінцеве значення функції помилки. Наприклад, як зміна ваги між нейронами вплине на кінцеву помилку. Це можна визначити через звичайну похідну:

$$\frac{\partial C_0}{\partial w^{(L)}} \tag{1.14}$$

В цьому записі, $\partial w^{(L)}$ варто розуміти як крок на числовій прямій, що змінює значення ваги (наприклад, зміна ваги на 0.1). Відповідно ∂C_0 — це результуюча зміна помилки від зміни значення ваги $\partial w^{(L)}$.

Таким чином, відповідно до алгоритму обчислення функції помилки, спочатку треба розглядати зміну $z^{(L)}$ від $w^{(L)}$. Потім зміну $a^{(L)}$ від зміни $z^{(L)}$, і в кінці, для обчислення власне помилки, зміну $C^{(L)}$ від $a^{(L)}$ (рис. 1.11).

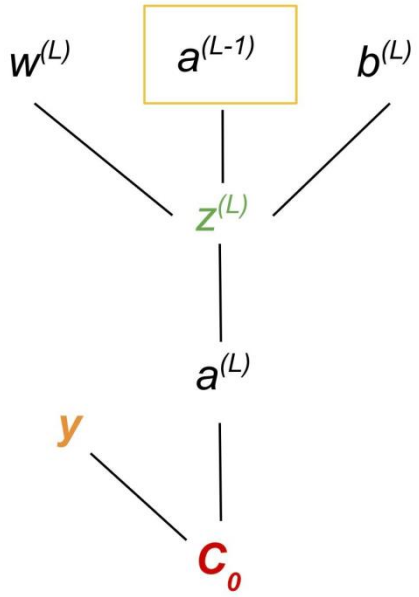


Рисунок 1.11 — Зв'язок між параметрами, що впливають на значення функції помилки

Це є ланцюжковим правилом, коли перемноження всіх цих співвідношень дасть відношення зміни одного з параметрів мережі на функцію помилки [9]. В даному прикладі, цим параметром є значення ваги між двома останніми нейронами $w^{(L)}$:

$$\frac{\partial C_0}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C_0}{\partial a^{(L)}}, \tag{1.15}$$

де C_0 — значення функції помилки;

$z^{(L)}$ — зважена сума;

$a^{(L)}$ — активація нейрону.

Далі необхідно обчислити всі похідні у виразі. Похідна від функції помилки:

$$\frac{\partial C_0}{\partial a^{(L)}} = 2(a^{(L)} - y) \tag{1.16}$$

Оскільки

$$C_0 = (a^{(L)} - y)^2 \tag{1.17}$$

Помітно, що розмір похідної є пропорціональним різниці між виходом мережі та бажаним значення Y. Тому, якщо вихід не відповідає бажаному значенню, то навіть його невелика зміна буде досить сильно впливати на функцію помилки.

Похідна $a^{(L)}$ від $z^{(L)}$ — це просто похідна від функції сигмоїди, або, наприклад, іншої не лінійної функції, що використана для нормалізації зваженої суми.

$$\frac{\partial a^{(L)}}{\partial z^{(L)}} = \sigma'(z^{(L)}) \tag{1.18}$$

Похідна $z^{(L)}$від $w^{(L)}$:

$$\frac{\partial z^{(L)}}{\partial w^{(L)}} = a^{(L-1)} \tag{1.19}$$

Оскільки:

$$z^{(L)} = w^{(L)} a^{(L-1)} + b^{(L)} \tag{1.20}$$

У випадку з останньою похідною варто пояснити, що зміна значення ваги між двома останніми нейронами показує, наскільки сильним є передостанній рівень мережі. Саме таким чином в нейронних

мережах пояснюється біологічний принцип “нейрони, що працюють разом над однією задачею — пов’язані між собою”.

Все це разом дасть похідну від функції помилки для кожного прикладу з набору тренувальних даних, які потім сумуються і усереднюються:

$$\frac{\partial C}{\partial w^{(L)}} = \frac{1}{n}\sum_{k=0}^{n-1}\frac{\partial C_k}{\partial w^{(L)}}, \tag{1.21}$$

де ∂C — середнє значення похідної для функції помилки;

∂C_k — похідна функції помилки для кожного окремого прикладу.

Звісно ж, це лише один компонент вектору градієнта, який собою являє часткові похідні від функції помилки по всім значенням ваг та зміщень, що наявні в нейронній мережі:

$$\nabla C = \begin{bmatrix} \frac{\partial C}{\partial w^{(1)}} \\ \frac{\partial C}{\partial b^{(1)}} \\ \vdots \\ \frac{\partial C}{\partial w^{(L)}} \\ \frac{\partial C}{\partial b^{(L)}} \end{bmatrix} \tag{1.22}$$

Навіть якщо це лише одна з усіх необхідних часткових похідних, обчислення впливу інших параметрів на функцію помилки вже буде простішим.

Окрім ваг, в нейронній мережі можна ще змінювати значення зміщень для кожного нейрону. Тому, в формулі можна замінити лише

похідну від зваженої суми по $w^{(L)}$ на похідну від зваженої суми по b (значення зміщення для нейрону).

$$\frac{\partial C_0}{\partial b^{(L)}} = \frac{\partial z^{(L)}}{\partial b^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C_0}{\partial a^{(L)}} \tag{1.23}$$

Якщо подивитися на формулу зваженої суми (2.6), то ця похідна буде дорівнювати 1, оскільки зміщення — це константа:

$$\frac{\partial z^{(L)}}{\partial b^{(L)}} = 1\sigma'\left(z^{(L)}\right)2(a^{(L)} - y) \tag{1.24}$$

Саме звідси пішла ідея зворотного поширення помилки, адже можна побачити, наскільки чутливою є помилка з активації попереднього нейрону, або попереднього рівня нейронів. А саме, ось ця частинна похідна буде характеризувати чутливість зваженої суми до попередньої активації:

$$\frac{\partial C_0}{\partial a^{(L-1)}} = \frac{\partial z^{(L)}}{\partial a^{(L-1)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C_0}{\partial a^{(L)}} \tag{1.25}$$

Значення даної частинної похідної буде дорівнювати $w^{(L)}$, бо

$$z^{(L)} = w^{(L)} a^{(L-1)} + b^{(L)} \tag{1.26}$$

Хоча при навчанні мережі і немає можливості вплинути безпосереднього на активацію, одна все-рівно це треба відслідковувати, оскільки тепер можна повторювати правило ланцюга в зворотному напрямку, для попередніх шарів нейронної мережі. Це робиться для того, щоб проаналізувати чутливість помилки до попередніх ваг та зміщень.

Даний приклад може здаватися дуже простим, оскільки кожен рівень нейронної мережі містить лише один нейрон, а в реальних мережах складність обчислень буде зростати за експонентою. Але насправді, при збільшенні кількості нейронів все не дуже змінюється. Будуть додаватися

лише індекси для відслідковування нейронів та впливу їх активацій, зміщень на функцію помилки.

Окрім того, що кожна активація буде позначена рівнем ($a^{(L)}$, $a^{(L-1)}$), додатково вводяться змінні для нумерації активацій в межах кожного рівня, наприклад $a_0^{(L)}$, $a_1^{(L)}$ тощо (рис. 1.12).

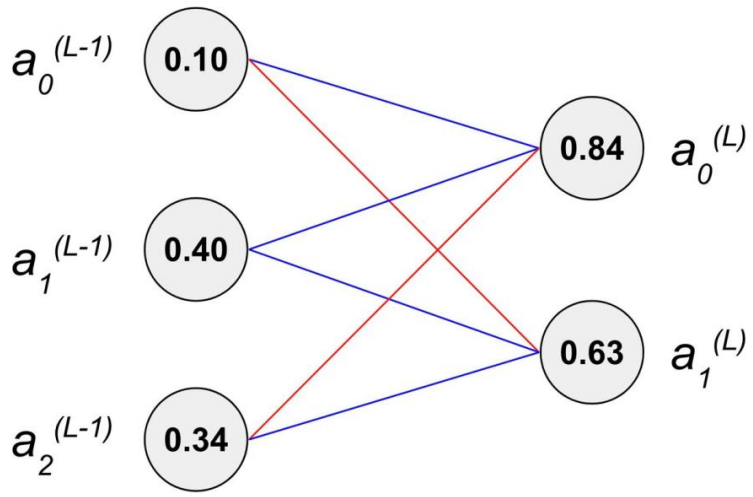


Рисунок 1.12 — Нумерація нейронів в багаторівневій нейронній мережі

Для подальшого пояснення варто ввести індекси останнього рівня, позначивши їх літерою *j* та індекси передостаннього рівня, позначивши їх буквою *k* (рис. 1.13).

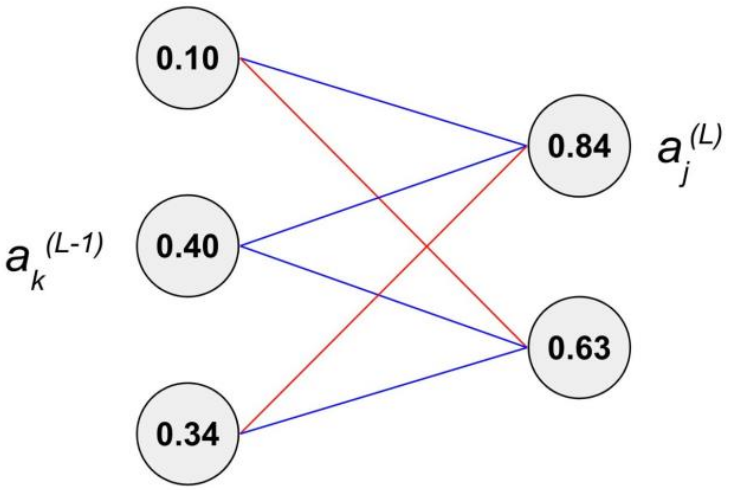


Рисунок 1.13 — Позначення нейронів через індекси на різних рівнях

В такому випадку помилка буде визначатися як сума всіх помилок нейронів останнього рівня, в порівнянні з еталонними значеннями активації y_i:

$$C_0 = \sum_{j=0}^{n_L-1} (a_j^{(L)} - y_j)^2 \tag{1.27}$$

Оскільки така нейронна мережа може мати багато зв'язків між нейронами сусідніх рівнів, то варто позначати ваги між нейронами відповідно до індексів цих самих нейронів. Наприклад, вага між нейронами двох суміжних рівнів з відповідними індексами j та k буде позначена як w_{jk} (рис. 1.14):

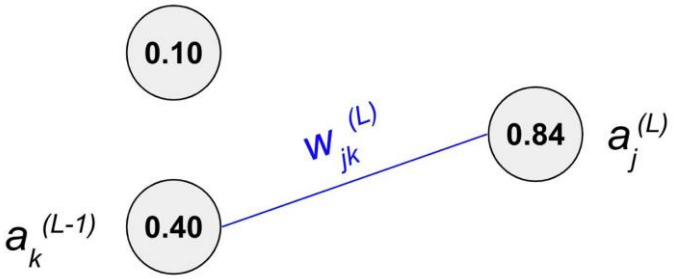


Рисунок 1.14 — Позначення ваг між нейронами через відповідні індекси

В такому випадку ваги між нейронами будуть становити собою матрицю відповідного розміру j x k, з якою комп'ютер буде простіше виконувати обчислення.

Вираз похідної за ланцюговим правилом, що описує чутливість помилки до певної ваги по суті виглядає так само. Те що змінюється, так це похідна від помилки по кожній активації на рівні $(L-1)$:

$$\frac{\partial C_0}{\partial a_k^{(L-1)}} = \sum_{j=0}^{n_L-1} \frac{\partial z_j^{(L)}}{\partial a_k^{(L-1)}} \frac{\partial a_j^{(L)}}{\partial z_j^{(L)}} \frac{\partial C_0}{\partial a_j^{(L)}} \tag{1.28}$$

Суть полягає у тому, що у випадку з великою кількістю нейронів на кожному рівні, один і той самий нейрон може по-різному впливати на функцію помилки (рис. 1.15):

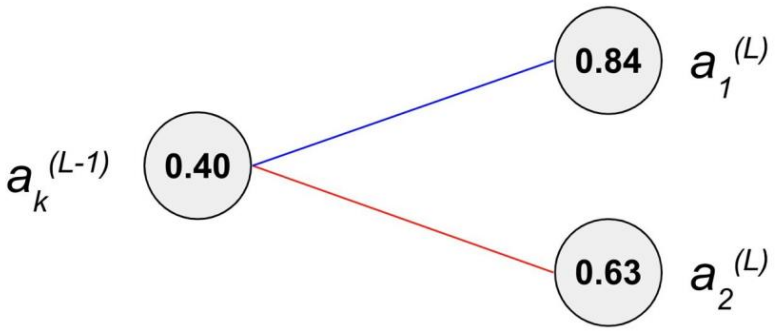


Рисунок 1.15 — Вплив одного нейрону на різні нейрони наступного рівня в залежності від міцності зв'язків між ними

Як видно із попереднього рисунку, нейрон $a_k^{(L-1)}$ впливає відразу на 2 нейрони із наступного рівня, при чому ваги можуть мати різне значення, відповідно кожен з нейронів останнього рівня буде мати свій вплив на функцію помилки. Тому їх вплив на функцію помилки потрібно підсумувати, що і було описано в формулі (2.20).

Дізнавшись, наскільки функція помилки є чутливою до активації на передостанньому рівні, можна повторити процес для всіх ваг та зміщень на цьому рівні, і так далі від останнього до початкових рівнів нейронної мережі. Похідні будуть визначати компоненти вектору градієнта, що допомогає мінімізувати помилку мережі при кожній наступній ітерації. Це і є реалізація алгоритму зворотного поширення помилки, що лежить в основі навчання нейронних мереж.

1.9 Висновки до розділу

Технологія глибинного навчання лежить в основі ефективного вирішення задачі класифікації зображень. Концепція сучасних штучних нейронних мереж є дуже схожою до біологічних нейронних мереж, через це після навчання з учителем штучні нейромережі можуть показувати досить хороший результат. Суть навчання полягає у мінімізації помилки системи, на кожній ітерації отриманий результат порівнюється з істинним значенням активації нейронів. Мінімізація здійснюється шляхом застосування математичних методів, а саме методу зворотного поширення помилки та градієнтного спуску. Нейронна мережа навчається на тренувальних даних до тих пір, поки помилка системи не стане мінімальною, після цього отримані значення ваг і зміщень здатні правильно передбачити результат на даних, що не використовувалися в процесі навчання.

2 ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ

2.1 Застосування згорткових нейронних мереж в ImageNet

Клас згорткових нейронних мереж — це саме ті нейронні мережі, з яких почався масовий інтерес до машинного навчання та прикладного застосування даної технології для вирішення справжніх задач. Інші класи нейронних мережі існували вже досить давно, вони використовувалися теж певною мірою, однак деякі з них не надто краще вирішують прикладні задачі, ніж звичайні класичні методи (наприклад, алгоритмічний підхід), інколи застосування таких нейронних мереж може давати навіть гірший результат, особливо на табличних даних. Однак все змінилося на конкурсі ImageNet. Суть конкурсу полягає в тому, що існує велика кількість графічних зображень з досить хорошою якістю, здебільшого всі зображення мають високу роздільну здатність, а також для кожного рисунку відомо, що саме зображено на ньому. Іншими словами, кожний файл графічного зображення має мітку-класифікатор, за яким всі зображення можна розділити на певні класи. Теоретично, на зображенні можуть бути об’єкти з різних класів, оскільки це досить поширений випадок при фотографуванні, редагуванні зображень тощо. Задача, яка ставиться перед учасниками конкурсу — сказати, що саме представлено на кожному зображенні. Звісно ж, з використанням комп’ютерного програмування. Така задача була досить складною 10 років тому назад, і якщо створена програма могла правильно класифікувати хоча б деякі об’єкти, що представлені на зображенні, при чому вірогідність приналежності до певного класу могла бути не вище 60-70 %, то вже таке вирішення задачі було досить успішним.

Однак, все змінилося в 2012 році, коли згорткова нейронна мережа для розпізнавання і класифікації графічних образів досягла помилки в

процентному відношенні лише 16 %. Варто зауважити, що ще в 2011 році дуже хорошим був результат помилки в 25 %, і лише через кілька років після застосування згорткових нейронних мереж помилка впала до раніше неможливих кількох відсотків. Більше того, дослідження 2015 року показали, що в деяких задачах проекту ImageNet, який оцінює ефективність роботи тих чи інших комп'ютерних програм при обробці графічних зображень, методи застосування згорткових нейронних мереж є кращими за людські здібності розпізнавання образів. Звісно ж, комп'ютерна програма може лише вказати приналежність образу до того чи іншого класу (категорії), в той час як людина може розпізнати тисячі різних класів і додатково вказати контекст приналежності образу, однак, незважаючи на це, успіх згорткових нейронних мереж став переломним моментом в сфері машинного навчання та штучного інтелекту.

До появи згорткових нейронних мереж, зазвичай, застосовували алгоритмічний підхід для виявлення тих чи інших ознак на графічному зображенні. Наприклад, контури тих чи інших шаблонів, зміни кольорів на різних ділянках зображення тощо. Це було далеко не точним вирішенням задачі класифікації графічних зображень, більше того, використання таких класичних методів вимагало дуже багато часу для опрацювання кожного зображення. Відповідно, було досить складно опрацювати відеопотік, який собою представляє набір окремих зображень в різні моменти часу.

Варто зазначити, що згорткові нейронні мережі існували і раніше, однак кілька факторів не давали можливості для їх масового застосування та розвитку. По-перше, це обмежені обчислювальні ресурси комп'ютерів. Відеокарти були не надто потужні до початку 2010-х років для обробки великої кількості зображень з досить високою роздільною здатністю. По-друге, важливим фактором для успішного навчання і розвитку будь-якої нейронної мережі, що використовує алгоритм навчання з учителем, є

наявність великого набору тренувальних даних. Раніше не було можливості створити досить потужні хмарні сховища, що будуть тримати в собі велику кількість графічних зображень. З розвитком індустрії мобільних телефонів та цифрових зображень в цілому, почали з'являтися різні веб-сайти, ресурси, або навіть сховища з цифровими графічними зображення. ImageNet, власне кажучи, і є таким ресурсом. Це найбільша база анотованих зображень, яка призначена для розробки і тестування методів машинного зору та розпізнавання образів. ImageNet містить понад 10 мільйонів зображень, що розподілені на класи, в залежності від змісту зображення, а також кожен графічний образ має координати на зображенні у вигляді прямокутника (рис. 2.1). З початку 2010 року ведеться проект ILSVRC (англ. ImageNet Large Scale Visual Recognition Challenge — Компанія по широкомасштабному розпізнаванню графічних образів в ImageNet [10]), в рамках якого різні програмні продукти щороку змагаються в класифікації і розпізнавання зображень та сцен з використанням бази зображень ImageNet.

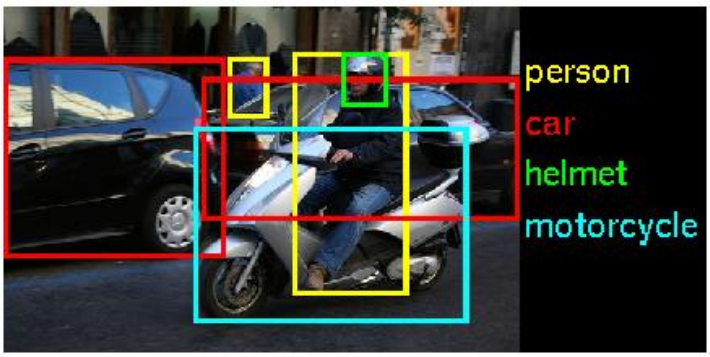


Рисунок 2.1 — Приклад класифікованого зображення в системі ImageNet

З появою більш потужних відеокарт в 2012 вдалося створити відому згорткову нейронну мережу AlexNet [11], яка видавала помилку всього лише 15 %, що на 10 % краще ніж результати інших програмних реалізацій розпізнавання образів, які приймали участь у змаганні на базі ImageNet. За своєю структурою нейронна мережа містить 8 рівнів.

AlexNet вважається однією з найвпливовіших наукових публікацій в сфері розпізнавання образів та комп’ютерного зору. Дана реалізація стала початком розвитку інших нейронних мереж та досліджень, що прискорювали досягнення у сфері машинного навчання. За даними Google Scholar, публікація наукової статті з нейронною мережею AlexNet була використана в якості цитати або посилання більше ніж 80 000 разів станом на 2021 рік.

Після появи AlexNet, якщо говорити глобально, ніяких інших технологій або алгоритмічних підходів не використовується в конкурсі ImageNet. Станом на зараз, учасниками змагань та дослідниками використовуються згорткові нейронні мережі, їх нові версії та аналоги AlexNet, або, наприклад, це можуть комбінації уже існуючих згорткових нейронних мереж, для яких додатково створюються ще певні класифікатори на вищому рівні. Це дає змогу покращити результат, знизивши відсоток помилки при розпізнаванні зображень, однак варто розуміти, що загальний підхід до машинного навчання і реалізації комп’ютерного зору залишається незмінним впродовж останніх десяти років. Такими темпами, починаючи в період з 2012 до 2017 року, якість розпізнавання та правильного передбачення класу зображень в рамках проекту ILSVRC виросла в 10 разів, про що свідчить значення помилки навчання тих чи інших сучасних нейронних мереж. Більше того, використання згорткових нейронних мереж вирішує прикладну задачу класифікації зображень, суть якої полягає у виявленні схожих образів і присвоєння відповідних міток кожному зображенню [12]. В цій задачі нейронна мережа не здатна самостійно назвати клас, якщо в процесі її навчання не використовувалися схожі приклади, однак такий підхід дає змогу швидко опрацювати велику кількість даних, розділивши їх на класи за певними ознаками. Дуже часто навіть самі ознаки нейронна мережа обирає самостійно (це можуть бути контури, поєднання або ж переходи

кольорів тощо). І найцікавіший факт полягає в тому, що навіть при вирішенні цієї задачі групою людей, яким надається певний обмежений проміжок часу на класифікацію зображень, при наявності складних графічних образів значення помилки буде набагато вищим, ніж отримане в результаті опрацювання зображень нейронною мережею. Саме через це нейронні мережі дуже часто асоціюються з комп’ютерним зором, оскільки це був перший тріумф в їх застосуванні.

2.2 Проблеми використання класичних методів розпізнавання

В чому все-таки полягає ефективність застосування згорткових нейронних мереж, на відміну від використання розглянутого в першому розділі підходу з перетворенням зображення в одномірний вектор і його подальше опрацювання нейронною мережею? Насправді, цей метод також працює досить гарно, однак лише для простих задач розпізнавання (невеликий розмір зображення, хороша якість, чітке розміщення графічного образу на зображенні тощо). У випадку чорно-білих зображень, задача взагалі дуже спрощується, оскільки на вході буде лише одновимірна матриця (при кольоровому зображенні додаткові рівні створюються за рахунок кодування, наприклад, RGB, при якому вхідна матриця буде мати три рівні для червоного, зеленого та синього кольорів відповідно).

Першим суттєвим недоліком перетворення зображення в одновимірний вектор є те, що повністю втрачається інформація про оточення кожного пікселя [13]. Іншими словами, після перетворення зображення у вектор нейронна мережа не буде мати можливості використати інформацію про розташування деяких пікселів поруч на зображенні, що є досить суттєвою втратою. Пікселі стать не пов'язаними між собою.

Це не означає що пікселі стають абсолютно окремими у векторі, все-таки зберігається їх порядок і якщо, наприклад, пікселі знаходяться поруч або у певній ділянці зображення, то можна ними оперувати за допомогою індексів, але це потребує додаткових обчислень та відповідного програмного коду (рис. 2.2). В такому випадку нейронна мережа буде заново вчитись відновлювати зв’язки між пікселями в процесі навчання. Інакше кажучи, мережа буде розуміти що пікселі з індексами n та m є сусідніми і їх потрібно аналізувати разом.

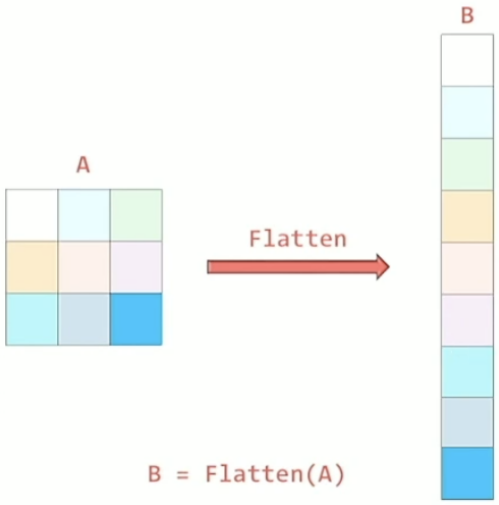


Рисунок 2.2 — Перетворення зображення в одномірний вектор

Інший дуже важливим момент полягає в тому, що часто зображення, які потребують класифікації і будуть оброблятися нейронною мережею, можут мати інший ракурс на графічний об’єкт, або різний кут повороту самого об'єкта. Деякі частини зображення можуть мати хороше освітлення і гарну якість, в той час як в іншій частині освітлення буде гіршим і контури об'єкта не можна буде так чітко виділити серед пікселів.

Таким чином, графічний образ буде розміщуватися в різних частинах зображення і може, наприклад, бути навіть обрізаний при збільшенні. Нейронна мережа повинна вміти знайти все той самий клас зображень, який представлений набором пікселів в абсолютно хаотичних

варіантах повороту, зміщення тощо. При одновимірному векторі, що подається на вхід, ваги кожного наступного нейрону після першого рівня будуть мати свої значення, як і зміщення, і таким чином нейрони не будуть пов’язані між собою, при навчанні потрібно буде зробити набагато більше ітерацій, щоб правильно вирішувати таку задачу [14]. Це не є оптимальний підходом, оскільки обрабка зобраежень є досить вимогливими процесом до обчислювальних можливостей машини. І якщо вибірка зображень є малою, то взагалі вирішення цієї задача ставиться під сумнів.

Якщо аналізувати складність обчислень і саму структуру нейронної мережі, що буде працювати з одновимірним масивом на вході, то зі збільшенням розміру зображення кількість нейронів в мережі буде зростати пропорціонально, а час навчання буде зростати за експонентою. Знову ж таки, це пов’язано зі слабким зв’язком між нейронами та необхідністю їх навчання окремо (обчислення зміщень та ваг для кожного нейрону на кожній ітерації навчання). Навіть при досить невеликих розмірах зображення, враховуючи той факт що нейронна мережа буде містити кілька рівнів, то кількість параметрів для налаштування може становити десятки тисяч. І більшість з цих параметрів будуть вимагати додаткових обчислень, причому невідомо, чи дасть це потрібний результат, оскільки ітерація може забрати ресурси і час навчання, намагаючись виділити той чи інший контур на зображенні. Відповідно, це викликає додаткове навантаження на відеокарту, і при ускладненні структури мережі обчислювальних ресурсів може бути недостатньо. Завжди має бути запас потужності для проведення досліджень, адже можливо є сенс додати ще один рівень в структурі мережі тощо. Але тут варто розуміти, що ускладнення структури може мати і негативний ефект, оскільки збільшується кількість локальних мінімумів і при навчанні нейронна мережа може почати видавати поганий результат.

У випадку класичних класичних методів проблема аналогічна — велика кількість вхідних ознак, які сильно корельовані один з одним, причому кожна ознака окремо несе дуже малу кількість інформації. Дерева рішень можуть бути одним із варіантів реалізації класифікації зображень, але знову ж таки, їм просто не вистачить інформації для правильного розділення на класи.

2.3 Інтуїтивне осмислення задачі розпізнавання зображень

Як було описано вище, при обробці зображення звичайними лінійними нейронними мережами на вході подається вектор із пікселів. По суті, значення вектору будуть виступати параметрами деякої функції, що має видавати результат після обробки. Цією функції, власне, і виступає нейронна мережа (рис. 2.3):

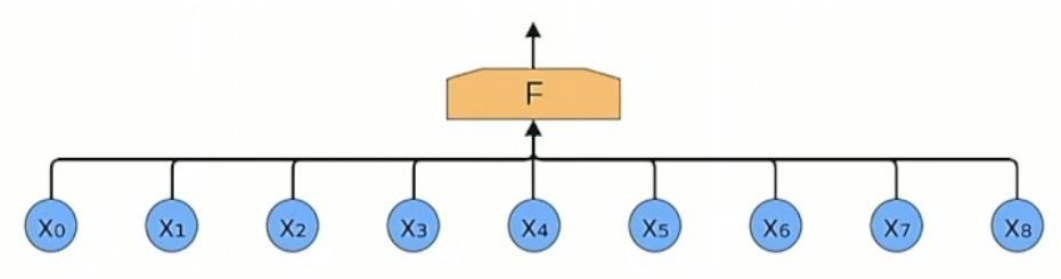


Рисунок 2.3 — Концепція нейронної мережі як функції

Тепер варто розглянути дещо оптимальніший варіант, при якому входи нейронної мережі будуть певним чином локально пов’язані. Можна уявити зовсім малу нейронну мережу, на вході якої буде лише 4 пікселі, але мережа буде точно знати, що вони знаходяться поруч. Ця мала нейронна мережа буде намагатися розпізнати той чи інший графічний образ, або його окремий контур чи шаблон. Звісно ж, в реальних задачах розпізнавання практично неможливо ідентифікувати той чи інший шаблон лише за чотирма пікселями. Той чи інший контур, графічний образ

кодується великими групами пікселів. Однак це хороший приклад для розуміння принципу роботу згорткових нейронних мереж та суті операції згортки (рис. 2.4).

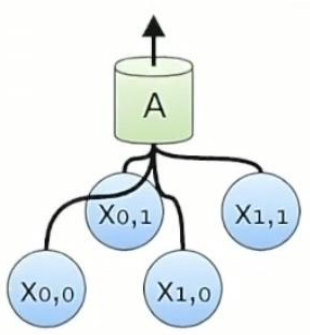


Рисунок 2.4 — Нейронна мережа для пошуку простих шаблонів

Далі можна застосувати дану нейронну мережу до всіх ділянок зображення, і таким чином визначати, чи дійсно присутній шуканий графічний образ на окремих ділянках, що дасть змогу визначити чи наявний образ на зображенні в цілому (рис. 2.5).

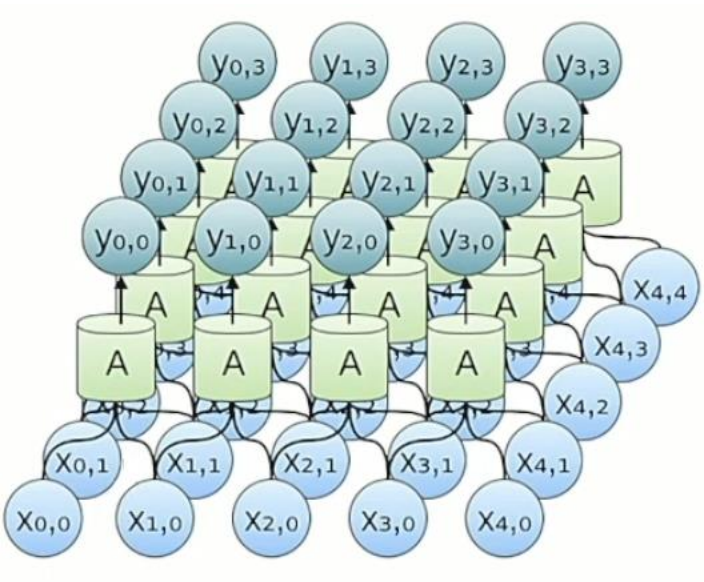


Рисунок 2.5 — Пошук шаблону по всьому зображенню

В результат буде отримана нейронна мережа з всього лише чотирма вагами і зміщеннями для кожного нейрону. Однак, для повного вирішення задачі розпізнавання образів цього ще дуже мало. На даному етапі

отримані ознаки є більш виразними з кожної ділянки зображення, і їх кількість зменшилася. Це вже має позитивний ефект, оскільки далі можна застосувати класичний підхід і побудувати лінійну нейронну мережу, що буде працювати з цими ознаками, час її навчання буде набагато менший відповідно до меншої кількості нейронів, а саме навчання буде більш ефективним, оскільки отримані ознаки є більш “яскравими” [15].

Очевидно що всього лише за чотирма пікселями неможливо розпізнати графічний образ. Але навіть цей простий приклад може вирішувати задачу шляхом передбачення тих чи інших ліній на зображенні, контурів тощо. Отримані значення після обробки кожної групи пікселів будуть слугувати вхідним вектором або матрицею для нейронної мережі (рис. 2.6).

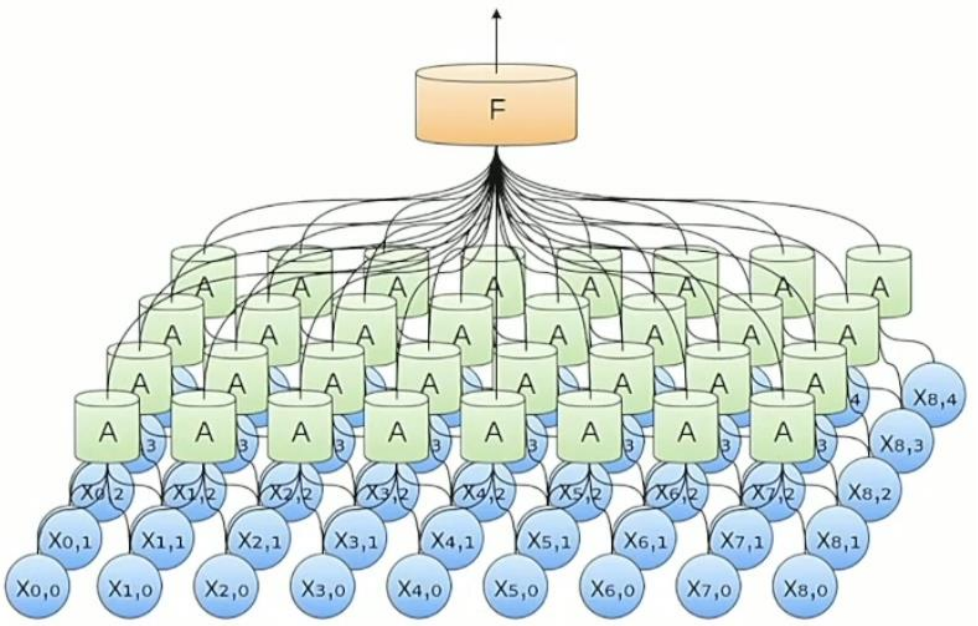


Рисунок 2.6 — Концепція однорівневої згорткової нейромережі

На даному етапі варто застосувати функцію активації, оскільки це полегшить роботу мережі на наступному рівні. Як було описано в першому розділі, це може бути функція сигмоїди або ReLU. До отриманої структури можна додати ще один рівень, який по суті має ще більше абстрагувати ознаки зображення і виділяти лише потрібні образи. Варто

зауважити, що вже отримується ієрархічна структура, в якій мережа А виділяє якусь локальну осанку (наприклад, наявність лінії), мережа В буде працювати на основі взаємопов'язаних ознак з мережі і таким чином, може робити передбачення чогось більш складного за отриманими ознаками ліній, переходу кольорів, контурів тощо. Все це можна в кінці пропустити через деяку мережу F, що буде агрегувати інформація по всьому зображенню для виявлення тих чи інших графічних образів.

Даний опис являє собою інтуїтивний підхід до вирішення проблеми не пов'язаних пікселів при використанні класичних методів. Насправді, отримана структура буде працювати досить повільно і не є реальним рішенням задачі класифікації зображень. До того ж, така структура може розпізнавати якусь ознаку, а не їх комбінацію. Однак дана теорія лягла в основу для використання згорткових нейронних мереж при створенні комп'ютерного зору.

2.4 Згортка

Поняття згортки з'явилося в математиці задовго до існування будь-яких нейронних мереж. Згортка — це математична операція двох функцій *f(t)* та *g(t)*, яка дозволяє отримати третю функцію:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(t-\tau)g(\tau)d\tau \qquad (2.1)$$

Головна властивість згортки полягає у тому, що фур'є-образ згортки пропорційний добутку фур'є-образів двох функцій. Згортки та пов'язані з ними операції дуже широко використовуються в науці, математиці та інженерії в цілому. При обробці зображень, в більшості випадків будь-який фільтр реалізований саме через операцію згортки (підвищення різкості, розмиття, виявлення контурів тощо). Фотографічний термін Боке

також являє собою операцію згортки, в цьому випадку несфокусована фотографія є згорткою справжнього (чіткого) зображення з функцією лінзи.

Для знайомства з операцією згортки простіше всього розглянути одновимірну згортку, яка буде працювати з одновимірним сигналом. Одновимірний сигнал — це просто масив чисел. Шаблон, з яким буде співставлятися сигнал, являє собою також одновимірний масив. Цей масив називається ядром згортки.

При обробці сигналу його елементи перемножаються з елементами ядра згортки, після чого результат сумується (рис. 2.7):

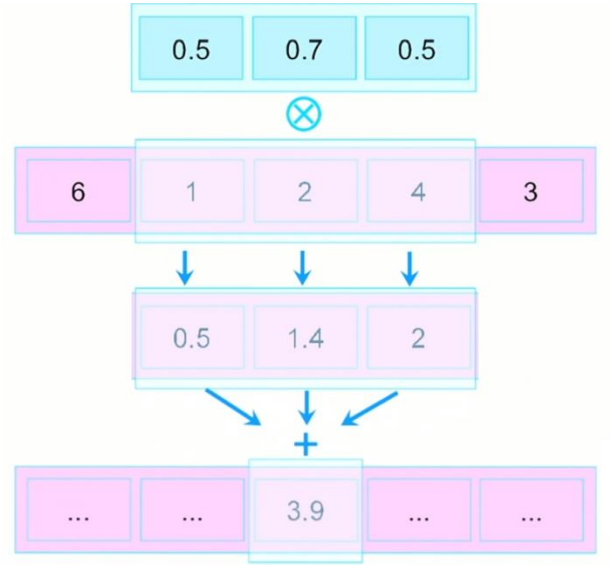


Рисунок 2.7 — Приклад операції згоркти на одновимірному векторі

Ядро згортки можна налаштувати таким чином, щоб отриманий результат після операції згортки був найбільшим лише в тій ділянці сигналу, де знаходиться необхідний шаблон. Таким чином згортка буде максимально виділяти потрібну частину сигналу, дана операція може бути трактована як своєрідний фільтр.

Для застосування операції згортки на зображеннях варто розглянути 2D випадок, коли на вході буде матриця з пікселів зображення замість одновимірного сигналу. У випадку чорно-білих зображень інформація

буде передаватися у вигляді звичайної матриці. При цьому ядро згортки теж має бути двовимірним, тобто матрицею відповідного розміру n x n.

Ядро поелементно перемножається зі значеннями матриці на кожному кроці, і отриманий результат є сумою добутків. Ця сума і передається далі у вигляді елементів нової матриці. Щоб заповнити цю матрицю, потрібно покроково зміщувати ядро по елементам початкової матриці і проводити операції добутку та сумування (рис. 2.8).

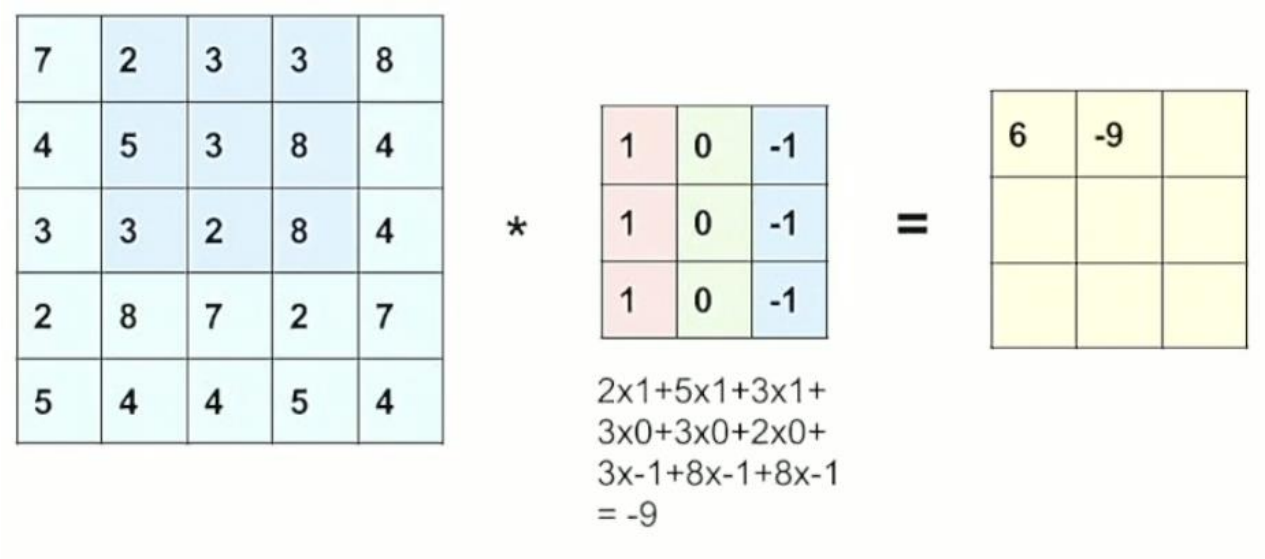


Рисунок 2.8 — Операція згортки на прикладі двомірної матриці

Як уже було описано вище, за таким алгоритмом можна застосувати різні фільтри для зображення, наприклад розмиття або підвищення різкості. Все залежить від налаштування елементів матриці ядра. Наприклад, якщо потрібно розмити зображення, то варто значення матриці зображення розділити на певне число в залежності від ступеня розмиття. В такому випадку всі елементи матриці ядра згортки будуть мати значення n^{-1}, де n — ступінь розмиття. Або при підвищенні різкості зображення ядро має максимально підсилювати центральний елемент і відрізняти його від сусідніх пікселів. Якщо уявити фільтр розміром 3 на 3, то центральний елемент буде мати значення n — ступінь підвищення різкості зображення, а всі інші елементи можна заповнити значеннями -1.

Як відомо, реальні зображення зазвичай мають певний тип кодування для передачі кольорів. Це означає, що вхідна матриця не може бути одновимірною. Для RGB кодування каналів буде 3 для передачі значень червоного, зеленого та синього кольорів відповідно. Це значить що для кожного каналу буде окремо виконуватися операція згортки, причому значення матриці ядра повинні бути різними для кожного кольору. Це пояснюється тим, що нейронна мереж повинна вміти розпізнавати специфічні зображення. Наприклад, квіти червоного кольору. В такому випадку немає сенсу однаково налаштовувати ядро для трьох каналів, оскільки реакція нейронної мережі на синій та зелений кольори має бути мінімальною (інколи в таких випадках значення елементів матриці ядра може дорівнювати нулю). Кожен фільтр буде шукати свій шаблон на зображенні (рис. 2.9).

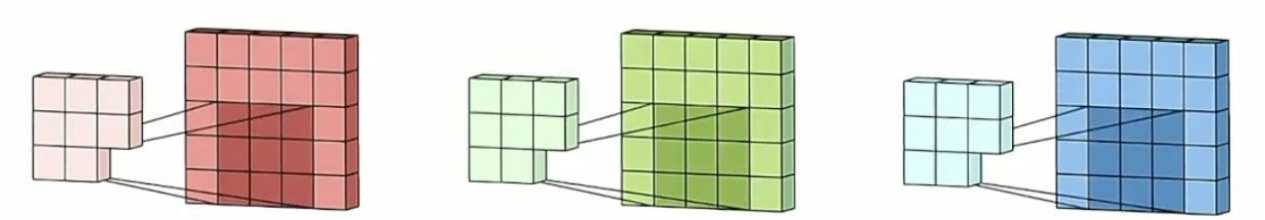


Рисунок 2.9 — Застосування різних фільтри для операції згортки по різним каналам зображення [16]

На наступному етапі отримані матриці після операції згортки сумуються, тобто отримується одна матриця ознак, елементами якої є сума елементів матриць операції згортки червоного, зеленого та синього каналів. Таким чином можна отримати сумарну реакцію нейронної мережі з налаштованими фільтрами каналів на вхідне зображення. Якщо, наприклад, зображення не містить кольорів синього відтінку, то очевидно що матриця після операції згортки по синьому каналу буде мати нульові або максимально наближені до нуля елементи. Якщо фільтри змогли виявити той чи інший колір або шаблон, то операція сумування буде

давати зважений результат для подальшої обробки на наступних рівнях нейронної мережі (рис. 2.10).

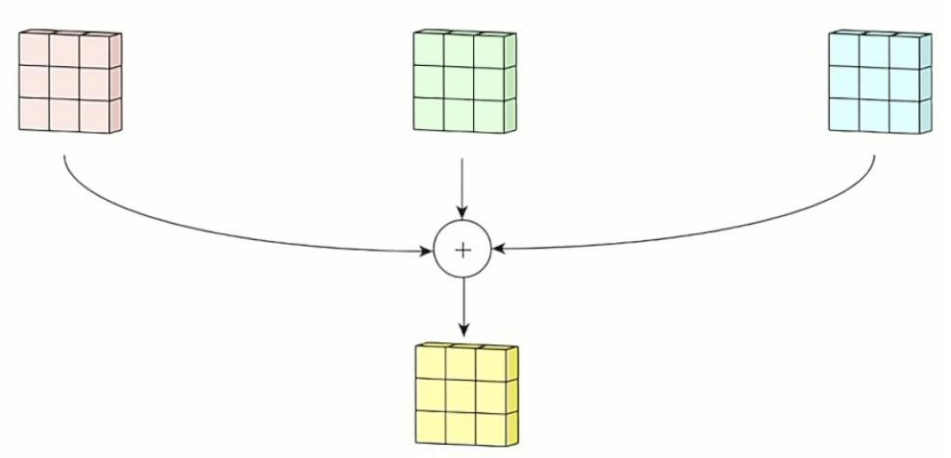


Рисунок 2.10 — Сумування результатів операції згортки по різним каналам зображення

Таким чином із трьохканального сигналу буде отриманий одноканальний. На даному етапі це не є принциповим, однак по аналогії до лінійних нейронних мереж, до отриманої матриці може додаватися певне зміщення (bias). Це константа, яка слугує для того, що змістити той самий сигнал в ту чи іншу область функції активації (рис. 2.11):

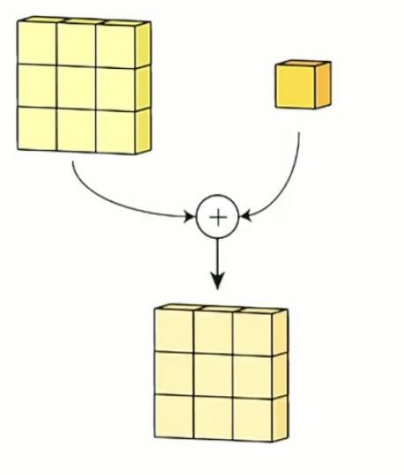


Рисунок 2.11 — Додавання зміщення до результату операції згортки

Після отримання одноканального сигналу, до нього можна застосувати певне обмеження чи перетворення. Наприклад, якщо яскравість пікселя на зображенні вимірюється в діапазоні від 0 до 255 згідно кодування RGB, то можна на зображенні виділити самі яскраві ділянки шляхом обмеження діапазону можливих значень тільки від 240, наприклад, до 255. Таким чином, після опрацювання зображення фільтром результатом буде зображення із конкретними точками або набором пікселів на тій ділянці, де був знайдений шаблон [17].

Фактично, це і є застосування передавальної функції ReLU. Станом на зараз, ReLU є найбільш популярною передавальною функцією для глибинних нейромереж. Дана функція описується формулою:

$$f(x) = x^+ = \max(0, x), \tag{2.2}$$

де x — це вхідне значення нейрона.

Графік функції ReLU зображнний на рис. 2.12:

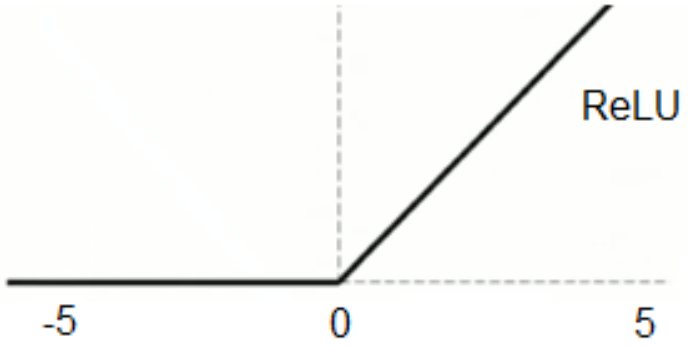


Рисунок 2.12 — Графік функції активації ReLU

Також ReLU може бути параметричною в залежності від специфіки використання. Можна змінювати нахил кривої, тобто буде введений додатковий коефіцієнт активації.

$$f(x) = x^+ = \max(0, ax), \tag{2.3}$$

2.5 Згортковий шар

Першим шаром згорткової нейронної мережі завжди є згортковий шар. Операція згортки застосовується до вхідних даних, а результат передається на наступні рівні. При роботі із зображеннями, на вході згорткового шару є двомірна матриця, відповідно, результатом операції згортки також буде матриця. На останніх згорткових шарах результатом є одномірний вектор, що характеризує приналежність зображення до того чи іншого класу.

Аналогічно до лінійних нейронних мереж, де в процесі навчання мережа змінювала параметри (ваги та зміщення нейронів), згорткові нейронні мережі також здатні до навчання з учителем. Вся мережа так само виступає диференціальною функцією оцінки: від необроблених пікселів зображення на вході до оцінок класу на виході. Всі методики для навчання звичайних лінійних мереж також застосовуються і до згорткових нейромереж. Але параметрами виступаються саме значення ядра згортки, тобто значення матриці згорткового шару. Цю матрицю неможливо підібрати для успішного розпізнавання зображень мережею. Можливо, підбір параметрів допоможе краще зрозуміти суть роботи згорткових нейронних мереж на простих прикладах (зображення простих геометричних фігур, таких як квадрат, коло, трикутник тощо), однак для налаштування мережі при роботі з реальними зображеннями єдиним варіантом підбору елементів матриці ядра є процес навчання з учителем. На початковому етапі значення елементів ядра — це випадкові числа, так само як було в лінійних нейронних мережах для значень ваг та зміщень. В залежності від типу проблеми, що вирішується згортковою нейромережею, а також від того, які функції досліджуються, можна застосовувати різні види згорток.

Фільтрів для навчання може бути багато, оскільки кожен окремий фільтр розпізнає лише характерний для нього шаблон (графічний образ або його частину) [18]. Фактично, фільтри будуть малими нейронними мережами, що проходять по всьому зображенню і результатом їх роботи є карта ознак (англ. "feature map"). Фільтри та карти ознак є двома основними елементами, що забезпечують роботу згорткових нейронних мереж (рис. 2.13).

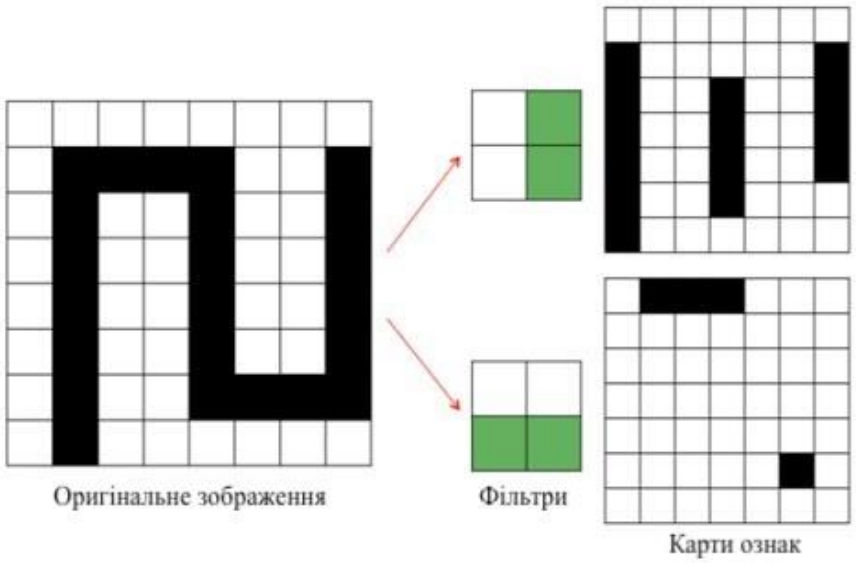


Рисунок 2.13 — Операція згортки по зображенню та отримані карти ознак

Кожен фільтр після операції згортки видає карту ознак, відповідно чим більше фільтрів застосовується в нейронній мережі, тим більше буде карт ознак на виході після першої операції згортки. Також кількість карт ознак залежить від типу кодування графічного зображення. Наприклад, при RGB кодуванні кількість карт ознак буде втричі більшою в порівнянні, наприклад, із чорно-білим зображенням.

В сукупності карти ознак можна трактувати як багатоканальне зображення на виході після операції згортки. За замовчуванням, до результату операції згортки додається зміщення (bias), кожна карта ознак має своє значення зміщення. Після додавання зміщення карта ознак опрацьовується тієї чи іншою функції активації (як було описано вище,

зазвичай це ReLU). Отримана матриця і є результатом на згортковому шарі, який передається на наступні рівні нейронної мережі (рис. 2.14):

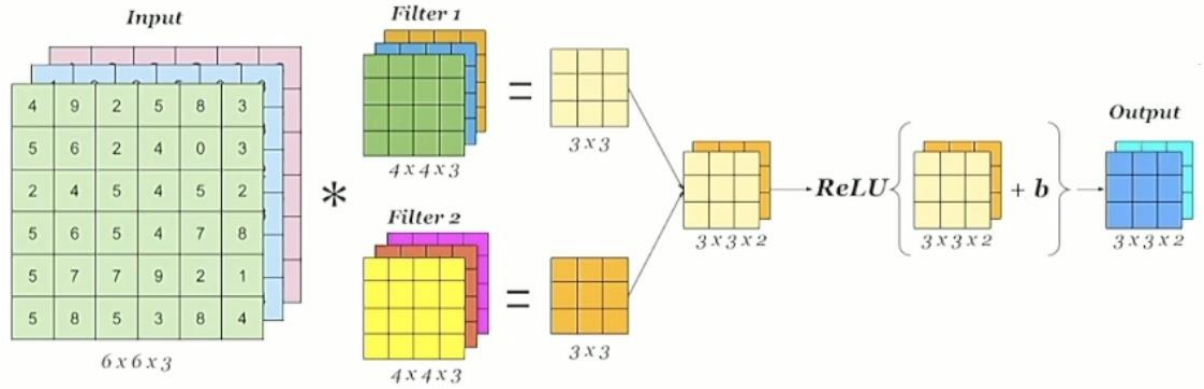


Рисунок 2.14 — Шар згортки [19]

На згортковому рівні нейронної мережі є ще кілька важливих факторів, які безпосередньо впливають на якість розпізнавання і значення або механізм цих факторів можна налаштовувати.

Першим фактором є крок, з яким здійснюється операція згортки по вхідному зображенню. В англомовній термінології цей термін називається “stride”, з англ. — крок. Операція згортки може виконуватися з кроком в 1, 2 або n пікселів. При збільшенні кроку розмір матриці карти ознак буде пропорційно зменшуватися в порівнянні з розміром вхідного зображення. Це може дуже корисним як мінімум для того, щоб зменшити розмір зображення. При великому розмірі вхідної матриці доведется дуже багато разів виконувати операцію згортки, щоб отримати відносно малий розмір карти ознак і передати її на наступний рівень нейронної мережі.

Також варто зауважити, що з практичної точки зору буде правильним припущення про те, що пікселі, які знаходяться поруч на зображенні, можуть бути однакові або відносно ідентичні [20]. Це означає що немає сенсу робити операцію згортки з кроком в один піксель, адже це не дасть виразності елементам на карті ознак. Тому при роботі з великими зображенням цілком правильно підібрати крок операції згортки більше за

1 піксель (інколи це значення може бути навіть 10, 20 пікселів тощо). Приклад використання різних значень кроку зображено на рис. 2.15:

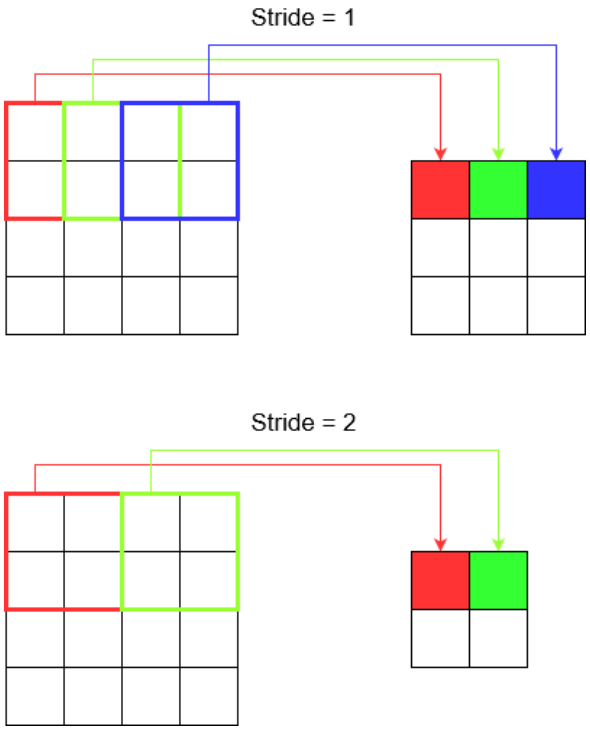


Рисунок 2.15 — Вплив кроку (stride) на розмір карти ознак

Другим важливим параметром може бути налаштування відступів (padding). При операції згортки може втрачатися інформація про зовнішні пікселі на зображенні. Тому дуже часто використовують підхід із додаванням одного або кількох пікселів по краях вздовж усього периметру зображення (один або кілька пікселів — залежить від розміру матриці ядра згортки та обраного кроку). В більшості випадків ці крайні пікселі заповнюються нулями.

Для чого і в яких випадках застосовується Padding? По-перше, при виконанні операції по всьому зображенню крайні пікселі враховуються лише один раз, в той час як пікселі всередині зображенню при малому кроці будуть приймати участь в згортці кілька разів. Це означає, що інформація з країв зображення втрачається або її вплив на карту ознак

буде несуттєвим. Завдяки використанню методу Padding крайні пікселі будуть приймати участь в операції згортки кілька разів. По-друге, варто розуміти, що завжди при здійсненні операції згортки, карта ознак буде матрицею меншого розміру, ніж вхідне зображення. Це добре для стиснення інформації при роботі з великими зображеннями, однак бувають задачі, при яких розмір результуючої матриці (карти ознак) має бути таким самим, як і розмір вхідного зображення. Це можна компенсувати завдяки додаванню пікселів по краям зображення, тобто використовуючи метод Padding. В деяких випадках замість використання нулів для заповнення доданих пікселів можуть бути використані значення крайній пікселів зображення [21]. Таким чином Padding буде збільшувати зображення, при чому контекст залишиться той самий, єдине що інформація з країв стане більш впливовою на карту ознак. Приклад застосування методу Padding зображено на рис. 2.16:

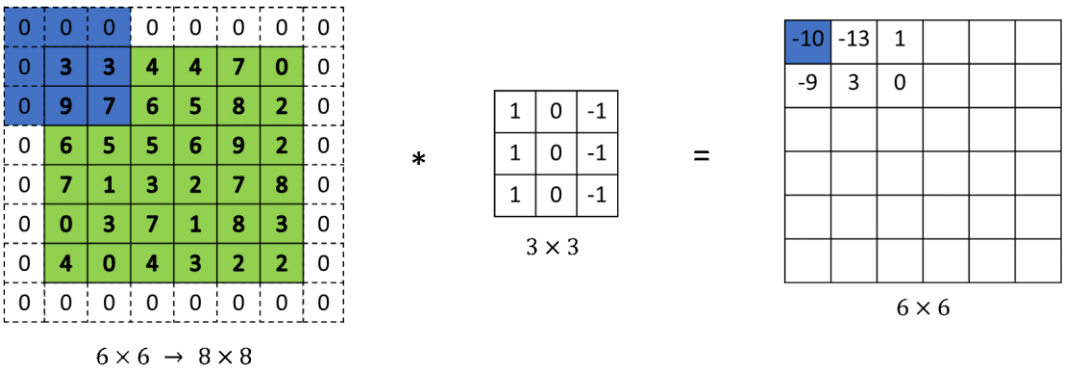


Рисунок 2.16 — Приклад використання Padding

2.6 Рецептивне поле

При проектуванні структури згорткової нейронної мережі завжди постає питання вибору розміру матриці ядра, тобто фільтру. В першу чергу це залежить від потенційного розміру вхідного зображення, з яким

буде оперувати нейронна мережа, а також від складності чи необхідної точності пошуку ознак на зображення.

Рецептивне поле — це і є розмір матриці ядра, тобто набір пікселів на зображенні, що можуть нести інформацію про ту чи іншу ознаку графічного образу. Щільність рецептивного поля — це кількість інформації, яка здатна поміститися в межах групи пікселів, що розглядається. Після операції згортки до отриманих сигналів зазвичай застосовується лінійне перетворення, після чого сигнали переходять на наступний рівень нейронної мережі. Так відбувається процес ущільнення рецептивного поля. Варто розуміти, що це ущільнення або зменшення розміру вхідного зображення. Після кожної операції згортки отримується матриця, кожен елемент якої відповідає групі пікселів початкового зображення. Тобто кожен елемент сигналу після згортки є узагальненням певної частини вхідного зображення (рис. 2.17).

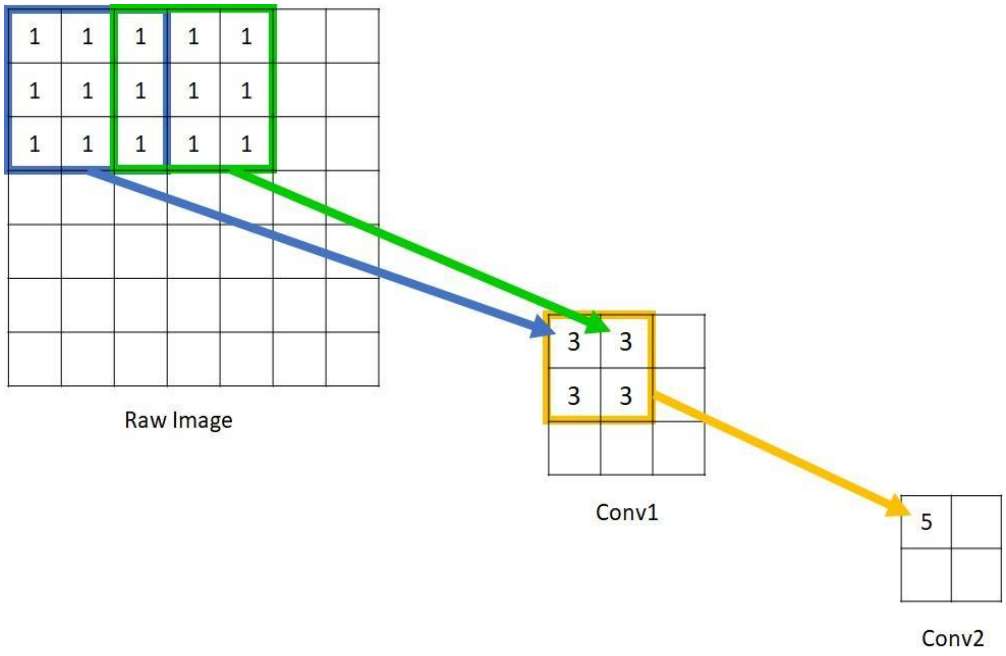


Рисунок 2.17 — Ущільнення рецептивного поля

Ущільнення рецептивного поля дає можливість згортковим рівням мережі перетворювати низькорівневі характеристики (лінії, границі) в

ознаки більш високого рівня (криві, шаблони, текстури). Таким чином нейронна мережа формує все більш складні детектори високорівневих характеристик (фрагментів графічного образу, їх поєднання на зображенні).

Шари початкових рівнів переходять в стан активації при наявності низькорівневих елементів, а шари більш глибоких рівнів вже працюють з високорівневими елементами, щоб розпізнати, наприклад, геометричну фігуру, тварину, обличчя тощо. Згортков нейронна мережа зазвичай починається з невеликох кількості фільтрів, які розпізнають елементарні ознаки на зображенні, і поступово переходить до більшого числа фільтрів, кожен з яких займається розпізнаванням високорівневих образів.

Все завершується пулінговим рівнем (від англ. “pooling” — об’єднання, ущільнення), який фактично перетворює матриці малого розміру в єдиний піксель. Цей піксель, власне кажучи, і є детектором характеристики вхідного зображення з дуже ущільненим рецептивним полем. Пулінговий рівень по суті теж є згорткою, однак для передачі сигналу на наступний рівень не використовується множення матриць рецептивного поля та фільтру, як це виконується на звичайному згортковому рівні (рис. 2.18). З певної групи пікселів може братися один за методом вибору максимального значення (max pooling), або може вираховуватися середнє арифметичне значення цієї групи (average pooling).

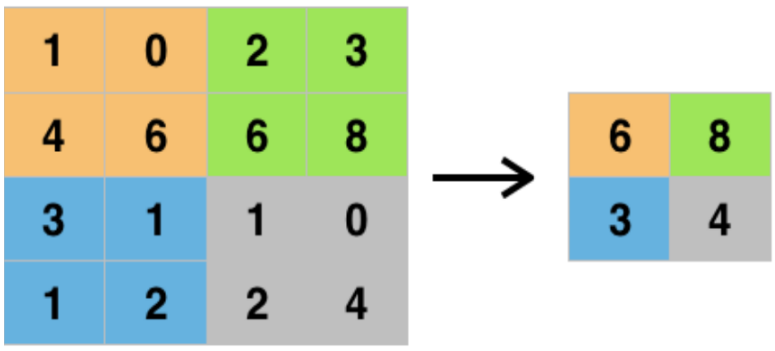


Рисунок 2.18 — Приклад здійснення операції ущільнення

Це є дуже ефективним методом зменшення розміру зображення, оскільки ніяких нових параметрів не додається в структурі нейронної мережі. Окрім стиснення зображення, пулінговий рівень несе в собі ще кілька додаткових переваг. По-перше, завдяки операції ущільнення збільшується розмір рецептивного поля, оскільки групу пікселі з рецептивного поля можна зберегти з невеликим відхиленням через вибір максимального або середнього значення. Деякі пікселі з групи рецептивного поля можуть не мати такого суттєвого впливу, щоб зберігати кожен з них. Наприклад, для згорткової нейронної мережі не є суттєвим кут нахилу того чи іншого образу на зображенні, або колір заднього фону. Навіть є важливим місцеположення ознаки, головне її наявність на зображенні в певній ділянці. По-друге, ущільнення через операцію пулінг допомагає боротися з шумом на зображенні, тому що при наявності шуму є сенс вибрати з групи пікселів найяскравіший і проігнорувати інші, через це найпоширенішим методом є саме max pooling. Ще одним позитивним результатом використання пулінгу можна назвати стійкість нейронної мережі до невеликих зміщень або поворотів зображення. Як уже було описано вище, при наявності пулінгового шару для нейронної мережі не є суттєвим конкретне місцеположення шаблону чи характеристики графічного образу, головне його наявність на зображенні в цілому.

Однак, очевидним недоліком використання пулінговому шару є втрата частини інформації через ігнорування менш яскравих пікселів. Через це в деяких архітектурах згорткових нейронних мереж пулінговий шар не використовується. Існують методи заміни пулінгового шару зі збереженням того ж самого результату — нейронна мережа буде так само стійкою до місцеположення графічного образу або його окремої характеристики на зображенні.

Можна помітити, що згорткова нейронна мережа будує візуальну ієрархію вхідного графічного образу. В процесі навчання ці візуальні ієрархії уточнюються, накопичуються і зберігаються в параметрах матриці кожного ядра, тобто фільтру. Починаючи з аналізу низькорівневих характеристик і, використовуючи їх для побудови візуальних ієрархій, згорткова нейронна мережа після навчання може виділяти складні графічні концепції та шаблони: розпізнавати людей, тварин, емоції на обличчі тощо. Саме це робить дану архітектуру настільки ефективною в прикладних задачах аналізу і розпізнавання графічних образів.

2.7 Висновки до розділу

Використання згорткових нейронних мереж в задачах класифікації зображень є найбільш ефективним методом на сьогоднішній день, порівнюючи з класичним методами (дерева рішень, алгоритмічний підхід). Після появи згорктової нейронної мережі AlexNet довгий час всі спроби покращити точність розпізнавання фактично зводилися до вдосконалення саме цього типу архітектури. При поступовому ущільненні рецептивного поля, операція згортки дозволяє перейти від виявлення простих графічних образів чи їх характеристик до високорівневих шаблонів. Впродовж останніх років згорткові нейронні мережі демонструють високу точність розпізнавання образів при аналізі зображень.

3 РОЗРОБКА СИСТЕМИ

3.1 Опис бібліотек програмного забезпечення

Інтелектуальна система розпізнавання образів реалізована у вигляді комп’ютерної програми, яка здатна обробляти вхідні зображення і класифікувати їх за наявності тих чи інших образів. Програмне забезпечення розроблене мовою програмування Python. В першу чергу це пов’язано з наявністю готових бібліотек для побудови і навчання нейронних мереж. Ці бібліотеки добре документовані, є досить гнучними в налаштуваннях, а також дозволяють легко підключити створену нейронну мережу в якості функціоналу прикладної десктоп-програми з відповідним графічним інтерфейсом. Для розробки графічної частини програми використано фреймворк PyQt, який є найбільш поширеним і популярним інструментом створення інтерфейсу користувача мовою програмування Python. Фреймворк є повністю безкоштовним, поєднує в собі підходи розробки класичного фреймворку Qt для реалізації програм в середовищах різних операційних систем, таких як Windows, MacOS, Linux, Android тощо.

Варто зауважити, що машинне навчання і мова Python є досить пов’язаними в сучасних дослідженнях та розробках відповідних інформаційних систем. Як було зазначено вище, основною причиною є спеціальні бібліотеки, що розробляються і підтримуються саме з використанням цієї мови програмування. До вагомих факторів, що спонукають дослідників використовувати саме Python можна також віднести високу обчислювальну здатність мови, простий синтаксис, кросплатформеність, легкість створення прототипів. Сама мова є досить гнучкою з точки зору підходу програмування. Наприклад, у розробника є вибір між об’єктно-орієнтованим підходом та скриптами. Більше того, Python особливо зручна мова грамування для тих розробників, які пишуть

(або звикли писати) більшу частину коду з допомогою IDE. Перераховані фактори дають пояснення, чому Python так широко і активно застосовується в сфері машинного навчання і створенні відповідних алгоритмів навчання нейронних мереж, в тому числі згорткових.

Для реалізації архітектури нейронної мережі, її ефективного навчання та раціонального використання обчислювальних ресурсів машини необхідно мати інструмент введення та обробки математичних операції. Таким інструментом є бібліотека NumPy. Саме через неї реалізовуються всі математичні операції над векторами та матрицями в процесі навчання та безпосередньої роботи нейронної мережі. Містить в собі безліч готових методів для різного роду математичних операцій, і є схожою до мови програмування і пакету Matlab.

Для аналізу ефективності роботи нейронної мережі необхідно порівнювати отримані результати класифікації зображень з істинними значенням, аналізувати відсоток помилки тощо. Зазвичай, будь-який аналіз схожого типу неможливий без використання графіків та діаграм. Саме з цієї метою розроблена система використовує бібліотку Matplotlib, яка представляє собою низькорівневий набір файлів для побудови графіків функцій та двомірних діаграм. Ця бібліотека є зручною для побудови будь-якого типу графіків. Однак, є і недолік її використання, яким полягає у тому, що для реалізації складних візуалізацій процесів, наприклад, необхідно написати набагато більше коду, в порівнянні з аналогами, серед яких є той самий пакет Matlab.

Основним інструментом реалізації нейронної мережі в системі є бібліотека Keras. Це дуже широко розповсюджена бібліотека з активною підтримкою, що робить машинне навчання настільки простим в реалізації, наскільки це можливо в межах сучасних мов програмування. Раніше використання глибинних нейронних мереж було проблемою. Існувала складність використання тих чи інших програмних середовищ,

пов’язана зі складними інтерфейсами інтеграції, налаштування нейронної мережі. Було кілька концепцій по спрощенню та вдосконаленню високорівневого API для побудови моделей нейронних мереж. Keras саме і є одним із провідних API такого типу, для створення і роботи з глибинними нейромережами різних архітектур. Даний інструмент написаний мовою Python і підтримує кілька серверних обчислювальних механізмів. Keras був створений для того, щоб бути зручним користувачеві, реалізує модульний підхід, може бути легко розширений. Іншими словами, інтерфейс був створений для людей, а не для обчислювальної машини. Нейронні рівні, функції ваг, оптимізатори, схеми ініціалізації, функції активації — це приклади окремих модулів, які можна комбінувати і налаштовувати для створення нових моделей. Можна легко додати нові функції і класи. Моделі визначаються і описуються в коді Python, а не в окремих файлах конфігурації моделей, що є досить зручним підходом з точки зору архітектури проекту. Власне Keras не виконує низькорівневих обчислювальних операцій, таких, наприклад, як згортка, для цього бібліотека використовує свій власний внутрішній інструментарій з інших пакетів [22]. Також обчислення можна реалізувати на сервері, для цього є відповідні інтерфейси взаємодії всередині Keras.

Модель — основна структура даних в Keras. Доступні два типи моделей: послідовна модель (Sequential model) та власна клас (Model), що можуть бути використані з функціональним API бібліотеки. Послідовна модель представляє собою лінійний набір рівнів, які можна легко описати та налаштувати. Ініціалізація кожного рівня вимагає лише одного рядка коду, так само як і компіляція (визначення процесу навчання), власне навчання, оцінка (розрахунок втрат і метрик), прогнозування вихідних даних навченої моделі тощо. Завдяки цьому є можливість застосування скриптів для реалізації тієї чи іншої архітектури нейронної мережі, її

модифікація чи зміна структури є інтуїтивно зрозумілою для програміста. Приклад ініціалізації рівнів та базових обчислювальних операцій з використання послідовної моделі в Keras зображено на рис. 3.1:

```
import keras
from keras.models import Sequential
from keras.layers import Dense

#Створення послідовної моделі, що буде містити в собі основні рівні нейронної мережі
model = Sequential()

#Основні рівні (Dense layers) реалізують наступні операції:
#         output = activation(dot(input, kernel) + bias)
#Активації пропорційні до значень останнього рівня нейронної мережі

#Активації та функція помилки може бути налаштована через прості класи чи константи
model.add(Dense(units=64, activation='relu', input_dim=100))
model.add(Dense(units=10, activation='softmax'))

#Метод 'compile' налаштовує процес навчання мережі
model.compile(loss='categorical_crossentropy',
              optimizer='sgd',
              metrics=['accuracy'])

#Метод 'fit' реалізує процес навчання через групи з набору тренувальних даниї
model.fit(x_train, y_train, epochs=5, batch_size=32)

#Обчислення втрат та метрик для моделі, що навчається
loss_and_metrics = model.evaluate(x_test, y_test, batch_size=128)

#Прогнозування виходів на останньому рівні нейронної мережі
classes = model.predict(x_test, batch_size=128)
```

Рисунок 3.1 — Скріншот програмного коду, що реалізує базові операції з використанням послідовної моделі Keras

Послідовна модель в Keras є досить простою, однак її використання обмежене топологією моделі (кількістю можливих входів та виходів). Для створення більш складних моделей існує функціональний API в Keras, який дозволяє будувати комплексні моделі з різною кількістю входів та

виходів, орієнтовані ациклічні графи (DAG) та абстрактні моделі рівнів. Функціональний інтерфейс Keras використовує ті ж самі рівні, що і послідовні моделі, однак дозволяє більш гнучко їх налаштовувати і об'єднувати [23]. В функціональному API спочатку створюються рівні, потім з них створюється модель, що навчається при компіляції коду. Обчислення і прогнозування є аналогічними, як і в послідовній моделі на Рисунку 3.1, тому ці фрагменти коду пропущені на рис. 3.2, що відображає приклад використання функціонального інтерфейсу в Keras:

```
1  from keras.layers import Input, Dense
2  from keras.models import Model
3
4  #Ініціалізація тензору (багатовимірного масиву)
5  inputs = Input(shape=(784,))
6
7  #Екземпляр рівня виклакається для кожного масиву вхідних
       даних і, відповідно, повертає новий масив значень після
       обчислень і лінійних перетворень функцією активації
8  x = Dense(64, activation='relu')(inputs)
9  x = Dense(64, activation='relu')(x)
10 predictions = Dense(10, activation='softmax')(x)
11
12 #Створення моделі що містить вхідний рівень та три основні
       рівні обчислень
13 model = Model(inputs=inputs, outputs=predictions)
14 model.compile(optimizer='rmsprop',
15                loss='categorical_crossentropy',
16                metrics=['accuracy'])
17 model.fit(data, labels)  #Початок навчання
```

Рисунок 3.2 — Скріншот програмного коду, що створює модель з використанням функціонального інтерфейсу Keras

Наведені приклади використовують лише звичайні щільні шари (Dense layers). Окрім них Keras має широкий вибір попередньо визначених рівнів, а також підтримує написання власних шарів. Шари згортки (використання фільтра для отримання карти ознак по зображенню) виконуються від обчислень над одновимірних масивом до складни реалізацій операції згортки при обробці багатовимірних матриць. Для реалізації нейронної мережі, що розпізнає графічні образи і класифікує зображення, зазвичай використовується операція згортки над звичайною матрицею.

В Keras наявні інструменти для реалізації операції Pooling (зменшення розміру вхідних даних), що описана в попередньому розділі. Дана операція може виконуватися над вхідними даними різної розмірності (від звичайних векторів до матриць), а також підтримує різні варінти реалізації, такі як вибір максимального значення чи усереднення значень пікселів певної ділянки рецептивного поля. Локально з’єднані шари взаємодіють між собою аналогічно до згорткових рівнів, за винятком того, що ваги не впливають на активації в сусідніх рівнях. Keras надає 7 стандартних наборів даних для глибинного навчання через клас keras.datasets. Ці дані включають зображення з різних категорій, таких як рукописні літери та цифри, малі зображення різних класів, близько 25 тисяч відгуків на фільми тощо. Всі ці дані можуть бути використані для налагодження нейронної мережі або для створення простих прикладів використання Keras і реалізації алгоритмів глибинного навчання на базі даної бібліотеки. Репозиторій прикладів Keras містить понад 40 зразків різних моделей та архітектур, серед яких є згорткові нейронні мережі, моделі розпізнавання тексту і послідовностей, генеративні моделі, приклади обробки звуків.

Також при розробці системи використано бібліотеку TensorFlow. Це провідна бібліотека з відкритим кодом, що створена для розробки і

розгортання найсучасніших програм з використанням машинного навчання [24]. TensorFlow активно використовується вченими, розробниками ПЗ і викладачами. Як було описано вище, це платформа з відкритим кодом для машинного навчання, що використовує графи потоків даних (англ. “dataflow graph”). Вузли графа представляють собою математичні операції, а ребра графа — багатовимірні масиви даних (тензори), що протікають між вузлами графа. Ця гнучка архітектура дозволяє описати алгоритми машинного навчання у вигляді графу зв’язаних операцій. Приклад простого графу потоків даних зображено на рис. 3.3:

$$h = ReLU(Wx + b)$$

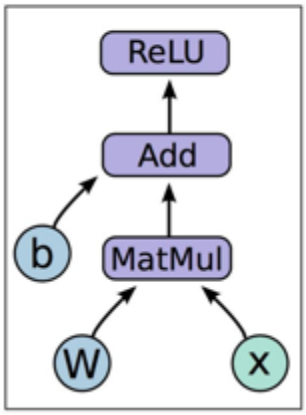


Рисунок 3.3 — Приклад графу потоків даних, що виконує обчислення значення *h* за наведеною формулою

Алгоритми, що запрограмовані у вигляді графів потоків даних, можна навчати та виконувати на графічних та звичайних процесорах різних платформ без переписування коду під кожен тип процесора або операційної системи. Це означає, що алгоритми можуть працювати на портативних девайсах, персональних комп’ютерах та швидко обчислювальних серверах. З самого початку платформа TensorFlow була розроблена командою Google Brain для проведення досліджень в області машинного навчання глибинних нейронних мереж, однак система

виявилась настільки універсальною, що стало можливим її використання і в інших сферах. Як початковий розробник TensorFlow, Google продовжує активну підтримку платформи і прискорює темпи її подальшої розробки. Наприклад, нещодавно команда Google розробила онлайн-центр обміну великою кількістю різних моделей, створених користувачами.

TensorFlow можна використовувати для розробки моделей і систем, що вирішують широкий діапазон прикладних задач, включаючи обробку і розпізнавання природних мов, розпізнавання зображень, рукописного вводу. Також з використанням цієї платформи можна будувати різні обчислювальні моделі, наприклад для вирішення диференційних рівнянь в частинних похідних.

Робочий процес в TensorFlow складається з трьох окремих етапів, а саме: попередньої обробки даних, побудова моделі, навчання моделі робити правильний прогноз (у випадку задачі розпізнавання образів — правильно класифікувати вхідне зображення). Платформа вводить вхідні дані у вигляді багатовимірного масиву. Ці дані називається тензорами. Після цього будується обчислювальний граф, що визначає потік даних для навчання моделі. При використанні архітектури з TensorFlow навчання може проводитися з використанням локальних обчислювальних ресурсів машини, або у віддаленому центрі обробки даних. В обох випадках процес навчання прискорюється за рахунок обробки тензорів графічним процесором.

Ключові переваги платформи TensorFlow полягають в її здатності виконувати низькорівневі операції на багатьох платформах. До цих операцій відноситься автоматичне обчисленні градієнтів складних функцій, масштабування графів потоків даних тощо. Представляючи Keras в якості високорівневого API і миттєву компіляцію в якості альтернативи парадигми потоків даних, писати код завжди зручно і легко на базі даної платформи.

Також TensorFlow містить велику кількість допоміжних інструментів та функцій. Наприклад, TensorBoard, що дозволяє користувачам візуально спостерігати процес навчання, будувати базові обчислювальні графіки і метрики для цілей налагодження параметрів нейронної мережі та оцінки продуктивності моделі (рис. 3.4). TensorBoard — це уніфіковане середовище візуалізації даних для TensorFlow і Keras.

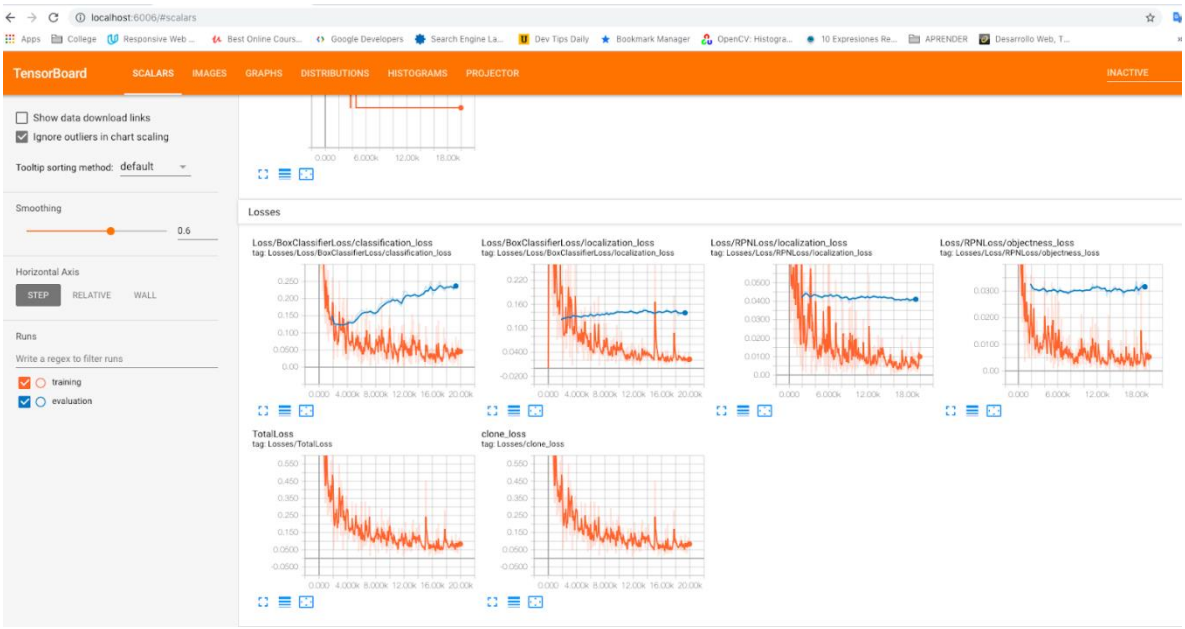


Рисунок 3.4 — Візуалізація результатів навчання нейронної мережі з використанням інструменту TensorBoard

3.2 Архітектура згорткової нейронної мережі

Згорткова нейронна мережа є головним компонентом системи, що розробляється. Саме від неї залежить точність і швидкість розпізнавання графічних образів, а також ефективність використання обчислювальних ресурсів в процесі навчання. Створена нейронна мережа містить 3 згорткових рівні, 3 рівні що виконують операцію Pooling відповідно після кожного згорткового рівня, 1 рівень локально з’єднаних нейронів (на цьому рівні результати операцій згоркти та pooling перетворюються в одномірний вектор), а також 2 повнозв’язні шари, після яких знаходиться

останній рівень нейромережі. Активації нейронів останнього рівня будуть визначати приналежність графічного образу до того чи іншого класу зображень. Клас зображення обирається через найбільшу активацію на останньому рівні. Спрощена структурна схема представленої в системі згорткової нейронної мережі зображена на рис. 3.5:

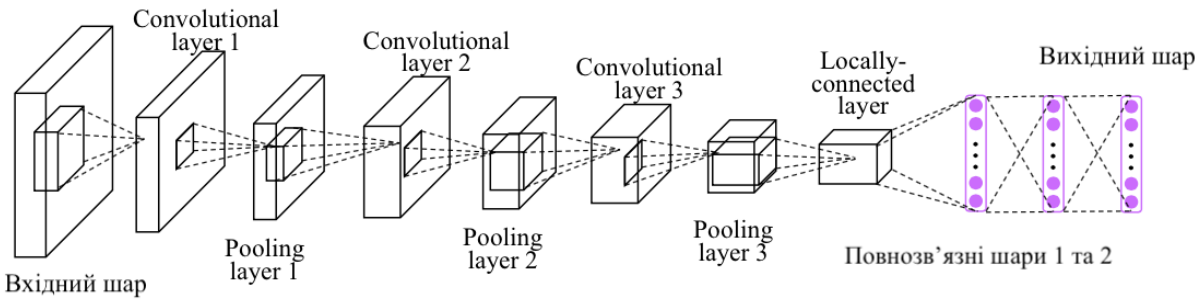


Рисунок 3.5 — Спрощена структурна схема згорткової нейронної мережі

Повнозв'язні шари в згорткових нейронних мережах не варто плутати з повнозв'язними нейронними мережами — класичною архітектурою нейронної мережі, в якій всі нейрони попереднього рівня з'єднуються з нейронами наступного рівня. Як було описано в попередньому розділі, класична архітектура виявилась неефективною для вирішення прикладних задач комп'ютерного зору. На Рисунку 3.5 видно, що в складі згорткової нейронної мержі є також повнозв'язні шари, на вхід першого з яких приходить результат згортки та об'єднання, і потім ці шари приймають рішення про кінцеву класифікацію.

Для того, щоб передати результати операції згортки до повнозв'язних шарів, використовується проміжний рівень, який зазвичай в англомовній літературатурі називається "Fully connected input layer", або "Flatten". На цьому рівні відбувається перетворення вхідних даних в єдиний вектор, який слугує входом для наступного етапу обробки даних. Перший повнозв'язний рівень бере вхідні дані з аналізу карт ознак і

застосовує ваги, щоб спрогнозувати правильний клас зображення. Повнозв’язний останній рівень дає кінцеві вірогідності для кожної мітки або класу зображень. Структурна схема прихованих шарів кінцевої класифікації зображена на рис. 3.6:

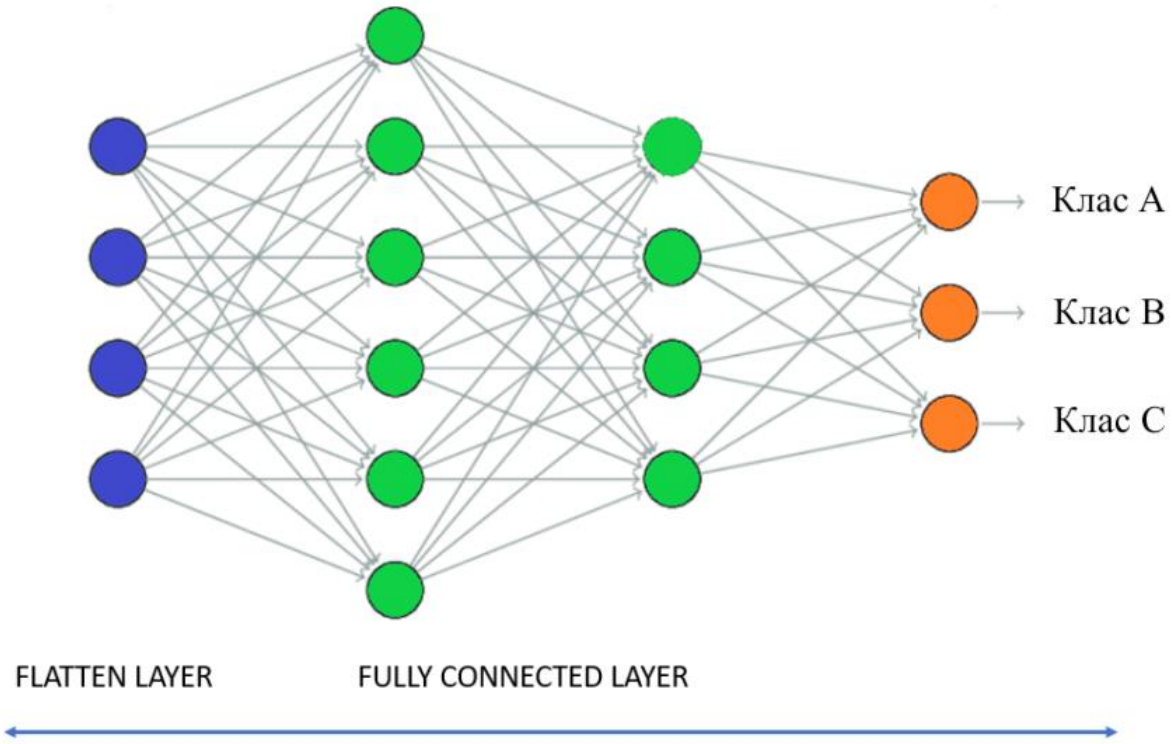


Рисунок 3.6 — Структурна схема повнозв’язних шарів нейронної мережі

Повнозв’язний шар в стурктурі згорткових нейронних мереж — це аналогія прихованих шарів в звичайних лінійних нейронних мережах. Цей рівень представляє собою комбінацію афінного перетворення та нелінійної функції. Перетворення площини (зображення) називається афінним, якщо воно є взаємно однозначним і відображенням будь-якої прямої є пряма. Взаємно однозначне перетворення переводить кожну точку площини (зображення) P в іншу точку площині (зображення) P’, таким чином, що кожній точці P відповідає якась точка P’. Поворот, відображення, розтягування і стиснення зображення — все це приклади

афінних перетворень площини. А згадана вище нелінійна функція — це, зазвичай, ReLU або сигмоїда.

Комбінація афінного перетворення та нелінійної функції створена для того, щоб максимально підсилити активацію вірогідно правильного нейрону останнього рівня, і знизити активації всіх інших нейронів, для того щоб нейронна мережа могла чітко вказати кінцевий клас графічного образу. Можна додати кілька таких шарів в залежності від глибини, з якою буде працювати створена модель класифікації. Варто розуміти, що це повністю залежить від набору тренувальних даних, їх кількості та якості.

Якщо розглянути структурну схему згорткової нейронної мережі, то видно, що початкові шари складаються лише з операцій згортки, pooling, об’єднання. Вони слугують для того, щоб дістати карти ознак із вхідного зображення. В загальному, на основі тренувальних даних можна додавати різні перестановки і комбінації цих шарів. Вихідний рівень згорткової нейронної мережі складається з нелінійної функції перетворення та алгоритму обчислення функції помили.

Програмна реалізація даної архітектури виконана повністю через функціональний інтерфейс Keras то послідовну модель. Вхідні зображення мають розмір 300 на 300 пікселів. Очевидно, що перш ніж нейронна мережа розпочне роботу над класифікацією зображенням, його треба перетворити у матрицю, яка є математичним представленням цього зображення. Оскільки система повинна вміти працювати не тільки з чорно-білими, а й з кольоровими зображеннями, то на вході першого згорткового рівня має бути тривимірна матриця для кодування Reg, Green та Blue каналів. В якості функції активації після кожного згорткового рівня використана функція ReLU. Розмір ядра згортки на кожному рівні — 3 на 3 пікселі. Операція pooling (зменшення карти ознак) виконується з

малих рецептивних полей розміром 2 на 2 пікселі за методом вибору максимального значення (оператор в Keras має назву MaxPooling2D).

Лістинг створення моделі через функціональний інтерфейс Keras представлено на рис. 3.7:

```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense

model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(300, 300, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2), padding = 'same'))

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(128, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(1))
model.add(Activation('sigmoid'))

model.compile(loss = 'binary_crossentropy',
              optimizer = 'rmsprop',
              metrics = ['accuracy'])
```

Рисунок 3.7 — Лістинг коду створення архітектури згорткової нейронної мережі через функціональний інтерфейс в Keras

Кількість карт ознак після першого згорткового рівня становить 32, після другого і третього згорткових рівнів — 64 та 128 відповідно. Для перетворення результатів операції згортки та pooling в одновимірний вектор в Keras існує оператор “Flatten()”. Далі йдуть 2 повнозв’язні шари, кількість нейронів на кожному з яких становить 64, кінцева функція активації — сигмоїда.

3.3 Реалізація попередньої обробки зображень

Більшість інтелектуальних систем, що реалізують технологію комп’ютерного зору, не здатні працювати із зображеннями без їх попередньої обробки. Більше того, попередня обробка зображень дає можливість зменшити час навчання нейронної мережі, підвищити точність розпізнавання на реальних зображеннях, а також підвищити ефективність використання обчислювальних ресурсів.

Створена в даній роботі згорткова нейронна здатна опрацьовувати вхідні дані у вигляді тривимірної матриці розміром 300 на 300 пікселів. Це досить оптимальний розмір, оскільки при використанні найбільш поширених форматів PNG та JPEG якість зображення та деталей на ньому залишається досить високою при вказаній розмірності. Звісно, для впровадження системи для опрацювання потокового відео або високоякісних фото необхідно буде внести невеликі зміни в налаштування нейронної мережі, щоб мати можливість використовувати вхідні матриці великих розмірів. При такому підході немає гарантії, що точність розпізнавання підвищиться, однак при наявності на зображенні декількох або багатьох графічних образів різних класів, очевидно що при більшій розмірності зображення буде втрачатися менше деталей, контурів, шаблонів. Набір тренувальних даних, що був використаний для навчання нейронної мережі, містив зображення лише потрібної

розмірності, тобто 300 на 300 пікселів. Однак, для роботи і опрацювання зображень поза набором тренувальних даних в програмі реалізований метод стиснення зображення мовою Python з використанням бібліотеки Pillow. Дана бібліотека дає можливість легко застосувати базові фільтри для зображення, змінити розмір, контраст, яскравість тощо. Бібліотека Pillow підтримує широкий перелік форматів вхідних файлів, ефективне використання обчислювальних ресурсів та досить потужні можливості обробки зображень в цілому. Основні методи бібліотеки (ядро) призначені для швидкого доступу до даних, які зберігаються одразу в декількох форматах пікселів. Саме це реалізує потужну основу для загального підходу обробки зображень. Реалізація стиснення зображення до потрібного розміру займає лише кілька рядків коду, лістинг коду наведений на рис. 3.8:

```
from PIL import Image

needed_size = 300
needed_dpi = 1

filename = 'dog_class_01.png'
new_filename = 'dog_class_01_DOWNSCALED.png'

img = Image.open(filename)
width, height = img.size

new_size = int(needed_dpi * needed_size)
img = img.resize((new_size, new_size), resample=Image.BICUBIC)
img.save(new_filename, dpi=(print_dpi, print_dpi))
```

Рисунок 3.8 — Реалізація стиснення вхідного зображення до потрібного розміру з метою подальшої обробки нейронною мережею

Таким чином, можна попередньо опрацьовувати всі вхідні зображення і приводити їх до одного формату та розміру. У випадку великої кількості вхідних зображень або при навчанні нейронної мережі

можна застосувати багатопотоковий підхід обробки зображень ще до початку навчання, для того щоб потім зекономити обчислювальні ресурси комп’ютера при виконанні згортки, операції pooling тощо.

В розробленій системі впроваджено можливість застосування оператору Собеля для попередньої обробки зображення та при його класифікації. Оператор Собеля може бути використаний для обробки вхідного зображення, якщо для нейронної мережі одним із основних критеріїв класифікації є саме контури графічних об’єктів, адже після його застосування конутри стають явно видимі і зникає вся зайва інформація. Оператор Собеля використовує ядра розміром 3 на 3 і згортає зображення для обчислення значень частинних похідних по вертикалі та по горизонталі. Приклад обробки зображення з використанням оператору Собеля наведено на рис. 3.9:

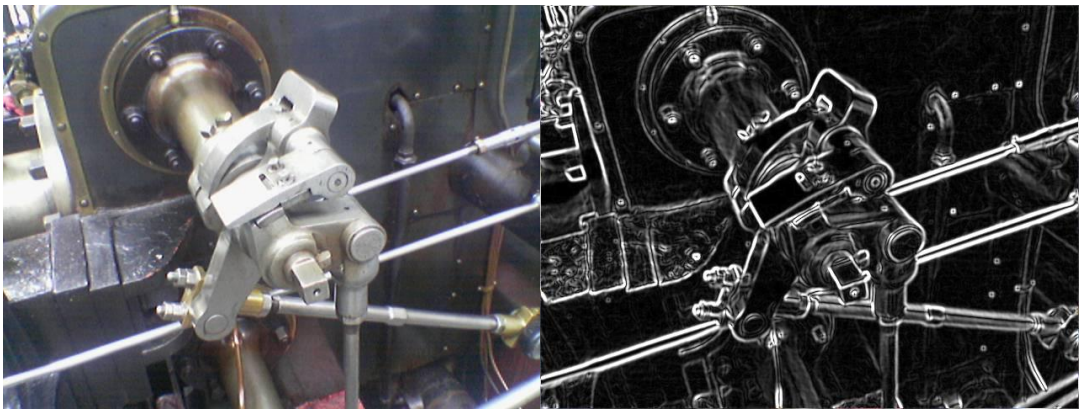


Рисунок 3.9 — Приклад застосування оператору Собеля

Математично оператор Собеля описується наступною формулою:

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * A \quad \text{та} \quad G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * A \qquad (3.1)$$

де G_y та G_x — два зображення, на яких кожна точка містить наближеі похідні по *x* та по *y*;

A — матриця пікселів вхідного зображення.

Після виконання операції згортки для пікеселів в напрямку осі абсцис та ординат, можна обчилити наближене значення величини градієнту шляхом використання отриманих на попередньому етапі наближених значень похідних:

$$G = \sqrt{G_x^2 + G_y^2} \tag{3.2}$$

Значення отриманої матриці G і будуть формувати нове зображення з явно виділеними контурами. Оператор Собеля є одним із найкращих алгоритмів пошуку і виділення контурів та границь на зображенні, він часто зостосовується як один із кроків більш складних і точних алгоритмів, наприклад, таких як алгоритм Кенні.

Реалізація оператору Собеля для попереденьої обробки вхідного зображення представлена на рис. 3.10:

```
1  import numpy as np
2  import cv2
3  import argparse
4  import matplotlib.pyplot as plt
5  import convolution
6  import gaussian_blur
7
8  def sobel_processing(imagePath):
9      image = cv2.imread(imagePath)
10     image = gaussian_blur(image, 9, verbose=False)
11     sobel_processing(image)
12
13     filter = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]])
14
15     new_image_x = convolution(image, filter, true)
16     new_image_y = convolution(image, np.flip(filter.T, axis=0), true)
17
18     gradient_matrix = np.sqrt(np.square(new_image_x) + np.square(new_image_y))
19     gradient_matrix *= 255.0 / gradient_matrix.max()
20
21     return gradient_matrix
```

Рисунок 3.10 — Лістинг коду, що реалізує оператор Собеля

3.4 Алгоритм навчання нейронної мережі

На початковому етапі нейронна мереж є не налаштованою, всі параметри матриць фільтрів, ваг та зміщень необхідно обчислити в процесі навчання. В загальному визначенні під навчанням розуміють ітеративний процес представлення нейронній мережі графічного образу із набору тренувальних дани, і отримана класифікація порівнюється з бажаним виходом. Отримана різниця між отриманим результатом та істинним значенням класифікації буде визначати помилку навчання. Як було описано в другому розділі, процес навчання зводиться до мінімізації функції помилки. Для лінійний нейронних мереж прямого поширення це досягається шляхом коригування вагових коефіцієнтів синаптичних зв’язків між нейронами, для згорткових нейронних мереж коригування проводиться таким самим шляхом для останніх повноз’язних рівнів, а для згорткових рівнів можуть коригуватися значення матриць фільтрів, що виконують операція згортки по зображенню.

Для самого останнього рівня нейронної мережу найпростіше здійснити налаштування після кожної ітерації, оскільки істинні значення вже відомі (при кожній ітерації навчання нейронно мережа має інформація про клас зображення, що подається на вхід, відповідно, можна легко трансформувати ці дані в активації нейронів останнього рівня). Однак для нейронів попередній рівнів налаштування не є таким очевидним. Для навчання нейронної мережі використовується алгоритм зворотного поширення помилки, який описаний в першому розділі. Даний алгоритм є першим і основним з прикладної точки зору для навчання багаторівневих нейронних мереж.

В якості методу мінімізації функції помилки застосовується метод градієнтного спуску. Завдяки обчисленню градієнта функції в точці, можна легко визначати напрямок руху функції, за яким її значення буде збільшуватися або зменшуватися. Для мінімізації функції

використовується від'ємне значення градієнту в точці. Графічне представлення процесу мінімізації функції помилки через градієнт зображено на рис. 3.11:

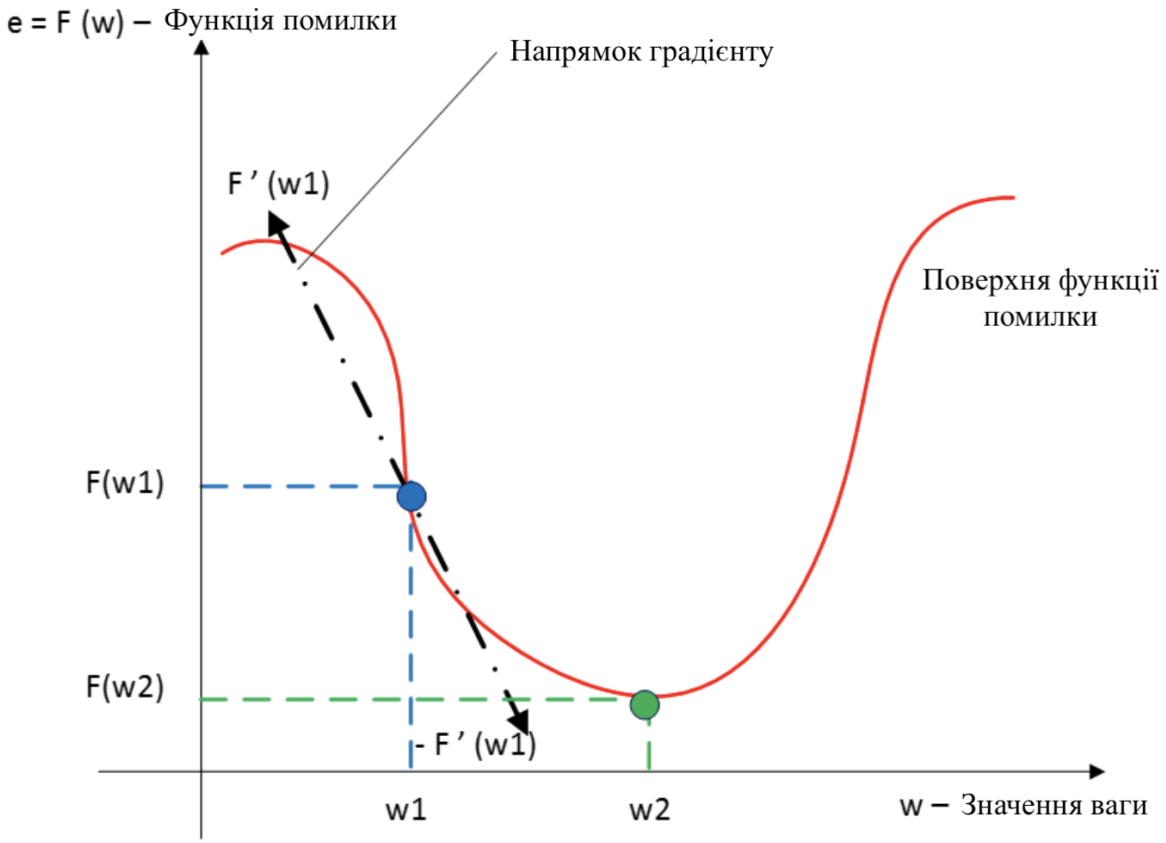


Рисунок 3.11 — Графічне представлення методу градієнтного спуску

Однак, для обчислення значення градієнту функції на першій ітерації навчання, все-одно потрібно мати якісь значення ваг та зміщень нейронів. Їх можна ініціалізувати випадковими значеннями. Проте такий підхід має суттєвий недолік — обчислений в подальшому градієнт може мінімізувати функцію, однак отриманий мінімум буде одним із локальних, не глобальний. Це означає, що після всіх наступних ітерацій функція помилки не зможе досягти свого теоретично мінімального значення, і відсоток хибних розпізнавать неможливо буде зробити меншим певного значення, що буде отримано в кінці навчання нейронної мережі.

Одним із можливих варіантів вирішення цієї проблеми є покращенням методу ініціалізація значень ваг та зміщень на першій ітерації навчання. В процесі навчання створеної нейронної мережі застосовано ітеративний підхід до визначення початкових значень ваг та зміщень. Ці значення ініціалізуються випадковими в кілька ітерацій, після кожної ініціалізації нейронна мережа опрацьовує 30 зображень, таким чином починається процес її навчання, однак він не завершується повністю, тобто лише певний відсоток зображень опрацьовується із випадково ініціалізованими значеннями ваг та зміщень. Коли нейронна мережа виконає обробку цих зображень, то помилка буде мінімізуватися зі змінами ваг та зміщень на кожній ітерації. Їх кінцеві значення зберігаються, і потім процес починається заново, тобто виконується ініціалізація випадковими значеннями, проте ці значенням повинні мати значне відхилення від тих, що були використані на першій ітерації. Знову нейронна мережа опрацьовує 30 зображень і кінцеві значення ваг та зміщень зберігаються.

Після заданої кількості таких ітерацій ваги та зміщення усереднюються, тобто знаходиться середнє арифметичне значення для кожного числа. І лише після цього нейронна мережа починає процес навчання на повному наборі тренувальних даних.

Цей підхід може здаватися не ефективним з точки зору обчислювальних ресурсів та додаткових витрат часу ще до початку навчання нейронної мережі, однак суть алгоритму полягає в тому, щоб градієнт з більшою вірогідністю наближував функцію саме до глобального мінімуму, а не використовував локальний мінімум в якості орієнтиру, до якого треба прямувати. Наприклад, при першій ітерації значення ваг та зміщень ініціалізовані випадковими числами, і градієнт функції помилки починає їх змінювати з метою зменшими значення цієї ж функції, однак він прямує до локального мінімуму. При наступній

ітерації ваги та зміщення ініціалізується теж випадковими числами, але більш вдало, таким чином, що градієнт функції помилки прямує до глобального мінімуму. Усереднені значення ваг та зміщень після того, як перша та друга ітерації опрацюють 30 зображень, зможуть спрямувати функцію помилки саме до глобального мінімуму, таким чином точність розпізнавання буде максимально високою в межах структури нейронної мережі та набору тренувальних даних. Алгоритм навчання зображений на рис. 3.12. Повний алгоритм навчання наведений у Додатку А.

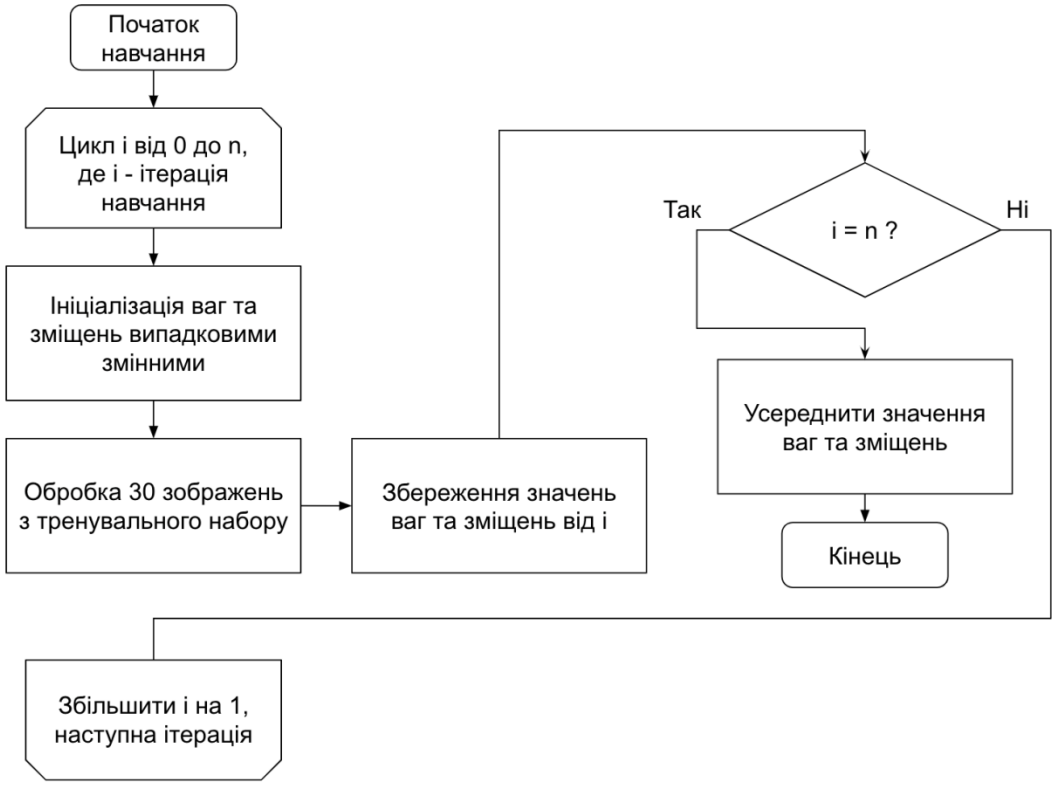


Рисунок 3.12 — Ітеративний алгоритм ініціалізації значень ваг та зміщень

Окрім ітеративного алгоритму ініціалізації значень, подальший процес навчання виконується за невеликими групами зображень з набору тренувальних даних (по 40 зображень в кожній). В кожній групі містяться зображення з різних класів, завдяки чому нейронна мережа буде більш стійкою до помилки при роботі з реальними даними, оскільки клас

вхідного зображення не буде відомий заздалегідь і, по суті, може бути будь-яким. Після опрацювання нейронною мережею кожної вибірки, отримані значення ваг та зміщень усереднюються між всіма групами. Демонстрація алгоритму навчання при розподіленні тренувальних даних на вибірки зображена на рис. 3.13:

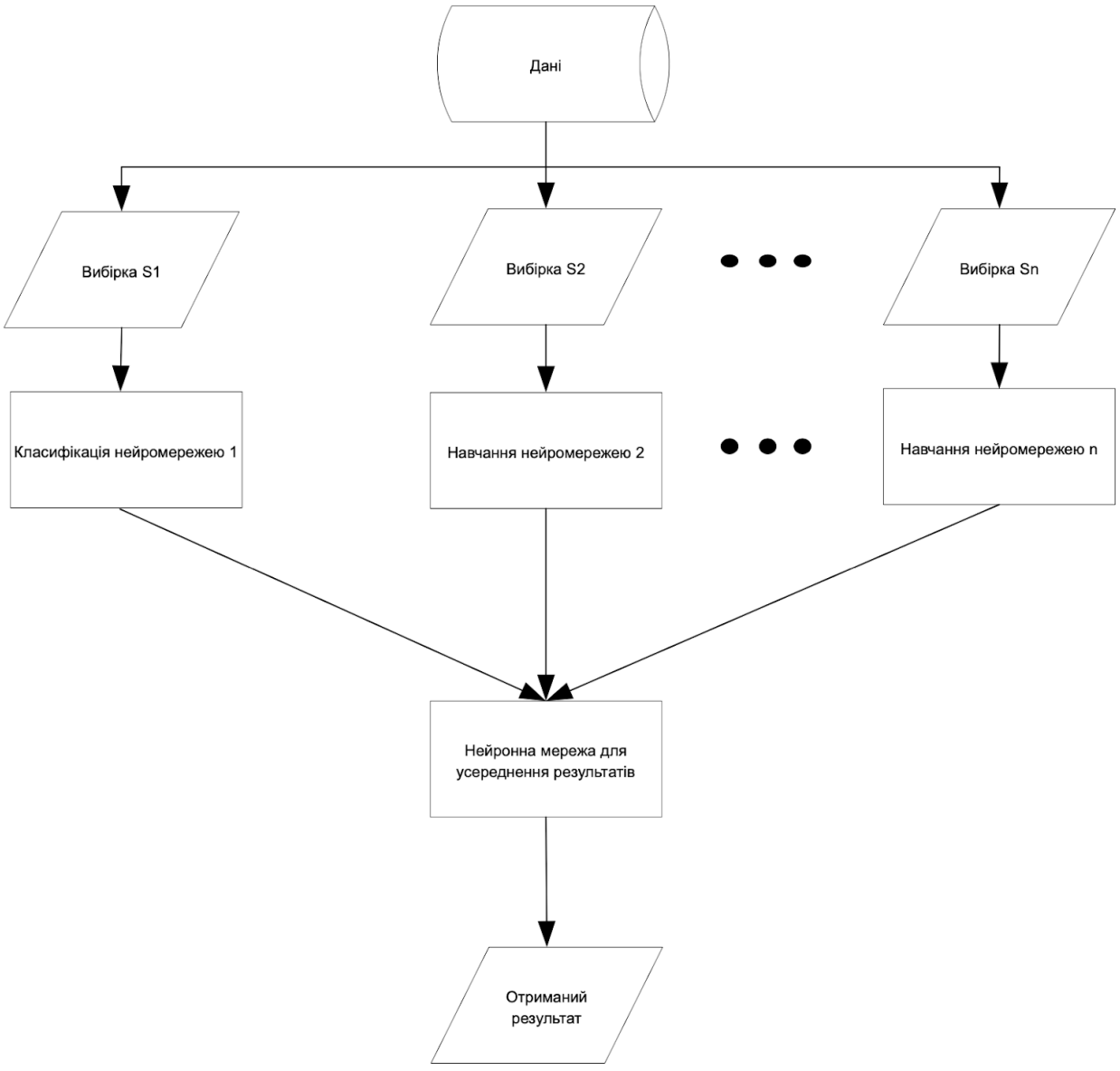


Рисунок 3.13 — Демонстрація алгоритму навчання при розподіленні тренувальних даних на вибірки

3.5 Графічний інтерфейс програми

Графічний інтерфейс розроблений на базі бібліотеки PyQt. Головне вікно містить прості елементи управління для тестування системи шляхом розпізнавання графічних образів на обраному користувачем зображенні. Перед початком роботи з програмою необхідно завантажити значення елементів матриць фільтрів, що виконують операцію згортки по зображенню, а також завантажити ваги нейронів для повнозв'язних шарів нейронної мережі. Ці значення завантажуються з єдиного файлу, до якого записуються всі дані на етапі навчання нейронної мережі. Розширення файлу — ".h5", це один із варіантів формату HDF, що означає ієрархічний формат даних. Цей формат був створений для зберігання великої кількості цифрової інформації. Даний формат дуже часто використовується в машинному навчанні, оскільки кількість параметрів нейронної мережі може досягати кількох десятків тисяч (ваги нейронів, зміщення, елементи матриць фільтрів для згорткових нейронних мереж тощо). Вміст файлів HDF організовано подібно до ієрархічної файлової системи, і для доступу до окремих записів застосовуються шляхи, схожі з POSIX-синтаксисом (наприклад, "home/directory/record"). Метадані зберігаються у вигляді іменованих атрибутів.

Для завантаження конфігураційного файлу з розширенням ".h5" на головному екрані розміщена відповідна кнопка, за якою користувач може вибрати файл через локальний провідник на комп'ютері. Аналогічний функціонал для вибору графічного зображення. До програми можна завантажити зображення із розширенням PNG та JPG/JPEG. Після завантаження, зображення з'являється на центральній частині головного вікна.

Якщо завантажено конфігураційний файл і зображення, то можна зробити класифікацію графічного образу через відповідну кнопку в

нижній частині екрану. Якщо зображення було успішно опрацьоване нейронною мережею, то результат класифікації буде виведений в нижній частині вікна. Система виводить клас зображення, тобто приналежність графічного образу до тієї чи іншої групи, а також точність розпізнавання, яка виражається вірогідністю приналежності образу до цієї групи. Вірогідність розпізнавання є пропорційною до найбільшого значення активації нейрону на останньому повнозв'язному рівні нейронної мережі. Кількість нейронів останнього рівня дорівнює кількості класів графічних образів, що можуть розпізнаватися системою. Нейронна мережа має активувати один із нейронів цього рівня найбільшим значенням. Графічний інтерфейс вікна після успішної класифікації зображення представлений на рис. 3.14:

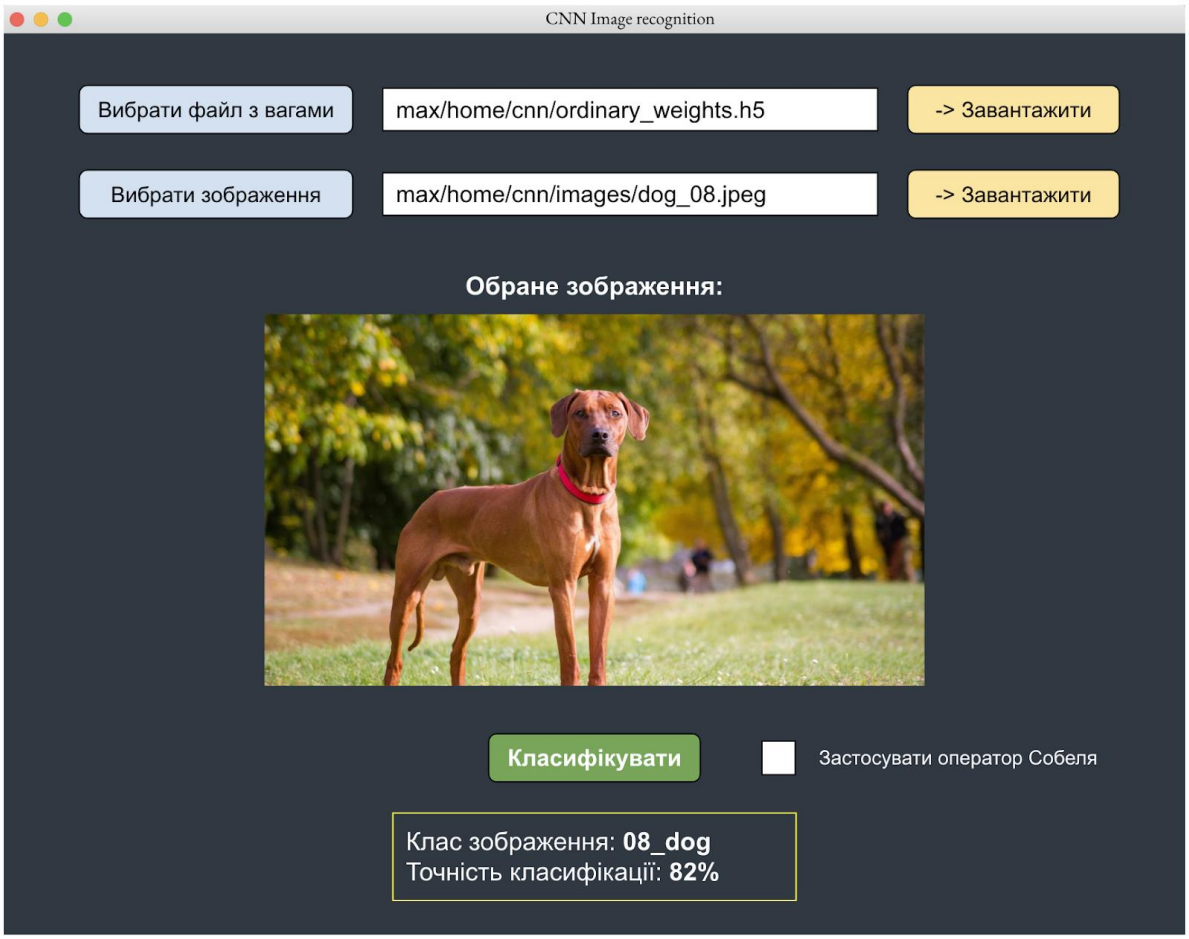


Рисунок 3.14 — Головне вікно програми після успішного розпізнавання

Для демонстрації застосування оператору Собеля, на рис. 3.15 обрано інше зображення. Система успішно розпізнає графічний образ автомобіля і правильно класифікує зображення, оскільки дана група образів була використана в процесі навчання нейронної мережі.

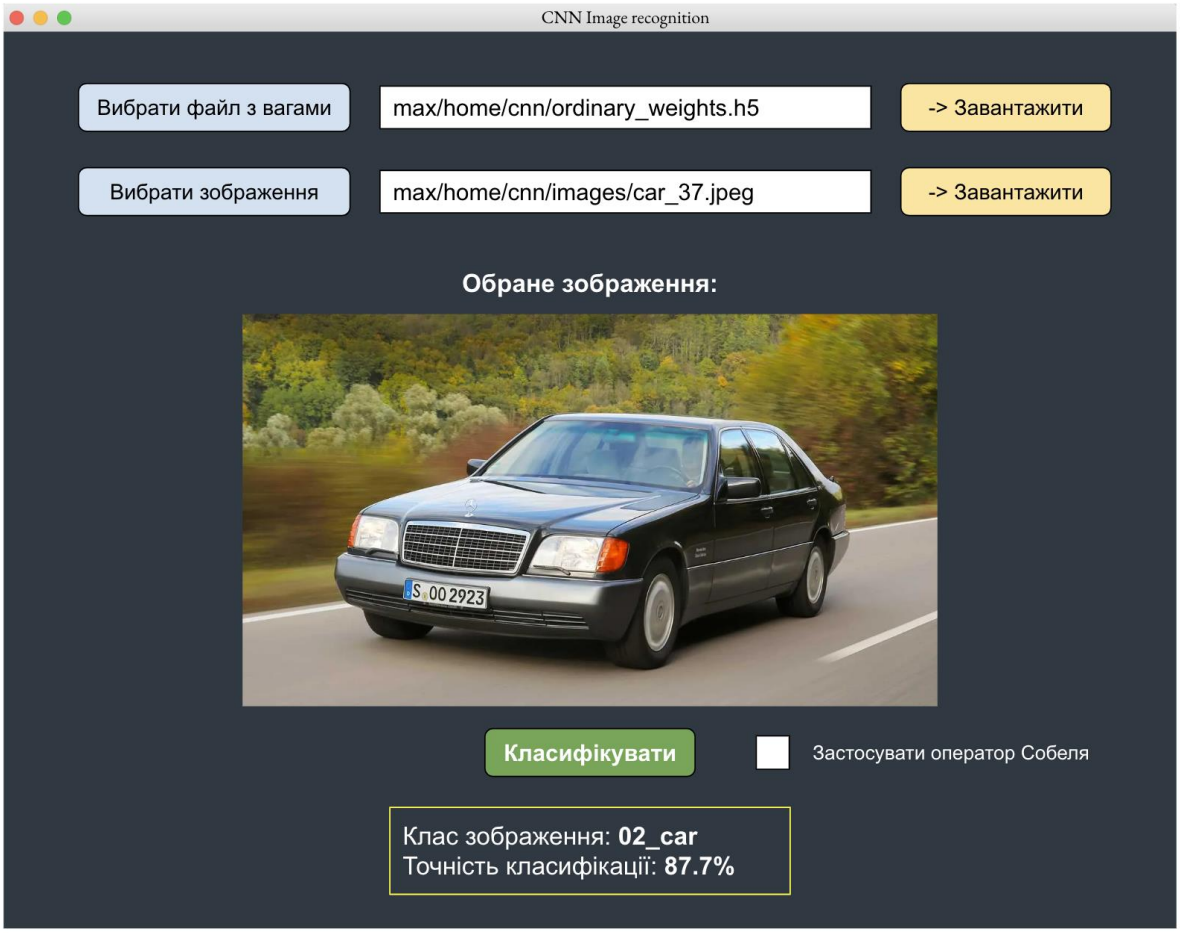


Рисунок 3.15 — Демонстрація розпізнавання програмою графічного образу автомобіля

Як видно з попереднього рисунку, точність розпізнавання становить 87.7%. Для підвищення точності, на головному вікні програми знаходиться CheckBox, який дозволяє встановити налаштування для наступної обробки зображення із застосуванням оператору Собеля. При наступній класифікації, зображення спочатку буде опрацьоване відповідними згортками оператору Собеля, і лише потім отримана матриця пікселів буде подаватися на вхід нейронної мережі. Приклад

застосування оператору Собеля до зображення автомобіля, що було використано вище, наведено на рис. 3.16:

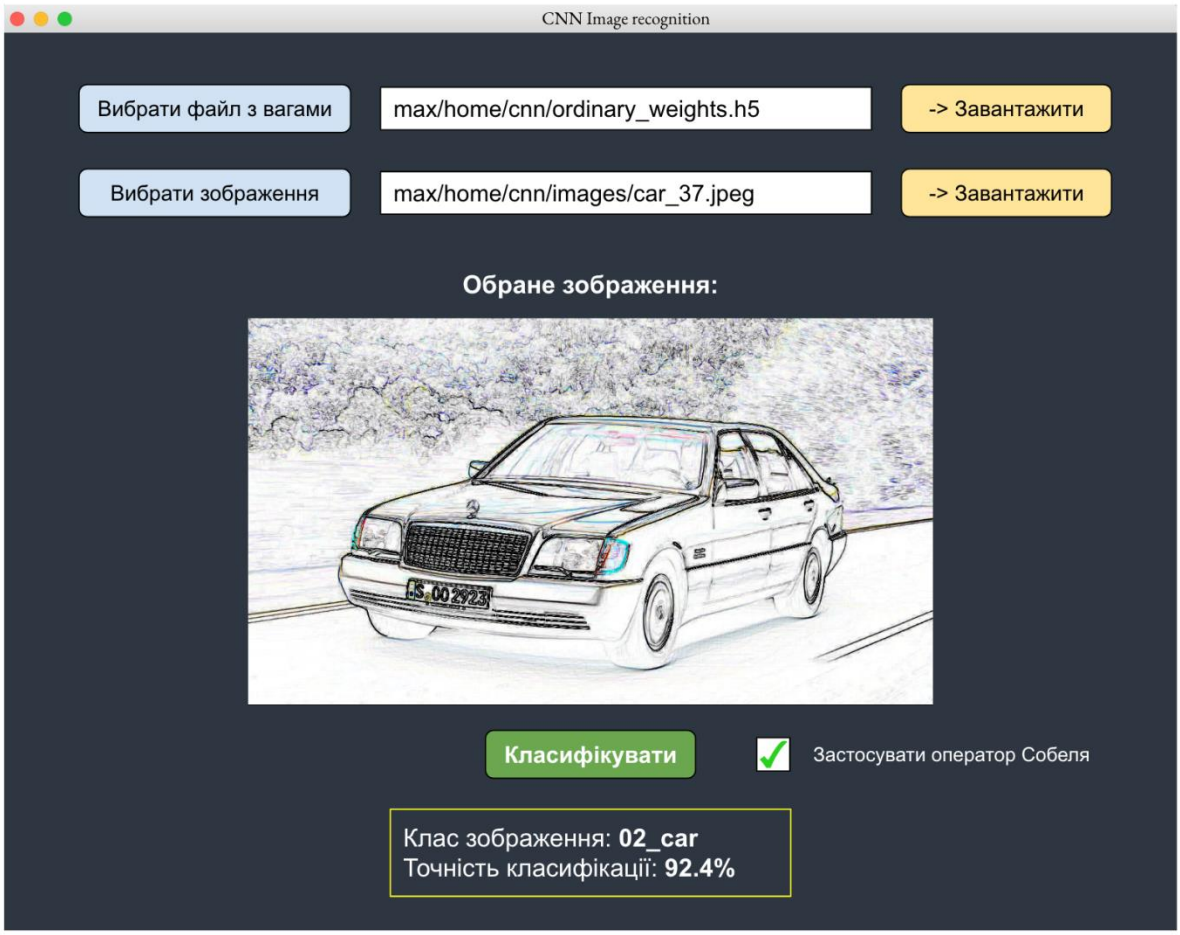


Рисунок 3.16 — Демонстрація розпізнавання програмою графічного образу автомобіля із застосуванням оператору Собеля

На рис. 3.16 можна побачити, що точність класифікації становить 92,4%, на рис. 3.15 точність становила 87,7%. З цього можна зробити висновок, що для даної групи графічних образів застосування оператору Собеля є ефективним методом підвищення точності розпізнавання.

Конкретно для даного зображення, точність передбачення класу графічних образів збільшилася на 4,7%. Варто зауважити, що цей показник може збільшуватися або зменшуватися в залежності від класу зображень. Для деяких графічних образів виділення контурів є суттєвим, в той час як для інших застосування оператору Собеля може не дати

позитивного ефекту. Висновки про ефективність попередньої обробки зображення встановлюються експериментальним шляхом в процесі тренування згорткової нейронної мережі.

3.6 Оцінка точності розпізнавання образів системою

Точність розпізнавання визначається відношенням правильно класифікованих зображень до всього набору тренувальних даних в процесі навчання. З кожною наступною ітерацію цей показник збільшується, в сам процес навчання продовжується до тих пір, поки помилка розпізнавання не досягне мінімального значення. В розробленій нейронній мережі навчання завершувалось, коли помилка розпізнавання досягала значення менше ніж 10%. Скріншот лістингу програмного коду, що ініціалізує процес навчання, наведено на рис. 3.17:

```
import mnist
import numpy as np
from conv import Conv3x3
from maxpool import MaxPool2
from softmax import Softmax

train_images = mnist.train_images()[:300]
train_labels = mnist.train_labels()[:300]
test_images = mnist.test_images()[:300]
test_labels = mnist.test_labels()[:300]

conv = Conv3x3(8)                  # 28x28x1 -> 26x26x8
pool = MaxPool2()                  # 26x26x8 -> 13x13x8
softmax = Softmax(13 * 13 * 8, 10) # 13x13x8 -> 10

def forward(image, label):
  out = conv.forward((image / 255) - 0.5)
  out = pool.forward(out)
  out = softmax.forward(out)

  loss = -np.log(out[label])
  acc = 1 if np.argmax(out) == label else 0

  return out, loss, acc
```

Рисунок 3.17 — Ініціалізація процесу навчання нейронної мережі

Після опрацювання кожної групи зображень із набору тренувальних даних, було проведено обчислення втрат та точності розпізнавання. Сам процес навчання виконувався в 3 епохи (рис. 3.18).

```
for epoch in range(3):
  print('--- Epoch %d ---' % (epoch + 1))

  # Перемішування вхідних даних
  permutation = np.random.permutation(len(train_images))
  train_images = train_images[permutation]
  train_labels = train_labels[permutation]

  # Поаток навчання епохи
  loss = 0
  num_correct = 0
  for i, (im, label) in enumerate(zip(train_images, train_labels)):
    if i > 0 and i % 100 == 99:
      print(
        '[Step %d] Past 100 steps: Average Loss %.3f | Accuracy: %d%%' %
        (i + 1, loss / 100, num_correct)
      )
      loss = 0
      num_correct = 0

    l, acc = train(im, label)
    loss += l
    num_correct += acc

```

Рисунок 3.18 — Виконання однієї епохи навчання

Одна епоха навчання — це коли набір тренувальних даних проходить через нейронну мережу в прямому і зворотному напрямку лише 1 раз. У випадку застосування великих наборів тренувальних даних, їх розділяються на групи, які називаються “batches”. Варто пам’ятати, що в будь-якому випадку застосовується обмежений набір даних для тренування, і недостатньо виконати лише одну епоху для повного навчання нейронної мережі. Може здаватися, що збільшення кількості епох призводить до збільшення точності розпізнавання, однак все залежить саме від цієї кількості. При великій кількості епох нейронна мережа “перенавчається” в негативному сенсі. Зі збільшенням кількості

епох, ваги та зміщення нейронної мережі змінюються все більшу кількість разів. Однак, не існує єдиного правила для визначення оптимальної кількості епох навчання. Для різних наборів тренувальних даних цей показник буде відмінним. Але очевидним є факт, що кількість епох навчання залежить від різноманітності даних. Якщо набір тренувальних даних містить велику кількість класів графічних образів, то в такому випадку кількість епох навчання має бути теж відносно великою.

Консольний вивід точності та значень функції втрат кожної епохи навчання зображено на рис. 3.19:

```
MNIST CNN initialized!
--- Epoch 1 ---
[Step 100] Past 100 steps: Average Loss 2.254 | Accuracy: 18%
[Step 200] Past 100 steps: Average Loss 2.167 | Accuracy: 30%
[Step 300] Past 100 steps: Average Loss 1.676 | Accuracy: 52%
[Step 400] Past 100 steps: Average Loss 1.212 | Accuracy: 63%
[Step 500] Past 100 steps: Average Loss 0.949 | Accuracy: 72%
[Step 600] Past 100 steps: Average Loss 0.848 | Accuracy: 74%
[Step 700] Past 100 steps: Average Loss 0.954 | Accuracy: 68%
[Step 800] Past 100 steps: Average Loss 0.671 | Accuracy: 81%
[Step 900] Past 100 steps: Average Loss 0.923 | Accuracy: 67%
[Step 1000] Past 100 steps: Average Loss 0.571 | Accuracy: 83%
--- Epoch 2 ---
[Step 100] Past 100 steps: Average Loss 0.447 | Accuracy: 89%
[Step 200] Past 100 steps: Average Loss 0.401 | Accuracy: 86%
[Step 300] Past 100 steps: Average Loss 0.608 | Accuracy: 81%
[Step 400] Past 100 steps: Average Loss 0.511 | Accuracy: 83%
[Step 500] Past 100 steps: Average Loss 0.584 | Accuracy: 89%
[Step 600] Past 100 steps: Average Loss 0.782 | Accuracy: 72%
[Step 700] Past 100 steps: Average Loss 0.397 | Accuracy: 84%
[Step 800] Past 100 steps: Average Loss 0.560 | Accuracy: 80%
[Step 900] Past 100 steps: Average Loss 0.356 | Accuracy: 92%
[Step 1000] Past 100 steps: Average Loss 0.576 | Accuracy: 85%
--- Epoch 3 ---
[Step 100] Past 100 steps: Average Loss 0.367 | Accuracy: 89%
[Step 200] Past 100 steps: Average Loss 0.370 | Accuracy: 89%
[Step 300] Past 100 steps: Average Loss 0.464 | Accuracy: 84%
[Step 400] Past 100 steps: Average Loss 0.254 | Accuracy: 95%
[Step 500] Past 100 steps: Average Loss 0.366 | Accuracy: 89%
[Step 600] Past 100 steps: Average Loss 0.493 | Accuracy: 89%
[Step 700] Past 100 steps: Average Loss 0.390 | Accuracy: 91%
[Step 800] Past 100 steps: Average Loss 0.459 | Accuracy: 87%
[Step 900] Past 100 steps: Average Loss 0.316 | Accuracy: 92%
[Step 1000] Past 100 steps: Average Loss 0.460 | Accuracy: 87%

--- Testing the CNN ---
Test Loss: 0.5979384893783474
Test Accuracy: 0.78
```

Рисунок 3.19 — Консольний вивід точності та значень функції втрат

Для візуалізація процесу навчання та його ефективності використано інструмент TensorBoard. Точність розпізнавання зручно представити у вигляді звичайного графіку, на якому по осі абсцис будуть відкладені ітерації навчання, а по осі ординат — точність на кожній ітерації. Графічне представлення точності розпізнавання образів нейронною мережею в процесі навчання представлений на рис. 3.20. Синій графік демонструє точність розпізнавання при обробці зображень із тренувального набору даних, помаранчевий — тестування нейронної мережі на зображеннях, що не використовувалися в процесі навчання.

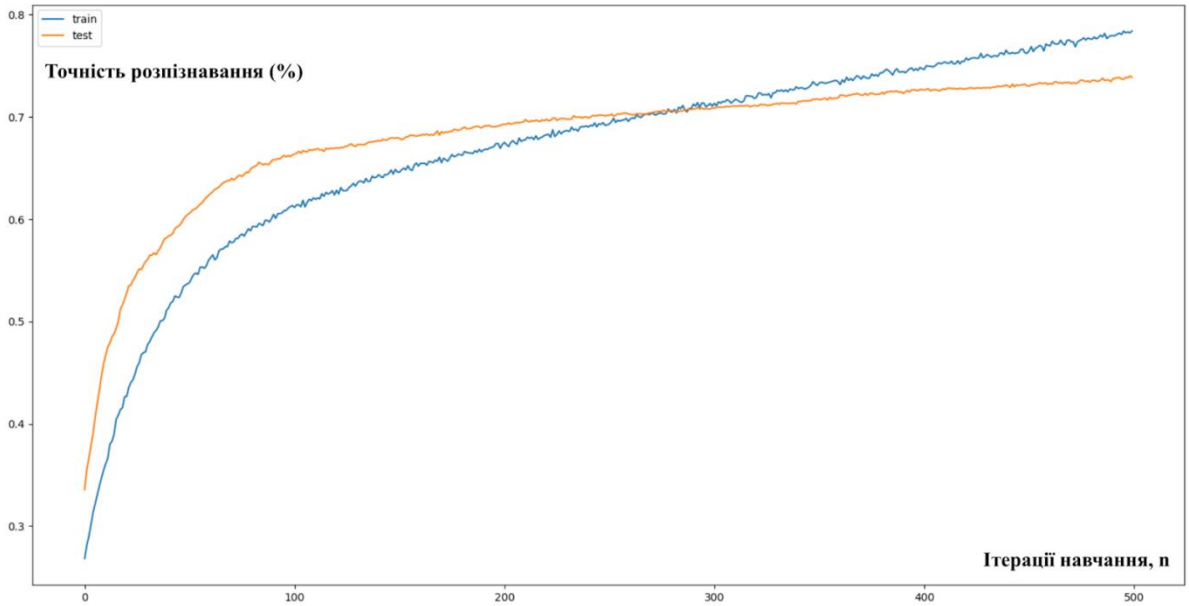


Рисунок 3.20 — Точність розпізнавання зображень згортковою нейронною мережею в процесі навчання

На основі візуалізації навчання, можна зробити висновок про досить ефективне розпізнавання графічних образів нейронною мережею. Помилка розпізнавання на перших ітерація була дуже великою, і система могда прогнозувати правильну приналежність образу до того чи іншого класу зображень з вірогідністю не більше 40%. На останніх ітерація помилка розпізнавання становила близько 10%, відповідно, вірогідність

правильного передбачення досягла близько 90%. Звісно ж, при використанні зображень поза набором тренувальних даних, цей показник є дещо гіршим, про що свідчить розташування помаранчевого графіку нижче синього на рис. 3.20. Однак, це є звичайною практикою при тестування нейронних мереж.

Також було проведено дослідження впливу застосування оператору Собеля при попередній обробці зображення на точність розпізнавання. Результати дослідження наведено на рис. 3.21.

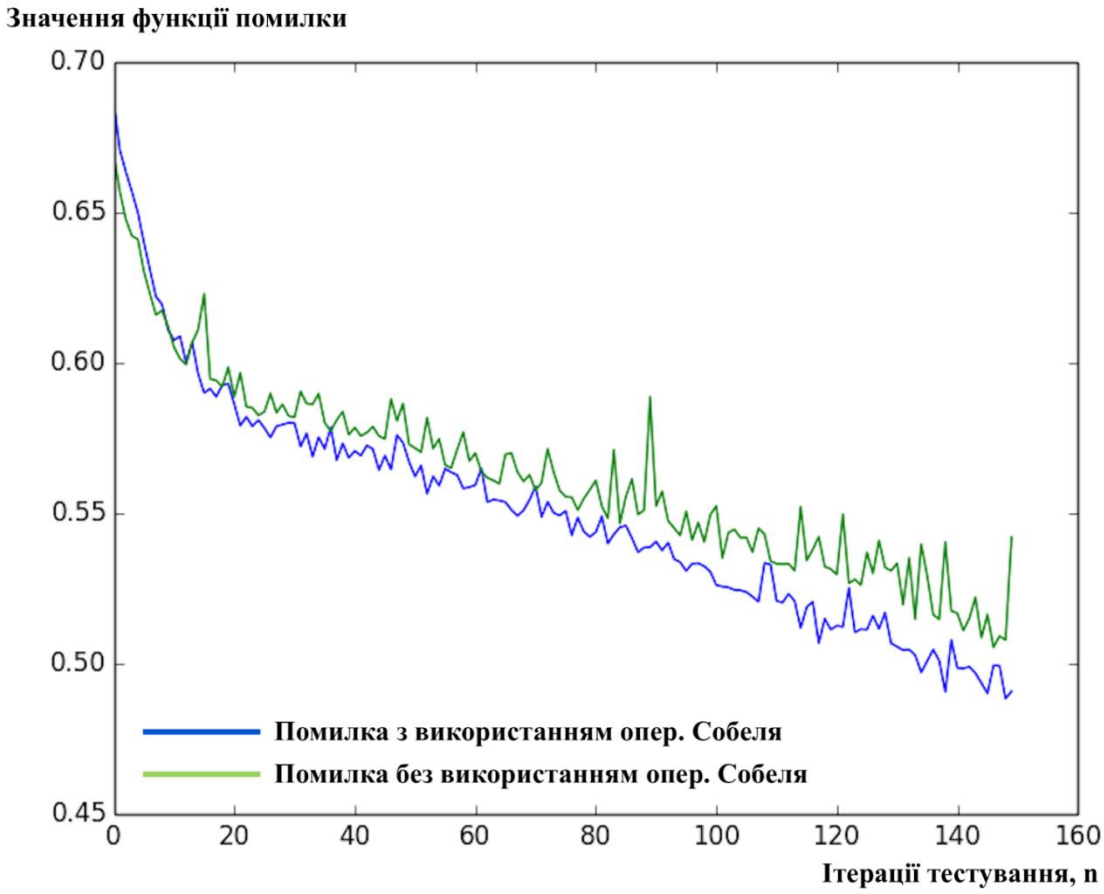


Рисунок 3.21 — Дослідження впливу застосування оператору Собеля на точність розпізнавання

Як видно із попередньої візуалізації, застосування попередньої обробки зображення може значно підвищити точність розпізнавання і, відповідно, знизити значення функції помилки. При застосування

оператору Собеля до вхідних зображень значення помилки зменшується в середньому на 5-7%, що і демонструє попередній графік.

3.7 Висновки до розділу

Точність розпізнавання графічних образів та ефективність використання обчислювальних ресурсів комп’ютера в процесі навчання залежить від багатьох факторів, серед яких головними є: архітектура згорткової нейронної мережі, кількості шарів, алгоритм навчання, застосування методів попередньої обробки вхідних зображень.

Запропонований алгоритм навчання нейронної мережі, який полягає в ітеративній ініціалізації початкових значень ваг та зміщень нейронів, має досить високу точність навчання, про що демонструють візуалізації функції помилки.

Також можна легко підвищити точність через використання тих чи чи інших методів попередньої обробки зображень. В розробленій системі є можливість застосувати оператор Собеля до вхідного зображення, який допомагає чітко виділити контури графічних образів на матриці пікселів. Цей метод збільшив точність розпізнавання в середньому на 5%.

Розроблений графічний інтерфейс програми дозволяє виконати тестування системи шляхом розпізнавання графічного образу на вхідному зображенні, яке обирається користувачем.

ВИСНОВКИ

Застосування згорткових нейронних мереж дає можливість технології комп’ютерного зору перейти на якісно новий рівень. Дана нейромережева архітектура вирішує прикладні задачі, починаючи від ідентифікації особи за обличчям і закінчуючи прогнозуванням складних медичних діагнозів.

В магістерській дисертації розроблено інтелектуальну систему розпізнавання графічних образів, яка дозволяє виконувати класифікацію зображень. Проаналізовано можливі варіанти підвищення точності розпізнавання, та запропоновано алгоритм ітеративної ініціалізації параметрів нейронної мережі на перших етапах навчання. Також в якості методу зменшення помилки розпізнавань, в системі реалізована попередня обробка зображень через застосування оператору Собеля, який дає можливість чітко виділити контури графічного об’єкту на вхідному зображенні.

Запропоновані в магістерській дисертації методи підвищення точності розпізнавання графічних образів є досить ефективними, про що свідчать результати досліджень в процесі навчання нейронної мережі. Точність розпізнавання графічних образів системою досягає 90%.

В порівнянні зі здібностями звичайних нейронних мереж прямого поширення, згорткові нейронні мережі більш раціонально використовують обчислювальні ресурси комп’ютера. Нейронна мережа прямого поширення займається побудовою абстрактних векторів характеристик зображення, для цього потрібна набагато вища обчислювальна потужність і дуже велика кількість навчальних даних. Архітектурі згорткових нейронних мереж набагато краще вдається розпізнавати графічні образи на зображеннях, оскількт, на відміну від багаторівневого персептрона, враховується двовимірна топологія

зображення. Завдяки цьому досягається стійкість нейронної мережі до зміщень, змінам масштабу та кута повороту графічного об’єкту на вхідному зображенні.

Незважаючи на очевидні переваги та велику кількість досліджень, питання надійності згорткових нейронних мереж та точності розпізнавання графічних образів є досить актуальним при побудові сучасних інформаційних систем. Згорткові нейронні мережі лежать в основі багатьох програм, сучасних пристроїв та інформаційних систем. Виправлення існуючих недоліків різних архітектур згорткових нейронних мереж неодмінно призведе до розширення їх застосування в інших сферах життя людини.

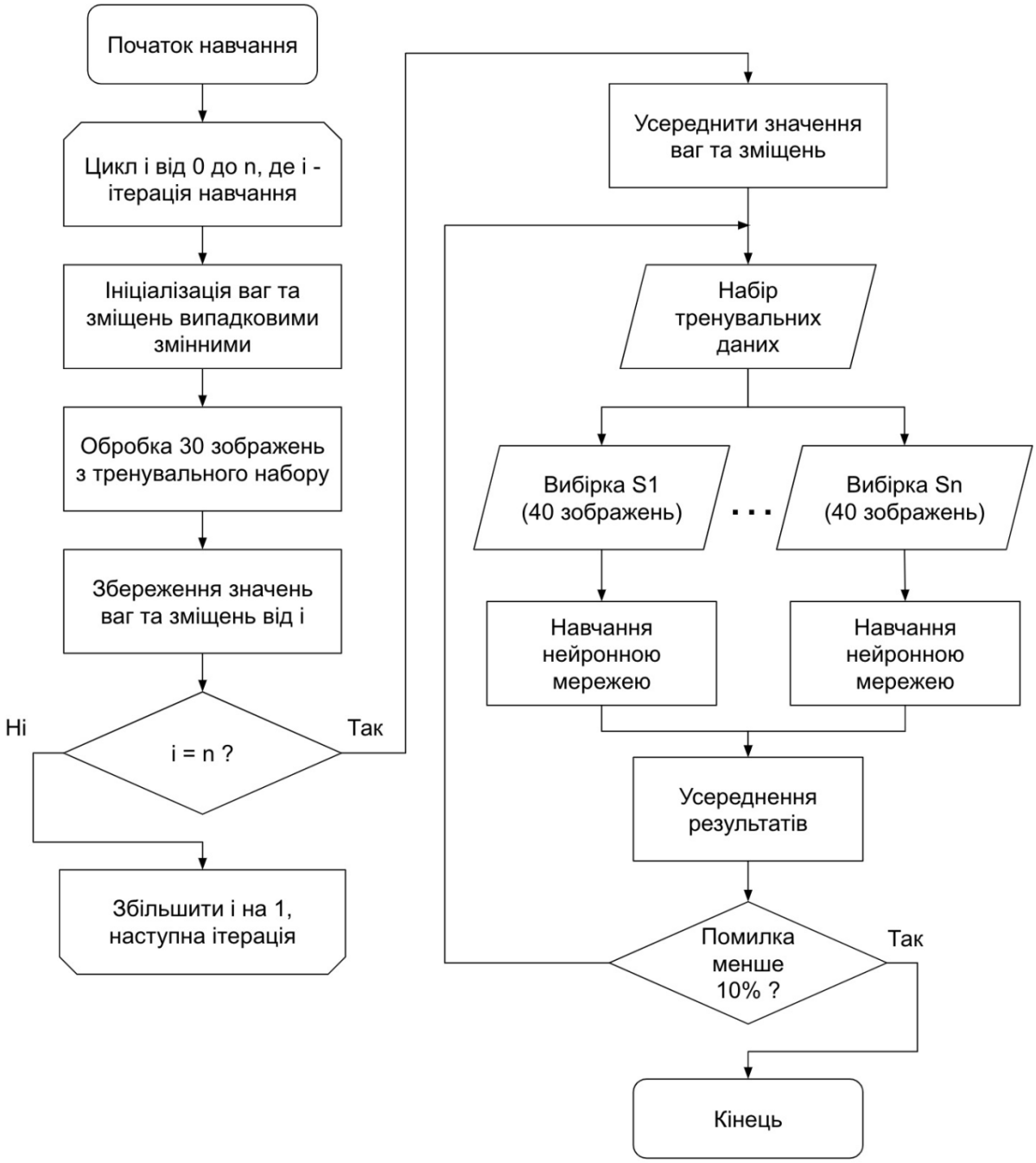
ПЕРЕЛІК ПОСИЛАНЬ

1. Зображення демонстрації роботи комп'ютерного зору [Електронний ресурс] – Режим доступудо ресурсу: https://medium.com/mlearning-ai/computer-vision-fundamentals-and-opencv-overview-9a30fe94f0ce
2. Каллан, Р. Нейронні мережі: Короткий довідник / Р. Каллан. - М .: Вільямс І.Д., 2017. - 288 с.
3. Deep Neural Networks for Image Recognition and Classification Problems [Електронний ресурс] – Режим доступу до ресурсу: http://itcm.comp-sc.if.ua/2017/Sineglazov.pdf.
4. Ніколенко С. Глибоке навчання / С. Ніколенко, А. Кадурін, Е. Архангельська // СПб .: Питер, 2018. - 480 с.
5. Зображення візуалізації грідієнтного спуску [Електронний ресурс] – Режим доступу до ресурсу: https://www.machinelearningmastery.ru/ gradient-descent-algorithm-and-its-variants-10f652806a3/
6. Редько В.Г. Еволюція, нейронні мережі, інтелект: Моделі і концепції еволюційної кібернетики / В.Г. Редько // М .: Ленанд, 2019. - 224 с.
7. Korchenko A. Neural network models, methods and tools for assessing the security parameters of Internet-oriented information systems: monograph / A.Korchenko, I. Tereykovsky, N. Karpinsky, S. Tynimbaev. K.: TOV "Our Format".2016. - 275 p.
8. Марр Д. Зор: Інформаційний підхід до вивчення подання та обробки зорових образів. – М.: Радио и связь, 1987.
9. Theodoridis S., Koutroumbas K., Pattern Recognition. New York: Elsevier Science, 2008 [Електронний ресурс]. Режим доступу: https://books.google.com/books?id=QgD-3Tcj8DkC

10. ImageNet Large Scale Visual Recognition Challenge [Електронний ресурс] – Режим доступу до ресурсу: https://www.image-net.org/challenges/LSVRC/
11. Jerry Wei AlexNet: The Architecture that Challenged CNNs [Електронний ресурс] – Режим доступу до ресурсу: https://towardsdatascience.com/alexnet-the-architecture-that-challenged-cnns-e406d5297951
12. Згорткова нейронна мережа [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Згорткова_нейронна_мережа.
13. Szegedy C. Deep neural networks for object detection / C. Szegedy, A. Toshev, D. Erhan. // NIPS. – 2013.
14. Krizhevsky A. ImageNet classification with deep convolutional neural networks / A. Krizhevsky, I. Sutskever, G. E. Hinton. // Advances in Neural Information Processing Systems (NIPS). – 2012. – No25. – С. 1097–1105.
15. Vedaldi A. Convolutional neural networks for matlab / A. Vedaldi, K. Lenc. // Proceedings of the 23rd Annual ACM Conference on Multimedia Conference. – 2015.
16. Зображення операції згортки [Електронний ресурс] – Режим доступу до ресурсу: https://neurohive.io/ru/osnovy-data-science/glubokaya-svertochnaja-nejronnaja-set/
17. Y. Le Cun and Yoshua Bengio. Convolutional networks for images, speech, and time series. In Michael A.Arbib, editor, The Handbook of Brain Theory and Neural Networks, pages 255–258. MITPress, Cambridge, Massachusetts, 1995.
18. Szegedy C. Going deeper with convolutions / C. Szegedy, W. Liu, Y. Jia. // CVPR. – 2015.
19. Зображення шару згортки [Електронний ресурс] – Режим доступу до ресурсу: https://neurohive.io/ru/osnovy-data-science/glubokaya-svertochnaja-nejronnaja-set/

20. Zeiler M. Visualizing and Understanding Convolutional Networks / M. Zeiler, R. Fergus. // ICCV. – 2015.
21. Neurocomputer architecture based on spiking neural network and its optoelectronic implementation / Oleh K. Kolesnytskyj; Vladislav V. Kutsman; Krzysztof Skorupski; Mukaddas Arshidinova, Proc. SPIE 11176, Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments 2019, 1117609 (6 November 2019); doi: 10.1117 / 12.2536607
22. Eijaz Allibhai Building a Convolutional Neural Network (CNN) in Keras [Електронний ресурс] – Режим доступу до ресурсу: https://towardsdatascience.com/building-a-convolutional-neural-network-cnn-in-keras-329fbbadc5f5
23. Nielsen MA Neural Networks and Deep Learning [Електронний ресурс]. Режим доступу: http://www.neuralnetworksanddeeplearning.com
24. What is TensorFlow? The machine learning library explained [Електронний ресурс] – Режим доступу до ресурсу: https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html
25. Методичні вказівки до виконання магістерських дисертацій освітньо- наукової програми «Інтегровані інформаційні системи» для студентів спеціальності 126 «Інформаційні системи та технології» /Уклад. Н.Б. Репнікова.- К.: НТУУ «КПІ ім.Ігоря Сікорського», 2020.- 46 с.
26. ДСТУ 3008-95. Державний стандарт України. Документація. Звіти усфері науки і техніки. Структура і правила оформлення. – Введ. 23.02.95 - К.:Держстандарт України, 1995. – 37 с.

ДОДАТОК А Блок-схема алгоритму навчання нейронної мережі



ДОДАТОК Б Наукова стаття

УДК 004.043
Ткаченко М.С., Сокульський О.Є.
НТУУ "КПІ ім. Ігоря Сікорського"
ФІОТ, Кафедра ІСТ

ЗАСТОСУВАННЯ R-CNN ПРИ АВТОМАТИЧНОМУ ПОЗИЦІОНУВАННІ ОБ'ЄКТІВ ЧЕРЕЗ НЕЙРОМЕРЕЖЕВИЙ АНАЛІЗ ГРАФІЧНИХ ДАНИХ

Проведено аналіз сучасних підходів, які можуть бути використані при розробці методів оптичного моніторингу, що здійснюється на основі графічних даних систем відеоспостереження і мобільних електронних пристроїв, а також нейромережевого аналізу. Вказано на пріоритет у застосуванні при розробці нейромережевих алгоритмів згорткової нейромережевої архітектури «Fast R-CNN», яка включає у себе машину опорних векторів і модель регресії. Запропонована у рамках дослідження схема характеризується мінімальною ресурсомісткістю процесу обробки графічних даних, що досягається шляхом вирішення проблеми вибіркового пошуку великого об'єму областей інтересу для кожного зображення та оцінки вектора ознак для кожної області. Показано, що відповідна архітектура може бути ефективно використана для завдання розпізнавання візуальних об'єктів, їх класифікації та позиціонування відповідно вирішення задачі геолокації. Розглянуто методики оптимізації зазначеної системи відповідно цільових показників точності машинного аналізу за умов обмеження навантаження на обчислювальний ресурс.

Ключові слова: *згорткові нейронні мережі, машинний аналіз, позиціонування, область інтересу, машина опорних векторів, модель регресії, функція втрат.*

Вступ

Сучасна система суспільних взаємин, що є типовою для розвинених та розвиткових країн характеризується високою мобільністю громадян у рамках професійної та рекреаційної діяльності. Це суттєво розширює

можливості для організації гнучкої та масштабованої економічної моделі міста, а також інфраструктурного і логістичного проектування, але водночас ускладнює контроль і запобігання потенційних правопорушень та координацію громадян, особливо у рамках втілення парадигми інклюзивності (доступності громадських і приватних об’єктів для людей з вадами зору, нейробіологічними особливостями, тощо). Очевидно, що на загальному рівні вказана проблема може бути вирішена через застосування інформаційних технологій. На сьогоднішній день ефективним підходом у зазначеній галузі є використання у інфраструктурі оптичного моніторингу, що базується на масивах даних систем відеоспостереження та мобільних електронних засобів громадян (смартфони, планшети, окуляри доповненої реальності), а також підсистем машинного аналізу на основі нейромережевих алгоритмів. Це надає можливість ефективно відновити графічні дані отримані мінітюаризованими системами фотореєстрації [1, 2], виділити візуальні об’єкти, провести їх класифікацію, побудувати тривимірну сцену зображення та визначити положення окремого візуального об’єкту у рамках тривимірної сцени [3, 4]. Однією з ключових проблем застосування нейромережевих архітектур є високе навантаження на обчислювальний ресурс за умов вирішення задач високої складності, що є неприйнятним при потребі обробки великих масивів даних у режимі реального часу. При цьому вказуються переваги вирішення задачі сегментації та аналізу графічних даних через застосування згорткової нейромережевої архітектури (Convolutional Neural Network, CNN), що функціонує подібно до зорової кори мозку людини [5, 6], зокрема зазначається ***актуальність*** впровадження у рамках дослідження CNN на основі областей (Region Based Convolutional Neural Network, R-CNN) для виконання селективного пошуку візуальних об’єктів зображення [6-8].

Аналіз сучасних досліджень і публікацій присвячених проблемам впровадження алгоритмів машинного аналізу масивів графічних даних на основі CNN надав можливість оцінити найбільш актуальні підходи [5, 6], що можуть бути застосовані у даній галузі. Статичні результати експериментальних досліджень по практичній реалізації зазначених алгоритмів для систем обробки графічних і відеоданих вказують на те, що робота моделей типу CNN [5-8] потребує використання надмірного обчислювального ресурсу, і тому у багатьох випадках не може бути застосовано при роботі у режимі реального часу. Тому основна увага була приділена до моделей, що базуються на основі R-CNN, що включають у себе машину опорних векторів (Support Vector Machine) і засоби регресійного аналізу (Regression Model, RM), зокрема «Fast R-CNN» [6, 7] і «Faster R-CNN» [8, 9], «Mask R-CNN» [10, 11] і архітектура YOLO [12, 13], що попри більшу ресурсоємність надає можливість найбільш ефективно виділити візуальні об’єкти та проводити їх класифікацію. Як показали дослідження алгоритми машинного аналізу на базі відповідних нейромережевих архітектур здатні вирішувати широке коло задач з аналізу графічних даних, але при цьому не була розглянута проблема обробки даних систем фотореєстрації з широкого кола систем фотореєстрації, які характеризуються різним рівнем якості на рівні апаратної платформи, так і режиму зйомки, що розглядається як ***невирішена частина загального дослідження.***

Таким чином, **метою дослідження** стала розробка методологія побудови нейромережевих алгоритмів на основі архітектури R-CNN, що включає у себе вирішення завдання машинного аналізу графічних даних з метою позиціонування візуальних об’єктів у просторі міської інфраструктури у режимі реального часу.

1. Адаптація архітектури згорткової нейромережі до задачі позиціонування візуальних об’єктів

Розглянемо базову схему аналізу графічних даних систем відеоспостереження з метою позиціонування методами машинного аналізу на основі нейромережевих алгоритмів (рис. 1).

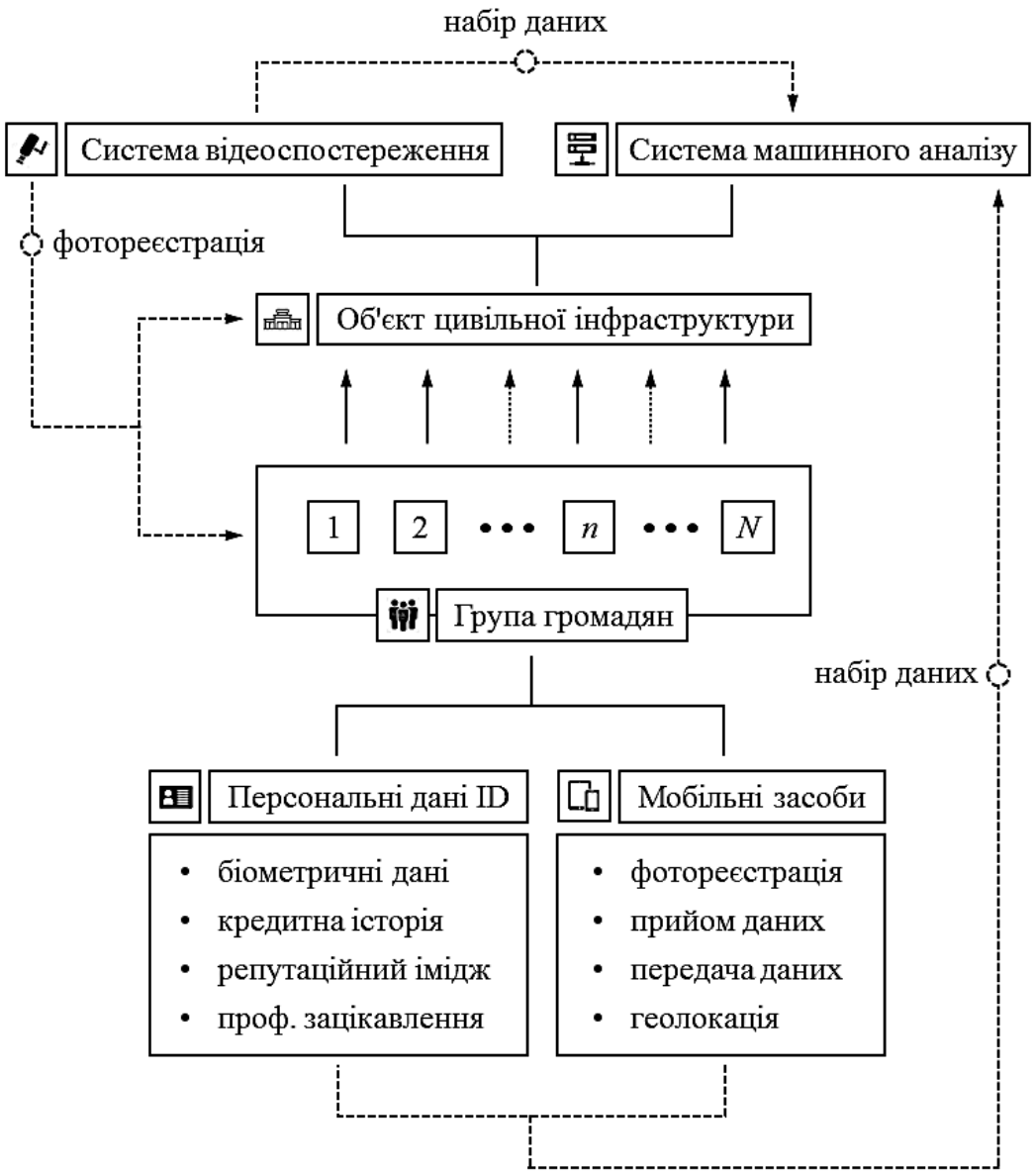


Рисунок 1 — Базова схема аналізу графічних даних систем відеоспостереження з метою позиціонування

Система відеоспостереження проводить фото- та відеореєстрацію громадян, що знаходяться у рамках об’єкту цивільної інфраструктури

(громадський заклад, вокзал, аеропорт, торгівельно-розважальний центр, тощо) і передає масив даних у систему машинного аналізу. У свою чергу, система машинного аналізу виділяє характерні риси присутніх у рамках об’єкту (обличчя, фігура, поведінка) і співставляє з біометричними даними, що знаходяться у відкритому доступі. На основі системи прийому передачі даних мобільних електронних пристроїв громадян система машинного аналізу додатково уточнює вірогідність знаходження відповідної людини у зазначеному закладі через дані геолокації і доступні біометричні дані, а також отримує додатковий блок графічних даних, що підлягає аналізу. На основі персональних даних громадянина (кредитна історія, репутаційний імідж, професійні та інші зацікавлення) система може оцінити ризик правопорушення, надати допоміжну інформацію, а також сформувати актуальний рекламний блок. Зазначена схема наочно вказує на великий об’єм вхідних даних, нетривіальність задачі побудови системи машинного аналізу, а також необхідність роботи у режимі реального часу. Як було вказано вище, зазначена задача найбільш ефективно вирішується через застосування архітектури CNN, алгоритми на основі якої забезпечують часткову стійкість при аналізі до змін масштабу або зсуву окремого зображенні, зміни ракурсу фото- і відеозйомки, а також значної частини оптичних аберацій. Моделі нейромережевої архітектури, що базуються на CNN включають у себе локальні рецептивні поля, що забезпечують локальну двовимірну зв’язність нейронів для спрощення архітектури, загальні вагові коефіцієнти для детектування ознак у локальних областях кадру і зменшення навантаження на обчислювальний ресурс, просторові підвибірки для ієрархічної організації архітектури і паралельні процеси проведення обчислень для налаштування багатоканальна система обчислень, що зменшує час обробки вхідних даних при роботі у режимі реального часу. Основною перевагою CNN є проведення аналізу на

основі ядра згортки (Convolution Kernel, CK), яке представляє собою матрицю набагато меншої розмірності $M_{CK}: X_{CK} \times Y_{CK}$ ніж матриця вхідного зображення $M_0: X_0 \times Y_0$ (де $X_{CK} \ll X_0$ і $Y_{CK} \ll Y_0$, причому зазвичай $X_{CK} = Y_{CK}$), що формується на етапі навчання CNN. Нейромережевий алгоритм, що базується на архітектурі CNN полягає у скануванні за допомогою M_{CK} зображення M_0 та формування у багатоканальному режимі карти виділених ознак. Крок з яким здійснюється сканування (крок згортки ΔK) при цьому є дещо меншим за розмірність матриці M_{CK} ($\Delta K < X_{CK}$ і $\Delta K < Y_{CK}$) для забезпечення часткового перекриття. Розглянемо класичну квадратну форму ядра згортки, де $K = X_{CK} = Y_{CK}$, а K^2 визначається як вага ядра згортки. Відповідно, загальна кількість параметрів N_F, що визначається е результаті виконання процедури згортки розраховується як добуток ваги ядра згортки, кількості каналів N_I та кількості каналів згортки N_I (Convolutional Features, CF):

$$N_F = N_I \cdot N_{CF} \cdot K^2. \quad (1)$$

Також адаптація алгоритмів на основі CNN зумовлює включення процедури субдискретизації для зменшення розмірності карт ознак, що надає можливість додатково зменшити навантаження на обчислювальний ресурс і збільшити інваріантність роботи алгоритму по відношенню до масштабу вхідних даних. Такими чином, загальна архітектура прихованих шарів, представляє собою багатошарову структуру, що складається з багатокомпонентного елементу «шар згортки - шар активації - шар субдискретизації». Після формування карти ознак на базі вхідного масиву даних та ядра згортки, її елементи зменшуються і надалі кожен з них проходить процедуру згортки, у результаті чого реалізується багатоканальна обробка даних і організується ієрархія ознак. Навчання

CNN організується за схемою «навчання з учителем», що включає застосування підготовленої вибірки даних та застосування методу зворотного поширення помилки, яке здійснюється через пошук екстремуму середньоквадратичного значення похибки. Слід зазначити, що для представленої задачі також актуальними є схеми «навчання без вчителя», зокрема такі, що використовують базу вибірок випадкових фрагментів зображень навчальної вибірки [14], застосування при навчанні алгоритмів на основі обмежених машин Больцмана [15], імовірнісних моделей [16], нейромереж розрідженого кодування [17], тощо.

Слід вказати, що попри налаштованість алгоритмів на основі CNN на роботу з графічними даними, вони не можуть бути використані при аналізі великих об’ємів даних у режимі реального часу за умов обмеження обчислювального ресурсу системи, тому у даному випадку актуально адаптувати моделі, що базуються на архітектурі R-CNN.

2. Розробка алгоритмів розпізнавання, класифікації і позиціонування візуальних об’єктів на основі архітектури R-CNN

Нейромережеві алгоритми на основі архітектури R-CNN як підклас алгоритмів CNN реалізують селективний пошук візуальних об’єктів через поділ матриці вхідного зображення на так звані області інтересу (RoI: Regions of Interest), по відношенню до яких виконуються класичні CNN-алгоритми за схемою розглянутою у попередньому розділі. Базова схема роботи відповідного нейромережевого алгоритму включає у себе наступні етапи (рис. 2):

- тренування базової CNN на основі навчальної вибірки з застосуванням допоміжних алгоритмів;
- селективний пошук RoI, що містять візуальні об’єкти, які можуть бути актуальні для класифікації;
- уніфікація розміру RoI відповідно до параметрів наявної архітектури нейромережі;

- видалення тих RoI, що відповідно класифікації не містять актуальних візуальних об’єктів;
- налаштування нейромережі відповідно до уніфікованих RoI;
- виділення ознак для кожного з RoI, що супроводжується оцінкою функціонального вектора на основі метода опорних векторів (Support Vector Machine, SVM).

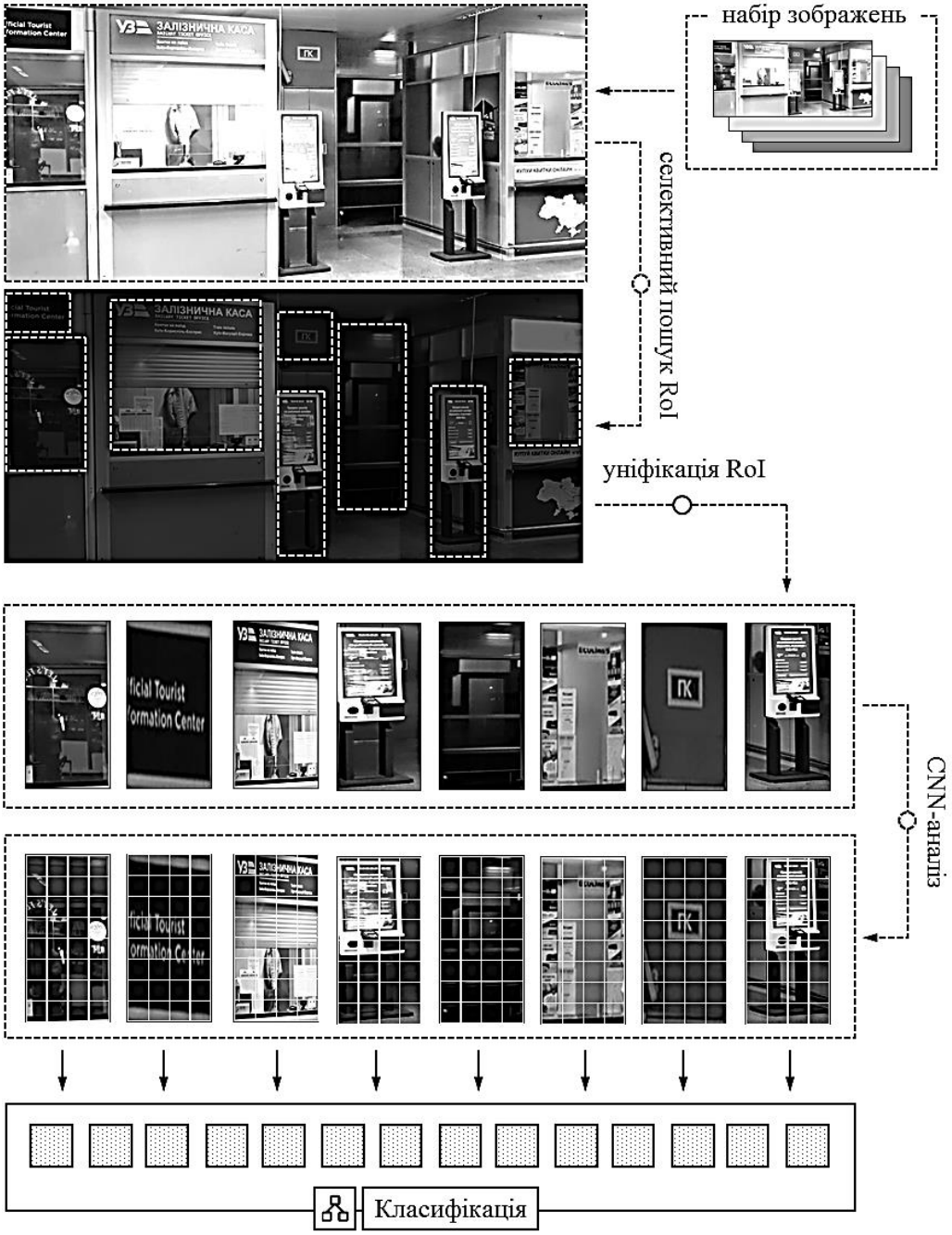


Рисунок 2 — Базова схема класифікації за допомогою алгоритму на основі R-CNN

При цьому SVM визначається для кожного класу ознак, де позитивні вибірки до схеми визначаються як запропоновані RoI, для котрих показник перетину областей (Intersection over Union, IU) є більшим за порогове значення κ_{IU}. Характерно, що навчання для класичної моделі R-CNN є ресурсомістким і тривалим процесом, яке включає у себе виконання процедури вибіркового пошуку для надзвичайно великої кількості окремих RoI для кожної матриці зображення та оцінку вектора ознак для кожної зазначеної області. Відповідна процедура включає застосування трьох моделей:

- CNN для класифікації матриці зображення;
- класифікатор SVM для що ідентифікації візуальних об’єкти,
- модель регресії, яка використовується для оптимізації селективного пошуку.

Адаптація архітектури R-CNN по відношенню до задачі дослідження зумовлює зменшення ресурсомісткості нейромережевого алгоритму з метою зменшення часу обробки даних. Це реалізується на основі класу нейромережевих моделей «Fast R-CNN», що об’єднують CNN, SVM та регресійну модель в рамках однієї структури. Модель «Fast R-CNN» передбачає застосування CNN до всієї матриці зображення, що через селективний відбір поділяє матрицю на набір ознак, яка застосовується для вивчення класифікації візуальних об’єктів. Визначення пулінгового шару RoI як найбільш важливий етап реалізації алгоритму «Fast R-CNN», дозволяє конвертувати ознаки області зображення розміром $X \times Y$ у вікні фіксованого розміру$x \times y$. Таким чином, область має бути розділена на матрицю підвікон розмірності $(X/x) \times (Y/y)$, а відповідна вдосконалена модель «Fast R-CNN» складається з послідовного виконання наступних операцій:

- попередня підготовка CNN відповідно завдання класифікації зображень;

- проведення оцінки через селективний пошук RoI для зазначеної кількості областей;
- адаптація CNN через заміну пулінгового шару пулінговим шаром RoI, на основі якого отримуються спільні вектори ознак;
- адаптація CNN через заміну останнього повнозв'язного шару (Fully Connected Layer, FCL) і останньої функції Softmax, що представляє собою нормовану експоненційну функцію для шару n на FCL і функцію Softmax фонового класу $(n+1)$;
- оцінка класу $(n+1)$ на основі функції Softmax з отриманням значення функції дискретного розподілу ймовірностей для кожного RoI;
- прогнозування відносно вихідних зміщень RoI для кожного класу N класів за допомогою регресійної моделі.

Формалізація відповідної схеми R-CNN для побудови математичної моделі для побудови математичної моделі зумовлює введення наступних системи позначень:

- набір міток класу $n \in [1; N]$;
- дискретний набір значень розподілу ймовірності $P_n \in [P_1; P_N]$, що можуть бути розраховані на сонові функції Softmax;
- істинні значення функції обмежувальної коробки (Bounding Box Function, BB) як $B\left(B_{x\downarrow}, B_{y\downarrow}, B_{x\uparrow}, B_{y\uparrow}\right)$;
- прогнозовані набори значень функції обмежувальної коробки як $B_n: B_{x\downarrow}(n), B_{y\downarrow}(n), B_{x\uparrow}(n), B_{y\uparrow}(n)$.

На основі зазначених величин може бути визначена функція втрат (Loss Function, LF) як сума функції класифікації втрат (Classification Loss Function, CLF) та функції прогнозування втрат на основі BB:

$$F_L = F_{CL}(P_n) + F_{BB}(B_n), \tag{2}$$

а F_{CL} та F_{BB}, у свою чергу, на основі робастної функції втрат (Robust Loss Function, RLF) розраховуються наступним чином:

$$\begin{cases} F_{CL}(P_n) = -\log(P_n) \\ F_{BB}(B_n) = \sum_n F_{RL}(\Delta B_n) \cdot \end{cases} \quad (3)$$

Функція втрат у рамках даного дослідження розглядається як цільовий показник, що може бути визначено на кількісному рівні. Пошук глобального мінімуму зазначеної функції надає можливість звести задачу оптимізації нейромережевої архітектури «Fast R-CNN» відповідно поставленої задачі виділення та класифікації візуальних об'єктів матриці зображення до стандартної математичної задачі. Зазначена схема може бути розширена для моделей побудованих на основі «Fast R-CNN», як то моделей «Faster R-CNN», «Mask R-CNN», тощо.

Висновки

У результаті проведеної роботи було визначено актуальні підходи по розробці нейромережевих алгоритмів на основі архітектури згорткових нейромереж класу R-CNN, що, зокрема, включало у себе оптимізацію машинного аналізу графічних даних з метою позиціонування візуальних об'єктів у просторі міської інфраструктури у режимі реального часу з мінімальними навантаженням на обчислювальний ресурс апаратно-програмної платформи.

При цьому у рамках дослідження було розроблено:

- базову схему аналізу графічних даних отриманих на основі системи відеоспостереження та мобільних електронних пристроїв з метою позиціонування та геолокації;
- схему виділення і класифікації візуальних об'єктів за допомогою нейромережевих алгоритмів на основі архітектури R-CNN;
- математичну модель, що може бути використана для оптимізації та оцінки ефективності процесу виділення і класифікації візуальних об'єктів за допомогою нейромережевих алгоритмів на основі архітектури «Fast R-CNN».

1. Ye H.Y.H., Gao, Z.G.Z., Qin, Z.Q.Z., & Wang, Q.W.Q. Near-infrared fundus camera based on polarization switch in stray light elimination. *Chinese Optics Letters, 11*(3), 2013, pp. 031702 031705.
2. Zou J., Li Z., Guo Z. & Hong D. Super-Resolution Reconstruction of Images Based on Microarray Camera. *Computers, Materials & Continua*, *60*(1), 2019, pp. 163–177.
3. Elzein, N. M., Fakherldin, M., Abaker, I., Abdulmajid, M. (2019). ANN-based Performance Analysis on Human Activity Recognition. *2019 4th International Conference and Workshops on Recent Advances and Innovations in Engineering (ICRAIE).* doi:10.1109/icraie47735.2019.9037749.
4. Reyes-Ortiz, J.L., et al. Human Activity and Motion Disorder Recognition: towards smarter Interactive Cognitive Environments. *in ESANN*. 2013. Citeseer.
5. Zhang, H., Wan, S., Yue, L., Wu, Z., & Zhao, Y. (2015). A new fast object detection architecture combing manually-designed feature and CNN. *2015 8th International Congress on Image and Signal Processing (CISP)*.
6. Wang, X., Ma, H., & Chen, X. (2016). Salient object detection via Fast R-CNN and low-level cues. *2016 IEEE International Conference on Image Processing (ICIP)*.
7. Guan, T., & Zhu, H. (2017). Atrous Faster R-CNN for Small Scale Object Detection. *2017 2nd International Conference on Multimedia and Image Processing (ICMIP)*.
8. Adam, B., Zaman, F., Yassin, I., Abidin, H., & Rizman, Z. (2018). Performance evaluation of faster R-CNN on GPU for object detection. *Journal of Fundamental and Applied Sciences, 9* (3S), 909.
9. Kızıloluk, S., & Sert, E. (2022). Hurricane-faster R-CNN-JS: Hurricane Detection with faster R-CNN using artificial Jellyfish Search (JS) optimizer. *Multimedia Tools and Applications.* https://doi.org/10.1007/s11042-022-13156-9
10. Wei, X., Xie, C., Wu, J., & Shen, C. (2018). Mask-CNN: Localizing parts and selecting descriptors for fine-grained bird species categorization. *Pattern Recognition, 76*, 704-714.
11. Levy, C., Ćiprijanović, A., Drlica-Wagner, A., Mutlu-Pakdil, B., Nord, B., & Tanoglidis, D. (2021). Detecting low surface brightness galaxies with mask R-CNN. *Fourth Workshop on Machine Learning and the Physical Sciences (NeurIPS 2021).* https://doi.org/10.2172/1825283
12. Du, J. (2018). Understanding of Object Detection Based on CNN Family and YOLO. Journal of Physics: *Conference Series*, 1004, 012029.
13. Yan, W., Liu, T., & Fu, Y. (2021). Yolo-tight: An efficient dynamic compression method for Yolo Object Detection Networks. *2021 13th International Conference on Machine Learning and Computing.* https://doi.org/10.1145/3457682.3457740.
14. Orhan, S., Bastanlar, Y. (2017). Effect of patch based training on object localization with convolutional neural networks. *2017 25th Signal Processing and Communications Applications Conference (SIU)*.
15. Ghebrechristos, H., Gita, G. A. (2018). Expediting Training Using Information Theory Based Patch Ordering Algorithm. *2018 International Conference on*

Computational Science and Computational Intelligence (CSCI). doi:10.1109/csci46756.2018.00224.

16. Wang, Z. (2017). Temporal-Related Convolutional-Restricted-Boltzmann-Machine Capable of Learning Relational Order via Reinforcement Learning Procedure. *International Journal of Machine Learning and Computing, 7* (1), 1-8. doi:10.18178/ijmlc.2017.7.1.610.
17. Jamieson, A. R., Drukker, K., Giger, M. L. (2012). Breast image feature learning with adaptive deconvolutional networks. *Medical Imaging 2012: Computer-Aided Diagnosis.* doi:10.1117/12.910710.

Application of R-CNN in automatic in-door positioning through ANNs analysis of graphical data

The analysis of modern approaches which can be used at development of modern methods of optical monitoring which is carried out on the basis of graphic data of mobile electronic devices and the neural network analysis is carried out. It is indicated priority in application in the development of ANNs algorithms of convolutional neural network architecture "Fast R-CNN", which includes a machine of reference vectors and a regression model. The scheme proposed in the study is characterized by the minimum resource intensity of the graphic data processing, which is achieved by solving the problems of selective search of a large volume of regions of interest for each image and estimating the feature vector for each region. It is shown that the corresponding architecture can be effectively used for the task of recognizing visual objects, their classification and positioning, respectively, to solve the problem of geolocation.

Keywords: convolutional neural networks, machine analysis, in-door positioning, region of interest, reference vector machine, regression model, loss function.

ДОДАТОК В Наукова стаття

УДК 004.043
Ткаченко М.С., Сокульський О.Є.
НТУУ "КПІ ім. Ігоря Сікорського"
ФІОТ, Кафедра ІСТ

ПРИНЦИПИ ОРГАНІЗАЦІЇ ПРОЦЕДУРИ МАШИННОГО АНАЛІЗУ НА ОСНОВІ ЗГОРТКОВОЇ НЕЙРОМЕРЕЖЕВОЇ АРХІТЕКТУРИ

Проведено огляд галузей застосування алгоритмів машинного аналізу, що базуються на моделі згорткової нейромережі. Визначено базову архітектуру згорткової нейромережі: організацію шарів нейромережі, принципи вибору функції активації та схему розрахунку функції втрат. Запропоновано комплексну методологію, що надає можливість провести організацію, налаштування та оптимізацію алгоритмів машинного аналізу, що базуються на моделі згорткової нейромережі відповідно цільових показників ефективності точності нейромережевого аналізу та навантаження на обчислювальний ресурс апаратно-програмної платформи загального комплексу.

Ключові слова: *згорткові нейронні мережі, функція активації, функція втрат, ініціалізація параметрів, регуляризація ваги, ітераційні алгоритми оптимізації, цільова функція,*

Вступ

На сьогоднішній день галузі застосування алгоритмів машинного аналізу, що базуються на моделі згорткової нейромережі (Convolutional Neural Network, CNN) включають у себе попередню і пост-обробку графічних даних, виділення і класифікацію візуальних об’єктів, побудову тривимірної сцени, сегментацію матриці зображення, тощо. Це вказує на можливість їх застосування при роботі з широким колом задач, як то машинний аналіз медичних фотоданих [1, 2], обробка даних супутникової і аеро-зйомки [3, 4], організація систем аутентифікації за біометричними

показниками і запобігання правопорушень зі застосуванням даних систем відеореєстрації [5, 6], розпізнавання текстових блоків, представлених у растровому вигляді [7, 8], тощо. Ефективність вирішення поставлених відповідно показників точності та адаптивності системи машинного аналізу, а також навантаження на обчислювальний ресурс і часу обробки даних залежить від особливостей організації нейромережевої архітектури та підходів, що використовуються у процесі навчання CNN.

Аналіз сучасних досліджень і публікацій присвячених проблемам впровадження алгоритмів машинного аналізу масивів графічних даних на основі CNN вказав на основні підходи, що використовуються при класифікації зображень (моделі LeNet-5, AlexNet, ZFNet, VGGNet, GoogLeNet, ResNet і DenseNet) нейромережевими алгоритмами [9-15], виділення візуальних об’єктів через застосування нейромережевої архітектури R-CNN, Fast R-CNN, Faster R-CNN, SPP-Net, Mask R-CNN і YOLO [16-21], а також методів сегментації матриці зображення на основі повнозв’язної CNN (Fully Convolutional Network, FCN) та програмними додатками DeepLab, Deconvnet, SegNet, DeepMask, SharpMask, U-Net, PANet та TensorMask [22-31]. Проведений аналіз вказав на ***актуальність*** вирішення завдання побудови загальних підходів, що базуються на визначенні кількісних цільових показників, по організації нейромережевих алгоритмів машинного аналізу великих масивів даних у режимі реального часу. Відсутність у представлених дослідженнях універсальної методології, що надає можливість сформувати принципи розробки зазначених алгоритмів, розглядається, відповідно, як ***невирішена частина загального дослідження***.

Таким чином, **метою дослідження** стала розробка цілісної методології побудови нейромережевих алгоритмів на основі архітектури CNN, що характеризуються високою точністю машинного аналізу та якістю обробки вхідних даних за умов мінімізації навантаження на

обчислювальний ресурс та часу обробки даних, які можуть бути ефективно використані при роботі з широким колом задач.

1. Особливості організації архітектури згорткової нейромережі

Аналіз базової схеми CNN вказує, що переваги даної архітектури при роботі з графічними даними (узагальнення різнорідних фото-даних і виділення ознак глибокого рівня, зокрема розташування візуального об’єкту та оцінка естетичності складових зображення), пов’язані з організацією структури CNN, як то наявністю повнозв’язних (Fully Connected, FC) шарів і моделлю прямого розповсюдження (Deep Feed-Forward Architecture, DF-FA), що може бути ефективно організована як нейромережа глибинного навчання [32, 33]. В загальному вигляді архітектура CNN глибинного навчання складається з сандвіч-структури згорткових та пулінгових шарів, що організовані у відповідні набори, де кожен наступний набір виділяє ознаки з більшим рівнем абстракції. Як показано на рис. 1, якщо сандвіч-структура складається з N наборів, то загальна кількість шарів, що включатиме вхідний і вихідний, а також FC-шар, розраховується як $(2N + 3)$.

Можна вказати, що переваги нейромережевих алгоритмів на основі архітектури CNN при відновленні, обробці і аналізі графічних даних можуть бути формалізовані наступним чином:

1. Функція розподілу ваги, яка зменшує кількість параметрів і надає можливість уникнути типової проблеми перенавчання нейромережі;

2. Навчання шарів, що відповідають за виділення і класифікацію ознак у рамках однієї процедури, що збільшує цілісність та точність проведення процедури машинного аналізу;

3. Спрощена схема модифікації та масштабування CNN при зміні та розширенні набору поставлених задач.

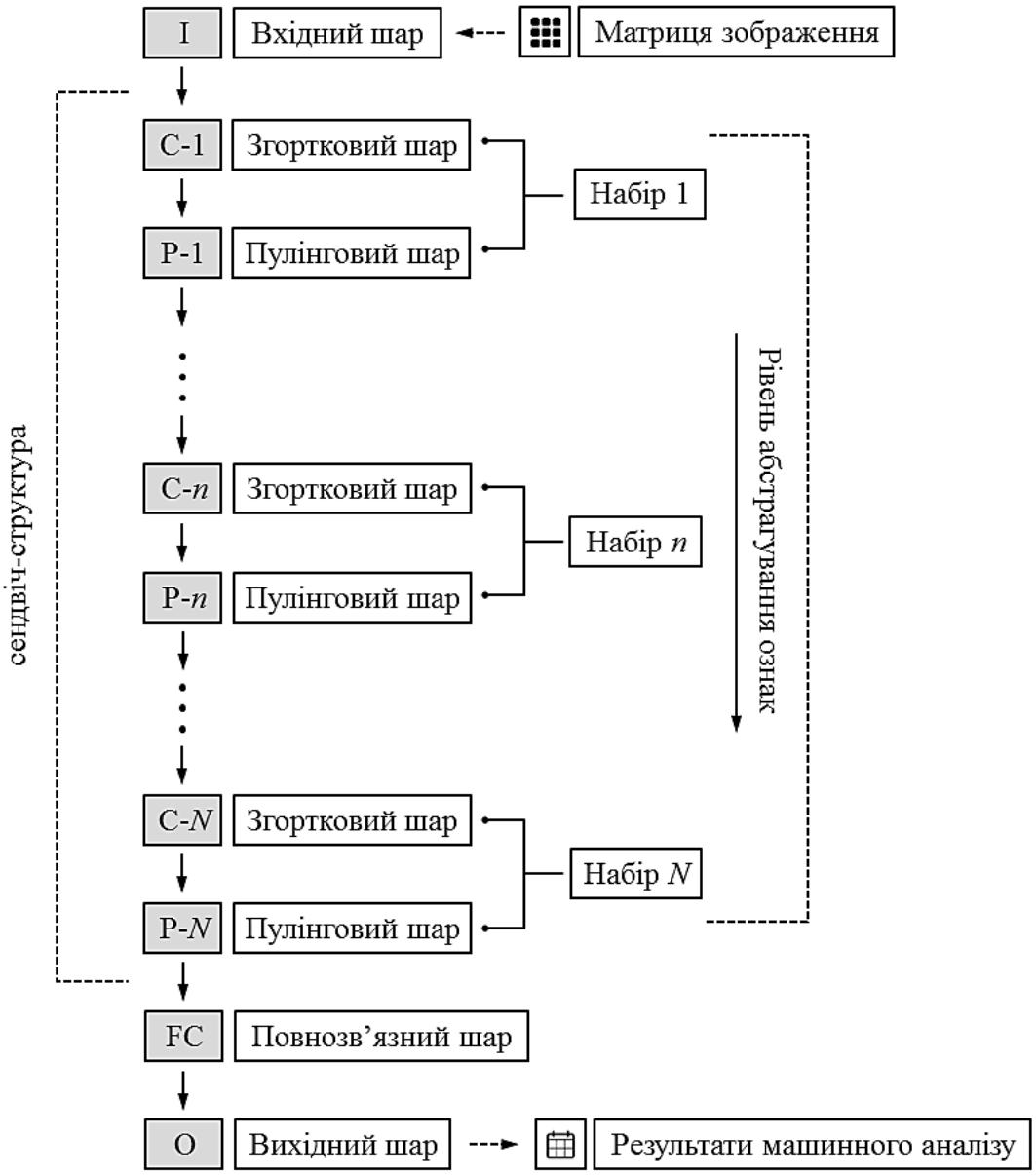


Рисунок 1— Базова структура згорткової нейромережі глибинного навчання

На рівні кожного загорткового шару, що характеризується розміром ядра згортки (Convolutional Kernel, CK), що виконує роль фільтра, здійснюється процедура згортки зображення відповідно заданої метрики для побудови карти ознак (Feature Map, FM). При цьому CK представляє собою матрицю значень, що є вагами ядра, налаштування яких відповідає процесу навчання. Під час обробки графічних даних матриця зображення розбивається на колірні канали і надалі обробляється по багатоканальній

схемі, що кратно збільшує швидкість машинного аналізу. Для кожного каналу на основі СК проводиться сканування матриці зображення через зміщення СК на відповідний крок згортки (Convolutional Stride, CS). Слід зазначити, що розмір CS має бути меншим СК для забезпечення часткового перекриття у процесі сканування. CS є аргументом цільових функцій вибір якого відповідає задачі дослідження: при збільшенні CS зменшується навантаження на обчислювальний ресурс і час обробки даних, але водночас зменшується розмірність FM, що, очевидним чином, негативно впливає на ефективність машинного аналізу. Недоліком базового підходу, представленого вище, є те, що периферійні елементи матриці зображення (пікселі) приймають участь у меншій кількості згорток. Це може бути вирішено через застосування операції доповнення зображення (Convolution Padding, CP), що застосовується безпосередньо перед проведення процедури згортки. Згортки, у яких застосовується CP з метою збереження розмірності матриці зображення класифікуються як «однакові» (Same Convolution, SC), а згортки з нульовим доповненням — «правильними» (Valid Convolution, VC).

Відповідно проведеної формалізації виконання процедури згортки, розмірність карти ознак $X_C \times Y_C$, що отримується на виході загорткового шару при розмірності матриці вхідного зображення $X \times Y$ визначається наступними чином:

$$\begin{cases} X_C = \left\lfloor 1 + \dfrac{X - S_{CK} + S_{CP}}{S_{CS}} \right\rfloor \\ Y_C = \left\lfloor 1 + \dfrac{Y - S_{CK} + S_{CP}}{S_{CS}} \right\rfloor \end{cases}, \quad (1)$$

де S_{CK} — розмірність матриці СК, S_{CP} — доповнення згортки, а S_{CS} — розмір CS. Алгоритм розрахунку може бути розширено для прямокутного СК, через перехід для розрахунку X_C від S_{CK} до X_{CK}, а для Y_C від S_{CK} до

Y_{CK}. Загальною перевагою застосування згорткових шарів у нейромережевій архітектурі є розріджена зв'язність (тобто відсутність повного набору зв'язків між всіма нейронами сусідніх шарів) та розподіл вагових коефіцієнтів, що у даному випадку не є унікальними для двох окремих нейронів, а відповідає виключно їх зв'язку з елементами вхідних даних, як то пікселями матриці вхідного зображення. Це значно спрощує виконання процедур навчання та машинного аналізу відповідно показників навантаження на обчислювальний ресурс та часу обробки запиту.

Також, як це показано на рис. 1 необхідним елементом архітектури CNN є пулінгові шари, що використовуються для стиснення карт ознак, як додатковий засіб зменшення навантаження на обчислювальний ресурс. Ефективність операції пулінгу визначається через співвіднесення рівня стиснення та мінімізації втрат значимих ознак та точності класифікації. Подібно до операції згортки, операція пулінгу характеризується розміром ядра пулінгу (Pooling Kernel, PK) та кроком сканування (Pooling Stride, PS). Поза відповідних значень S_{PK} і S_{PS}, у якості аргументу цільової функції розглядається і сама функція, на основі якої здійснюється операція пулінгу, як то $F_{\mathrm{P}\uparrow}$ — пулінг на основі максимального значення (Max Pooling, MP), $F_{\mathrm{P}\downarrow}$ — пулінг на основі мінімального значення (Max Pooling, MP), F_{AP} — пулінг на основі середнього значення (Average Pooling, AP) та інші підходи, відповідно поставленої задачі, зокрема гібридні схеми [32, 33]. У свою чергу, передостанній шар нейромережевої архітектури — «FC» використовується у якості класифікатора, що об'єднує карти ознак з найвищим рівнем абстрагування. Як це показано на рис. 1 відповідний шар є повнозв'язним по відношенню до останнього пулінгового шару «P-N» і вихідного шару «O».

2. Вибір функції активації та функції втрат згорткової нейромережі

На цільові показники нейромережевого аналізу також впливає вибір функції активації (Activation Functions, AF), що на основі суми вхідних даних нейронів з урахуванням зміщення (за наявності нейронів зміщення) визначає умову спрацювання окремого нейрона. У CNN глибинного навчання для представлення нелінійного відображення між масивом даних на вході та на виході використовується нелінійна AF. Також слід зазначити для організації навчання CNN за методом зворотного поширення помилки використовується диференційована AF. Такими чином, актуальний набір AF, що розглядається у рамках дослідження, включає у себе такі як:

- сигмоїда ($A_{Sig}(x)$);
- функція Tanh ($A_{Tanh}(x)$);
- функція ReLu ($A_{ReLu}(x)$), а також побудовані на її основі функції «Leaky ReLu» ($A_{LR}(x)$) і «Noisy ReLU» ($A_{NR}(x)$).

Наведемо математичне представлення зазначених функцій у рамках математичного апарату, що використовується у даному дослідженні:

$$\begin{cases} A_{Sig}(x) = (1 + e^{-x})^{-1} \\ A_{Tanh}(x) = (e^{x} - e^{-x})/(e^{x} + e^{-x}) \\ A_{ReLu}(x) = \max\limits_{x}(0; x) \\ A_{LR}(x) = \begin{cases} x \text{ при } x > 0 \\ \mu x \text{ при } x \le 0 \end{cases} \\ A_{NR}(x) = \max\limits_{x}(x + b), \text{де } b \sim F\big(0; \sigma(x)\big) \end{cases} . \quad (2)$$

У свою чергу, цільовий показник точності машинного аналізу визначається на основі функції втрат (Loss Function, LF), а відповідно пошук її глобального мінімуму надає можливість на кількісному рівні вирішити задачу оптимізації нейромережевого алгоритму. На загальному рівні функція втрат базується на співвіднесенні істинних (умовно

істинних) параметрів як набору P_i з результатами роботи CNN як набору P_i', де $i \in [1; I]$, причому сама реалізація даної процедури залежить від класу задач, яка виконується на рівні застосування відповідного нейромережевого алгоритму [32, 33].

Розглянемо наступні форми представлення функції втрат, що можуть бути використані у рамках представленого дослідження:

- $L_{SM}(P, P')$ — функція «Soft-Max»;
- $L_E(P, P')$ — функція втрат на основі евклідової метрики (Euclidean Loss Function, ELF);
- $L_H(P, P')$ — кусково-лінійна функція втрат (Hinge Loss Function, HLF).

Аналогічно, математичне представлення зазначених функцій у рамках математичного апарату, що використовується у даному дослідженні може бути проведено наступним чином:

$$\begin{cases} L_{SM}(P, P') = -\sum_{i=1}^{I} (P_i \cdot \log(P_i')) \\ L_E(P, P') = \dfrac{\sum_{i=1}^{I} (P_i - P_i')^2}{2I} \\ L_H(P, P') = \sum_{i=1}^{I} \Big(\max\big(0, \mu - P_i' \cdot (2P_i - 1)\big)\Big) \end{cases} . \quad (3)$$

Вибір функції втрат та експериментальне визначення часу обробки запиту при фіксованій архітектурі апаратно-програмного комплексу формують повний набір цільових функцій. Через варіювання аргументів цільових функцій з метою пошуку глобальних мінімумів проводиться процедура оптимізація архітектури CNN.

3. Розробка, налаштування та оптимізація алгоритмів машинного аналізу, що базуються на моделі згорткової нейромережі

Алгоритм налаштування та оптимізації процедури машинного аналізу, що базуються на архітектурі CNN у загальному вигляді складається з наступного набору етапів (рис. 2):

1. Визначення типової архітектури апаратно-програмної платформи, що дозволяє оцінити обчислювальний ресурс системи машинного аналізу на основі CNN;
2. Визначення типових параметрів вхідних даних (розмірність матриці зображення $X \times Y$ та кількість зображень, що підлягають аналізу на одиницю часу $\overline{K}$, K_{max});
3. Архітектура CNN глибинного навчання, що визначається через шари «I», «C-*n*», «P-*n*», «FC» і «O»;
4. Параметри згортки: розмірність матриці ядра згортки, розмір доповнення згортки, розмір кроку сканування;
5. Параметри пулінгу: розмірність матриці ядра пулінгу, розмір кроку сканування, а також функція, на основі якої здійснюється операція пулінгу;
6. Вибір функції активації і параметрів функції активації;
7. Розрахунок функції втрат машинного аналізу та визначення часу обробки вхідного запиту;
8. Мінімізація цільових функцій втрат та часу обробки через корегування параметрів нейромережевого алгоритму.

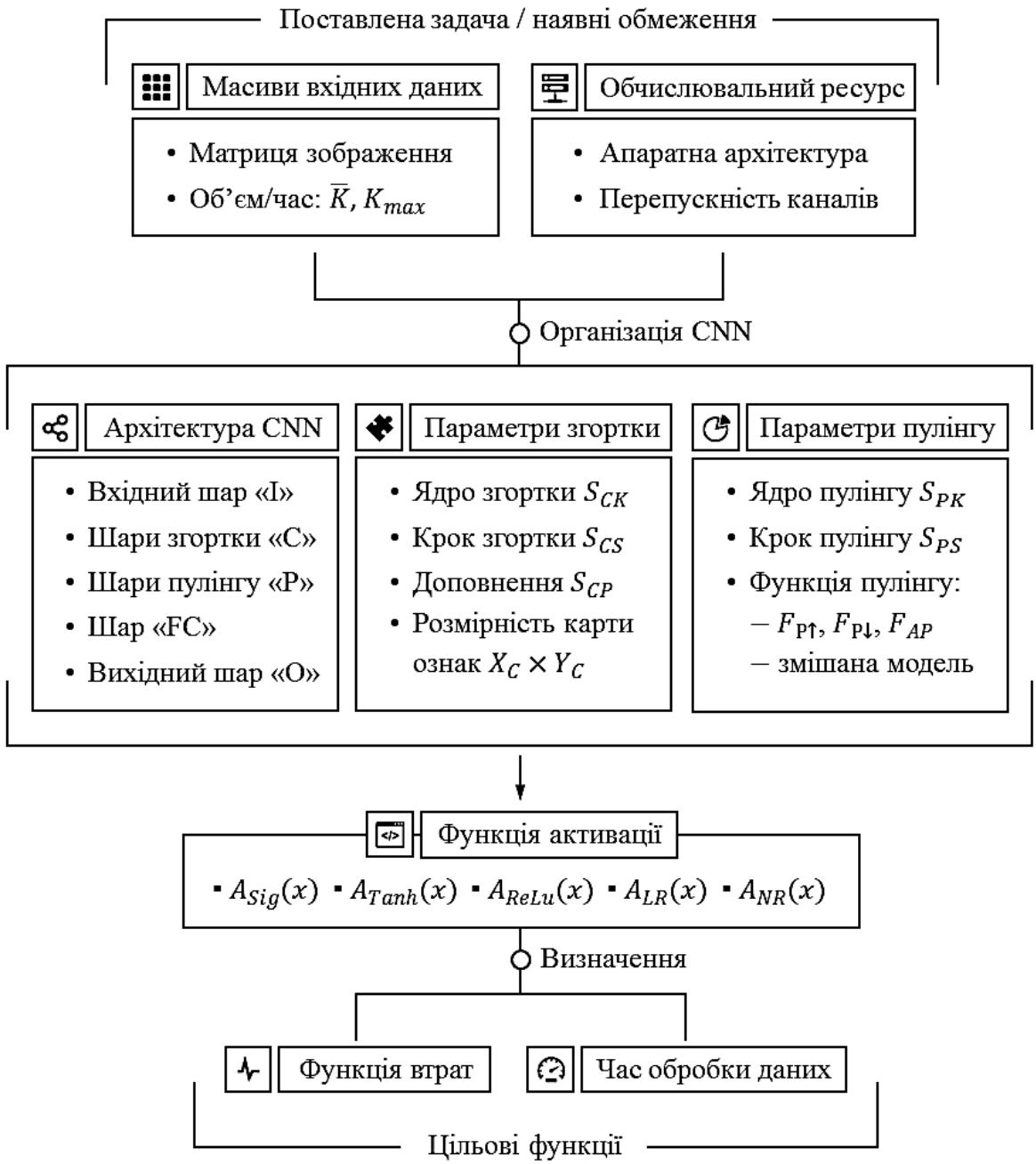


Рисунок 2 — Схема налаштування та оптимізація алгоритмів машинного аналізу, що базується на архітектурі згорткової нейромережі

Розширення даного алгоритму можливо через включення у параметри цільових функцій особливості навчання нейромережевих алгоритмів та попередню обробку вхідних даних, що у свою чергу

призводить до необхідності включення у цільові функції час навчання та час попередньої обробки.

Висновки

У результаті проведеної роботи було визначено принципи розробки цілісної та універсальної методології побудови нейромережевих алгоритмів на основі архітектури CNN, що характеризуються високою точністю машинного аналізу за умов мінімізації часу обробки даних при наявних обмеженнях на обчислювальний ресурс апаратно-програмної платформи.

При цьому у рамках даного дослідження було проведено:

- визначення принципів побудови структури згорткової нейромережі глибинного навчання;
- формалізацію математичного апарату проведення процедури згортки;
- формалізацію математичного апарату проведення процедури пулінгу;
- модель організації процедури налаштування та оптимізації алгоритмів машинного аналізу, що базується на архітектурі згорткової нейромережі, на рівні кількісних показників.

1. Zhao, Y., Ge, F., & Liu, T. (2018). Automatic recognition of holistic functional brain networks using iteratively optimized convolutional neural networks (IO-CNN) with weak label Initialization. *Medical Image Analysis, 47*, 111–126. https://doi.org/10.1016/j.media. 2018.04.002.
2. Li, B., Keikhosravi, A., Loeffler, A. G., & Eliceiri, K. W. (2021). Single image super-resolution for whole slide image using convolutional neural networks and self-supervised color normalization. *Medical Image Analysis, 68*, 101938. https://doi.org/10.1016/j.media. 2020.101938.
3. Maggiori, E., Tarabalka, Y., Charpiat, G., & Alliez, P. (2017). Convolutional neural networks for large-scale remote-sensing image classification. *IEEE Transactions on Geoscience and Remote Sensing, 55*(2), 645–657. https://doi.org/10.1109/tgrs.2016. 2612821.
4. Ji, C., & Tang, H. (2020). Number of building stories estimation from monocular satellite image using a modified mask R-CNN. *Remote Sensing, 12* (22), 3833. https://doi.org/10.3390/rs12223833.

5. Rasti, P., Uiboupin, T., Escalera, S., & Anbarjafari, G. (2016). Convolutional Neural Network Super resolution for face recognition in surveillance monitoring. *Articulated Motion and Deformable Objects*, 175–184. https://doi.org/10.1007/978-3-319-41778-3_18.
6. Kumar, S., & Singh, S. K. (2020). Occluded thermal face recognition using bag of CNN. *IEEE Signal Processing Letters, 27,* 975–979. https://doi.org/10.1109/lsp.2020.2996429.
7. Opitz, M., Diem, M., Fiel, S., Kleber, F., & Sablatnig, R. (2014). End-to-end text recognition using local ternary patterns, mser and deep convolutional nets. *2014 11th IAPR International Workshop on Document Analysis Systems.* https://doi.org/10.1109/ das.2014.29.
8. Wang, Z.-R., Du, J., & Wang, J.-M. (2020). Writer-aware CNN for parsimonious HMM-based offline handwritten Chinese text recognition. *Pattern Recognition, 100*, 107102. https://doi.org/10.1016/j.patcog.2019.107102.
9. Verdhan, V. (2021). Image classification using LeNet. *Computer Vision Using Deep Learning, 67–101*. https://doi.org/10.1007/978-1-4842-6616-8_3.
10. Zhang, X., Pan, W., & Xiao, P. (2018). In-vivo skin capacitive image classification using AlexNet Convolution Neural Network. *2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC).* https://doi.org/10.1109/icivc.2018.8492860.
11. Kaddoun, S. S., Aberni, Y., Boubchir, L., Raddadi, M., & Daachi, B. (2021). Convolutional Neural Algorithm for palm vein recognition using ZFNet architecture. *2021 4th International Conference on Bio-Engineering for Smart Technologies (BioSMART).* https://doi.org/10.1109/biosmart54244.2021.9677799.
12. Chaudhari, S., Sardar, V., Rahul, D. S., Chandan, M., Shivakale, M. S., & Harini, K. R. (2021). Performance analysis of CNN, Alexnet and vggnet models for drought prediction using satellite images. *2021 Asian Conference on Innovation in Technology (ASIANCON).* https://doi.org/10.1109/asiancon51346.2021.9545068.
13. Teymournezhad, K., Azgomi, H., & Asghari, A. (2022). Detection of counterfeit banknotes by security components based on image processing and GoogLeNet Deep Learning Network. *Signal, Image and Video Processing.* https://doi.org/10.1007/s11760-021-02104-z.
14. M., N. K. (2020). Breast cancer classification of image using modified ResNet. Journal of Advanced Research in Dynamical and Control Systems, 12(3), 134–140. https://doi.org/10.5373/jardcs/v12i3/20201175.
15. Li, G., Zhang, C., Lei, R., Zhang, X., Ye, Z., & Li, X. (2019). Hyperspectral remote sensing image classification using three-dimensional-squeeze-and-excitation-densenet (3D-Se-DenseNet). *Remote Sensing Letters, 11*(2), 195–203. https://doi.org/10.1080/2150704x. 2019.1697001
16. Wang, X., Ma, H., & Chen, X. (2016). Salient object detection via Fast R-CNN and low-level cues. *2016 IEEE International Conference on Image Processing (ICIP)*.
17. Adam, B., Zaman, F., Yassin, I., Abidin, H., & Rizman, Z. (2018). Performance evaluation of faster R-CNN on GPU for object detection. *Journal of Fundamental and Applied Sciences, 9* (3S), 909.

18. Kızıloluk, S., & Sert, E. (2022). Hurricane-faster R-CNN-JS: Hurricane Detection with faster R-CNN using artificial Jellyfish Search (JS) optimizer. *Multimedia Tools and Applications.* https://doi.org/10.1007/s11042-022-13156-9.
19. Wei, X., Xie, C., Wu, J., & Shen, C. (2018). Mask-CNN: Localizing parts and selecting descriptors for fine-grained bird species categorization. *Pattern Recognition, 76,* 704-714.
20. Yan, W., Liu, T., & Fu, Y. (2021). Yolo-tight: An efficient dynamic compression method for Yolo Object Detection Networks. *2021 13th International Conference on Machine Learning and Computing.* https://doi.org/10.1145/3457682.3457740.
21. Du, J. (2018). Understanding of Object Detection Based on CNN Family and YOLO. Journal of Physics: *Conference Series*, 1004, 012029.
22. Qin, P. (2019). Fully convolutional-based dense network for lung nodule image retrieval algorithm. *International Journal of Performability Engineering*. https://doi.org/10.23940/ ijpe.19.01.p33.326336.
23. Cai, Y., & Li, Q. (2021). DeepLab network for Meteorological Trough Line Recognition. *2021 4th International Conference on Sensors, Signal and Image Processing.* https://doi.org/10.1145/3502814.3502820.
24. Mukherjee, A., Chakraborty, S., & Saha, S. K. (2019). Detection of loop closure in slam: A DeconvNet based approach. *Applied Soft Computing, 80*, 650–656. https://doi.org/10.1016/j.asoc.2019.04.041.
25. Saood, A., & Hatem, I. (2020). Covid-19 lung CT image segmentation using deep learning methods: UNET vs. segnet. https://doi.org/10.21203/rs.3.rs-56882/v2.
26. Wang, P., Xiong, L., & Dan, B. (2021). Surface vortex image segmentation in KR desulfurization based on improved SegNet model. 2021 China Automation Congress (CAC). https://doi.org/10.1109/cac53003.2021.9728231.
27. Son, S.-B., Jung, J.-U., Oh, H.-S., & Jung, Y.-chul. (2020). DeepMask: Face masking system using deep neural networks on real-time streaming. *Journal of Institute of Control, Robotics and Systems, 26* (6), 423–428. https://doi.org/10.5302/j.icros.2020.20.0029.
28. Trullo, R., Petitjean, C., Ruan, S., Dubray, B., Nie, D., & Shen, D. (2017). Segmentation of organs at risk in thoracic CT images using a SharpMask architecture and conditional random fields. *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017).* https://doi.org/10.1109/isbi.2017.7950685.
29. Hu, X., & Yang, H. (2020). Dru-Net: A novel U-Net for biomedical image segmentation. IET Image Processing, 14(1), 192−200. https://doi.org/10.1049/iet-ipr.2019.0025.
30. Wang, K., Liew, J. H., Zou, Y., Zhou, D., & Feng, J. (2019). Panet: Few-shot image semantic segmentation with prototype alignment. *2019 IEEE/CVF International Conference on Computer Vision (ICCV).* https://doi.org/10.1109/iccv.2019.00929.
31. Chen, X., Girshick, R., He, K., & Dollar, P. (2019). TensorMask: A foundation for dense object segmentation. *2019 IEEE/CVF International Conference on Computer Vision (ICCV).* https://doi.org/10.1109/iccv.2019.00215.

Principles of machine analysis procedure organization based on convolutional neural network architecture

The areas of application of machine analysis algorithms based on the convolutional neural network model are reviewed. The basic architecture of the convolutional neural network is determined: the organization of neural network layers, the principles of activation function selection and the scheme of loss function calculation. The formalization of the learning process of the convolutional neural network based on preprocessing of data, parameters initialization, weights regularization and iterative optimizer algorithms selection is carried out. A complex methodology is proposed, which provides an opportunity to organize, configure and optimize machine analysis algorithms based on the model of convolutional neural network in accordance with the target performance efficiency of neural network analysis and the load on the computing resource of the general complex hardware and software platform.

Keywords: convolutional neural networks, activation function, loss function, parameters initialization, weights regularization, iterative optimizer algorithms, target function.