1МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики
(повне найменування назва факультету)

дискретного аналізу та інтелектуальних систем
(повна назва кафедри)

Магістерська робота

«Використання мапи глибини в обробці зображень»

Виконав: студент групи ПМіМ-23с
спеціальності
122 «Комп'ютерні науки»
(шифр і назва спеціальності)

Жаворонков М. А.
(підпис) (прізвище та ініціали)

Керівник Притула М. М.
(підпис) (прізвище та ініціали)

Рецензент Селіверстов Р. Г.
(підпис) (прізвище та ініціали)

Львів – 2022

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет Прикладної математики та інформатики
Кафедра Дискретного аналізу та інтелектуальних систем
Спеціальність 122 «Комп'ютерні науки»
(шифр і назва)

«ЗАТВЕРДЖУЮ»
Завідувач кафедри Притула М. М.

"31" серпня 2022 року

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Жаворонкову Микиті Андрійовичу
(прізвище, ім'я, по батькові)

1.Тема роботи «Використання мапи глибини в обробці зображень»

керівник роботи проф. Притула Микола Миколайович док. фіз.-мат. наук,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені Вченою радою факультету від **" 13" вересня 2022 року № 15**

2. Строк подання студентом роботи **12.12.2022р.**

3. Вихідні дані до роботи

1. Наукові видання з питань обробки зображень і комп'ютерного зору
2. мови програмування Rust, Python
3. бібліотеки Image-rs, Rayon, PyQt, Qt
4. середовище програмування Visual Studio Code

4. Зміст магістерської роботи (перелік питань, які потрібно розробити)

1. Швидкий та надійний метод отримання мапи глибини зображення
2. Алгоритм виділення областей на мапі глибини
3. Програмний застосунок для використання методів обробки зображень до виділених областей

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

1. Приклади роботи програмного застосунку
2. Зображення для пояснення роботи методів отримання мапи глибини
3. Можливі способи отримання вхідних даних до методів пасивної тріангуляції

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання **12.12.2022 р.**

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1.	Вивчення предметної області	10.05.2022 — 20.05.2022	
2.	Розробка алгоритму виділення глибини і дискретизації зображення	21.05.2022 — 06.07.2022	
3	Покращення швидкодії алгоритму дискретизації зображення	07.07.2022 — 14.07.2022	
4.	Застосування дискретизованого зображення для отримання мапи глибини	15.07.2022 — 02.08.2022	
5.	Ознайомлення з операціями обробки зображень	03.08.2022 — 10.08.2022	
6.	Реалізація операцій обробки зображень в застосунку	11.08.2022 — 23.08.2022	
7.	Дослідження бітових масок	24.08.2022 — 04.09.2022	
8.	Реалізація локальних методів обробки зображень з використанням бітових масок	05.09.2022 — 26.09.2022	
9.	Реалізація алгоритму кластеризації для отримання бітової маски	27.09.2022 — 13.10.2022	
10.	Створення графічного інтерфейсу для програмного застосунку	14.10.2022 — 29.10.2022	
11.	Оформлення роботи	30.10.2022 — 15.11.2022	

Студент _____ Жаворонков М. А.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Притула М. М.
(підпис) (прізвище та ініціали)

Реферат

Актуальність теми дослідження. Глибокі нейронні мережі нині стають одним із найпопулярніших методів машинного навчання. Вони показують кращі результати в порівнянні з альтернативними методами в таких областях, як розпізнавання мови, обробка природної мови, комп'ютерний зір [1], медична інформатика [2] та ін. виділяючи з даних важливі ознаки, необхідні для вирішення завдання. В альтернативних алгоритмах машинного навчання ознаки мають виділятися людьми, існує спеціалізований напрямок досліджень – інженерія ознак (feature engineering). Однак при обробці великих обсягів даних нейронна мережа справляється з виділенням ознак набагато краще ніж людина.

Модель штучних нейронних мереж була запропонована в 1943 [4], а сам термін глибоке навчання (deep learning) став широко використовуватися тільки починаючи з 2006 [5, 6]. До цього застосовувалися терміни завантаження глибоких мереж (loading deep networks) [7, 8] та навчання глибокої пам'яті (learning deep memories) [9].

Зростання популярності глибоких нейронних мереж, що відбувається останні кілька років, можна пояснити трьома чинниками. По-перше, відбулося суттєве збільшення продуктивності комп'ютерів, у тому числі прискорювачів обчислень GPU (Graphics Processing Unit), що дозволило навчати глибокі нейронні мережі значно швидше і з більш високою точністю [10]. Раніше обчислювальних потужностей не вистачало для навчання складної мережі, придатної для вирішення практичних завдань. По-друге, був накопичений великий обсяг даних, необхідний для навчання глибоких нейронних мереж. По-третє, розроблено методи навчання нейронних мереж, що дозволяють швидко та якісно навчати мережі, що складаються зі ста і більше шарів [11], що раніше було неможливо через проблему зникаючого градієнта та перенавчання. Поєднання трьох факторів призвело до суттєвого прогресу в навчанні глибоких нейронних мереж та їх практичному використанні, що дозволило глибоким

нейронним мережам зайняти позицію лідера серед методів машинного навчання.

Мета та завдання дослідження. Метою даної роботи є оптимізація в навчанні глибоких моделей.

Для досягнення поставленої мети у роботі необхідно виконати низку завдань:

- розкрити методи навчання глибоких моделей;
- окреслити математичні аспекти навчання глибоких моделей;
- запропонувати алгоритм реалізації навчання глибоких моделей;
- розробити структуру системи розпізнавання об’єктів на базі штучної нейронної мережі;
- здійснити навчання штучної нейронної мережі;
- виконати реалізацію системи розпізнавання об’єктів на базі штучної нейронної мережі;
- здійснити верифікацію результатів дослідження.

Об’єкт та предмет дослідження. Об’єктом роботи є глибокі нейронні мережі.

Предметом виступає процес навчання глибоких нейронних мереж.

Методи дослідження. В роботі використані теорія штучних самоорганізуючих нейронних мереж для графічних масивів бази даних із застосуванням процесу розпізнавання без вчителя, теорія ймовірностей та статистичний аналіз, теорія кодування/декодування і дискретних систем, концептуального аналізу, теорія множин і кластеризації/категоризації об'єктів, методи алгоритмізації, методи комп'ютерного моделювання, а також об'єктно-орієнтованого програмування (ООП).

Наукова новизна отриманих результатів. Наявні в арсеналі алгоритми в основному використовують не адаптивні алгоритми навчання. На основі проведених досліджень в даній роботі запропоновано дослідження методів навчання глибоких нейронних мереж.

Наукова новизна полягає у формулюванні методики і проведення дослідження методів навчання глибоких нейронних мереж.

Структура роботи. Структуру роботи складають: перелік умовних скорочень, вступ, три розділи, висновки та список використаних джерел. Загальний обсяг роботи становить 108 сторінок.

ЗМІСТ

РОЗДІЛ 1 ТЕОРЕТИЧНІ АСПЕКТИ НАВЧАННЯ ГЛИБОКИХ МОДЕЛЕЙ 8
1.1 Короткий огляд літератури за темою 8
1.2 Аналіз поняття «штучна нейронна мережа» 10
1.3 Принципи формування глибоких моделей 14
1.4 Проблематика по постановка завдань дослідження 22
Висновки до розділу 26
РОЗДІЛ 2 МЕТОДИ ТА МОДЕЛІ НАВЧАННЯ ГЛИБОКИХ МОДЕЛЕЙ 27
2.1 Методи навчання глибоких моделей 27
2.2 Математичні аспекти навчання глибоких моделей 44
2.3 Алгоритм реалізації навчання глибоких моделей 55
Висновки до розділу 65
РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ НА БАЗІ ШТУЧНОЇ НЕЙРОННОЇ МЕРЕЖІ 66
3.1 Розробка структури системи розпізнавання об'єктів на базі штучної нейронної мережі 66
3.2 Навчання штучної нейронної мережі 76
3.3 Реалізація системи розпізнавання об'єктів на базі штучної нейронної мережі 77
3.4 Верифікація результатів дослідження 85
Висновки до розділу 93
ВИСНОВКИ 95
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 97

РОЗДІЛ 1
ТЕОРЕТИЧНІ АСПЕКТИ НАВЧАННЯ ГЛИБОКИХ МОДЕЛЕЙ

1.1 Короткий огляд літератури за темою

Питання дослідження штучного інтелекту вивчали чимало як зарубіжних так і вітчизняних вчених. До їх числа можна віднести Галину Машлій, Ольгу Мосій та Мар'яну Пельчер [125], які дослідили особливості управління у період розвитку штучного інтелекту як в Україні, так і за її межами. Опрацювали наукові підходи до тлумачення терміна «штучний інтелект». Встановлено, що на сьогодні не існує єдиного і точного визначення досліджуваного поняття. Авторами визначено, що за останні роки спостерігаються надзвичайно високі темпи розвитку науки, техніки та технологій, спрямованих та розроблення та впровадження у практичну діяльність людини штучного інтелекту. Науковцями зазначено, що це неминуче призведе до значних соціальних та економічних змін, у тому числі подальшої оптимізації управлінських процесів; наголошено на необхідності врахування ризиків, що можуть виникати для людства у зв'язку з розвитком штучного інтелекту; проаналізовано перспективи впровадження штучного інтелекту в Україні, задля з'ясування яких викладено результати проведеного опитування менеджерів вітчизняних підприємств щодо готовності практичного застосування новітніх наукових розробок; наведено конкретні пропозиції в сфері управління розвитком штучного інтелекту.

О.Ю. Бусол [126] розглянув думки відомих вчених зі світовим ім'ям та інших вчених у галузі філософії, кібернетики, футурології, космології, фізики щодо небезпеки створення штучного інтелекту для людства у майбутньому. Акцентовано увагу на відсутності нових підходів до створення системи контролю, насамперед етичного характеру, над штучним інтелектом.

Машинне навчання (machine learning) – це процес машинного аналізу підготовлених статистичних даних для пошуку закономірностей і створення на

їх основі потрібних алгоритмів (налаштування параметрів нейронної мережі), які потім будуть використовуватися для прогнозів.

Розрізняють 3 основних підходи до машинного навчання [1]:

- навчання з учителем;
- навчання з підкріпленням;
- навчання без вчителя (самонавчання).

Основними проблемами перед масовим застосуванням графів закономірностей і пошуку аналогій при аналізі графічних даних методами машинного навчання є наступні 2 аспекти:

1. Наявність великої кількості графічних даних для навчання, їх різноплановість та приналежність.

2. Виявлення об'єктів на зображенні.

Без вивірених і якісних графічних даних система аналізу не працюватиме, саме вони є першою серйозною складністю для впровадження.

А. С. Довбиш, В. І. Зимовець, М. В. Бібик [127] дослідили метод інформаційно-екстремального машинного навчання системи функціонального діагностування технічного стану складної машини з оптимізацією ієрархічної структури вхідних даних. Автори показали, що на функціональну ефективність машинного навчання системи функціонального діагностування суттєво впливає розміщення в ієрархічній структурі класів розпізнавання, які характеризують технічний стан машини та її вузлів. При цьому для кожної страти ієрархічної структури накладаються обмеження на кількість класів розпізнавання, що дозволяє зменшити ступінь їх перетину в просторі діагностичних ознак.

Актуальні проблеми Data Mining розкриті у роботі [3]. У посібнику розглянуто актуальні проблемі Data Mining. Виклад зосереджено на задачах класифікації, кластеризації та побудови асоціативних правил.

А.Г. Кривохата, О.В. Кудін, М.В. Давидовський, А.О. Лісняк [128] описали принципи застосування ансамблевого навчання в задачах класифікації акустичних даних.

Д.В. Ланде, І.Ю. Субач, Ю.Є. Бояринова [129] розглядають базові питання теорії і практики інтелектуального аналізу даних: алгоритми, моделі, задачі класифікації, кластерного аналізу, пошуку, глибинного аналізу даних (Data Mining), теорії складних мереж (Complex Networks), а також приводяться відомості, необхідні для математичного і комп'ютерного моделювання та аналізу складних систем і мереж в сфері кібербезпеки.

Також варто відмітити роботи так вчених як: М. М. Шаркаді, М. В. Роботишин, М. М. Маляр, М.С. Лавренюк та О.М. Новіков, А. Ю. Тітова, Д.Є. Іванов, Л.В. Зубик, A.Y. Gladun, Y. V. Rogushena, P. Ghamisi, J. Plaza, Y. Chen, J. Li, A. J. Plaza, James D. Miller, T. Chen, C. Guestrin та інших.

Структурні моделі сьогодення включають кілька напрямків представлення даних, найбільш популярною є модель на основі штучних нейронних мереж (artificial neural network, ANN).

1.2 Аналіз поняття «штучна нейронна мережа»

Структурні моделі сьогодення включають кілька напрямків представлення даних, найбільш популярною є модель на основі штучних нейронних мереж (artificial neural network, ANN) [4].

Ідею штучної нейронної мережі (ШНМ) вперше описали математик Уолтер Піттс і нейропсихолог Уоррен Маккалок в 1943 році [5]. ШНМ – це обчислювальна модель з величезним числом паралельно діючих процесів, об'єднаних безліччю зв'язків. Її вузли – нейрони розподіляються в мережі за рівнями – шарами. Спочатку інформація надходить на нейрони вхідного шару. Нейрони вхідного шару приймають цей сигнал і передають його далі нейронам прихованого або внутрішнього шару. У прихованому шарі відбувається основна обробка даних. Після чого інформація відправляється в останній шар – вихідний. Число прихованих шарів і кількість нейронів вибирається залежно від завдання, яке вирішує мережу обсягу даних і доступних обчислювальних ресурсів.

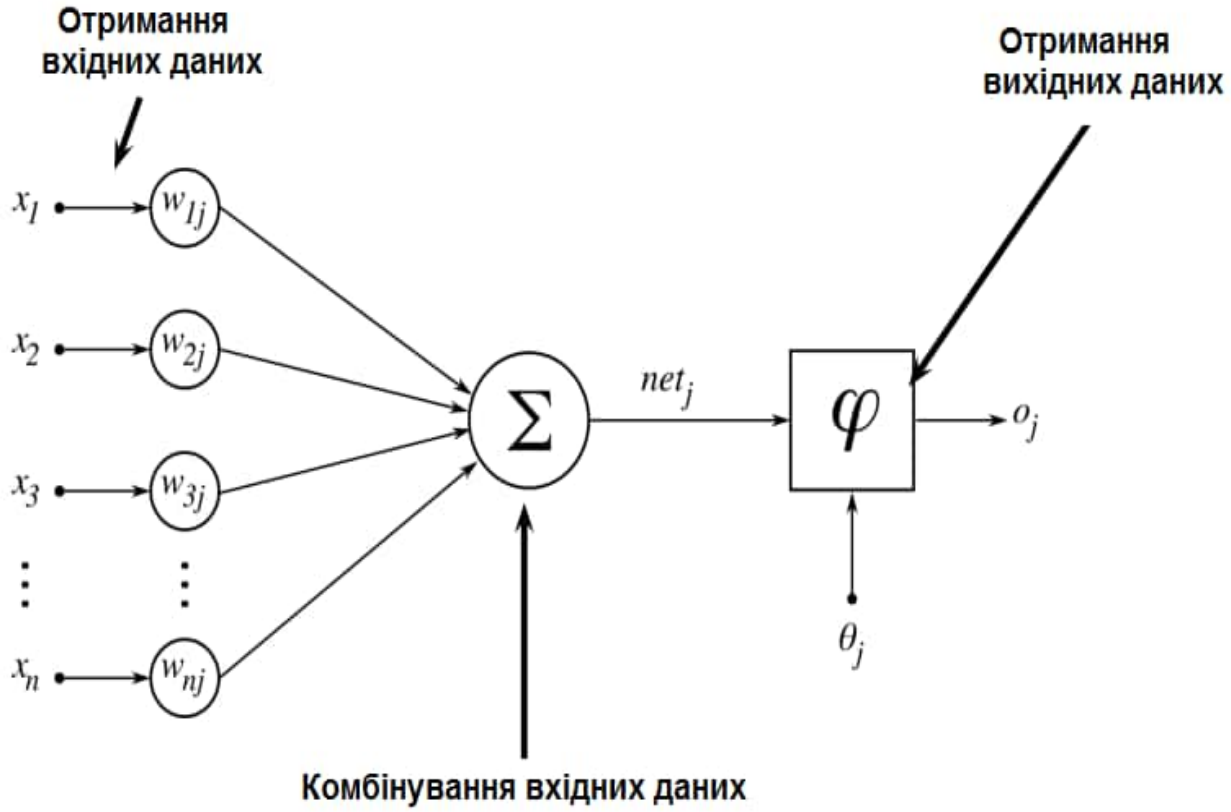


Рисунок 1.1 – Модель нейронної мережі [6, 7]

Модель нейронної мережі наведено на рис. 1.1.

Модель визначення нейрону:

$$S = \sum_i w_i x_i \qquad (1.1)$$

Структура тришарової нейронної мережі виглядає наступним чином (рис. 1.2).

Найбільш поширений спосіб отримання нейрону S через функцію активації:

$$A(x) = \frac{2}{1+e^{-x}} - 1 \qquad (1.2)$$

Вихід вузла k у другому шарі:

$$O_k = A\left(\sum_i w_{ik}^{1\to 2} x_i\right) \tag{1.3}$$

Вихід вузла j у вихідному шарі:

$$O_j = A\left(\sum_k w_{kj}^{2\to 3} x_k\right) \tag{1.4}$$

$$O_j = A\left[\sum_k w_{kj}^{2\to 3} A\left(\sum_i w_{ik}^{1\to 2} x_i\right)\right] \tag{1.5}$$

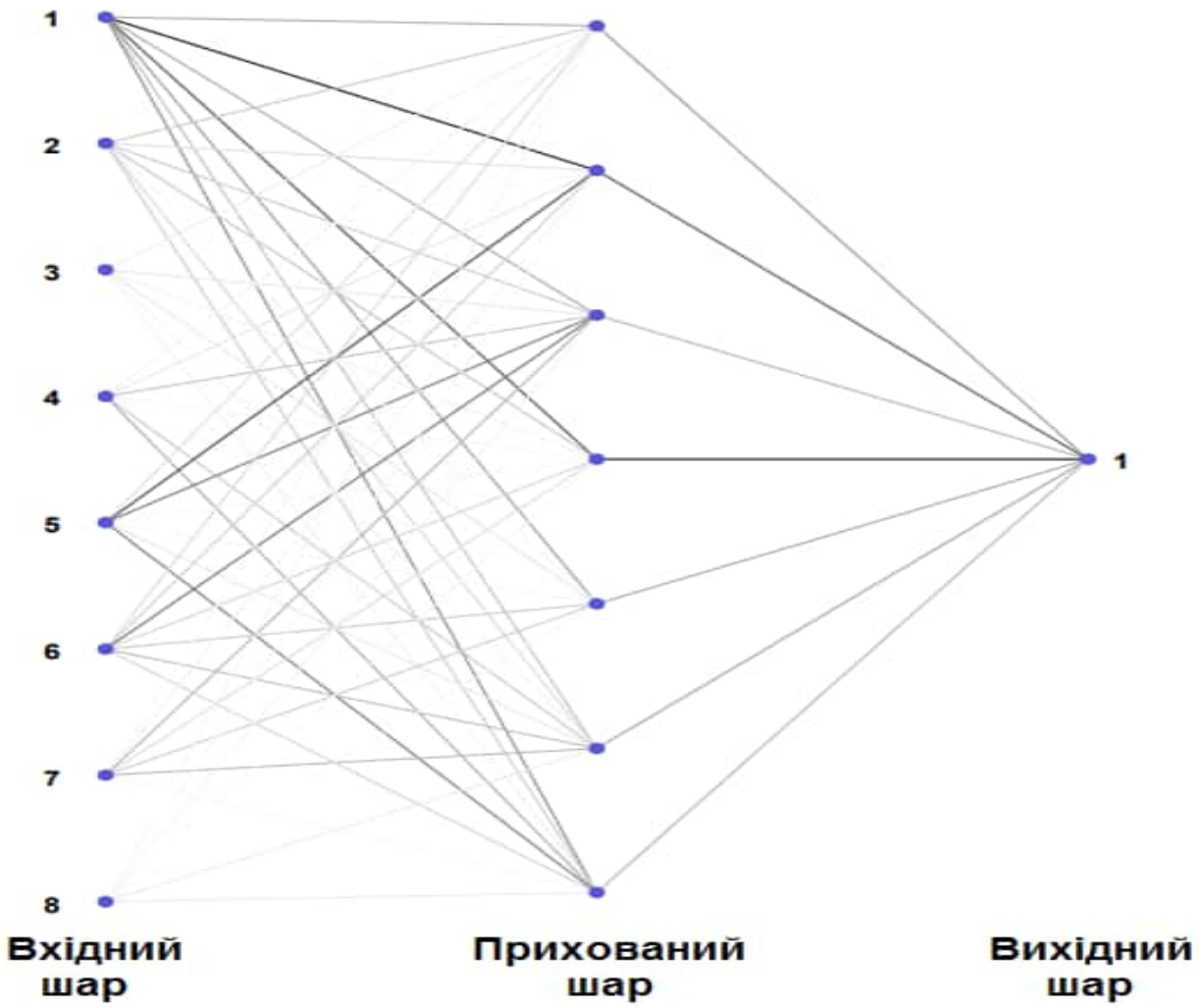


Рисунок 1.2 – Структура тришарової нейронної мережі [8, 9]

На рис. 1.2 наведено структуру тришарової нейронної мережі прямого поширення. При виконанні даної мережі сигнал проходить пошарово в одному напрямку – від входу до виходу. На першому вхідному шарі нейрони не змінюють сигнал, а поширюють його на нейрони другого прихованого шару, на якому сигнали оброблюються та формується вихідний нейрон.

У випадку використання більшої кількості шарів, логіка викладення аналогічна. Всі знання зберігаються у вагах.

Наступним кроком є навчання нейромережі. Навчання нейромережі – це процес, заснований на відомих прикладах подій, які знаходять для набору ваг, та які мінімізують функцію втрат [10]. Навчання є найголовнішим процесом у використанні нейронних мереж, так як саме від навчання найбільшою мірою залежить кінцевий результат.

Перш ніж використовувати нейронну мережу для планованого завдання, потрібно знайти найкращий набір ваг. Потрібна деяка міра [11], яка показує, який набір ваг найкраще виконує функцію втрат.

В якості функції втрати для нейронних мереж зазвичай використовується середньоквадратична помилка.

Квадратична функція втрат:

$$\chi^2 = \sum_j w_j x_j^2 = \sum_j \frac{1}{2} \sum_i \left(T_{ji} - O_{ji} \right)^2 \qquad (1.6)$$

де T_{ji} - вихід мережі номер j;

O_{ji} – вірна відповідь мережі номер j

Але середньоквадратична помилка це не єдиний варіант, також є тенденція до використання середньої крос-ентропії з усіх навчальних вибірок.

Функція ентропії:

$$E_D = \sum_j w_j E_D^j = \sum_j \sum_i - \log\left(\frac{1}{2}\left(1 + T_{ji} O_{ji}\right)\right) \qquad (1.7)$$

Таким чином, функція ентропії дає підстави формально окреслити задачу навчання класифікатора: процедура навчання нейронної мережі це мінімізація функції втрати у просторі ваг.

Класифікацію у найпростішому випадку можна представити як подію, коли потрібен окремий вихідний вузол, за умови якщо вихід вище обраного значення, це клас 1 (сигнал), інакше клас 2 (фон).

Визначення величини значення умовної щільності ймовірності відбувається за умови наявності багатьох вузлів у вихідному шарі, кожен з яких відповідає на запитання, чи ймовірність більша за граничну величину. Вихідний вектор представляє собою інтегровану щільність імовірності.

На основі вищевикладеного, вірні твердження, що штучна нейронна мережа здатна здійснити моделювання нелінійної залежності майбутнього значення зовнішніх факторів.

Однак у випадку перекладу тесту ШНМ, навчена на одній ділянці ряду, у більшості випадках не дозволяє одержувати задовільні результати прогнозування на іншій ділянці, де характер тексту змінився. Таким чином обов’язково після кожної зміни структури текстового ряду потрібно нове навчання нейронної мережі.

Ще одним істотним недоліків ШНМ є форма інтерпретації отриманих результатів, вона є незрозумілою людині, однак цей істотний недолік відсутній в апараті нечіткої логіки.

1.3 Принципи формування глибоких моделей

Біологічна нейронна мережа – це мережа взаємозалежних нейронів. Кожен нейрон має так звані дендрити, які збирають інформацію з довкілля. Інформація надходить у нейрон у вигляді електричних/хімічних сигналів. Як

тільки нейрон отримує сигнал, він обробляє сигнал і, якщо досягає певного порога, випромінює вихідний сигнал через аксон, який підключений до наступного нейрону. Наступний нейрон після отримання сигналу робить те саме, і процес триває.

Штучна нейронна мережа (ANN) неясно натхнена біологічною нейронною мережею. Це набір пов'язаних штучних нейронів. Як і біологічний нейрон, штучний нейрон також приймає вхідні дані від одного нейрона, виконує деякі обчислення та передає сигнал іншому нейрону, який до нього підключений.

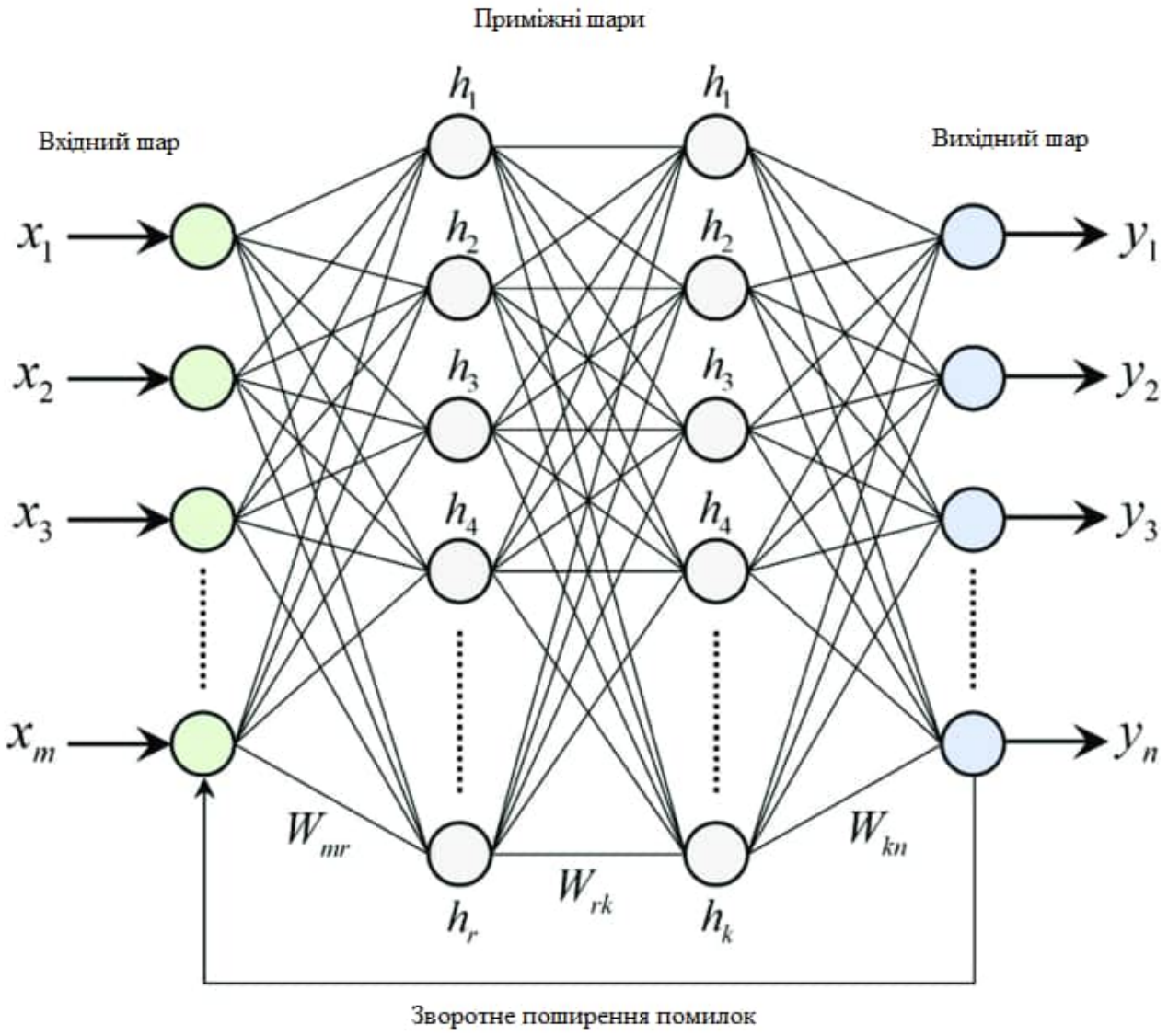


Рисунок 1.3 – Глибока нейронна мережа

Глибока нейронна мережа (DNN) – це штучна нейронна мережа з кількома рівнями між вхідним та вихідним рівнями. Кожен нейрон на одному шарі з'єднується з усіма нейронами наступного шару. Один або кілька шарів між вхідним та вихідним шарами називаються прихованими шарами.

Кожна сполука, яка з'єднує нейрон з одного шару з нейроном у попередньому шарі, має так звану вагу w, яка говорить про те, наскільки чутлива активація поточного нейрона до активації нейрона в попередньому шарі. Кожен нейрон у цьому шарі має щось, зване зміщенням b. Якщо пов’язувати з лінійною регресією то, член зміщення діє як перехоплювач c у $y = mx + c$. Якщо сума (mx) не перетинає поріг, але нейрон повинен активуватися, усунення буде скориговано, щоб знизити поріг цього нейрона, щоб змусити його спрацювати.

З рисунку 1.3 видно, що глибока нейронна мережа це функція $y = f(x)$, де x – це вхід, y – вихід. Усередині функції $f(x)$ вона викликає ланцюжок функцій, у якій виведення однієї функції передається іншою. Ці внутрішні функції – це приховані шари.

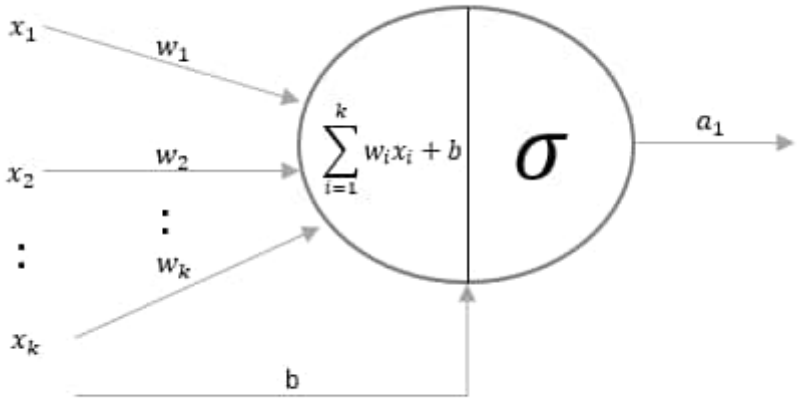


Рисунок 1.4 – Один штучний нейрон

Тепер збільшимо масштаб до одного штучного нейрона. Штучний нейрон складається із двох частин. У першій частині він бере вхідні дані з попереднього шару, що відповідають вазі та зсуву, а потім виконує лінійне

перетворення цих даних. Лінійне перетворення – це не що інше, як сума зважених вхідних даних та усунення.

Лінійне перетворення входів:

$$z = w_1 x_1 + w_2 x_2 + w_3 x_3 + \ldots + w_k x_k + b$$

$$z = \sum_{i=1}^{k} w_i x_i + b$$

У другій частині він перетворює це лінійне перетворення на нелінійне перетворення, використовуючи функцію активації, таку як сигмоїд, і випускає вихідний сигнал функції активації. Є й інші функції активації, такі як ReLu. Через цю комбінацію лінійних та нелінійних перетворень разом з кількома рівнями глибока нейронна мережа стає настільки потужною, що вона може обробляти будь-які складні дані.

Функція активації:

$$a = \sigma\left(\sum_{i=1}^{k} w_i x_i + b\right)$$

Сигмоїдна функція приймає виважену суму і перетворює значення від 0 до 1. Вона перетворює -infinity на 0 і + infinity на 1. Значення між 0 і 1 являє собою силу активації конкретного нейрона.

Сигмоподібна функція

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

Активацію нейрона на цьому шарі можна записати так:

$$a^L = \sigma\left(\sum_{i=1}^{k} w_i^L x_i^{L-1} + b^L\right)$$

У типовій нейронній мережі матимемо більше одного нейрона в даному шарі. Вищезгадане рівняння може бути представлене у вигляді матриці, що включає всі нейрони.

$$\left[a_1^L\ a_2^L\ a_3^L\ldots\ a_k^L\right]=\sigma\left(\left[w_{11}^L\ w_{12}^L\ w_{13}^L\ldots\ w_{1j}^L\ w_{21}^L\ w_{22}^L\ w_{23}^L\ldots\ w_{2j}^L\ w_{31}^L\ w_{32}^L\ w_{33}^L\ldots\ w_{3j}^L\ldots\ldots\right.\right.$$

Матрична форма активації лише на рівні L

Глибока нейронна мережа сама навчатиметься на наданих даних і використовуватиметься для прогнозування невидимих даних.

DNN має набір ваг та зміщень на кожному рівні. Активація нейрона залежить від відповідних ваг та зсувів. Таким чином, навчання на основі даних означає визначення найкращих ваг та зсувів мережі.

Щоб знайти ваги та зсунення, глибока нейронна мережа робить наступне:

1. Надає вагам та зміщенням деякі випадкові значення
2. Запускає навчальні дані (які мають вхідні та фактичні виходи) у мережі, використовуючи ці випадково призначені ваги та усунення. Під час цього вихідні дані функції активації в одному шарі будуть передаватися як вхідні дані на наступний рівень, поки не буде отримано вихідні дані з вихідного шару. Цей процес називається прямим розповсюдженням.
3. Початковий висновок з мережі завжди буде не точним та масштабним, оскільки використано випадкові ваги та зсуви. Необхідно обчислити помилку (різницю між передбаченням мережі та фактичним виходом), використовуючи якусь функцію вартості чи помилки.

Сума квадратів помилок

$$C=\sum_{i=1}^{n}\left(\hat{y}-y\right)^2$$

4. Оскільки всі нейрони в мережі зробили свій внесок у помилку, вказану вище, пропорція помилки (градієнт помилки) буде передана назад з вихідного шару у всі шари, за винятком вхідного, щоб можна було регулювати ваги та зміщення. Цей процес поширення помилки для коригування ваги і зміщень називається зворотним поширенням.

Оскільки функція вартості є функцією ваги і зміщення, градієнт помилки буде обчислюватися з використанням приватних похідних функції вартості ваги і зсувів.

Щоб краще зрозуміти, візьмемо просту мережу з одним вхідним шаром, одним прихованим шаром та одним вихідним шаром. Після першого проходу прямого розповсюдження матимемо помилку. Тепер потрібно передати пропорцію помилки всім нейронам у всіх шарах.

Спочатку обчислимо градієнт помилки для невеликої зміни ваги та зміщень на вихідному шарі. Для простоти запишемо функцію активації як функції.

$$z_0^L = \sum w_0^L a_0^{L-1} + b_0^L$$

$$a_0^L = \sigma\left(z_0^L\right)$$

$$C = \sum \left(a_0^L - y\right)^2$$

Знаходимо приватні похідні функції вартості C як за вагою, так і по зміщенню. Використовуючи правило ланцюга, похідну приватну від C і b можна записати наступним чином. Правило ланцюга

$$\frac{\partial C}{\partial w_0^L} = \frac{\partial C}{\partial a_0^L} \bullet \frac{\partial a_0^L}{\partial z_0^L} \bullet \frac{\partial z_0^L}{\partial w_0^L}$$

$$\frac{\partial C}{\partial b_0^L} = \frac{\partial C}{\partial a_0^L} \bullet \frac{\partial a_0^L}{\partial z_0^L} \bullet \frac{\partial z_0^L}{\partial b_0^L}$$

Приватні похідні кожного компонента у наведеному вище рівнянні рівні

$$\frac{\partial C}{\partial a_0^L} = \frac{\partial \left(a_0^L - y\right)^2}{\partial a_0^L} = 2\left(a_0^L - y\right)$$

$$\frac{\partial a_0^L}{\partial z_0^L} = \sigma'\left(z_0^L\right)$$

$$\frac{\partial z_0^L}{\partial w_0^L} = a_0^{L-1}$$

$$\frac{\partial z_0^L}{\partial b_0^L} = 1$$

Підставляючи зазначені вище значення приватних похідних рівняння правила ланцюга, градієнти помилок на вихідному шарі щодо ваг і зміщень рівні

$$\frac{\partial C}{\partial w_0^L} = 2\sum\left(a_0^L - y\right)\sigma'\left(z_0^L\right)\left(a_0^{L-1}\right)$$

$$\frac{\partial C}{\partial b_0^L} = 2\sum\left(a_0^L - y\right)\sigma'\left(z_0^L\right)$$

Для прихованого шару

$$z_h^{L-1} = \sum w_h^{L-1} x + b_h^{L-1}$$

$$a_h^{L-1} = \sigma\left(z_h^{L-1}\right)$$

$$a_0^L = \sigma\left(\sum w_0^L a_h^{L-1} + b_0^L\right)$$

$$\frac{\partial C}{\partial w_h^{L-1}} = \frac{\partial C}{\partial a_0^L} \bullet \frac{\partial a_0^L}{\partial z_0^L} \bullet \frac{\partial z_0^L}{\partial a_h^{L-1}} \bullet \frac{\partial a_h^{L-1}}{\partial z_h^{L-1}} \bullet \frac{\partial z_h^{L-1}}{\partial w_h^{L-1}}$$

$$\frac{\partial C}{\partial b_h^{L-1}} = \frac{\partial C}{\partial a_0^L} \bullet \frac{\partial a_0^L}{\partial z_0^L} \bullet \frac{\partial z_0^L}{\partial a_h^{L-1}} \bullet \frac{\partial a_h^{L-1}}{\partial z_h^{L-1}} \bullet \frac{\partial z_h^{L-1}}{\partial b_h^{L-1}}$$

$$\frac{\partial C}{\partial w_h^{L-1}} = 2\sum\left(a_0^L - y\right)\sigma'\left(z_0^L\right)\left(w_0^L\right)\left(\sigma'\left(z_0^{L-1}\right)\right)(x)$$

$$\frac{\partial C}{\partial b_h^{L-1}} = 2\sum\left(a_0^L - y\right)\sigma'\left(z_0^L\right)\left(w_0^L\right)\left(\sigma'\left(z_0^{L-1}\right)\right)$$

Примітка. Хоча L і L-1 представляють вихідний шар і прихований шар відповідно, використовуємо суб-нотацію «o» для виведення та «h» для прихованого шару, щоб бути чіткішим.

Так само можемо обчислити градієнт помилки на всіх прихованих шарах, якщо у нас їх більше одного. Оскільки маємо лише один прихований шар, зворотне поширення тут зупиняється.

5. Вищезгадане пряме і зворотне поширення буде виконуватися ітеративно, а ваги та зміщення коригуватимуться доти, доки не знайдемо оптимальні значення. Використовуватимемо алгоритм градієнтного спуску.

Градієнтний спуск є ітеративним методом оптимізації, що може знайти мінімум як функції. Він використовується, коли пошук оптимальних значень параметрів функції затруджений.

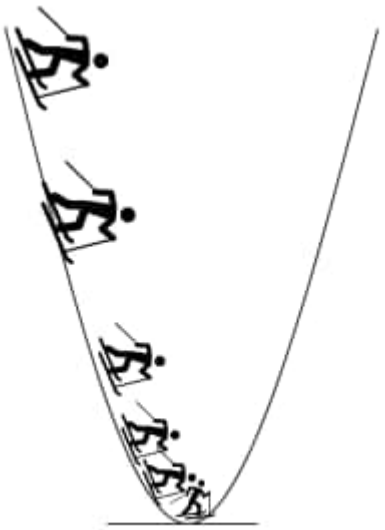


Рисунок 1.5 – Градієнтний спуск

Мета – знайти оптимальні значення ваги та зміщення, при яких функція вартості буде мінімальною. Наступні кроки, які використовують у алгоритмі градієнтного спуску.

1. Призначення випадкових значень для ваг **w** та зсувів **b** та постійне значення для швидкості навчання

2. Оновлення ваги та зсуву, використовуючи градієнт та швидкість навчання.

3. Повторення кроку 2, доки не буде знайдено мінімальне значення або досягнено максимальної кількості ітерацій.

1.4 Проблематика по постановка завдань дослідження

Нейронні мережі не завжди працюють так, як заплановано. Необхідно спланувати тренувальні дані та початкові значення ваги, а також спланувати вихідні значення. Найпоширенішою проблемою є насичення мережі. Воно виникає, якщо є великі значення сигналів, часто спровоковані великими початковими ваговими коефіцієнтами. Таким чином, сигнали потраплять в область близьких до нуля градієнту функції активації. Що, у свою чергу, впливає на здатність до навчання, а саме на підбір кращих коефіцієнтів.

1) Вхідні значення

Якщо використовувати у якості функції активації – сигмоїду, то при занадто великих значеннях вхідних даних, пряма буде виглядати, як пряма. Тому рекомендується задавати невеликі значення. Однак надто маленькі значення також погано позначатимуться на навчанні, оскільки точність комп'ютерних обчислень знижується. Тому рекомендується вибирати значення вхідних даних від 0.0 до 1.0. При цьому можна ввести зсув 0.01. [12].

На рисунку 1.6 видно, що зі збільшенням вхідних даних, здатність нейронної мережі до навчання знижується, оскільки сигмоїда майже випрямляється.

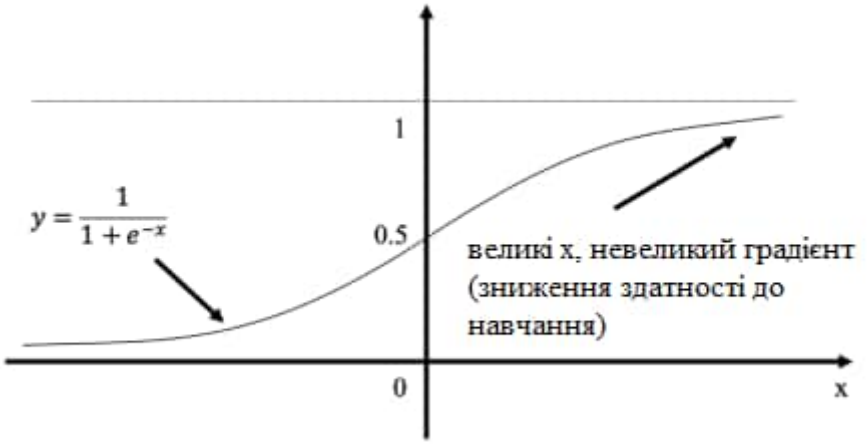


Рисунок 1.6 – Підготовка вхідних даних

2) Вихідні значення

Вихідні значення слід вибирати, залежно від обраної функції активації. Якщо вона не здатна забезпечувати значення понад 1.0, але вихідні значення необхідно отримати більше 1.0, то вагові коефіцієнти збільшуватимуться, щоб підлаштуватися під ситуацію. Але нічого не вийде, вихідні значення все одно не будуть більшими за максимальне значення функції активації. Тому вихідні значення слід масштабувати від 0.0 до 1.0. Так як граничні значення не досягаються, то рекомендується вибирати значення від 0.01 до 0.99. На рис. 1.7 продемонстровано це правило.

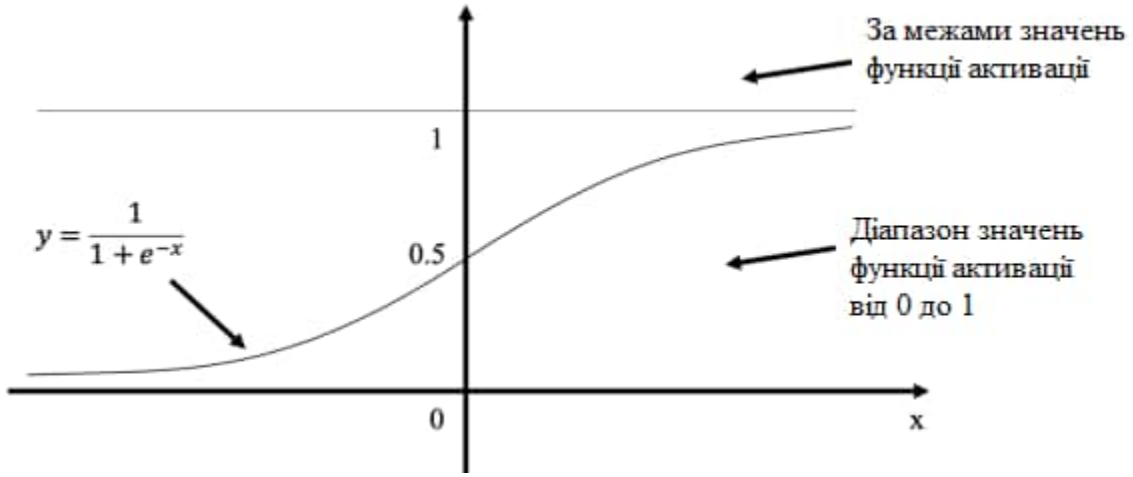


Рисунок 1.7 – Обмеження за вихідними значеннями

3) Випадкові початкові значення вагових коефіцієнтів

Найпростіший варіант у виборі початкових значень вагових коефіцієнтів – вибирати їх із діапазону від -1.0 до +1.0.

Але є підходи, які дозволяють визначити коефіцієнти залежно від конфігурації мережі. Мета полягає в тому, щоб якщо на вузол мережі надходить безліч сигналів та їх поведінка відома, то вагові коефіцієнти не повинні порушувати їхній стан. Тобто вага не повинна порушувати ретельну підготовку вхідних та вихідних значень, описаних у пунктах 1 та 2.

Якщо грубо описати правило, воно звучить так: «Вагові коефіцієнти ініціалізуються числами, що випадково вибираються з діапазону, які визначаються зворотною величиною квадратного кореня з кількості зв'язків, що ведуть до вузла» [14].

На рис. 1.8 ілюструються підходи вибору початкових ваг.

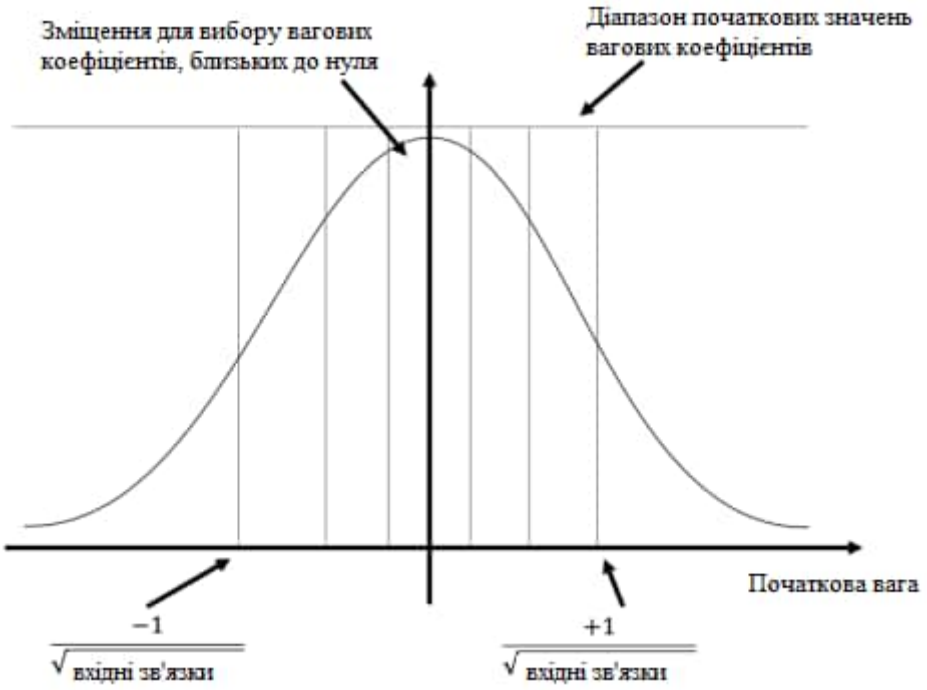


Рисунок 1.8 – Підходи до вибору вагових коефіцієнтів

Непотрібно ставити однакову вагу. Таким чином у вузли прийшли б однакові сигнали і вихідні значення вийшли б однаковими. І після оновлення ваг, їх значення однаково будуть рівними. Також не можна задавати нульові значення для вагових коефіцієнтів, оскільки вхідні значення в цьому випадку втрачають свою силу.

Щоб нейронні мережі працювали задовільно, необхідно вхідні та вихідні дані, а також початкові значення вагових коефіцієнтів задавати в залежності від структури нейронної мережі. Також перешкодою для найкращого навчання мережі є нульові значення сигналів та ваг. А значення вагових коефіцієнтів повинні відрізнятись один від одного.

Вхідні та вихідні значення мають бути масштабованими.

Метою даної роботи є оптимізація в навчанні глибоких моделей.

Для досягнення поставленої мети у роботі необхідно виконати низку завдань:

- розкрити методи навчання глибоких моделей;
- окреслити математичні аспекти навчання глибоких моделей;
- запропонувати алгоритм реалізації навчання глибоких моделей;

- розробити структуру системи розпізнавання об’єктів на базі штучної нейронної мережі;
- здійснити навчання штучної нейронної мережі;
- виконати реалізацію системи розпізнавання об’єктів на базі штучної нейронної мережі;
- здійснити верифікацію результатів дослідження.

Висновки до розділу

У рамках першого розділу здійснено дослідження поняття «нейронної мережі», розкрито основні функціональні принципи. Навчання нейронних мереж складне завдання по ряду причин. Нерідко процес пошуку адекватної нейромережевої моделі закінчується з нульовим результатом і дуже велику роль грає досвід розробника нейромережевих моделей. Для отримання нейромережевої моделі вирішальним завдання із заданим показником якості зазвичай необхідно пройти наступні кроки: спочатку необхідно підготувати дані, визначитися з типом мережі, визначити входи і виходи, вирішити задачу первісної структури мережі – шари і нейрони в них, далі необхідно навчити мережу, тобто підібрати коефіцієнти зв'язків між нейронами, перевірити навчену мережу на валідаційній вибірці і у підсумку перевірити в реальній роботі. При цьому всі кроки тісно пов'язані між собою і неякісне опрацювання по одному з них веде, в кінцевому рахунку, до тривалого навчання мережі або взагалі до отримання неправильно працюючої нейромережі. Існує велика кількість методів і алгоритмів попередньої підготовки даних, але всі вони в значній мірі спираються на досвід розробника.

РОЗДІЛ 2
МЕТОДИ ТА МОДЕЛІ НАВЧАННЯ ГЛИБОКИХ МОДЕЛЕЙ

2.1 Методи навчання глибоких моделей

Навчання нейронної мережі – це процес визначення ваги сполук між нейронами таким чином, щоб мережа наближала необхідну функцію із заданою точністю. Існує три підходи до навчання нейронних мереж [3]: навчання з вчителем (supervised learning), навчання без вчителя (unsupervised learning) та навчання з підкріпленням (reinforcement learning). Під час навчання з учителем на вхід мережі подаються набори вхідних сигналів (об'єктів), для яких заздалегідь відома правильна відповідь (навчальна множина). Ваги змінюються за певними правилами залежно від того, чи правильний вихідний сигнал видала мережа. Під час навчання без вчителя на вхід мережі подаються об'єкти, для яких правильний вихідний сигнал наперед не відомий. Навчання з підкріпленням передбачає наявність зовнішнього середовища, з яким взаємодіє мережа. Навчання відбувається на підставі сигналів, отриманих від цього середовища.

Нейронні мережі МакКаллока-Піттса не навчалися. Ваги для всіх входів нейронів мали бути задані заздалегідь.

Вперше ідею навчання нейронних мереж запропонував Дональд Хебб (Donald Hebb) у 1949 році [19]. Згідно з Хеббом, зв'язки нейронів, які активуються разом, повинні посилюватися, а зв'язки нейронів, які спрацьовують окремо один від одного, повинні слабшати. Хебб запропонував правила зміни ваги вхідних сигналів нейронів відповідно до того, правильну відповідь видавала мережа, чи ні [19] (навчання з учителем). А.В. Новіков довів збіжність запропонованого методу навчання нейрона на основі правил Хебба [20], за умови, що вибірка об'єктів лінійно роздільна. Згодом було запропоновано кілька аналогічних правил як для навчання з учителем [21-23], так і без учителя [24-28].

1970 року А.Г. Івахненко розробив метод групового обліку аргументів (group method of data handling) [29, 30], що дозволяє не лише обчислювати ваги зв'язків між нейронами, а й визначати кількість шарів у мережі та нейронів у них залежно від потреб прикладного завдання. Використовуючи підхід навчання з учителем, рівні мережі інкрементально будуються та навчаються на основі навчальної множини з використанням регресійного аналізу. Потім відбувається етап спрощення мережі із застосуванням окремої множини об'єктів з відомими правильними відповідями, яке не використовувалося при навчанні (перевірочна множина, validation set). Для виключення непотрібних нейронів із мережі використовується регуляризація. У роботі [30] описано застосування методу групового обліку аргументів на навчання глибокої нейронної мережі, що складається з восьми шарів. Метод широко використовувався практично [31–33].

В даний час для навчання нейронних мереж, у тому числі глибоких, використовується алгоритм зворотного розповсюдження помилки (error backpropagation algorithm), що ґрунтується на методі градієнтного спуску. Алгоритм було запропоновано в 1970 році в магістерській дисертації [34, 35] без зв'язку з нейронними мережами. Перше застосування цього алгоритму на навчання нейронних мереж описано у роботі [36], що вийшла 1981 року. Після цього з'явилося ще кілька робіт з цієї теми [37–39]. Алгоритм зворотного поширення помилки використовує навчання з учителем, для нього потрібна навчальна множина із заздалегідь відомими правильними відповідями. Вводиться міра помилки, яка визначає, наскільки вихідні значення мережі відрізняються від правильних відповідей. Потім міра помилки мінімізується за допомогою методу градієнтного спуску шляхом зміни значень ваг у мережі. Щоб оцінити, наскільки сильно кожна вага впливає на вихідне значення, розраховуються приватні похідні помилки по вагам. Потім проводиться зміна ваги на невеликі значення з урахуванням градієнта. Так повторюється доти, доки помилка на виході не скоротиться до допустимих значень. Початкові значення ваг нейронів у мережі задаються випадковим чином.

У глибокій нейронній мережі з декількома прихованими шарами проводиться розрахунок помилки, що передається від одного шару до іншого. У першому етапі розраховується значення помилки на виході нейронної мережі, де відомі правильні відповіді. Потім розраховується помилка на вході у вихідний шар мережі, яка використовуватиметься як помилка на виході прихованого шару. У такий спосіб розрахунок триває доти, коли буде відома помилка на вхідному шарі. Саме тому алгоритм має назву зворотне поширення помилки.

Можливо кілька варіантів реалізації навчання нейронних мереж за допомогою алгоритму зворотного розповсюдження помилки. При повному навчанні градієнт розраховується всіма об'єктами навчальної множини. Однак такий підхід часто не є ефективним у випадку, коли навчальна множина велика і для обробки всіх його елементів потрібен значний час. Альтернативний варіант – використання методу стохастичного градієнтного спуску, при якому ваги змінюються при обробці одного елемента навчальної множини (онлайн-навчання) або декількох елементів (навчання на пакетах або міні-вибірках). На практиці для навчання нейронних мереж найчастіше використовується саме метод стохастичного градієнтного спуску або його модифікації [40-42].

Нейронна мережа показана на рис. 1.3, називається повнозв'язковою. У такій мережі кожен нейрон наступного шару пов'язаний із усіма нейронами попереднього шару. Однак це не єдиний варіант з'єднання нейронів у мережу.

У 1980 Куніхіко Фукушима (Kunihiko Fukushima) запропонував архітектуру нейронної мережі, яка називається неокогнітрон [43]. Архітектура використовувала аналогію зі складними та простими клітинами в зоровій корі кішки [44]. Прості клітини спрацьовують у відповідь на прості візуальні сигнали, такі як орієнтація кордонів. Складні клітини менш залежні від просторового розташування сигналів і орієнтуються на більш загальні ознаки. У неокогнітроні простим клітинам відповідають згорткові шари (convolutional layers), а складним – шари підвиборки (subsampling layers). У згортувальних

шарах вікно згорткового вузла (convolutional unit) із заданим набором ваг (ядро згортки) переміщається по двовимірному масиву вхідних даних, наприклад, пікселям зображення (рис. 2.1). Значення елементів даних, що збігаються і ядро згортки перемножуються, отримані результати складаються і надходять у нейрон наступного шару. Всі згорткові вузли використовують однакові ядра згортки, тому для опису згорткової мережі потрібно трохи параметрів. Як правило, у згорткових шарах використовується не одне, а кілька ядер згортки.

Виходи згорткових шарів на неокогнітроні підключаються до входів шарів підвиборки. Причому до одного нейрону в шарі підвиборки підключаються кілька нейронів згорткового шару, зазвичай із квадратної області розміром 2 * 2 чи більше. Нейрони у шарі підвиборки спрацьовують у разі активності хоча б одного із вхідних сигналів. На цьому шарі важлива наявність самого сигналу, а не його конкретні координати, тому шари підвиборки менш чутливі до незначних зрушень та змін масштабу зображення. Навчання згорткових шарів у неокогнітроні проводиться за допомогою локальних алгоритмів навчання без вчителя, або ваги задаються заздалегідь [45, 46]. Для шарів підвиборки використовується просторове усереднення (spatial averaging) [43, 47]. Таким чином, незважаючи на те, що неокогнітрон є глибокою нейронною мережею, глибоке навчання в ньому не використовується..

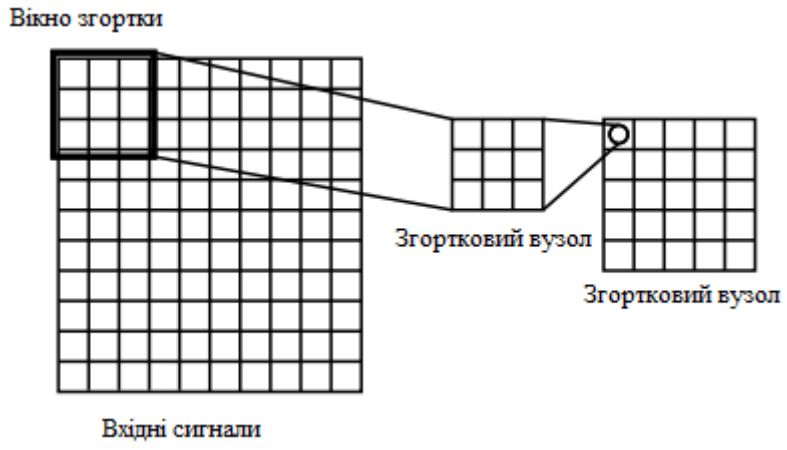


Рисунок 2.1 – Схема згорткового шару нейронної мережі неокогнітрон

В 1987 Дана Баллард (Dana Ballard) запропонував підхід до навчання нейронних мереж без вчителя на основі автокодувальника (autoencoder) [48]. Простий автокодувальник містить лише один прихований шар (рис. 2.2) з кодом h, який служить для представлення вхідного сигналу x. Автокодувальник містить функцію кодування f, яка використовується для перетворення вхідного сигналу код $h = f(x)$ і функцію декодування g, яка по коду відновлює значення вхідних сигналів $r = g(x)$.

Рисунок 2.2 – Схема простого автокодувальника

Автокодувальники застосовуються для зменшення розмірності оброблюваних даних. Для цього використовуються лінійні методи, такі як метод основних компонентів. За рахунок зниження розмірності автокодувальник виділяє найбільш значні характеристики даних.

Для навчання автокодувальників використовують метод рециркуляції [49]. Автокодувальник навчається таким чином, щоб на його виході були ті самі сигнали, як і на вході. Після навчання прихований шар простого автокодувальника вставляється в автокодувальник вищого рівня, який містить більше прихованих шарів. Таким чином, будується ієрархія автокодувальників у вигляді стека (рис. 2.3). У цьому розмірність даних зменшується на кожному рівні ієрархії. Подібна ієрархія може бути використана не тільки для автокодувальників, але й для інших методів навчання без вчителя [50, 51].

Ян Лекун (Yann LeCun) в 1989 застосував алгоритм зворотного поширення помилки для навчання мережі з архітектурою, дуже схожою на неокогнітрон [52]. Мережа містила згорткові шари з однаковими вагами (згортковими ядрами) та шари підвибірки. У цій роботі був представлений набір

даних MNIST, що містить рукописні цифри, розпізнавання яких згодом стало дуже популярним тестом в машинному навчанні. Глибокі згорткові мережі, навчені алгоритмом зворотного поширення помилки, показали хороші результати на розпізнаванні рукописних цифр [53] та відбитків пальців [54].

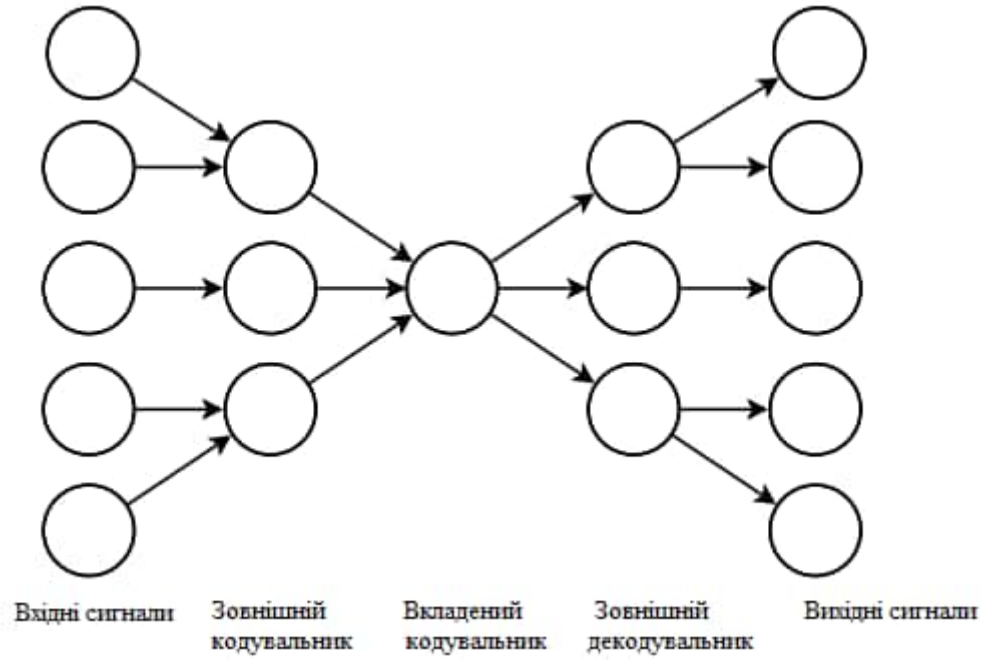


Рисунок 2.3 – Ієрархія автокодувальників

Наприкінці 80-х років XX століття стало зрозуміло, що одного алгоритму зворотного розповсюдження помилки замало ефективного глибокого навчання. Незважаючи на деякі успішні приклади [53, 54], використання більше одного прихованого шару рідко давало переваги на практиці [55-57]. Причина цього була сформульована 1991 року у роботі [58, 59] – проблема зникаючого градієнта (vanishing gradient problem). При використанні традиційних функцій активації, сигнали про поширювані помилки швидко стають дуже маленькими (або навпаки, надмірно великими). У практичних завданнях вони зменшуються експоненційно з кількістю шарів в мережі. Ця проблема також відома як проблема тривалої затримки (long time lag problem) [60, 61].

Один із підходів до вирішення проблеми зникаючого градієнта полягає у повній відмові від використання градієнта для навчання. Для деяких завдань

хороших результатів можна домогтися призначенням ваги випадковим чином [62]. Підхід до навчання Еволіно (Evoli№) [63] використовує лінійні методи визначення оптимальних ваг для вихідного шару і еволюцію визначення ваг прихованих шарів. Також можливе застосування методів універсального пошуку [64, 65].

Альтернативний підхід до вирішення проблеми градієнта, що зникає, – використання без гесіанної оптимізації (Hessian-free optimization) [66–70].

Дуже глибокий учитель, запропонований у роботі 1992 [71], забезпечує вирішення проблеми зникаючого градієнта та навчання мережі глибиною до сотень шарів за рахунок використання попереднього навчання без вчителя ієрархії рекурентних нейронних мереж. Кожна рекурентна мережа навчається окремо для того, щоб передбачити наступне значення, яке надійде на вхід [72, 73]. Після навчання лише помилково передбачені значення передаються у більш високий рівень мережі. Ця мережа працює вже на повільнішій часовій шкалі, за рахунок чого інформація стискається і кожній послідовності сигналів відповідає набір все менш і менш надлишкового кодування на більш глибоких рівнях мережі. Інша назва такої архітектури нейронної мережі – компресор історії (History Compressor), вона може стискати дані як у просторі, так і в часі. Існує також безперервний варіант компресора історії [74].

Проблему градієнта, що зникає, дозволяє вирішити інша архітектура рекурентної нейронної мережі – мережа довготривалої пам'яті (Long Short-Term Memory) [75–77]. Такі мережі містять вузли спеціального типу, які дозволяють запам'ятовувати значення тривалий термін. Блок мережі довготривалої пам'яті містить спеціальний нейрон, що використовується як осередок пам'яті (рис. 2.4). Вихід нейрона з'єднаний з його власним входом із одиничною вагою. За рахунок цього значення нейрона на кожному етапі перезаписується і таким чином зберігається. Управління нейроном виконується за допомогою трьох вентилів: вхідного, вихідного та вентиля забуття. При відкритому вхідному вентилі значення входу записується в комірку пам'яті. Якщо вхідний вентиль закритий, то вхідні сигнали не впливають на вміст комірки. Відкритий вихідний

вентиль дозволяє прочитати значення з комірки. Коли значення більше не потрібне, його можна стерти за допомогою вентиля забуття. Вентилі підключаються до інших вузлів нейронної мережі, які в процесі навчання визначають коли необхідно відкрити або закрити той чи інший вентиль.

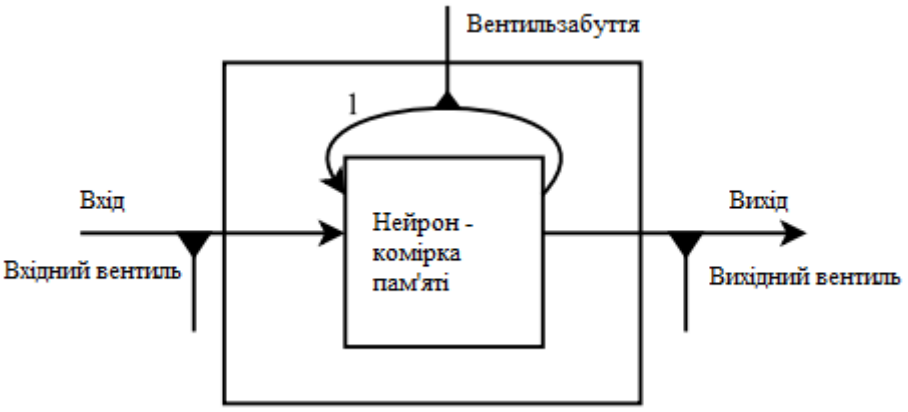


Рисунок 2.4 – Схема блоку мережі довготривалої пам'яті

Завдяки таким осередкам мережі довготривалої пам'яті можуть визначити важливість подій, що відбулися тисячі дискретних тимчасових кроків назад, і запам'ятати ці події. Рекурентні мережі, які використовувалися до цього, могли пам'ятати про подію не більше десяти тимчасових кроків. Проблема зникаючого градієнта в мережах довготривалої пам'яті вирішується за рахунок використання функції тотожності як активаційного та зворотного зв'язку з власним входом з вагою рівною одиниці (рис. 2.4). Оскільки похідна функції тотожності дорівнює одиниці, то помилка під час передачі через такі вузли неспроможна зникнути.

У 1992 році з'явилася архітектура нейронних мереж кресцептрон (cresceptron) [78], основою для якої послужив неокогнітрон. На відміну від неокогнітрона, кресцептрон змінює свою топологію під час навчання за аналогією з мережами, що використовують метод групового обліку аргументів [29]. Важлива ідея, запропонована в кресцептроні, - використання шарів вибору максимального елемента (max-pooling) замість шарів підвиборки з усередненням. Пізніші та вдосконалені версії Кресцептрона містили також

шари розмивання (blurring) зменшення залежності від становища об'єктів [79]. Шари максимального вибору зараз широко застосовуються в згорткових нейронних мережах. Однак для навчання сучасних згорткових мереж використовується алгоритм зворотного поширення помилки [80], що є ефективнішим [81].

У 2006 році Джеффрі Хінтон (Geoffrey Hinton) та Руслан Салахутдінов (Ruslan Salakhutdi№v) запропонували нову архітектуру нейронних мереж – глибокі мережі довіри (Deep Belief Networks) [5]. У цій архітектурі для вирішення проблеми градієнта, що зникає, використовувалася комбінація навчання з учителем і без вчителя. Глибока мережа довіри це стек обмежених машин Больцмана [82, 83]. Кожна така машина містить лише два шари нейронів: вхідний та прихований (рис. 2.5). З'єднання є лише між нейронами різних шарыв, на відміну машини Больцмана високого порядку [84], що може містити інші типи зв'язків.

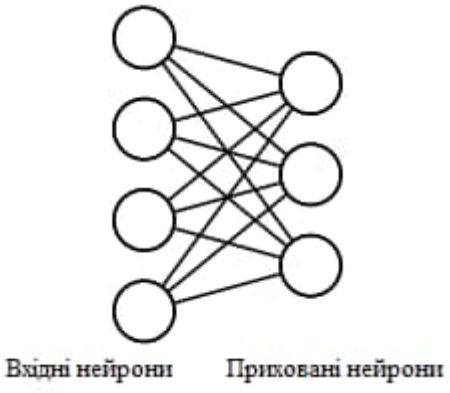


Рисунок 2.5 – Обмежена машина Больцмана

Кожна обмежена машина Больцмана отримує сигнали про подання шаблонів з попереднього рівня та намагається закодувати їх з використанням навчання без вчителя [6] (рис. 2.4). Після цього проводиться тонке налаштування всієї мережі за допомогою навчання з учителем з використанням алгоритму зворотного розповсюдження помилки. Навчена таким чином глибока мережа довіри показала хороші результати на тесті розпізнавання рукописних цифр MNIST [5], розпізнаванні фонем [85], пошуку документів [86] та інших задач. Схожим альтернативним підходом, заснованим на попередньому навчанні

без вчителя та подальшому точному налаштуванні шляхом навчання з вчителем, є використання стека автокодувальників [48, 87–90].

У 2006–2007 роках також відбувся розвиток глибокого навчання мереж згортання на основі навчання з учителем без попереднього навчання без вчителя. У роботі [80] вперше описано застосування алгоритму зворотного поширення помилки для навчання глибокої нейронної мережі з архітектурою, подібною до неокогнітрона і кресцептрону, що складається з шарів згортки, що чергуються, і максимального вибору. Така архітектура нейронних мереж активно використовується і сьогодні [91-95].

Істотний внесок у розвиток згорткових нейронних мереж внесла пропозиція використовувати напівлінійну функцію активації (rectified linear unit) [96, 97], яка визначається наступним чином: ReLU(x) = max(0, x). Така функція активації дозволяє позбавитися проблеми зникаючого градієнта, так як при позитивному значенні сигналу немає його зміни, на відміну сигмоїдальних функцій активації. З іншого боку, напівлінійна функція активації дозволяє скоротити час навчання нейронної мережі [98]. Нейрони, вихідний сигнал яких негативний, не беруть участь у розрахунках, а інакше потрібно виконувати лише лінійні обчислення

У 2010 році Ксав'є Глоро (Xavier Glorot) та Йошуа Бенджіо (Yoshua Bengio) у роботі [99] провели дослідження впливу методів початкової ініціалізації ваг та функцій активації на поширення сигналу в мережі як у прямому, так і у зворотному напрямку. Відповідно до дослідження, використання логістичної сигмоїдальної функції разом із початковою ініціалізацією ваг погано підходить до створення глибоких нейронних мереж, так як призводить до швидкого насичення. Функція активації гіперболічний тангенс такою проблемою не володіє через симетричність (середнє значення функції 0, область значення - (-1, 1)). Глоро та Бенджіо запропонували новий метод ініціалізації ваг нейронної мережі, який вони назвали нормалізованою ініціалізацією. Початкові значення ваги мережі W визначаються за такою формулою:

$$W \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_j+n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j+n_{j+1}}}\right]$$

де U – рівномірний розподіл на відрізку, n_j – кількість нейронів на поточному шарі мережі, n_{j+1} – кількість нейронів на наступному шарі мережі. Використання нормалізованої ініціалізації призводить до зниження насичення нейронів і сигнал про помилку поширюється значно краще.

Метод нормалізованої ініціалізації ваги у 2015 році був адаптований для напівлінійної функції активації. У роботі [100] пропонується визначати початкові ваги W наступним чином:

$$W \sim U\left[-\frac{2}{n_j}, \frac{2}{n_j}\right]$$

де U – рівномірний розподіл на відрізку, n_j – кількість нейронів на поточному шарі мережі. Метод нормалізованої ініціалізації ваги дозволив досягати якісного навчання глибоких нейронних мереж без необхідності використовувати попереднє навчання без вчителя.

Сергій Йоффе (Sergey Ioffe) та Крістіан Жегеди (Christian Szegedy) у 2015 році [101] запропонували використовувати в нейронних мережах спеціальні шари пакетної нормалізації (batch №rmalization), які дозволяють підвищити якість навчання глибокої нейронної мережі. У роботі [13] встановлено, що алгоритм зворотного розповсюдження помилки сходиться швидше, якщо вхідні дані нормалізовані. Однак при розповсюдженні сигналу по нейронній мережі його дисперсія змінюється, причому іноді значно, що негативно позначається на процесі навчання. Сергій Йоффе та Крістіан Жегеди запропонували виконувати нормалізацію не лише на вході до нейронної мережі, а й перед кожним шаром мережі. Нормалізація виконується окремо для кожного пакета даних (mini-batch) методу стохастичного градієнтного спуску (чи його модифікацій).

Пакет B містить m елементів вхідних даних x_i: $B = \{x_1, ..., x_m\}$. Нормалізовані значення визначаються за такою формулою:

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

де μ_B – середнє значення даних у пакеті, σ_B^2 – дисперсія, ϵ – константа, введена для стабільності методу. Потім, щоб зберегти виразність даних, виконується зсув та масштабування:

$$y_i = \gamma \hat{x}_i + \beta$$

де y_i – результуюче значення, γ і β – параметри, які визначаються в процесі навчання.

Пакетна нормалізація реалізується у вигляді шарів пакетної нормалізації, які можуть бути вставлені у необхідне місце в нейронній мережі, у тому числі кілька разів. Додатковою перевагою використання пакетної нормалізації є скорочення часу навчання та зниження перенавчання.

Методи нормалізованої ініціалізації ваг та шари пакетної нормалізації допомагають на практиці впоратися з проблемою градієнта, що зникає, і навчати глибокі нейронні мережі, що складаються з декількох десятків шарів. Це дозволило деяким авторам стверджувати, що проблема градієнта, що зникає, в даний час вирішена [11].

Компанія Google у 2014 році запропонувала нову архітектуру згорткових нейронних мереж Inception [102]. У цій архітектурі мережа будується з набору блоків, що містять комбінацію операцій згортки та підвибору різної розмірності. Такий підхід дозволяє уникнути перенавчання, а також знизити кількість параметрів мережі, які необхідно навчати, що знижує час навчання мережі. Першу версію блоку мережі архітектури Inception показано на рис. 2.6.

Мережа будується з кількох блоків Inception, що повторюються, які йдуть один за одним.

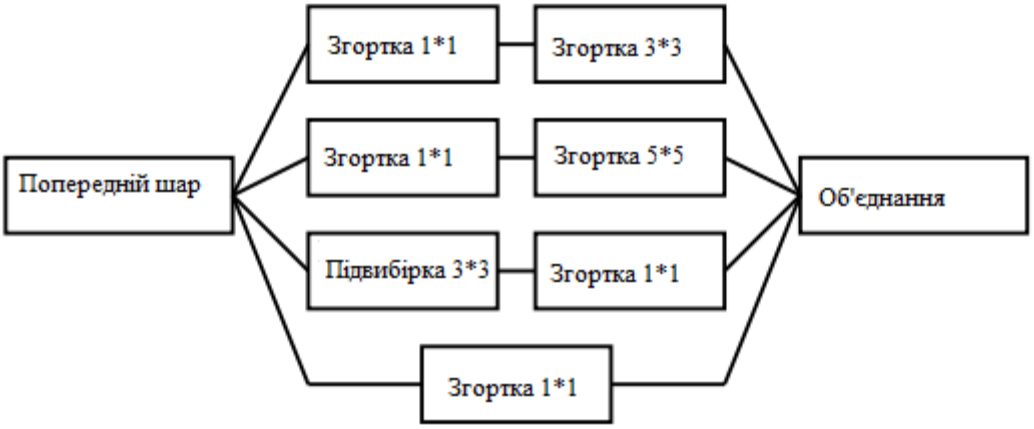


Рисунок 2.6 – Блок мережі Google Inception [102]

На основі запропонованої архітектури була створена та успішно навчена мережа GoogLeNet [102], що складається з 22 шарів. Мережа застосовується до завдань комп'ютерного зору. Згодом Google запропонувала ще кілька варіантів блоків архітектури Inception та мереж на її основі [103, 104].

У 2014 році також було запропоновано нову архітектуру рекурентних нейронних мереж – керовані рекурентні нейронні мережі (gated recurrent neural networks) [105, 106]. Такі мережі схожі на мережі довготривалої пам'яті, але в них не використовуються комірки пам'яті. Схема блоку керованої рекурентної мережі показано на рис. 2.7.

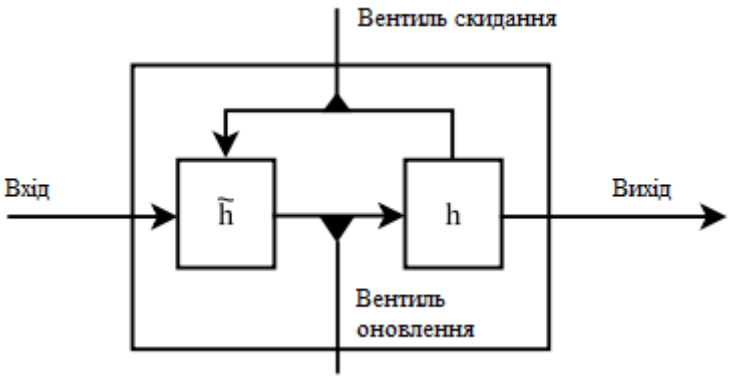


Рисунок 2.7 – Блок керованої рекурентної нейронної мережі [105]: h - значення в блоці

Блок керованої нейронної мережі, на відміну від мереж довготривалої пам'яті, не містить вихідного вентиля, тому значення в блоці завжди видається назовні. Використовується два типи вентилів: оновлення та скидання. Вентиль оновлення визначає, чи буде застосовуватися нове значення h~, а вентиль скидання задає, чи враховується поточне значення h при розрахунку нового значення. Керовані нейронні мережі добре показали себе у завданнях моделювання сигналів мови та поліфонічної музики, а також для автоматичного перекладу [106, 107]. Збільшення кількості шарів у нейронних мережах, навіть із використанням напівлінійних функцій активації, нормалізації початкових значень ваг і шарів нормалізації, який завжди призводить до підвищення якості навчання. При цьому причина не в проблемі градієнта, що зникає, і не в перенавченні, а в збільшенні помилки навчання мережі при зростанні кількості шарів [11, 108]. Для вирішення цієї проблеми компанія Microsoft у 2015 році запропонувала нову архітектуру та підхід до навчання нейронних мереж – залишкове навчання (residual learning). Архітектура використовує той факт, що нейронну мережу завжди можна зробити глибшою без зниження якості роботи шляхом додавання кількох шарів, які не змінюють сигналу. Мережа залишкового навчання, як і мережа Google Inception, будується з блоків, що повторюються. Схема блоку показано на рис. 2.8. Блок включає два звичайні шари нейронної мережі, ваги в яких навчаються за допомогою алгоритму зворотного поширення помилки, а також паралельний їм тотожний шар, що не змінює вхідних сигналів. Тотожний шар використовується, якщо не виходить навчити основні шари. Таким чином, мережа в процесі навчання сама визначає, скільки шарів потрібно для вирішення задачі.

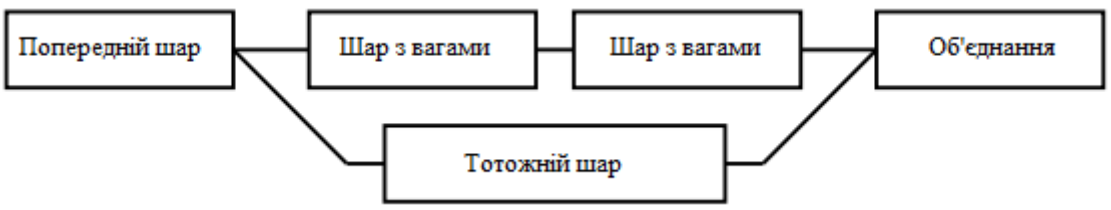


Рисунок 2.8 – Блок мережі залишкового навчання [11]

Компанії Microsoft вдалося зібрати з блоків мережі залишкового навчання глибиною 34, 50, 101 і навіть 152 шари, успішно навчити їх і застосувати для вирішення різних завдань комп'ютерного зору [11]. Компанія Google також почала використовувати підхід залишкового навчання в мережах архітектури Inception [104].

Нині створено велику кількість програмних систем на навчання глибоких нейронних мереж [109–111]. Серед найпопулярніших з них можна відзначити Caffe, Thea№, TensorFlow, Torch та CNTK. Їхні основні характеристики наведені в табл. 2.1.

Бібліотека Caffe [109] одна з найперших популярних систем глибокого навчання. Її розробили у центрі комп'ютерного зору та навчання у Берклі (Berkeley Vision and Learning Center), вихідні коди стали відкритими у 2014 році. Caffe включає найбільшу кількість готових до використання попередньо навчених моделей. Система Thea№ [112] створена в Університеті Монреалю, Канада. Thea№ розроблена на Python, але забезпечує високу продуктивність за рахунок того, що програма на Python автоматично перетворюється на програму C++, яка компілюється і потім виконується. TensorFlow [113] створена компанією Google у 2015 році і включає системи ефективної роботи з тензорами та потокової обробки даних на графі. Бібліотека Torch [114] розроблена мовою Lua та надає зручний високорівневий інтерфейс для створення програм машинного навчання, аналогічний MATLAB. Висока продуктивність забезпечується, так само як і Thea№, за рахунок інтеграції з мовою C. Автори Torch вважали за краще використовувати Lua замість Python через простоту інтеграції C і Lua. Компанія Microsoft розробила систему CNTK (Cognitive T oolkit) [115] та відкрила її вихідні коди у 2016 році.

Усі перелічені системи глибокого навчання нейронних мереж можуть використовувати прискорення навчання як багатоядерні процесори, і прискорювачі обчислень GPU (включаючи оптимізовану бібліотеку cuDNN). Причому істотною перевагою є те, що немає необхідності переробляти

програму, розпаралелювання на CPU та GPU виконується автоматично. Системи Caffe та Thea№ додатково підтримують прискорювачі Intel Xeon Phi, які також допомагають суттєво скоротити час навчання глибоких нейронних мереж [116]. Майже всі системи, крім Thea№, можна використовувати для розподілення нейронних мереж на обчислювальних кластерах.

На додаток до описаних вище систем можна відзначити також бібліотеку Keras [117], яка надає зручний та простий у використанні програмний інтерфейс для навчання глибоких нейронних мереж. Keras не є самостійною системою, а працює поверх Thea№, TensorFlow чи CNTK. У 2016 Keras включили до складу TensorFlow.

Заслуговують на увагу також і нові бібліотеки глибокого навчання, створені нещодавно, але які набирають популярності. Системи PaddlePaddle [118] (створена компанією Baidu) та MXNet [119] спеціально розроблені для навчання глибоких нейронних мереж на розподілених кластерах. Бібліотека Neon [120] розроблялася компанією Nervana. Після покупки Nervana компанією Intel, Neon стала однією з бібліотек, що швидко розвиваються, з підтримкою прискорювачів GPU і Intel Xeon Phi, а також великою кількістю вбудованих попередньо навчених нейронних мереж. MXNet та Neon, які показують хороші результати на тестах продуктивності [111, 121].

З моменту виникнення нейронних мереж відбулося багато змін у їхній архітектурі та методах навчання. В даний час домінуючими є два типи архітектур: згорткові мережі, які успішно застосовуються для задач комп'ютерного зору, і рекурентні мережі, що активно використовуються для обробки природної мови. Ранні згорткові мережі навчалися шляхом комбінації навчання з учителем та без вчителя з використанням автокодувальників та глибоких мереж довіри. Сучасні методи, такі як залишкове навчання, дозволяють використовувати тільки навчання з учителем і відмовитися від навчання, що прискорює та спрощує процес навчання. Також важливим напрямом у розвитку згорткових нейронних мереж є передача навчання (transfer learning) [122]. Цей підхід передбачає використання нейронних мереж, навчених

одних даних, на вирішення інших типів завдань. При цьому застосовується тонка настройка мережі і до навчання на даних від завдання, що є головним.

Таблиця 2.1 – Програмні системи навчання глибоких нейронних мереж

Властивість	Caffe	Thea№	TensorFlow	Torch	CNTK
Базова мова	C++	Python	C++	Lua	C++
API	C++ Python	Python	C++ Python	Lua Python	C# C++ Python
Багатоядерні CPU	+	+	+	+	+
GPU	+	+	+	+	+
Xeon Phi	+	+	-	-	-
Розподілене навчання	+	-	+	+	+
Розробник	Центр комп'ютерного зору та навчання Берклі	Університет Монреаля	Google	Ронан Колаберт	Microsoft
Відкриті коди	+	+	+	+	+
Навчені мережі	+	-	+	+	+

В результаті скорочується час навчання і розширюється сфера застосування попередньо навчених нейронних мереж. Перспективним також є спільне використання згорткових та рекурентних нейронних мереж з навчанням із підкріпленням [123].

Для завдань обробки природної мови, та більш загального випадку обробки послідовностей, нині використовуються рекурентні нейронні мережі. Серед них найефективнішими є мережі довготривалої пам'яті та керовані рекурентні нейронні мережі, так як вони дозволяють запам'ятовувати події, що цікавлять, на тривалий час [124]. Додатковою перевагою рекурентних мереж є можливість навчання без вчителя та без попередньо розміченого набору даних.

Широке поширення практичного застосування нейронних мереж є можливим завдяки наявності великої кількості готових рішень для навчання

глибоких нейронних мереж [109, 112–115, 117–120], у тому числі з можливістю використання сучасних багатоядерних процесорів, прискорювачів обчислень GPU та Intel Xeon Phi, також обчислювальних кластерів із розподіленою пам'яттю.

2.2 Математичні аспекти навчання глибоких моделей

Розгляд реалізації технології навчання буде здійснюватися на наборі графічних даних. Формування навчальної вибірки здійснюється за рахунок обробки зображень.

Таблиця 2.2 – Оператори, що використовуються у алгоритмі попередньої обробки зображення

b	c	$b \vee c$	$b \wedge c$	$b \Rightarrow c$	$b \leftrightarrow c$	$b \oplus c$
0	0	0	0	1	1	0
0	1	1	0	1	0	1
1	0	1	0	0	0	1
1	1	1	1	1	1	0

Реалізація морфологічних методів попередньої обробки зображення зумовлює включення таких логічних операторів як заперечення, диз'юнкція, кон'юнкція, імплікація та еквіваленція. Заперечення у даному випадку подається як одномісний сполучник з приєднаною функцією. Відповідно, якщо логічна змінна $a = 1$, її заперечення $\neg a = 0$, і, аналогічно, при $a = 0$, її заперечення $\neg a = 1$. Інші логічні оператори є двомісними сполучниками, принцип їх застосування подано у табл. 2.3.

У таблиці наведено приклади застосування операторів для логічних змінних b і c (диз'юнкція $b \vee c$, кон'юнкція $b \wedge c$, імплікація $b \Rightarrow c$ та еквіваленція $b \leftrightarrow c$). Крім того, для спрощення математичного апарату, що описує роботу

алгоритму попередньої обробки зображення, слід додати оператор антиеквіваленції $b \oplus c$.

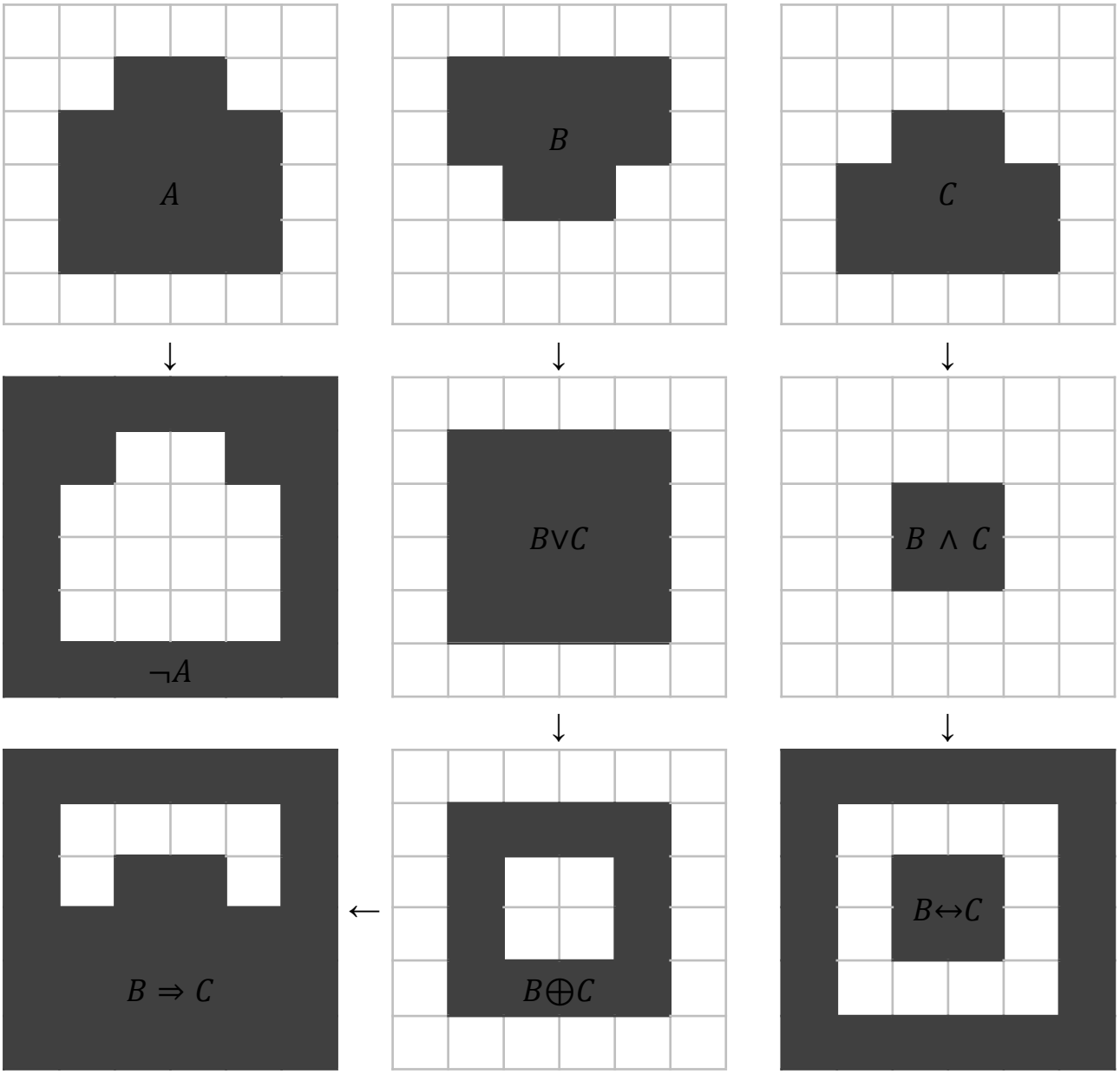


Рисунок 2.9 – Схема застосування логічних операторів при побудові морфологічних алгоритмів обробки бінарного зображення

Від змінних a, b і c можна перейти до множин A, B і C, що представляють собою двовимірні матриці бінарних зображень. Приклад застосування по відношенню до них логічних операторів для елементарних бінарних зображень A, B і C представлено на рис. 2.9.

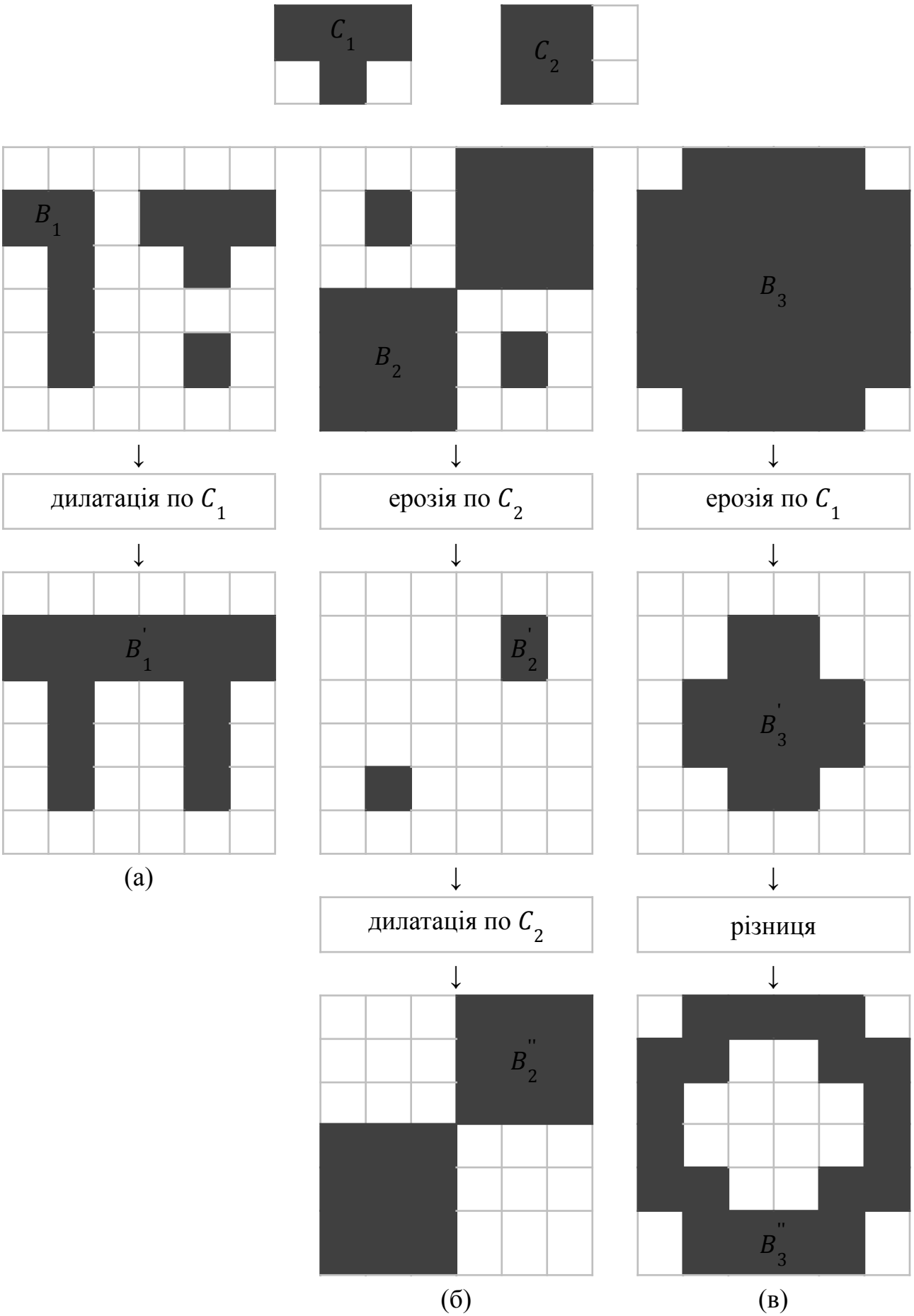

C_1
C_2
B_1
B_2
B_3
дилатація по C_1
ерозія по C_2
ерозія по C_1
B_1'
B_2'
B_3'
(а)
дилатація по C_2
різниця
B_2''
B_3''
(б)
(в)

Рисунок 2.10 – Обробка бінарного зображення на через впровадження процедур дилатації (а), ерозії (б) та виділення границь (в)

Крім того, математичний апаратж включатиме у себе оператор різниці (B/C - елементи B, що не входять у C), завдяки якому між зазначеними логічними операторами через систему рівнянь можна вказати наступні зв'язки:

$$\{B \oplus C = \neg(B \leftrightarrow C) \qquad B \oplus C = (B \vee C)/(B \wedge C)\, B \Rightarrow C = \neg B + B \wedge C \quad . \tag{2.1}$$

На основі заданого набору логічних операторів можна побудувати базові інструменти попередньої обробки бінарного зображення, що представлено на рис. 2.10 для множин вхідних зображень B_1, B_2 і B_3 та примітивів корекції C_1 і C_2. У рамках даного дослідження пропонується розглянути наступні процедури попередньої обробки зображення:

- процедура дилатації, що використовується при нівелювання розривів у елементах зображення;
- процедура ерозії, що використовується для видалення випадкових елементів зображення, що ймовірно є шумами;
- процедура виділення границь.

В основі побудови математичного апарату для проведення процедури дилатації лежать співвідношення на базі логічного оператору антиеквіваленції. Дилатація множини вхідного зображення *B* по множині *C*, яка є примітивом дилатації, представляє собою множину всіх переміщень *x*, при яких множини збігаються щонайменше в одному елементі (рис. 2.10-а). Аналогічно, ерозія множини *B* по примітиву *C* є множиною точок *x*, при зсуві у напрямку яких множина *C* повністю міститься у множині *B*. Після виконання процедури ерозії необхідно виконати процедури дилатації, щоб не втратити границі крупних елементів (рис. 2.10-б). Відповідно границя множина *B*, може бути отримана шляхом виконання операції ерозії *B* по *C*, і, після чого має бути визначена

різниця між множиною B і множиною, що є результатом виконання процедури ерозії (рис. 2.10-в).

Визначення зв'язних компонент між елементами зображення можна розглядати як операцію, що передує процедурі виділення візуальних об'єктів на базі нейромережевих алгоритмів. При цьому зв'язність визначається на базовому рівні через відповідність параметрів сусідніх компонентів, де саме поняття відповідності та максимальної відстані між сусідніми елементами задається на етапі побудови алгоритму.

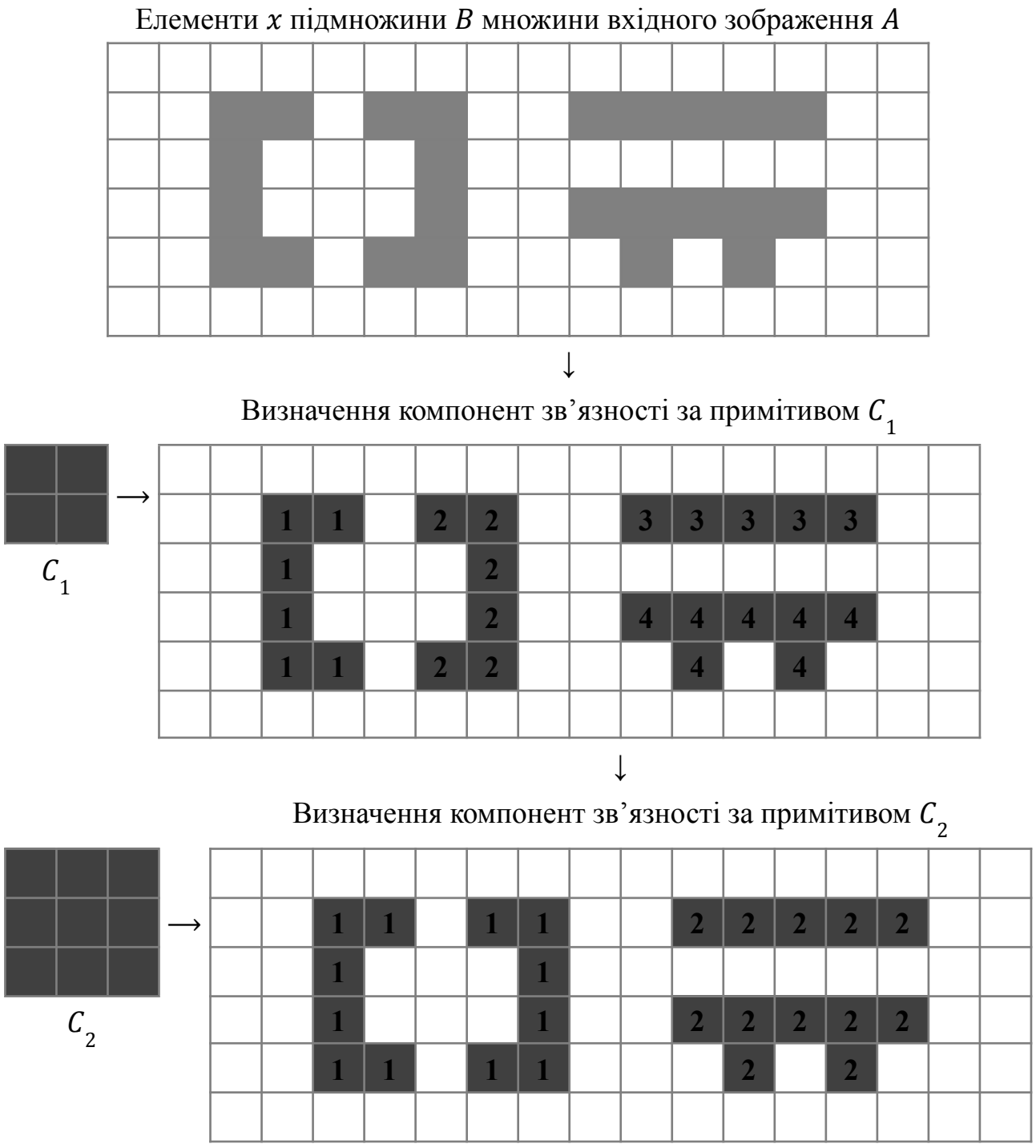


Рисунок 2.11 – Схема визначення компонент зв’язності відповідно до примітиву

Визначимо фундаментальне поняття зв’язності по відношенню до задачі, поставленої у рамках дисертаційного дослідження. Нехай матриця зображення представлена множиною A, підмножина якої B включає X елементів та підлягає аналізу на зв’язність відповідно заданого примітиву C. Вирішення такого завдання полягає у визначенні зв’язної компоненти для кожної з точок $x \in [1; X]$. В окремому випадку зв’язною компонентою може виступати вся підмножина B, тоді її слід віднести до зв’язної множини.

А рівні практичної задачі розглядається пошук множини зв’язних компонент $X^{'}$ для окремої точки $x^{'}$. У математичному вигляді відповідна множина може бути розрахована на базі відповідного виразу:

$$X^{'} \in \left[X_0; X_N\right], \text{ де } \{X_n = x^{'} \text{ для } n = 0 \quad X_n = \left(X_{n-1} \oplus C\right) \wedge B \text{ для } n \in [1; N] \ . \qquad (2.2)$$

Для наочності на рис. 2.11 представлено підмножину B зображення A, що включає у себе лише два класи елементів (умовно, їх можна розділити на нульові і ненульові). Перетин B і C на кожному кроці ітерації виключає з результатів проведення процедури дилатації ті значення, які припадають на нульові елементи. На представленому рисунку для декількох примітивів C показана процедура попередньої обробки зображення через виділення зв’язкових компонент, починаючи з елемента $x^{'}$ і закінчуючи ітерацією, на якій не виявлено жодної нової зв'язкової компоненти. У залежності від вибору примітиву зазначена процедура дозволяє виділяти як окремі елементи, так і їх групи (рис. 2.11).

Застосування порогових методів попередньої обробки зображення є одним з найбільш простих підходів, і при цьому він може бути надалі ефективно поєднаним з нейромережевими алгоритмами аналізу. Зазначена

схема розглядається як циклічне повторення наступного набору операцій: визначення або зміна порогового значення, застосування попередньої обробки зображення на базі порогового методу та поелементна перевірка зображення. Стандартною задачею, що надає можливість визначити ефективність алгоритму, є поділення зображення на області тла та області візуальних об’єктів (рис. 2.12).

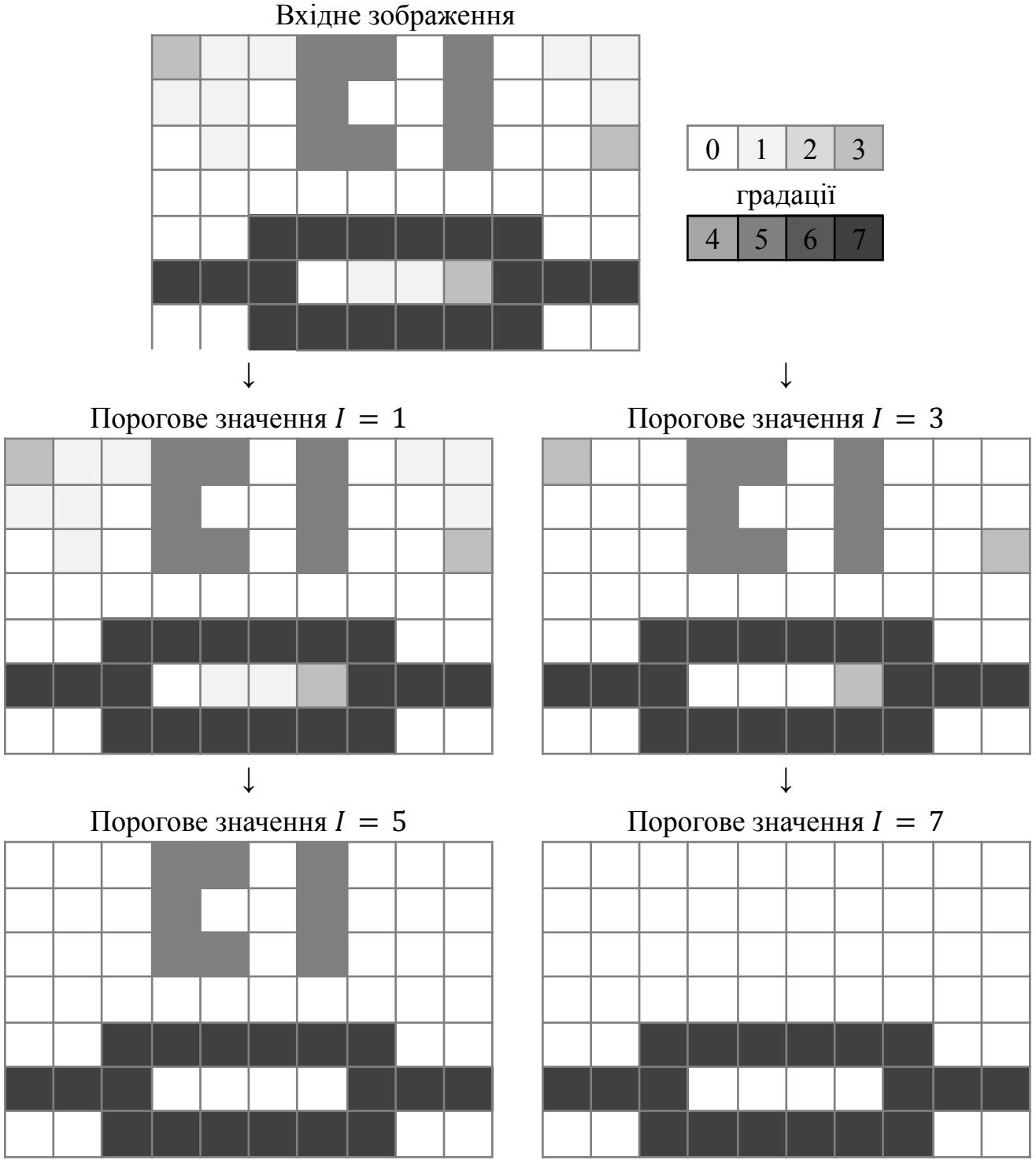


Рисунок 2.12 – Рівень обробки зображення в залежності від порогового значення

Представлена схема є суто ілюстративною і дає загальне уявлення про застосування порогових методів. Для реальних задач застосовуються алгоритми з глобальним або адаптивним порогом. Так у схемі на рис. 2.12. зображення як множина елементів $A: x_n \in [x_1; x_N]$, що характеризуються певним параметром $f(x_n)$, у результаті обробки зображення відповідно порогового значення f_T поділяється на підмножину елементів візуальних об'єктів $B: x_k \in [x_1; x_K]$ де $f(x_k) \geq f_T$ і підмножину елементів тла $C: x_m \in [x_1; x_M]$ де $f(x_n) < f_T$, причому $N = K + M$. Як показано на рисунку від вибору порогового значення f_T залежить якість попередньої обробки зображення: ефективність видалення шумів та цілісність структури об'єктів. Відповідно до введених позначень для алгоритму з глобальним порогом f_T є константою, а для алгоритму з адаптивним порогом f_T є функцією від x_n.

Алгоритм з глобальним порогом є більш простим у програмній реалізації. Він включає у себе наступні етапи (рис. 2.13):

- початкова оцінка порогового значення f_T;
- оцінка точності обрахунку порогового значення Δf_T
- відповідно до порогового значення f_T множина елементів зображення $A: x_n \in [x_1; x_N]$ поділяється на підмножину $B: x_k \in [x_1; x_K]$ де $f(x_k) \geq f_T$ і підмножину $C: x_m \in [x_1; x_M]$ де $f(x_m) < f_T$, де $N = K + M$;
- визначення середніх значень параметрів елементів для кожної з областей: $\overline{f}(x_k)$ і $\overline{f}(x_m)$, відповідно;
- корекція порогового значення, як $\frac{(\overline{f}(x_k)+\overline{f}(x_m))}{2}$;
- повернення до третього етапу, якщо різниця попереднього і корегованого значення більше Δf_T;

- завершення роботи алгоритму, якщо різниця попереднього і корегованого значення менше або дорівнює Δf_T.

Слід окремо зазначити, що вказаний алгоритм показує низьку ефективність при $K \ll M$ і $K \gg M$.

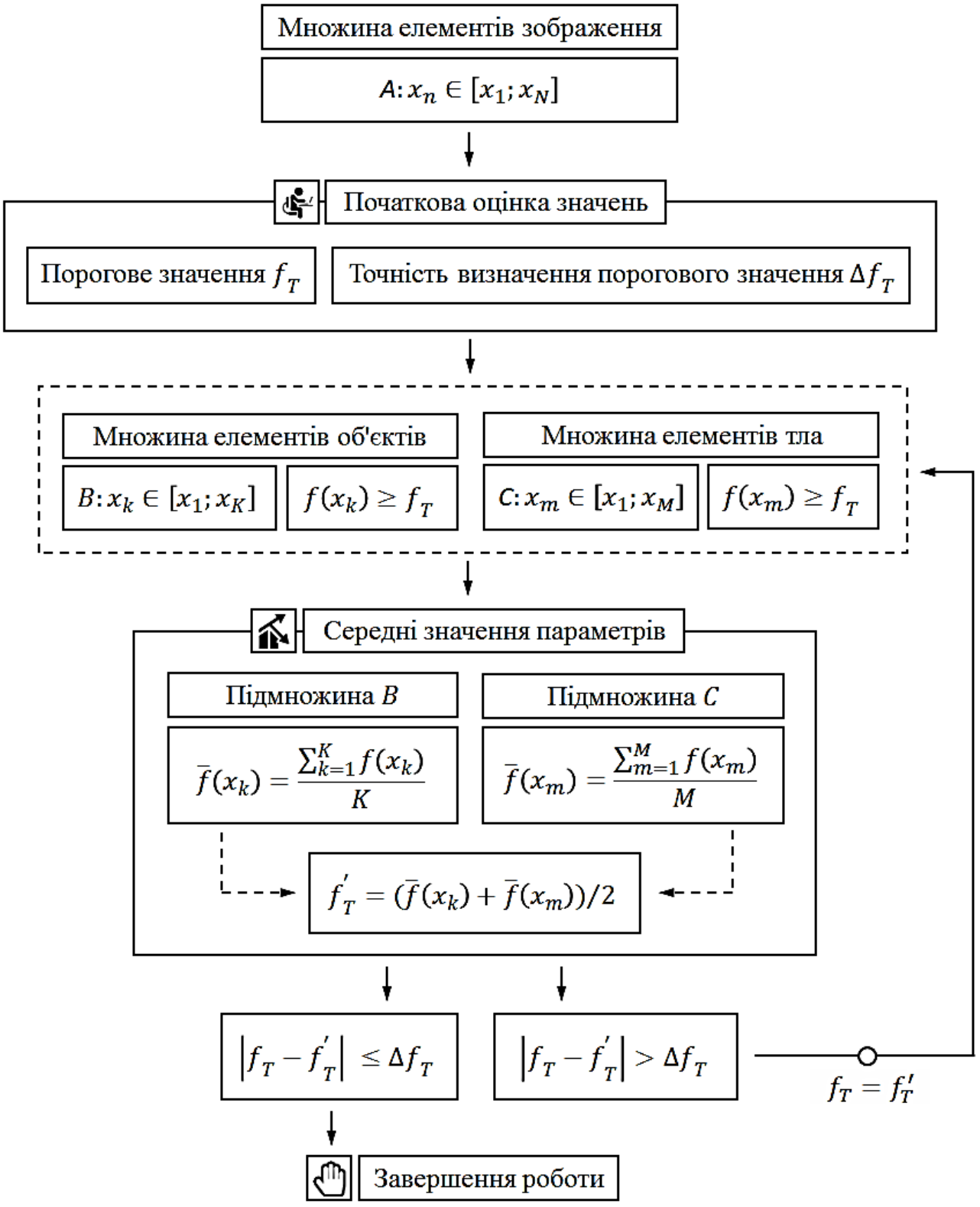
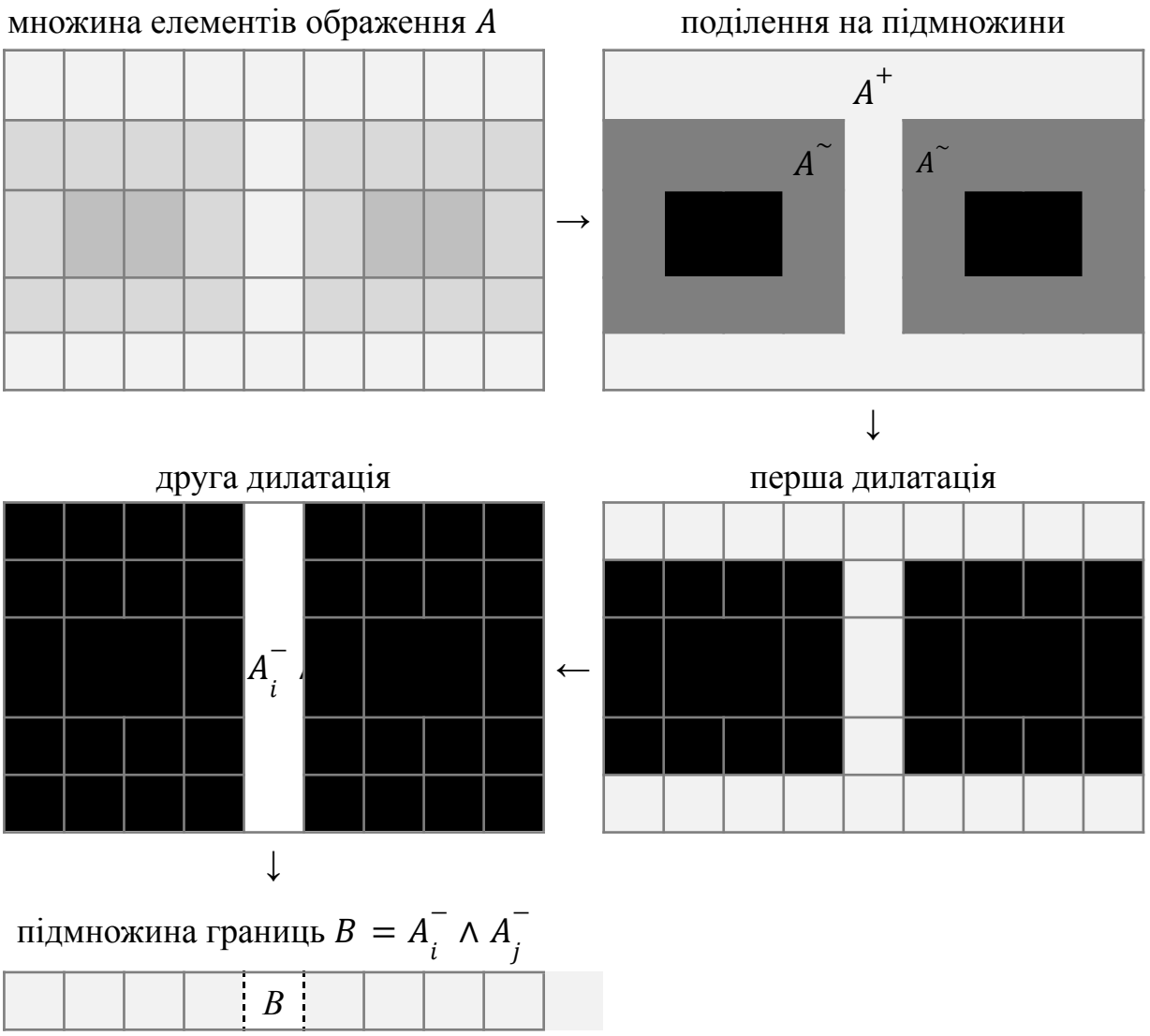


Рисунок 2.13 – Алгоритм обробки зображення з глобальним порогом

Якщо площа об’єктів є набагато меншою або набагато більшою ніж площа тла, зображення слід сегментувати, що відповідає застосуванню алгоритму з адаптивним порогом (також для цього можна використовувати інші методи попередньої обробки зображення), або замість обчислення середнього значення параметрів елементів зображення скористатися іншим алгоритмом, наприклад обчислити середні від мінімального та максимального значення.

У рамках дослідження пропонується використати в якості методів попередньої сегментації зображення або складових масиву відеопотоку методи математичної морфології, зокрема, метод водоподілу, а також методи текстурних дескрипторів.



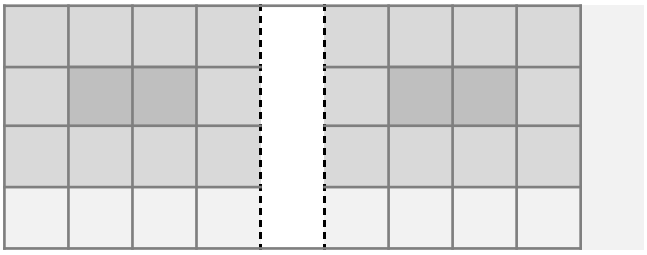


Рисунок 2.14 – Схема сегментації зображення методом водоподілу

Застосування методу водоподілу надає можливість отримати фіксовані результати сегментації вхідного зображення, границі яких не містять розривів. При цьому підхід є достатньо гнучким і у процесі модифікації розробник алгоритму може вводити додаткові обмеження. Алгоритм на основі зазначеного методу включає в себе наступні три етапи: (i) виявлення та усунення розривів у границях сегментів; (ii) порогова обробка зображення; (iii) обробка областей.

Сегментація відбувається шляхом поділу множина елементів зображення $A: x_n \in [x_1; x_N]$ відповідно до певного параметру $f(x_n)$, що їх характеризує, на три основні групи:

- елементи для яких параметр сягає максимального значення f_{max}: підмножина $A^+: x_m \in [x_1; x_M], \forall f(x_m) = f_{max}$;
- елементи для яких параметр сягає мінімального значення f_{min}: підмножина $A^-: x_k \in [x_1; x_K], \forall f(x_k) = f_{min}$;
- повний набір елементів, що не входять у першу і другу групи: підмножина $A^{\sim}: x_l \in [x_1; x_L], f_{min} < f(x_l) < f_{max}, L = N - (M + K)$.

При проведенні процедури сегментації на основі методу водоподілу проводиться розширення областей кожного з елементів підмножини A^- шляхом послідовного застосування морфологічних алгоритмів, зокрема дилатації, як це показано на рис. 2.14. На певному етапі розширені області елементів підмножини A^- перетинаються, і на місці їх перетину утворюються елементи підмножини границь $B = A_i^- \wedge A_j^-$. Надалі по відношенню до області B

алгоритм дилатації не застосовується. На практиці застосування методу водоподілу часто призводить до надлишкової сегментації, яка пов’язана з дефектами зображення. Обмеження кількості областей можливе шляхом попередньої обробки зображення і введенню маркерів як зв'язних компонент елементів візуальних об’єктів і тла зображення.

У свою чергу текстурний аналіз включає у себе два класи текстурних операторів:

- оператори, що визначають орієнтацію і масштаб візуального об’єкту зображення;
- оператори, що є незалежними від орієнтації і масштабу візуального об’єкту зображення (середнє значення і дисперсія).

Для побудови математичного апарату, що дозволяє описати виділену область зображення пропонується застосувати вирази для гістограми яскравості. Нехай область зображення складається з елементів $X: x_i \in \left[x_1; x_{I-1}\right]$ (одного рівня яскравості для кожного i), де I — кількість градацій яскравості, тоді його діаграма подається функцією $P\left(x_i\right)$. Статистичний розподіл гістограми оцінюється через центральний момент, тобто момент для центрованого розподілу. Центральний момент порядку n розраховується як:

$$\mu_n(x) = \sum_{i=0}^{I-1} \left(x_i - \sum_{i=0}^{I-1} \left(x_i \cdot P\left(x_i\right) \right) \right)^n \bullet P\left(x_i\right). \qquad (2.3)$$

Центральний момент другого порядку відповідає дисперсії гістограми, а центральний момент третього порядку – асиметрії гістограми.

2.3 Алгоритм реалізації навчання глибоких моделей

Побудові комплексного підходу по визначенню методологічних основ розпізнавання візуальних об’єктів у відеопотоці та наборах зображень на базі

нейромережевих алгоритмів має передувати розробка системи оцінки ефективності відповідних алгоритмів. У рооті були розглянуті групи актуальних на сьогоднішній день алгоритмів нейромережевого розпізнавання візуальних об’єктів, що надає можливість надалі запропонувати відповідну математичну модель, яка узагальнює принципи роботи нейромережевої архітектури, за рахунок чого можна визначити ефективність розпізнавання через розрахунок цільових функцій.

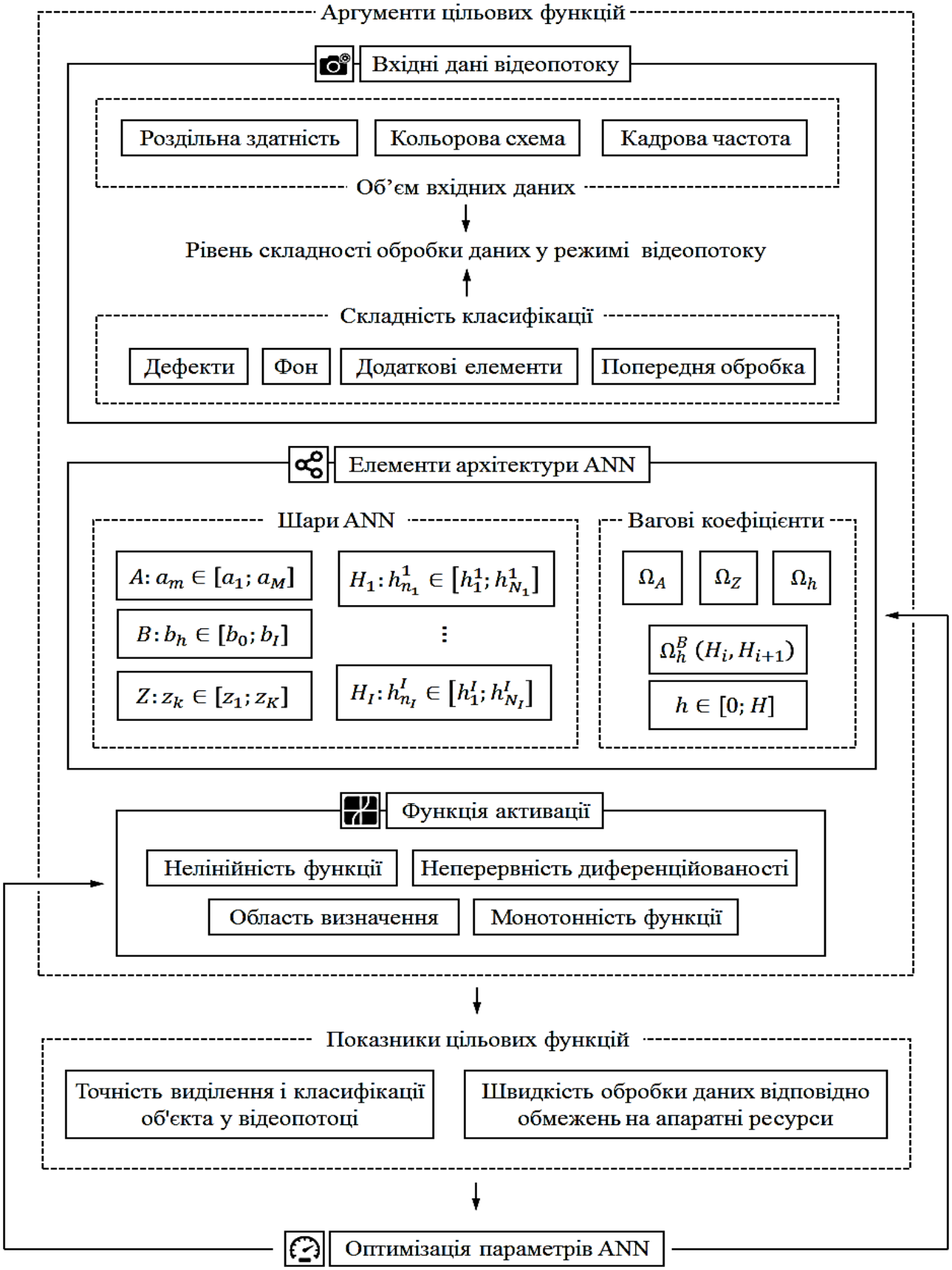


Рисунок 2.15 – Алгоритм оцінки та оптимізації параметрів нейромережевої системи розпізнавання образів у відеопотоці

Як було показано, в якості цільових функцій системи розпізнавання у загальному випадку мають виступати показники точності класифікації та швидкості обробки фіксованого об’єму даних відповідно обмеження на апаратні ресурси ANN-платформи.

Відповідно, аргументами цільових функцій виступатимуть ключові параметри архітектури ANN, процедури навчання, а також формат і об’єм вхідних даних, що обробляються і класифікуються на базі ANN, а отже визначають складність завдання класифікації і навантаження на апаратні ресурси системи. Загальну схему оцінки та оптимізації параметрів ANN-системи розпізнавання образів у відеопотоці показано на рис. 2.15.

У загальному випадку архітектура ANN включає у себе нейрони, що можуть бети поділені на такі групи як: нейрони вхідного шару, вихідного шару, прихованого шару або прихованих шарів, а також нейрони змішення, функції яких було розглянуто у попередньому розділі. Крім того, математична модель ANN має враховувати множини зв’язків між нейронами, опис яких залежить від типу архітектури ANN, що також слід узагальнити на етапі побудови відповідного математичного апарату. Для опису нейронів ANN пропонується ввести наступний набір матриць:

- множина нейронів вхідного шару як одномірна матриця A, що складається з набору змінних $a_m \in \{a_1, a_2, ...a_m, ...a_M\}$;
- множина нейронів вихідного шару як одномірна матриця Z, що складається з набору змінних $z_m \in \{z_1, z_2, ...z_k, ...z_K\}$;
- множина або множини нейронів прихованого шару або прихованих шарів як, відповідно, одномірна матриця H, що складається з набору змінних $h_n \in \{h_1, h_2, ...h_n, ...h_N\}$ або набір матриць $H_i \in \{H_1, H_2, ...H_i, ...H_I\}$, що складаються з наборів змінних $H_i: h^i_{n_i} \in \left\{h^i_1, h^i_2, ...h^i_{n_i}, ...h^i_{N_i}\right\}$;
- множина нейронів зміщення, що має бути представлена як одномірна матриця набору змінних $b_i \in \{b_0, b_1, ...b_i, ...b_I\}$, де змінна b_0 є нейроном зміщення

вхідного шару, а підмножина $\{b_1, b_2, ...b_i, ...b_I\}$ — нейрони зміщення прихованих шарів.

Підмножина змінних $[a_1; a_M]$ формується на базі даних відеопотоку, а $[b_0; b_I]$ - визначається при розробці нейромережевого алгоритму, у той час як підмножини $z_1; z_K$ і $\left\{\left[h_1^1; h_{N_1}^1\right], \left[h_2^2; h_{N_2}^2\right], ..., \left[h_1^i; h_{N_i}^i\right], ..., \left[h_1^I; h_{N_I}^I\right]\right\}$ обчислюються на основі нормалізованої функції активації від суми вихідних значень змінних нейронів попереднього шару помножених на вагові коефіцієнти. Вагові коефіцієнти, відповідно, також є аргументами цільових функцій, які у рамках математичного апарату системи оцінки ефективності слід представити у вигляді наборів одномірних матриць змінних. У більшості моделей ANN нейромережеві архітектуру можна описати через зв'язки нейронів кожного шару з наступним. Таким чином, модель складається з наступного набору матриць вагових коефіцієнтів:

- матриці $\Omega_A(A, H_1)$ зв'язків між вхідним та першим прихованим шаром: $\left[\Omega_A^1\left(a_1, H_1\right): \left\{\omega\left(a_1, h_1^1\right)...\omega\left(a_1, h_{N_1}^1\right)\right\}; \Omega_A^I\left(a_M, H_1\right): \left\{\omega\left(a_M, h_1^1\right)...\omega\left(a_M, h_{N_1}^1\right)\right\}\right]$
- матриці $\Omega_Z(H_I, Z)$ зв'язків між останнім прихованим та вихідним шаром: $\left[\Omega_Z^1\left(H_I, Z\right): \left\{\omega\left(h_1^I, z_1\right)...\omega\left(h_1^I, z_K\right)\right\}; \Omega_Z^K\left(H_I, Z\right): \left\{\omega\left(h_{N_I}^I, z_1\right)...\omega\left(h_{N_I}^I, z_K\right)\right\}\right]$;
- набори матриць зв'язків між прихованими шарами $\Omega_h\left(H_i, H_{i+1}\right)$ для $h \in [1; H]$;
- матриці зв'язків з нейронами зміщення Ω_h^B для $h \in [0; H]$.

Очевидно, що за умови більш складної архітектури ANN, як то рекурентного нейромережевого алгоритму, даний математичний апарат має бути доповнено додатковими наборами матриць вагових коефіцієнтів, але загальний принцип роботи залишиться незмінним. Поза наборів матриць нейронів та вагових коефіцієнтів, у якості аргументів цільових функцій виступають також функції активації, що можуть бути обрані індивідуально для кожного шару відповідно показників ефективності. Загальні міркування, що зумовлюють

вибір функції активації базуються на аналізі типових характеристик, як то показників нелінійності і монотонності функцій, області визначення та неперервності диференційованості, що визначає ефективність застосування методу градієнтного спуску.

Розроблена схема може бути використана для оцінки ефективності нейромережевих алгоритмів, що використовуються для широкого класу задач по виділенню візуального об’єкту у масиві даних відеопотоку, а також попередньої обробки вхідних даних. Математична модель, що покладена у її основу є достатньо гнучкою для модифікації та оптимізації відповідно поставленого завдання.

Сегментація окремого зображення або набору зображень даних відеопотоку, у загальному випадку розглядається як методика кластеризації множини елементів. Як було показано у оглядовій частині дисертаційної роботи, з метою вирішення вказаного завдання може бути застосовано широкий набір програмних методів і, зокрема, високу ефективність сегментації показують нейромережеві алгоритми.

Узагальнення нейромережевих методик сегментації зображення з метою побудови математичної моделі є комплексною задачею. У рамках даного дослідження пропонується узяти за основу нейромережевий метод нарощування областей, що далі може бути оптимізовано та модифіковано відповідно до особливостей кластеризаційних алгоритмів. Вказаний підхід, що може бути застосовано як для нейромережевих, так і для класичних програмних алгоритмів полягає у порівнянні характеристик окремого елементу зображення (яскравості відповідно до кожного з показників колірної схеми) з найближчими сусідніми елементами.

Базова схема виконання процедури сегментації зображення за допомогою нейромережевих алгоритмів, таким чином, складатиметься з наступних етапів:

1. Отримання вхідного зображення без попередньої обробки, що на рівні побудови математичного апарату представляється набором двовимірних матриць $A: \{A_n\}$, де $n \in [1; N]$, причому для $\forall n \ A_n: \{a_n(x, y)\}$ де $x \in [1; X]$, $y \in [1; Y]$. N — є

кількістю кольорів у кольоровій схемі вхідного зображення, а $X \times Y$ — його роздільною здатністю.

2. Випадковий вибір початкового елементу зображення з координатами $\left(x_0, y_0\right)$.
3. Перевірка сусідніх елементів з координатами, відповідно $\left(x_0 - 1, y_0 - 1\right)$, $\left(x_0 - 1, y_0 + 1\right)$; $\left(x_0 + 1, y_0 - 1\right)$, $\left(x_0 + 1, y_0 + 1\right)$, $\left(x_0 - 1, y_0\right)$, $\left(x_0, y_0 - 1\right)$, $\left(x_0, y_0 + 1\right)$ та $\left(x_0 + 1, y_0\right)$ на схожість по відношенню до початкового елементу на базі функції $F\left(F_1\left(a_1, a_1^{'}\right), \ldots, F_n\left(a_n, a_n^{'}\right) \ldots, F_N\left(a_N, a_N^{'}\right)\right)$, що співвідносить параметри двох елементів по кожному з каналів $n \in [1; N]$ колірної схеми вхідного зображення через застосування нейромережевого алгоритму.
4. Надання групі елементів, параметри яких відповідно функції F є достатньо близькими до початкового елементу і формування матриці початкового набору елементів першого сегменту $S_{01}: \left\{a_1^{01}, \ldots, a_i^{01}, \ldots a_{I_1}^{01}\right\}$. При цьому всі зазначені елементи $a_i^{01} \in \left[a_1^{01}; a_{I_1}^{01}\right]$ мають бути виключені з подальшого аналізу.
5. Пошук сусідніх елементів для набору S_{01}, що на рівні ітеративної процедури надає можливість сформувати повний набір елементів першого сегменту $S_1: \left\{a_1^1, \ldots, a_i^1, \ldots a_{I_1}^1\right\}$.
6. Повторення кроків (2) … (5) для всіх елементів зображення крім набору $a_i^1 \in \left[a_1^1; a_{I_1}^1\right]$ і формування набору матриць сегментів зображення $S: \left\{S_1, \ldots, S_j, \ldots S_J\right\}$, де J — загальна кількість сегментів.
7. Оцінка якості сегментації. Подальша робота з сегментованим зображенням або модифікація функції F (відповідно, зміна архітектури ANN та методу навчання) і повторення кроків (2) … (6).

Представлений алгоритм можна розглядати, як окремий випадок алгоритму оцінки та оптимізації параметрів нейромережевої системи розпізнавання образів у відеопотоці, що було представлено вище. Розроблена

методологія значною мірою обмежена за функціональністю, але при цьому є зручною з точки зору можливості визначення максимального рівня навантаження на апаратні ресурси, що визначається через максимальну кількість точок, які підлягають аналізу, так максимальну кількість сусідніх точок для однієї точки.

Окремим питанням, яке слід розглянути є навчальної вибірки для нейромережевого алгоритму. За умов побудови універсального алгоритму, коли дані про структуру зображення неможливо узагальнити пропонується сформувати навчальну вибірку зображень на базі псевдовипадкових послідовностей за колірною схемою, що відповідає формату вхідного зображення. Таким чином дані зображень вибірки можна розглядати як штучно генерований імпульсний шум.

Формалізація на рівні побудови математичного апарату процедури підготовки навчальної вибірки для навчання нейромережевого алгоритму виділення і класифікації об'єктів відеопотоку включає у себе наступні етапи:

1. визначення типу об'єктів навчальної вибірки та формату їх представлення (роздільна здатність $X \times Y$, кількість каналів колірної схеми N), а також типу навчання відповідно до завдання, що поставлено перед нейромережевим класифікатором;
2. підготовка масивів даних відеопотоку, що можуть бути представлені у вигляді двовимірних матриць зображень A: $\left\{A_{m,k}^{n}\right\}$ розміру $X \times Y$, де $n \in [1; N]$ відповідає каналу колірної схеми, $m \in [1; M]$ — кадру відеофрагменту, а $k \in [1; K]$ — номеру відеофрагменту.
3. обробка складових набору A: $\left\{A_{m,k}^{n}\right\}$ з метою виділення дефектів зображення, фону та додаткових елементів, і, відповідно отримання навчальної вибірки B: $\left\{B_{m,k}^{n}\right\}$;
4. класифікація об'єктів навчальної вибірки, що проводиться відповідно до типу навчання, що було обрано на етапі (1).

Характерно, що на сьогоднішній день, в області підготовки навчальної вибірки для нейромережевої класифікації об’єктів у відеопотоці проведено значний об’єм робіт, результати більшості яких є відкритими. У рамках даного дослідження, пропонується як розглянути можливість створити власну вибірку для навчання, так і використати вибірки, побудовані іншими дослідниками, та порівняти результат роботи ANN за схемою оцінки та оптимізації нейромережевих класифікаторів.

За інформаційним об’ємом накопичених зразків сучасні бази даних, що містять відкриті навчальні вибірки для нейромережевих алгоритмів можна поділити на п’ять основних груп, що складаються як з наборів окремих зображень, так і фрагментів відеоданих:

- навчальні вибірки розпізнавання обличчя [91-95], що включають у себе вибірки розпізнавання основних типів емоцій, віку, статі, та інших факторів (вибірки формувалися для різного рівня освітлення, повороту голови, тощо);
- навчальні вибірки розпізнавання дій людини [96-98], що включають у себе вибірки розпізнавання базових рухів, елементів соціальної взаємодії, дій, які становлять потенційну загрозу та т.п. складові;
- навчальні вибірки розпізнавання об’єктів супутникової та аеро-зйомки [107-110], які включають у себе широкий набір типових зразків областей, що групуються відповідно картографічної класифікації;
- навчальні вибірки розпізнавання об’єктів, що становлять професійний інтерес у вузьких областях науки, техніки, медицини та ін. [99-103];
- навчальні вибірки розпізнавання текстових елементів та умовних знаків, що використовуються для координації у просторі, регулювання транспортних потоків і навігації [104-106];

При виборі окремої вибірки слід враховувати відповідність її складових поставленому завданню по навчанню нейромережевого класифікатора, а також такі параметри як кількість елементів вибірки, відео-формат вибірки чи представлення її елементів як набору зображень, рівень зв’язності зображень, їх роздільна здатність і колірна схема, рівень попередньої обробки зображень,

виділення фону та додаткових елементів, попередня сегментація зображень і наявність міток, що дозволяють класифікувати складові вибірки.

Складність загальної задачі виявлення і розпізнавання предметів у візуальному потоці визначається потенційно високим ступенем мінливості об’єктів, фону і додаткових елементів, а також ракурсу і рівня освітлення тривимірної сцени. Тим більш нетривіальною задачею є узагальнення методології формування нейромережевого класифікатора, що виконує зазначений клас завдань. На даному етапі пропонується побудувати базову модель розробки згорткових нейромережевих класифікаторів, яку можна розширити для архітектури типу R-CNN, «Fast R-CNN» і «Faster R-CNN».

Базова математична модель включатиме у себе такі етапи як (i) генерація RoI як областей, що потенційно можуть містити об’єкт, (ii) формування карти ознак, (iii) класифікація об’єктів. Генерації RoI надає можливість знизити рівень складності виявлення об’єктів та, відповідно навантаження на апаратні ресурси системи. Даному етапу передує вибір маски детектору, що виділяє набір RoI. На рівні базової схеми обирається маска прямокутної рамки, що дозволяє виділити контури об'єкту. Показник кількості контурів, що утримуються у межах маски, прямо пропорційний ймовірності того, що маска містить об'єкт, який необхідно виділити, при цьому RoI складаються з областей з замкненими контурами. Модифікацією зазначеного алгоритму може бути ієрархічне угруповання подібних областей за певними ознаками на базі графів, ребра яких поєднують елементи зображення (вагові коефіцієнти ребер при цьому визначається як різниця у відповідних показниках зазначених елементів), що, у свою чергу, можна розглядати як селективний пошук. Виділені області надалі також групуються, згідно вагових коефіцієнтів, які для елементів у межах однієї групи мають бути меншими ніж для елементів різних груп. Таким чином зазначений алгоритм на першому етапі включає у себе сегментацію зображення, елементи якої групуються відповідно загальної ієрархії.

На наступному етапі формуються карти ознак потоку вхідних даних, що може бути представлено як набір двовимірних матриць зображень. Сформовані

на попередньому етапі RoI масштабуються відповідно архітектури згорткової нейромережі. Алгоритм оптимізації розширює RoI щоб утворити контур заданої товщини відповідно ядра CNN, яка на виході формує багатовимірний вектор ознак для кожної області, що потенційно утримує об’єкт. Нарешті виконується класифікація об'єктів для кожної RoI, для чого може бути використано методу опорних векторів (виявлення об’єктів на базі SVM-класифікатора), якому передує пошук локальних максимумів, що визначають контур об’єкту і виключають множинні RoI одного об’єкту.

Висновки до розділу

Другим розділом розкрито методи та моделі глибокого навчання. Окреслено математичну складову та описано алгоритм реалізації навчання глибоких моделей.

В даний час домінуючими є два типи архітектур: згорткові мережі, які успішно застосовуються для задач комп'ютерного зору, і рекурентні мережі, що активно використовуються для обробки природної мови. Ранні згорткові мережі навчалися шляхом комбінації навчання з учителем та без вчителя з використанням автокодувальників та глибоких мереж довіри. Сучасні методи, такі як залишкове навчання, дозволяють використовувати тільки навчання з учителем і відмовитися від навчання, що прискорює та спрощує процес навчання. Також важливим напрямом у розвитку згорткових нейронних мереж є передача навчання (transfer learning). Цей підхід передбачає використання нейронних мереж, навчених одних даних, на вирішення інших типів завдань. При цьому застосовується тонка настройка мережі і до навчання на даних від завдання, що є головним. В результаті скорочується час навчання і розширюється сфера застосування попередньо навчених нейронних мереж. Перспективним також є спільне використання згорткових та рекурентних нейронних мереж з навчанням із підкріпленням

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ НА БАЗІ ШТУЧНОЇ НЕЙРОННОЇ МЕРЕЖІ

3.1 Розробка структури системи розпізнавання об'єктів на базі штучної нейронної мережі

У рамках даної дипломної роботи, розроблено систему розпізнавання об'єктів на базі штучної нейронної мережі запропоновано, яка ґрунтується на низці моделей направлених на аналіз графічних даних до складу яких входить: модель налаштування програмного блоку системи реєстрації вхідного масиву графічних даних шляхом компенсації оптичних аберацій через побудову математичної моделі спотворень зображення і проведення процедури зворотної дисторсії, а також зменшення впливу статистичного шуму через побудову математичної моделі розподілу ймовірності і проведення процедури просторової фільтрації; модель попередньої обробки та попередньої сегментації графічних даних на основі морфологічних методів, що може бути застосована до матриць кольорових зображень через включення у методику алгоритмів виділення зв'язних компонент та порогових алгоритмів; модель оптимізації компонент загального комплексу по роботі з наборами зображень. Загальна множина описаних моделей, складає сімейство яке, має основну архітектуру. Ця архітектура дотримується гібридного підходу і складається з трьох підмереж кодування:

1) кодування особливостей графічних даних;

2) захоплення представлення графічних даних на рівні пікселів, а також особливостей використання фонового простору;

3) захоплення значення компонентів зображення у їх контексті. Крім того, рівень захоплення використовується для аналізу залежностей вихідних тегів.

Схематичне зображення запропонованої моделі наведено на рис. 3.1.

Рисунок 3.1 – Модель маркування послідовності графічної інформації

Нехай $V_{\text{зоб}}$, $V_{\text{озн}}$, $V_{\text{пік}}$ – набори типу зображення, ознак та пікселів відповідно; тут $|V|$ позначає розмір навченої вибірки V. У запропонованій моделі використовуються три види таблиць пошуку для відображення графічної інформації, ознак і типів використання. Позначимо їх як

$$L_{\text{зоб}} \in R\left|V_{\text{зоб}}\right| \times d_{\text{зоб}},\ L_{\text{озн}} \in R\left|V_{\text{озн}}\right| \times d_{\text{озн}},\ L_{\text{пік}} \in R\left|V_{\text{пік}}\right| \times d_{\text{пік}}, \quad (3.1)$$

де $d_{\text{зоб}}$, $d_{\text{озн}}$ і $d_{\text{пік}}$ – це довжини щільних векторів, що представляють зображення, ознаку та піксель графічної інформації відповідно. $L_{\text{зоб}}$ ініціалізується за допомогою попередньо навченого вбудовування зображення. $L_{\text{озн}}$ і $L_{\text{пік}}$ ініціалізуються випадковим чином зі значеннями, отриманими з рівномірного розподілу з діапазоном $[-\ 0,5; 0,5]$. Усі ці таблиці пошуку потім точно налаштовуються під час навчання.

Дана вхідна графічна інформація з n зображеннями $x\ =\ \{x_1,\ ..,\ x_n\}$, вона перетворюється на зображення, ознаки та набір пікселів:

$$x_{\text{зоб}} \in R^{n},\ x_{\text{озн}} \in R^{n\times nb_{\text{сег}}},\ x_{\text{пік}} \in R^{n} \qquad (3.2)$$

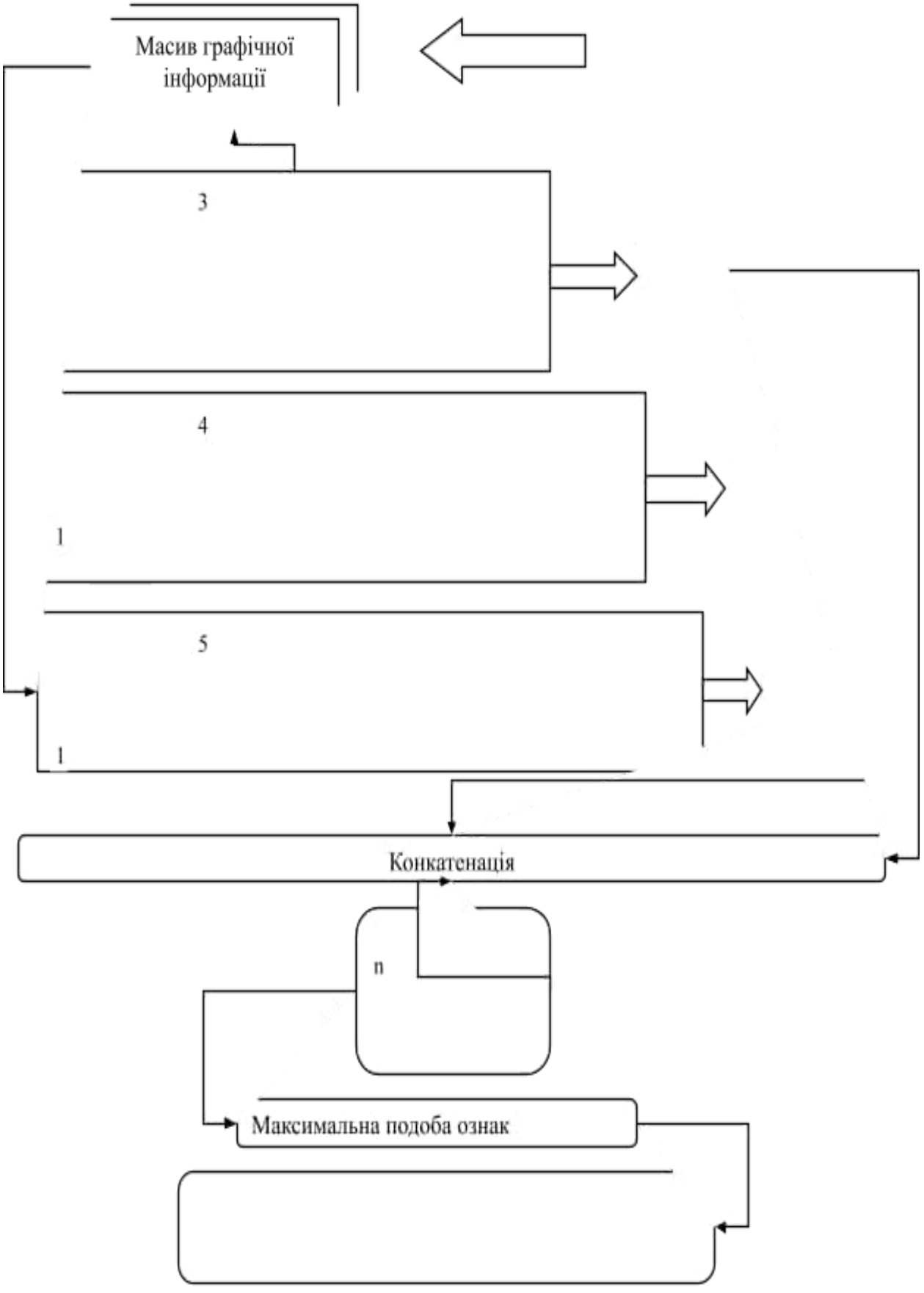
де $nb_{\text{озн}}$ позначає масив ознак у зображенні. Техніка заповнення використовується для того, щоб усі зображення мали максимально однакову кількість ознак. Після цього створюються масиви зображення, ознак та пікселів

$$e_{\text{зоб}} \in R^{n\times nd_{\text{зоб}}},\ e_{\text{пік}} \in R^{n\times nd_{\text{пік}}},\ e_{\text{озн}} \in\ R^{n\times nb_{\text{озн}}\times d_{\text{озн}}} \qquad (3.3)$$

шляхом пошуку $x_{\text{зоб}}, x_{\text{озн}}, x_{\text{пік}}$ у $L_{\text{зоб}}, L_{\text{озн}}$ і $L_{\text{пік}}$ відповідно.

За для досягнення максимально продуктивності, часто використовуються деякі додаткові функції, такі як зворотна дисторсія, функція фільтрації, а також функція фрагментації. Серед цих функцій функція просторової фільтрації не є зовнішньою, і вона дуже корисна для завдань обробки зображень, з метою визначення яскравості пікселів. Отже, у запропонованій моделі залишкова нейронна мережа використовується для захоплення особливостей фону зображення у лівому та правому контекстах (рис. 3.2).

Вхідні зображення перетворюються у форму використання загального фону, який кодує кожен піксель одним цілим значенням, що представляє використання фону цього зображення. Усі типи використання фонових ознак наведені в таблиці 3.1.


Масив графічної інформації
3
4
1
5
1
Конкатенація
n
Максимальна подоба ознак

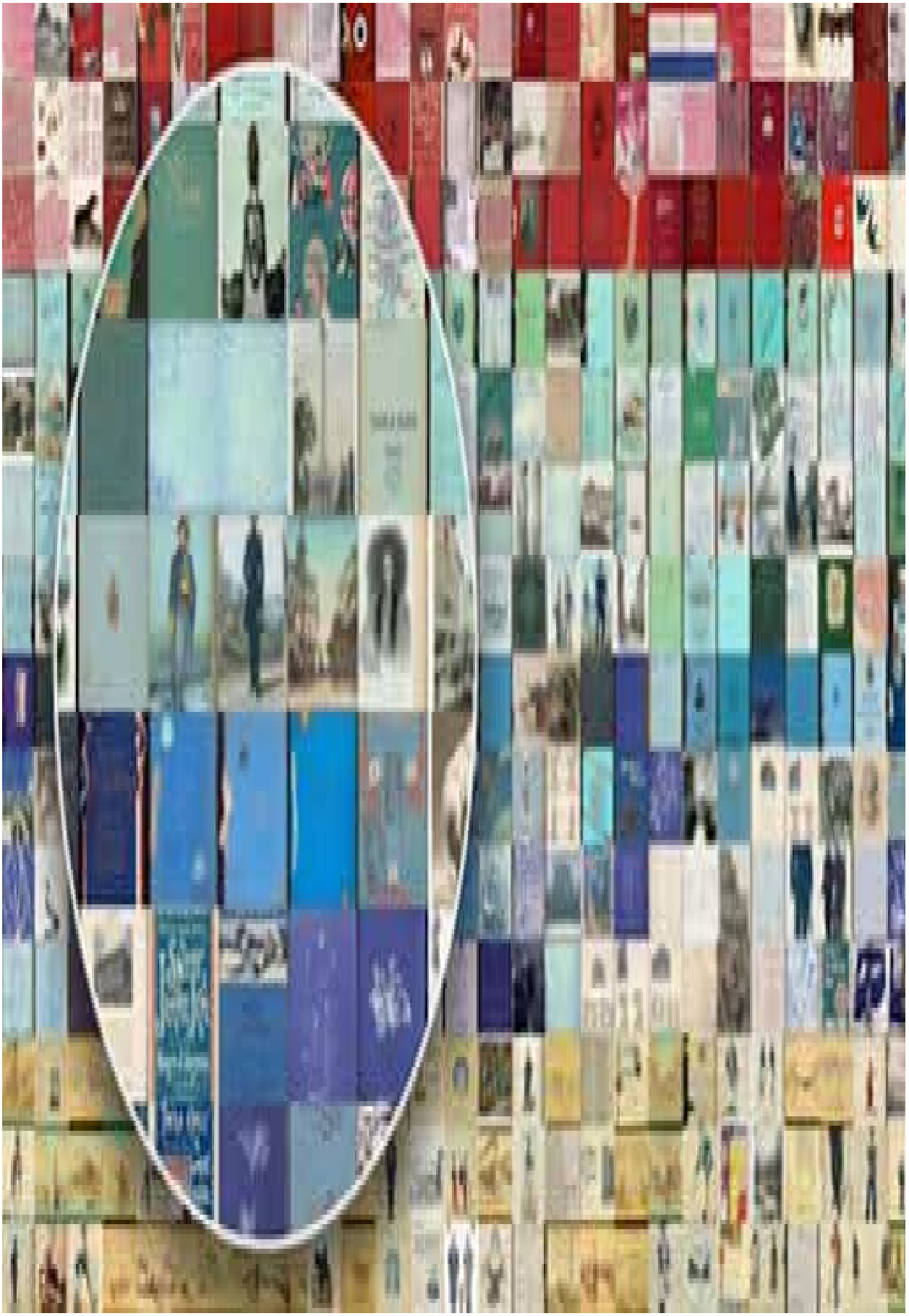


Рисунок 3.2 – Алгоритм вилучення ознак з графічної інформації за допомогою нейронної мережі

Таблиця 3.1 – Типи використання фонових ознак

Код Id	Фонові типи	Опис
0	тон	Максимально білий
0_1		Максимально чорний
1	яскравість	Максимально яскравий
1_1		Максимально тьмяний
2	контур	Чіткий
2_1		Розмитий
3	Інше	Зображення яке не належить до окреслених форматів

$e_{\text{пік}} \in R^{d_{\text{озн}}}$ є векторним поданням ознак вхідного зображення. Ці вектори подаються в дві мережі залишково нейронні мережі для захоплення контексту фону на рівні пропозиції та створення векторного представлення фону на основі ознак зображення:

$$\mathbf{e}_{\text{зоб}}^{\text{пік}} = \left[Q_{\vec{ResNet}(e_{\text{пік}})}, Q_{ResNet^{\leftarrow}(e_{\text{пік}})} \right] \tag{3.4}$$

де $Q_{\vec{ResNet}(e_{\text{пік}})}, Q_{ResNet^{\leftarrow}(e_{\text{пік}})}$ – позначають вихідні дані з прямого і зворотного шарів ResNet, відповідно;

[ ] – операція конкатенації.

Після обчислення трьох видів векторного представлення зображення $\mathbf{e}_{\text{зоб}}$, $\mathbf{e}_{\text{зоб}}^{\text{озн}}$, $\mathbf{e}_{\text{зоб}}^{\text{пік}}$, векторне представлення зображення є лише їхньою конкатенацією:

$$\mathbf{e} = \left[\mathbf{e}_{\text{зоб}}, \mathbf{e}_{\text{зоб}}^{\text{озн}}, \mathbf{e}_{\text{зоб}}^{\text{пік}} \right] \tag{3.5}$$

e потім подається в іншу залишкову нейронну мережу, щоб створити остаточне представлення зображення в контексті загального масиву графічної інформації:

$$r_{\text{зоб}} = \left[Q_{\vec{ResNet}(e)}, Q_{\overleftarrow{ResNet}(e)}\right] \quad (3.6)$$

Нехай nb_{tags} – це кількість тегів, попередньо визначених для даного завдання. Залишкова нейронна мережа використовується для створення десятків зображень у вхідному масиві графічної інформації відповідно до тегів

$$S_{f\,fnn}[n,\ nb_{tags}] \quad (3.7)$$

де $S_{f\,fnn}[i,\ j]$ представляє собою оцінку j-го тегу для і-го зображення.

Загальна робота системи розпізнавання об’єктів на базі штучної нейронної мережі ґрунтується на взаємній роботі окремих модулів, що входять до складу системи.

Архітектура системи розпізнавання об’єктів на базі штучної нейронної мережі наведена на рисунку 3.3.

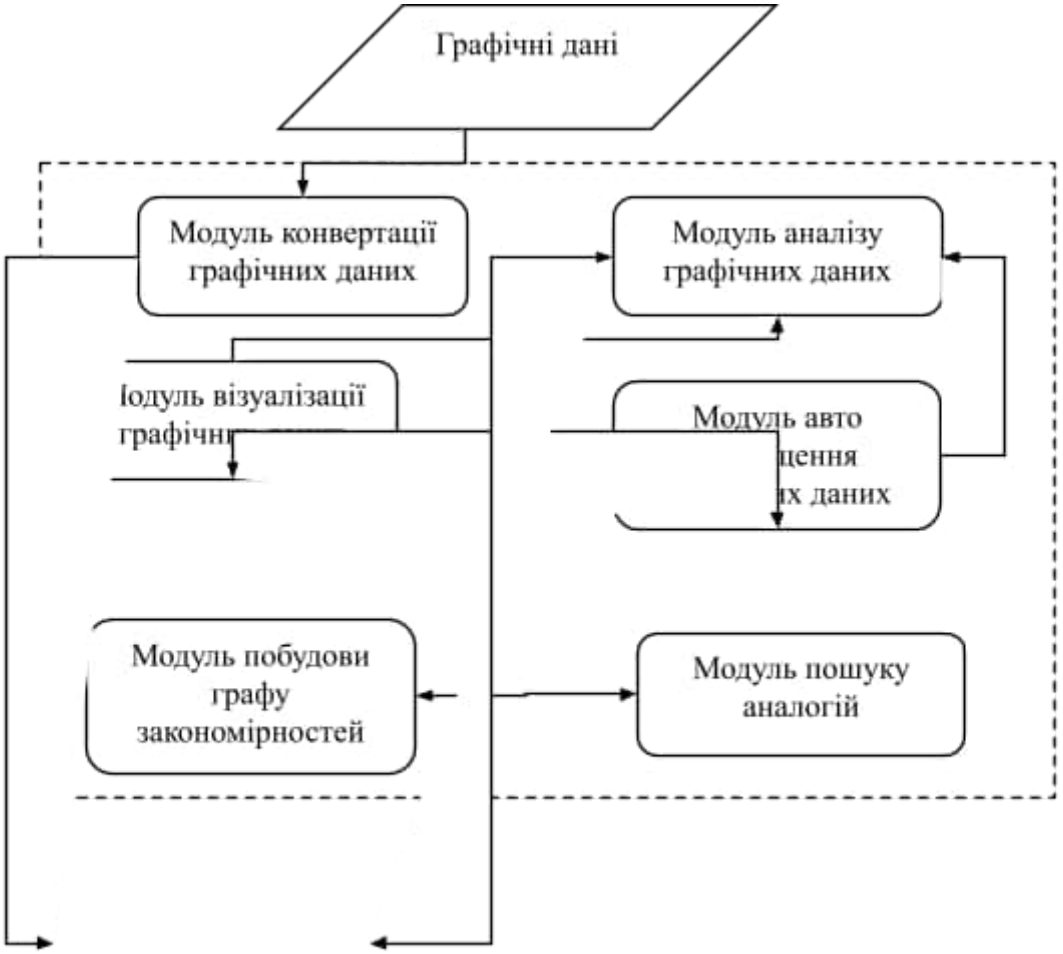


Рисунок 3.3 – Архітектура системи розпізнавання об’єктів на базі штучної нейронної мережі

Взаємодія наведених на рис. 3.3 модулів, яких у системі шість, відбувається за принципом послідовності. Модуль конвертації графічних даних конвертує початкову інформацію у формат доступний для подальшої роботи, всі дані зберігаються у базі зберігання графів закономірностей у спеціалізованому єдиному форматі. Модуль аналізу графічних даних у своєму складі має алгоритми для роботи з графічними даними, їх систематизації, розпізнавання та пошуку аналогій. Модуль візуалізації графічних даних реалізовує принципи відображення графів закономірностей побудованих на основі отриманих залежностей. Модуль авто розміщення графічних даних сприяє базовому

розміщенню графів закономірностей з можливістю подальшого аналізу структурної особливості графів. Модуль пошуку аналогій працює з конвертованим набором графічних даних та результат пошуку є фундаментальною основою побудови результуючого графу. Сховище візуалізації графів закономірностей реалізує принципи зберігання всіх отриманих графів.

Модуль пошуку аналогій реалізує алгоритм пошуку та здійснює структуризацію отриманих значень, та дозволяє:

- здійснювати пошук класових приналежностей;
- виконувати об’єднання схожих значень залежностей;
- реалізувати пошук у ширину кожної вершини графу;
- здійснювати пошук залежностей за заданими параметрами атрибутів.

Щодо об’єднання параметрів у відповідні класові залежності необхідною умовою є фіксація наборів атрибутів графу. Фільтрація класових залежностей відбувається у рамках заданого набору атрибутів.

3.2 Навчання штучної нейронної мережі

Завантажимо графічні дані, як набір даних CIFAR-10. Останній складається з 60000 кольорових зображень 32x32 у 10 класах, по 6000 зображень на клас. У рамках набору CIFAR-10 існує 50 000 навчальних зображень і 10 000 тестових зображень.

Набір даних розділений на п’ять навчальних пакетів і один тестовий пакет, кожен з 10 000 зображень. Тестовий пакет містить рівно 1000 випадково вибраних зображень з кожного класу. Навчальні пакети містять решту зображень у випадковому порядку, але деякі навчальні пакети можуть містити більше зображень з одного класу, ніж з іншого. Між ними навчальні групи містять рівно 5000 зображень з кожного класу.

Дані зображення – це масив uint8 розміром 10000x3072 numpy. Кожен рядок масиву зберігає кольорове зображення розміром 32x32. Перші 1024 записи містять значення червоного каналу, наступні 1024 – зелені, а останні 1024 – сині. Зображення зберігається в порядку великих рядків, так що перші 32 записи масиву є значеннями червоного каналу першого рядка зображення.

Мітки – список з 10000 чисел у діапазоні 0-9. Число в індексі і вказує на мітку і-го зображення в даних масиву.

У роботі побудуємо модель, яка порівнюватиме 2 зображення та дасть відповідь, схожі ці 2 зображення чи ні. Для цього обираємо 2 класи зображень із вихідного набору даних. Потім створимо новий набір даних з подібними і не схожими парами. Схожа пара – це пара, створена із зображень одного класу. Неподібна пара – це пара, створена із 2 зображень різних класів. Більше того, по-перше, трансформуємо зображення у відтінки сірого, по-друге, спробуємо підігнати модель зменшення розмірів, а потім створимо остаточну модель порівняння.

Існує багато методів зменшення розмірів. Найпопулярнішими з них є PCA, tSNE (T-розподілене вкладення стохастичної близькості). Проте скористаємося сучасним методом – UMAP. Апроксимація та проекція

рівномірного різноманіття – це метод зменшення розмірів, який можна використовувати для візуалізації подібно до t-SNE, але також для загального нелінійного зменшення розмірів.

Алгоритм заснований на трьох припущеннях щодо даних:

1. Дані рівномірно розподілені на рімановому багатообразі;
2. Ріманова метрика локально постійна (або може бути апроксимована як така);
3. Колектор локально з'єднаний.

Виходячи з цих припущень, можна моделювати різновид з нечіткою топологічною структурою. Вбудовування здійснюється шляхом пошуку низькорозмірної проекції даних, яка має якнайближчу еквівалентну нечітку топологічну структуру.

3.3 Реалізація системи розпізнавання об’єктів на базі штучної нейронної мережі

Створюємо функцію для завантаження файлів pickle (бінарних):

```
def unpickle(file):
    with open(file, 'rb') as fo:
        dict = pickle.load(fo, encoding='bytes')
    return dict
```

Набір даних містить інший файл, який називається batches.meta. Він також містить об’єкт словника Python. Та має такі записи: label_names – список з 10 елементів, який дає значущі імена числовим міткам у масиві labels, описаному вище. Наприклад, label_names[0] = "airplane", label_names[1] = "automobile" тощо.

Встановлюємо імена навчальних і тестових файлів:

```
list_train_files = ['data_batch_1','data_batch_2','data_batch_3',
'data_batch_4','data_batch_5'] # навчальні двійкові файли
list_test_files = ['test_batch'] # тестові двійкові файли
```

Завантажуємо дані навчання та випробування:

```
list_train_data = [unpickle('./cifar-10-batches-py/'+file_name)
for file_name in list_train_files]
list_test_data = [unpickle('./cifar-10-batches-py/'+file_name)
for file_name in list_test_files]
```

Тепер об’єднаємо всі дані в масив зображень і масив міток для навчальних і тестових наборів даних:

```
train_labels = [] # навчальні елементи
train_images = [] # підготовка зображень
# об’єднання
for data in list_train_data:
    train_images.append(data[b'data'])
    train_labels.append(data[b'labels'])
# конвертація масиву
train_images = np.vstack(train_images)
train_labels = np.hstack(train_labels)
```

Перевіряємо форму даних (тренувальні зображення):

```
train_images.shape
```

Висновок:

(50000, 3072)

Перевіряємо форму даних (тренувальні мітки):

```
train_labels.shape
```

Висновок:

(50000)

Отримано 50000 рядків (зображень). Робимо те ж саме для тестових даних:

```
test_labels = []
test_images = []
for data in list_test_data:
    test_images.append(data[b'data'])
    test_labels.append(data[b'labels'])
test_images = np.vstack(test_images)
test_labels = np.hstack(test_labels)
```

Кількість зображень і міток:

```
test_images.shape
```

Висновок:

(10000, 3072)

```
test_labels.shape
```

Висновок:

(10000,)

Таким чином, тестові дані містять 10000 зображень.

Обираємо лише клас 0 і клас 1 з навчальних і тестових частин (це може бути лише 2 класи):

```
flags = (train_labels==0)|(train_labels==1)
train_images = train_images[flags,]
train_labels = train_labels[flags,]
train_images.shape
```

Висновок:

(10000, 3072)

Отримуємо 10 000 зображень у навчальній частині. Робимо те ж саме для тестової частини:

```
flags = (test_labels==0)|(test_labels==1)
test_images = test_images[flags,]
test_labels = test_labels[flags,]
test_images.shape
```

Висновок:

(2000, 3072)

Маємо 2000 зображень.

Побудуємо кілька зображень, 9 перших зображень із зміною форми 32x32x3:

```
plt.rcParams['figure.figsize'] = [8,8] # розмір графіка
plt.rcParams['figure.dpi'] = 50 # DPI графіка
fig, axs = plt.subplots(3,3) # сітка графіків 3x3
# графік
k = 0
for i in range(3):
    for j in range(3):
        r = train_images[k][:1024].reshape(32,32)
        g = train_images[k][1024:2048].reshape(32,32)
        b = train_images[k][2048:].reshape(32,32)
        img = np.stack([r,g,b],axis=-1)
        axs[i,j].imshow(img), # reshape into 32x32x3
        k = k+1
plt.show()
```

Рисунок 3.4 – 9 перших зображень із зміною форми 32x32x3

Перетворимо ці зображення в градації сірого. Це перший крок процедури зменшення розмірів:

```
gray_train_images = []
for k in range(train_images.shape[0]):
    r = train_images[k][:1024]
    g = train_images[k][1024:2048]
    b = train_images[k][2048:]
    img = np.mean(np.float32(np.stack([r,g,b],axis=-1)),axis=-1)/255.
    gray_train_images.append(img)
gray_train_images = np.array(gray_train_images)
gray_train_images.shape
```

Висновок:

(10000, 1024)

Для набору тестових даних:

```
gray_test_images = []
for k in range(test_images.shape[0]):
    r = test_images[k][:1024]
    g = test_images[k][1024:2048]
    b = test_images[k][2048:]
    img = np.mean(np.float32(np.stack([r,g,b],axis=-1)),axis=-1)/255.
    gray_test_images.append(img)
gray_test_images = np.array(gray_test_images)
gray_test_images.shape
```

Висновок:

(2000, 1024)

Побудуємо ті самі зображення у форматі відтінків сірого:

```
plt.rcParams['figure.figsize'] = [8,8] # розмір графіка
plt.rcParams['figure.dpi'] = 50 # DPI графіка
fig, axs = plt.subplots(3,3) # сітка графіків 3x3
# графік
k = 0
for i in range(3):
    for j in range(3):
        img = gray_train_images[k].reshape(32,32)
        axs[i,j].imshow(img,cmap='gray'), # змінити форму на 32x32
        k = k+1
plt.show()
```

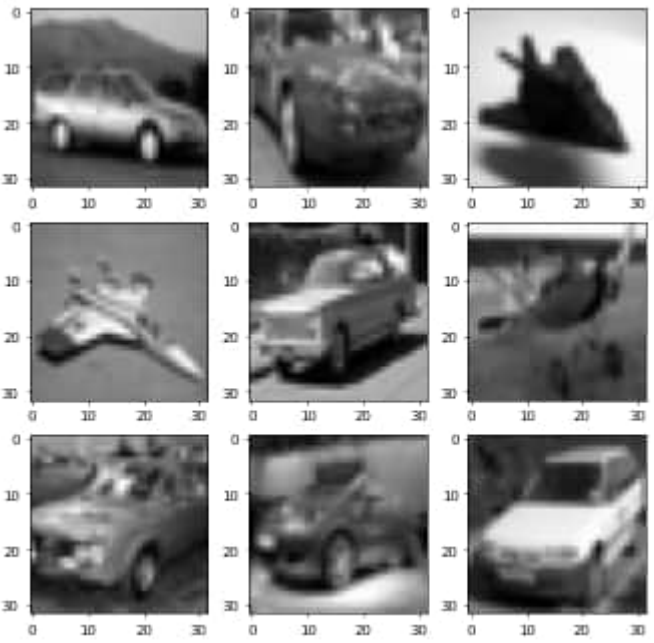


Рисунок 3.5 – Зображення у форматі відтінків сірого 32x32

Перевіряємо унікальні етикетки. Норма 2.

```
u_labels = np.unique(train_labels)
u_labels
```

Висновок:

масив ([0, 1])

Побудуємо модель зменшення розмірності за допомогою алгоритму UMAP. Вибираємо 2 вихідні компоненти. Є можливість вибрати менше або більше вихідних компонентів, але чим менше компонентів, тим більше інформації ми втратимо. Чим більше компонентів, тим складнішою буде модель. Складна модель не означає, що ця модель добре працюватиме з новими даними. Таким чином, обираємо 2 компоненти.

```
# побудувати та підігнати модель UMAP
dr_m = umap.UMAP(n_neighbors=16, # кількість сусідів
                 n_components=2, # 10 компонентів
                 min_dist=0.1, # щільність точок
                 random_state=0).fit(gray_train_images) # random_state
```

Перетворюємо навчальні та тестові набори даних у простір нижнього виміру (10 функцій):

```
train_X_t = dr_m.transform(gray_train_images)
test_X_t = dr_m.transform(gray_test_images)
```

Перевірка:

```
plt.rcParams['figure.figsize'] = [8,8]
plt.rcParams['figure.dpi'] = 100
plt.scatter(
    train_X_t[:, 0],
    train_X_t[:, 1],
    c = [sns.color_palette()[x] for x in train_labels])
plt.gca().set_aspect('equal', 'datalim')
plt.title('UMAP projection of the CIFAR-10 data set', fontsize=14)
plt.show()
```

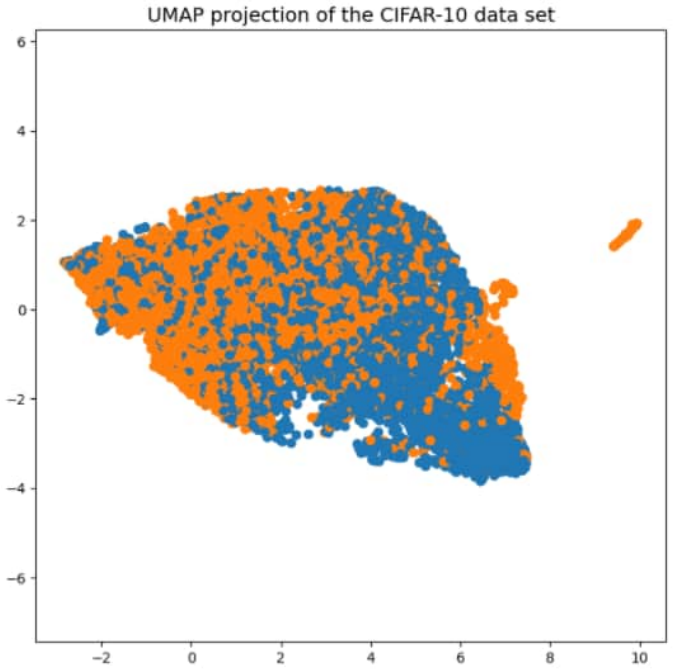


Рисунок 3.6 – Модель зменшення розмірності

Вихідні розміри:

```
print(train_X_t.shape)
print(test_X_t.shape)
```

(10000, 2)

(2000, 2)

Таким чином, вектори мають 10 компонентів. Далі будемо працювати з цими перетвореними даними.

Створимо новий набір навчальних даних, де 2 зображення подібності (скорочені вектори) будуть схожістю, якщо вони є одним і тим же класом. Встановлюємо результат 1. В іншому випадку встановлюємо результат 0. Вектор об’єкту буде абсолютною різницею між 2 зменшеними векторами.

```
n = 10000 # обсяг нового набору навчальних даних
train_X = [] # новий набір даних - абсолютні відмінності між 2 зменшеними векторами зображень
train_y = [] # нові мітки; 1 - подібність, 2 - неподібність
np.random.seed(0) # для відтворення тих самих результатів
for i in range(n):
    for label in u_labels:
        # виберіть 2 вектори подібності за випадковим принципом
        sim_index1, sim_index2 = np.random.permutation(np.where(label==train_labels)[0])[:2]
        # вибрати 1 вектор неподібності за випадковим принципом
        nsim_index  = np.random.permutation(np.where(label!=train_labels)[0])[0]
        # додати абсолютні відмінності між векторами подібності
        train_X.append(np.abs(train_X_t[sim_index1]-train_X_t[sim_index2]))
        train_y.append(1) # додати мітки
        # додати абсолютні відмінності між неподібними векторами
        train_X.append(np.abs(train_X_t[sim_index1]-train_X_t[nsim_index]))
        train_y.append(0) # додати мітки
train_X = np.array(train_X)
train_y = np.array(train_y)
train_X.shape
```

Висновок:

(40000, 2)

Набір даних готовий. Створюємо тестовий набір даних таким же чином:

```
n = 5000
test_X = []
test_y = []
np.random.seed(0)
for i in range(n):
    for label in u_labels:
        sim_index1, sim_index2 = np.random.permutation(np.where(label==test_labels)[0])[:2]
        nsim_index  = np.random.permutation(np.where(label!=test_labels)[0])[0]
        test_X.append(np.abs(test_X_t[sim_index1]-test_X_t[sim_index2]))
        test_y.append(1)
        test_X.append(np.abs(test_X_t[sim_index1]-test_X_t[nsim_index]))
        test_y.append(0)
test_X = np.array(test_X)
test_y = np.array(test_y)
test_X.shape
```

Висновок:

(20000, 2)

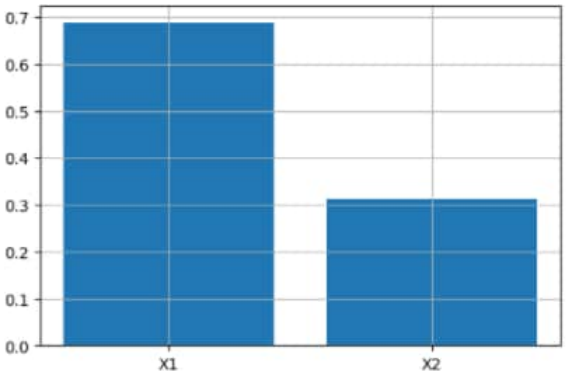


Рисунок 3.7 – Модель точності вибірки

З рисунка 3.7 видно, що «X1» є найважливішою характеристикою. Розглянемо модель точності, використовуючи навчальні та тестові набори даних:

```
pr_train_y = m.predict(train_X) # прогнозувати значення за даними навчання

pr_test_y = m.predict(test_X) # прогнозувати значення за тестовими даними
```

Точність в обох випадках:

```
1  accuracy_score(pr_train_y,train_y) # тренування точності
```

Вихід:

0,5529

```
1  accuracy_score(pr_test_y,test_y) # перевірка точності
```

Вихід:

0,5596

Точність набору тестових даних трохи вище.

Значення матриці, які були нормовані статистикою передбачення:

```
pd.DataFrame(np.round(confusion_matrix(pr_train_y,train_y, normalize = 'pred'),2),
             columns = ['0','1'],
             index = ['0','1'])
```

Вихід:

	0	1
0	0,5	0,39
1	0,5	0,61

```
pd.DataFrame(np.round(confusion_matrix(pr_test_y,test_y, normalize = 'pred'),2),
             columns = ['0','1'],
             index = ['0','1'])
```

Вихід:

	0	1
0	0,52	0,4
1	0,48	0,6

Матриця плутанини є дуже важливим інструментом для вивчення точності кожного класу. В ідеальному випадку необхідно мати 1 на головній діагоналі, а інші значення повинні бути 0. 0 клас (для навчальних і тестових наборів даних) має точність приблизно 50-52% і 1 60-61%.

3.4 Верифікація результатів дослідження

Повна статистика цих наборів даних наведена в таблиці 3.2.

Таблиця 3.2– Статистика наборів даних

Назва набору даних	Число класів	Число об’єктів	Опис
Canadian Institute For Advanced Research	10	60000	Фотографії об'єктів різних класів, кольорові зображення 32x32 пікселя
MNIST	10	70000	Рукописні цифри, чорно-білі зображення 32x32 пікселя
ImageNet	> 21000	> 14000000	Фотографії із зазначенням класів об'єктів на зображенні та їх позицій
Common Objects in Context	91	328000	Фотографії складних повсякденних

			сцен, що містять об'єкти в їхньому природному оточенні
FAce Semantic SEGmentation	19	340	Зображення осіб людей з різними кутами повороту як в оригіналі, так і в сегментованому вигляді
ImageNet Max Planck Institute	410	25000	Зображення повсякденної діяльності людей у різних позах

Усі експерименти проводилися на графічному процесорі NVIDIA GeForce GTX 1080 Ti з 11 ГБ оперативної пам’яті. Навчання на кожному вищезгаданому наборі даних займає від 1 до 3 годин. Якщо не вказано, поділ наборів даних на набори для навчання, перевірки та тестування виконується у співвідношенні 60%:20%:20%.

Експерименти діляться на три групи:

Експерименти в першій групі були спрямовані на оцінку системи розпізнавання об’єктів на базі штучної нейронної мережі на шести наборах даних.

Мета другого експерименту – проаналізувати вплив різних функцій введення на продуктивність системи розпізнавання об’єктів на базі штучної нейронної мережі в різних мовах.

Остання група експериментів мала на меті з’ясувати, наскільки добре система розпізнавання об’єктів на базі штучної нейронної мережі справляється з невеликими наборами даних.

Таблиця 3.3 – Продуктивність системи розпізнавання об’єктів на базі штучної нейронної мережі для набору даних CIFAR-10

Параметр пошуку / метрика	Образ людини	Образ тварини	Об’єкти цифр	Об’єкти природи	Об’єкти будівель	Різне	Об’єкти транспорту	Середній показник
Точність	96,12	97,36	90,56	93,65	95,02	94,15	98,74	95,0857143
Повнота	98,46	98,54	94,21	95,67	96,34	96,98	99,12	97,0457143
f-міра	97,02	98,01	92,55	94,64	95,98	95,88	99,00	96,1542857

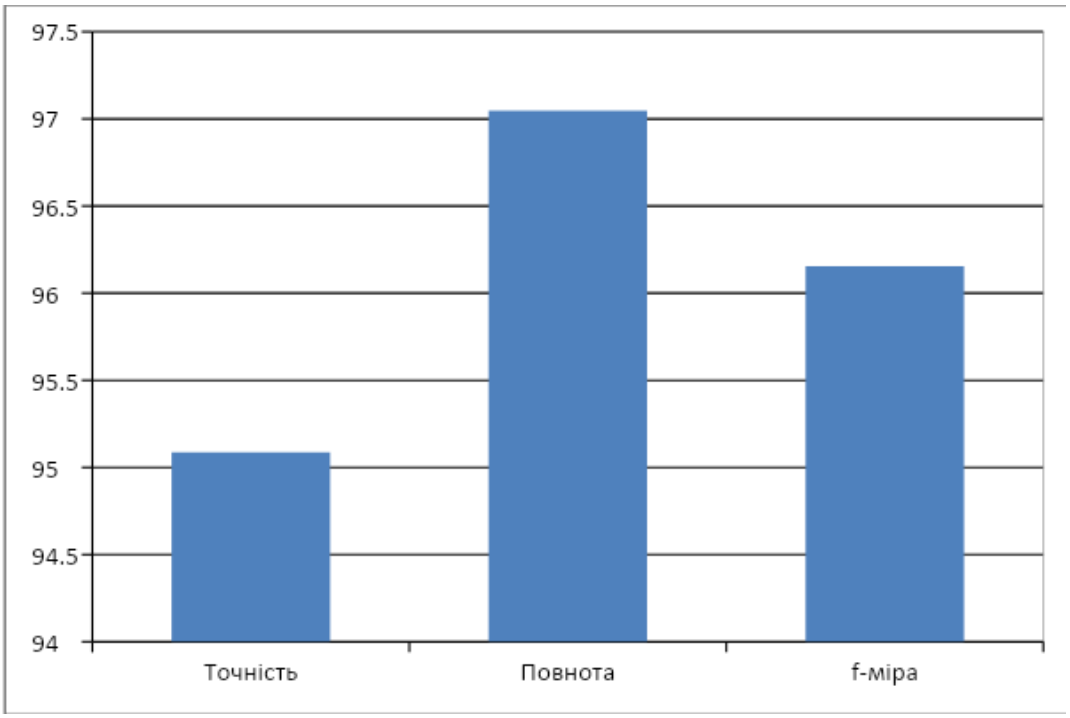


Рисунок 3.8 – Гістограма розподілу відсотка продуктивності за параметрами пошуку для системи розпізнавання об’єктів на базі штучної нейронної мережі щодо дослідження на наборі даних CIFAR-10

Таблиця 3.4 – Продуктивність системи розпізнавання об’єктів на базі штучної нейронної мережі на наборі даних MNIST

Параметр пошуку / метрика	Образ людини	Образ тварини	Об’єкти цифр	Об’єкти природи	Об’єкти будівель	Різне	Об’єкти транспорту	Середній показник
Точність	-	-	98,56	-	-	56,55	-	77,555
Повнота	-	-	99,21	-	-	66,68	-	82,945
f-міра	-	-	98,95	-	-	59,18		79,065

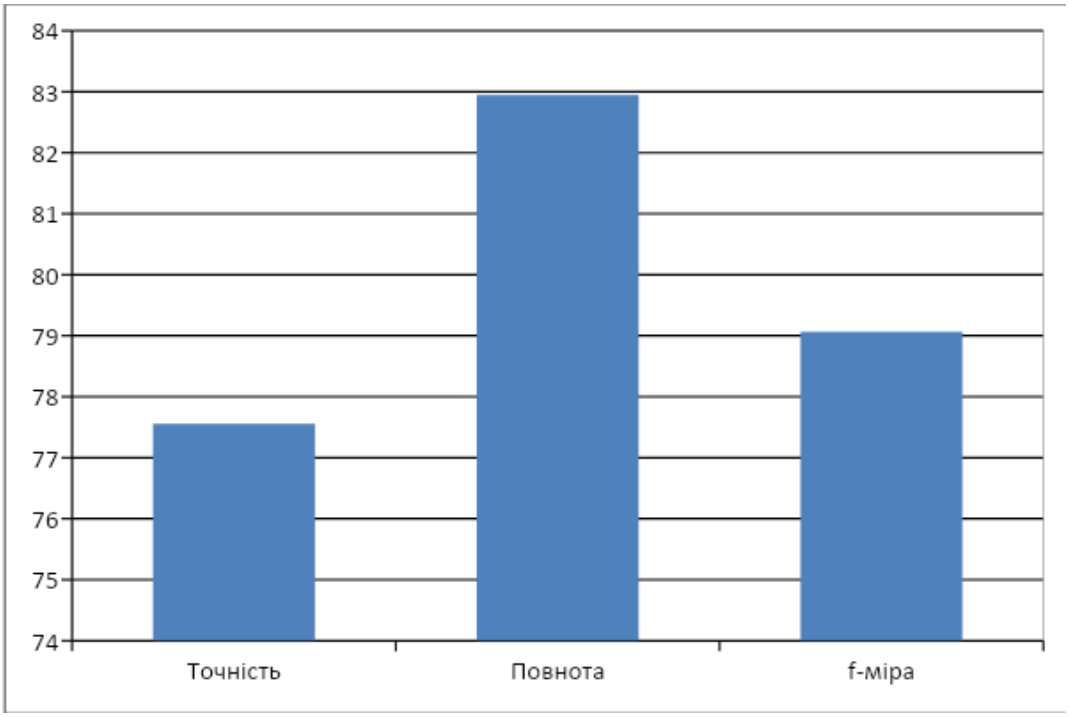


Рисунок 3.9 – Гістограма розподілу відсотка продуктивності за параметрами пошуку для системи розпізнавання об’єктів на базі штучної нейронної мережі щодо дослідження на наборі даних MNIST

Таблиця 3.5 – Продуктивність системи розпізнавання об’єктів на базі штучної нейронної мережі на наборі даних FAce Semantic SEGmentation із застосуванням методу перехресної перевірки

Параметр пошуку / метрика	Образ людини	Образ тварини	Об’єкти цифр	Об’єкти природи	Об’єкти будівель	Різне	Об’єкти транспорту	Середній показник
Точність	86,22	87,36	70,26	83,65	65,02	84,15	88,74	80,7714286
Повнота	88,36	88,54	74,21	85,57	76,64	86,68	89,29	84,1842857
f-міра	87,22	88,01	72,95	94,69	65,98	85,88	89,22	83,4214286

Перша група експериментів починається з оцінки системи розпізнавання об’єктів на базі штучної нейронної мережі на двох наборах даних: CIFAR-10 і MNIST. Результати експерименту показують, що модель досягла хороших показників. Таблиця 3.3 показує оціночні показники, розраховані на тестовому наборі даних CIFAR-10.

Таблиця 3.6 – Продуктивність системи розпізнавання об’єктів на базі штучної нейронної мережі у наборі даних FAce Semantic SEGmentation з перенавчанням на наборі даних CoCo

Назва методу / системи	Параметр пошуку / метрика	Образ людини	Образ тварини	Об’єкти цифр	Об’єкти природи	Об’єкти будівель	Різне	Об’єкти транспорту	Середній показник
Розроблена автоматизована	Точність	86,22	87,36	70,26	83,65	65,02	84,15	88,74	80,7714286

системи без попереднього навчання	Повнота	88,36	88,54	74,21	85,57	76,64	86,68	89,29	84,1842857
	f-міра	87,22	88,01	72,95	94,69	65,98	85,88	89,22	83,4214286
Розроблена автоматизована системи з перенавчанням на наборі даних CoCo	Точність	89,29	89,26	82,23	85,66	69,62	83,25	87,39	83,8142857
	Повнота	98,36	98,04	84,33	85,53	66,66	96,69	89,83	88,4914286
	f-міра	97,22	98,01	82,95	94,39	65,38	95,88	82,22	88,0071429

Наступний експеримент оцінює запропоновану модель на наборі даних FAce Semantic SEGmentation. Для цього експерименту використовується метод k-кратної перехресної перевірки через малий розмір набору даних. Результати експерименту, наведені в таблиці 3.5, вказують на те, що модель все ще демонструє хорошу продуктивність при навчанні на невеликому наборі даних. Щоб покращити продуктивність моделі в наборі даних FAce Semantic SEGmentation, її попередньо навчили на наборі даних CoCo. Це допомогло покращити продуктивність моделі приблизно на 4%, з 82,79% до 86,77% (для повного порівняння див. таблиці 3.6).

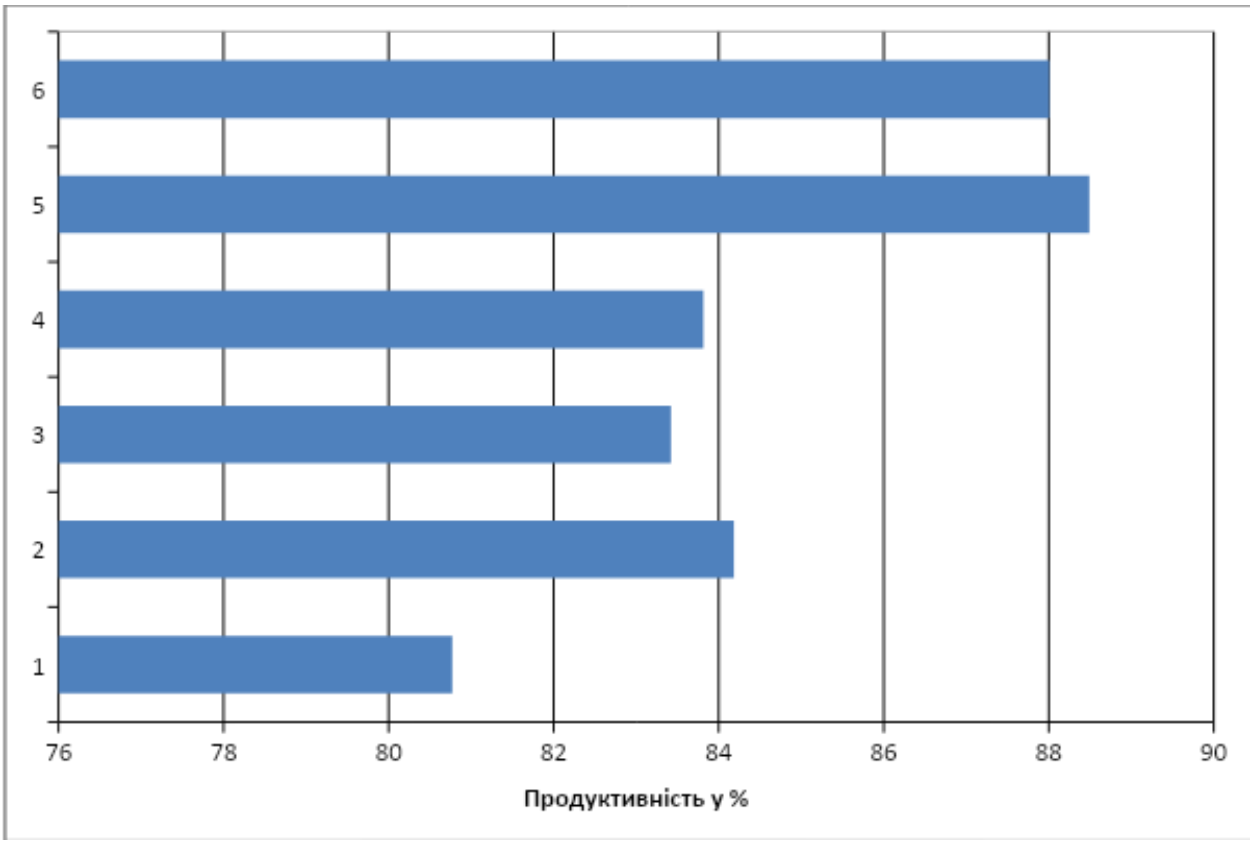


Рисунок 3.10 – Гістограма розподілу відсотка продуктивності за параметрами пошуку для системи розпізнавання об’єктів на базі штучної нейронної мережі у наборі даних FAce Semantic SEGmentation з перенавчанням на наборі даних CoCo

Таблиця 3.7 – Продуктивність системи розпізнавання об’єктів на базі штучної нейронної мережі на наборі даних ImageNet

Параметр пошуку / метрика	Образ людини	Образ тварини	Об’єкти цифр	Об’єкти природи	Об’єкти будівель	Різне	Об’єкти транспорту	Середній показник
Точність	89,67	92,69	87,96	83,65	79,65	94,21	88,11	87,9914286
Повнота	78,64	79,36	74,21	89,17	80,29	90,36	89,54	83,0814286
f-міра	90,11	79,54	92,95	92,39	86,52	94,25	89,96	89,3885714

Таблиця 3.8 – Продуктивність системи розпізнавання об’єктів на базі штучної нейронної мережі на наборі даних Max Planck Institute

Параметр пошуку / метрика	Образ людини	Образ тварини	Об’єкти цифр	Об’єкти природи	Об’єкти будівель	Різне	Об’єкти транспорту	Середній показник
Точність	92,03	98,54	78,25	89,21	78,65	87,95	89,36	87,7128571
Повнота	90,25	96,26	74,26	87,89	80,11	89,33	93,21	87,33
f-міра	91,66	93,14	79,22	88,98	79,24	88,54	92,45	87,6042857

Після цього запропонована модель тестується на наборах даних: ImageNet та Max Planck Institute. Результати цих експериментів наведені в таблицях 3.7, 3.8. Таблиця 3.9 показує продуктивність запропонованої моделі системи розпізнавання об’єктів на базі штучної нейронної мережі у порівнянні з деякими передовими моделями, що відображають зображення набору даних різних об’єктів. Оцінка за набором даних CoCo є останнім експериментом у першій групі.

Таблиця 3.9 – Продуктивність системи розпізнавання об’єктів на базі штучної нейронної мережі у порівняння з відомими системами

Назва системи / методу	Назва параметру		
	Точність	Повнота	f-міра
Метод попіксельного порівняння	87,57	88,77	89,42
Метод SURF	92,58	90,26	91,54
Метод WLSF	88,51	89,57	90,25
Метод порівняння текстури на зображеннях	93,65	**92,99**	92,54
Система розпізнавання об’єктів на базі штучної нейронної мережі	**95,62**	91,88	**95,33**

Через труднощі з пошуком фотографій складних повсякденних сцен, що містять об'єкти в їхньому природному оточенні, які складових набору даних для завдання розпізнавання використовується набір даних CoCo, розроблений для завдання сегментації різних поколінь зображень. У цьому експерименті замість характеристик об’єктів використовується інформація про сегментацію останніх. Результат тестування наведено в таблиці 3.10.

Таблиця 3.10 – Продуктивність системи розпізнавання об’єктів на базі штучної нейронної мережі на наборі даних CoCo

Параметр пошуку / метрика	Образ людини	Образ тварини	Об’єкти цифр	Об’єкти природи	Об’єкти будівель	Різне	Об’єкти транспорту	Середній показник
Точність	86,22	88,54	87,42	96,21	90,12	89,36	91,54	89,9157143
Повнота	88,69	86,54	87,33	92,12	82,17	98,11	90,99	89,4214286
f-міра	87,54	87,52	87,00	94,52	96,49	94,54	91,02	91,2328571

За наведеними результатами тестування видно, що система розпізнавання об’єктів на базі штучної нейронної мережі на наборі даних CoCo показує найбільший відсоток продуктивності на об’єктах природи 96,21%, найнижчий відсоток отримано про аналізі образів тварин.

Висновки до розділу

У третьому розділі розроблено систему розпізнавання об’єктів на базі штучної нейронної мережі запропоновано, яка ґрунтується на низці моделей направлених на аналіз графічних даних до складу яких входить: модель налаштування програмного блоку системи реєстрації вхідного масиву графічних даних шляхом компенсації оптичних аберацій через побудову

математичної моделі спотворень зображення і проведення процедури зворотної дисторсії, а також зменшення впливу статистичного шуму через побудову математичної моделі розподілу ймовірності і проведення процедури просторової фільтрації; модель попередньої обробки та попередньої сегментації графічних даних на основі морфологічних методів, що може бути застосована до матриць кольорових зображень через включення у методику алгоритмів виділення зв’язних компонент та порогових алгоритмів; модель оптимізації компонент загального комплексу по роботі з наборами зображень. Загальна множина описаних моделей, складає сімейство яке, має основну архітектуру.

ВИСНОВКИ

У межах даної роботи здійснено оптимізацію в навчанні глибоких моделей.

У роботі виконано низку завдань:

- розкрито методи навчання глибоких моделей;
- окреслено математичні аспекти навчання глибоких моделей;
- запропонувано алгоритм реалізації навчання глибоких моделей;
- розроблено структуру системи розпізнавання об’єктів на базі штучної нейронної мережі;
- здійснено навчання штучної нейронної мережі;
- виконано реалізацію системи розпізнавання об’єктів на базі штучної нейронної мережі;
- здійснено верифікацію результатів дослідження.

На основі вищевикладеного можна зробити наступний висновок:

Навчання нейронних мереж складне завдання по ряду причин. Нерідко процес пошуку адекватної нейромережевої моделі закінчується з нульовим результатом і дуже велику роль грає досвід розробника нейромережевих моделей. Для отримання нейромережевої моделі вирішальним завдання із заданим показником якості зазвичай необхідно пройти наступні кроки: спочатку необхідно підготувати дані, визначитися з типом мережі, визначити входи і виходи, вирішити задачу первісної структури мережі – шари і нейрони в них, далі необхідно навчити мережу, тобто підібрати коефіцієнти зв'язків між нейронами, перевірити навчену мережу на валідаційній вибірці і у підсумку перевірити в реальній роботі. При цьому всі кроки тісно пов'язані між собою і неякісне опрацювання по одному з них веде, в кінцевому рахунку, до тривалого навчання мережі або взагалі до отримання неправильно працюючої нейромережі. Існує велика кількість методів і алгоритмів попередньої підготовки даних, але всі вони в значній мірі спираються на досвід розробника.

В даний час домінуючими є два типи архітектур: згорткові мережі, які успішно застосовуються для задач комп'ютерного зору, і рекурентні мережі, що активно використовуються для обробки природної мови. Ранні згорткові мережі навчалися шляхом комбінації навчання з учителем та без вчителя з використанням автокодувальників та глибоких мереж довіри. Сучасні методи, такі як залишкове навчання, дозволяють використовувати тільки навчання з учителем і відмовитися від навчання, що прискорює та спрощує процес навчання. Також важливим напрямом у розвитку згорткових нейронних мереж є передача навчання (transfer learning). Цей підхід передбачає використання нейронних мереж, навчених одних даних, на вирішення інших типів завдань. При цьому застосовується тонка настройка мережі і до навчання на даних від завдання, що є головним. В результаті скорочується час навчання і розширюється сфера застосування попередньо навчених нейронних мереж. Перспективним також є спільне використання згорткових та рекурентних нейронних мереж з навчанням із підкріпленням.

Розроблено систему розпізнавання об’єктів на базі штучної нейронної мережі запропоновано, яка ґрунтується на низці моделей направлених на аналіз графічних даних до складу яких входить: модель налаштування програмного блоку системи реєстрації вхідного масиву графічних даних шляхом компенсації оптичних аберацій через побудову математичної моделі спотворень зображення і проведення процедури зворотної дисторсії, а також зменшення впливу статистичного шуму через побудову математичної моделі розподілу ймовірності і проведення процедури просторової фільтрації; модель попередньої обробки та попередньої сегментації графічних даних на основі морфологічних методів, що може бути застосована до матриць кольорових зображень через включення у методику алгоритмів виділення зв’язних компонент та порогових алгоритмів; модель оптимізації компонент загального комплексу по роботі з наборами зображень. Загальна множина описаних моделей, складає сімейство яке, має основну архітектуру.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. LeCun Y., Bengio Y., Hinton G. Deep Learning // Nature. 2015. Vol. 521. P. 436–444. DOI: 10.1038/nature14539.

2. Rav`ı D., Wong Ch., Deligianni F., et al. Deep Learning for Health Informatics // IEEE Journal of Biomedical and Health Informatics. 2017. Vol. 21, №. 1. P. 4–21. DOI: 10.1109/JBHI.2016.2636665.

3. Schmidhuber J. Deep Learning in Neural Networks: an Overview // Neural Networks. 2015. Vol. 1. P. 85–117, DOI: 10.1016/j.neunet.2014.09.003.

4. McCulloch W.S., Pitts W. A Logical Calculus of the Ideas Immanent in Nervous Activity // The Bulletin of Mathematical Biophysics. 1943. Vol. 5, №. 4. P. 115–133. DOI: 10.1007/BF02478259.

5. Hinton G., Salakhutdi№v R. Reducing the Dimensionality of Data with Neural Networks // Science. 2006. Vol. 313, №. 5786. P. 504–507. DOI: 10.1126/science.1127647.

6. Hinton G.E., Osindero S., Teh Y.-W. A Fast Learning Algorithm for Deep Belief Nets // Neural Computing. 2006. Vol. 18, №. 7. P. 1527–1554. DOI: 10.1162/neco.2006.18.7.1527.

7. S´ıma J. Loading Deep Networks Is Hard // Neural Computation. 1994. Vol. 6, №. 5. P. 842–850. DOI: 10.1162/neco.1994.6.5.842.

8. Windisch D. Loading Deep Networks Is Hard: The Pyramidal Case // Neural Computation. 2005. Vol. 17, №. 2. P. 487–502. DOI: 10.1162/0899766053011519.

9. Gomez F.J., Schmidhuber J. Co-Evolving Recurrent Neurons Learn Deep Memory POMDPs // Proc. of the 2005 Conference on Genetic and Evolutionary Computation (GECCO) (Washington, DC, USA, June 25–29, 2005), 2005. P. 491–498. DOI: 10.1145/1068009.1068092.

10. Ciresan D.C., Meier U., Gambardella L.M., Schmidhuber J. Deep, Big, Simple Neural Nets for Handwritten Digit Recognition // Neural Computation. 2010. Vol. 22, №. 12. P. 3207–3220. DOI: 10.1162/NECO_a_00052.

11. He K., Zhang X., Ren S., et al. Deep Residual Learning for Image Recognition // 2016 IEEE Conference on Computer Vision and Pattern Recognition (Las Vegas, NV, USA, 27–30 June 2016), 2016. P. 770–778. DOI: 10.1109/CVPR.2016.90. Обзор методов обучения глубоких нейронных сетей 42 Вестник ЮУрГУ. Серия «Вычислительная математика и информатика»

12. Rumelhart D.E., Hinton G.E., McClelland J.L. A General Framework for Parallel Distributed Processing // Parallel Distributed Processing: Explorations in the Microstructure of Cognition. 1986. Vol. 1, P. 45–76. DOI: 10.1016/B978-1-4832-1446-7.50010-8.

13. LeCun Y., Bottou L., Orr G.B. Efficient BackProp // Neural Networks: Tricks of the Trade. 1998. P. 9–50. DOI: 10.1007/3-540-49430-8_2.

14. Broomhead D.S., Lowe D. Multivariable Functional Interpolation and Adaptive Networks // Complex Systems. Vol. 2. P. 321–355. DOI: 10.1016/0167-6911(92)90025-N.

15. Stone M.N. The Generalized Weierstrass Approximation Theorem // Mathematics Magazine. 1948. Vol. 21, №. 4. P. 167–184. DOI: 10.2307/3029750.

16. Горбань А.Н., Дунин-Барковский В.Л., Кирдин А.Н. и др. Нейроинформатика. Новосибирск: Наука. 1998. С. 296.

17. Hornik K., Stinchcombe M., White H. Multilayer Feedforward Networks are Universal Approximators // Neural Networks. 1989. Vol. 2, №. 5. P. 359–366. DOI: 10.1016/0893-6080(89)90020-8.

18. Mhaskar H.N., Micchelli Ch.A. Approximation by Superposition of Sigmoidal and Radial Basis Functions // Advances in Applied Mathematics. 1992. Vol. 13, №. 13. P. 350–373. DOI: 10.1016/0196-8858(92)90016-P.

19. Hebb D.O. The Organization of Behavior. New York: Wiley. 1949. 335 p. DOI: 10.1016/S0361-9230(99)00182-3.

20. №vikoff A.B. On Convergence Proofs on Perceptrons // Symposium on the Mathematical Theory of Automata. 1962. Vol. 12. P. 615–622.

21. Rosenblatt F. The Perceptron: a Probabilistic Model for Information Storage and Organization in the Brain // Psychological Review. 1958. P. 65–386. DOI: 10.1037/h0042519.

22. Widrow B., Hoff M. Associative Storage and Retrieval of Digital Information in Networks of Adaptive Neurons // Biological Prototypes and Synthetic Systems. 1962. Vol. 1. 160 p. DOI: 10.1007/978-1-4684-1716-6_25.

23. Narendra K.S., Thathatchar M.A.L. Learning Automata – a Survey // IEEE Transactions on Systems, Man, and Cybernetics. 1974. Vol. 4. P. 323–334. DOI: 10.1109/tsmc.1974.5408453.

24. Rosenblatt F. Principles of Neurodynamics; Perceptrons and the Theory of Brain Mechanisms. 1962. Washington: Spartan Books. 616 p. DOI: 10.1007/978-3-642-70911-1_20.

25. Grossberg S. Some Networks That Can Learn, Remember, and Reproduce any Number of Complicated Space-Time Patterns // International Journal of Mathematics and Mechanics. 1969. Vol. 19. P. 53–91. DOI: 10.1512/iumj.1970.19.19007.

26. Kohonen T. Correlation Matrix Memories // IEEE Transactions on Computers. 1972. Vol. 100, №. 4. P. 353–359. DOI: 10.1109/tc.1972.5008975.

27. von der Malsburg C. Self-Organization of Orientation Sensitive Cells in the Striate Cortex // Kybernetik. 1973. Vol. 14, №. 2. P. 85–100. DOI: 10.1007/bf00288907.

28. Willshaw D.J., von der Malsburg C. How Patterned Neural Connections Can Be Set Up by Self-Organization // Proceedings of the Royal Society London B. 1976. Vol. 194. P. 431–445. DOI: 10.1098/rspb.1976.0087. А.В. Созыкин 2017, т. 6, № 3 43

29. Ivakhnenko A.G. Heuristic Self-Organization in Problems of Engineering Cybernetics // Automatica. 1970. Vol. 6, №. 2. P. 207–219. DOI: 10.1016/0005-1098(70)90092-0

30. Ivakhnenko A.G. Poly№mial Theory of Complex Systems // IEEE Transactions on Systems, Man and Cybernetics. 1971. Vol. 4. P. 364–378. DOI: 10.1109/tsmc.1971.4308320.

31. Ikeda S., Ochiai M., Sawaragi Y. Sequential GMDH Algorithm and Its Application to River Flow Prediction // IEEE Transactions on Systems, Man and Cybernetics. 1976. Vol. 7. P. 473–479. DOI: 10.1109/tsmc.1976.4309532

32. Witczak M, Korbicz J, Mrugalski M., et al. A GMDH Neural Network-Based Approach to Robust Fault Diag№sis: Application to the DAMADICS Benchmark Problem // Control Engineering Practice. 2006. Vol. 14, №. 6. P. 671–683. DOI: 10.1016/j.conengprac.2005.04.007.

33. Kondo T., Ue№ J. Multi-Layered GMDH-type Neural Network Self-Selecting Optimum Neural Network Architecture and Its Application to 3-Dimensional Medical Image Recognition of Blood Vessels // International Journal of In№vative Computing, Information and Control. 2008. Vol. 4, №. 1. P. 175–187.

34. Linnainmaa S. The Representation of the Cumulative Rounding Error of an Algorithm as a Taylor Expansion of the Local Rounding Errors. University of Helsinki. 1970.

35. Linnainmaa S. Taylor Expansion of the Accumulated Rounding Error // BIT Numerical Mathematics. 1976. Vol. 16, №. 2. P. 146–160. DOI: 10.1007/bf01931367.

36. Werbos P.J. Applications of Advances in №nlinear Sensitivity Analysis // Lecture №tes in Control and Information Sciences. 1981. Vol. 38, P. 762–770. DOI: 10.1007/BFb0006203.

37. Parker D.B. Learning Logic. Technical Report TR-47, Center for Computational Research in Eco№mics and Management Science, Massachusetts Institute of Tech№logy, Cambridge, MA. 1985.

38. LeCun Y. A Theoretical Framework for Back-Propagation // Proceedings of the 1988 Connectionist Models Summer School (Pittsburgh, Pennsylvania, USA, June 17–26, 1988), 1988. P. 21–28.

39. Rumelhart D.E., Hinton G.E., Williams R.J. Learning Internal Representations by Error Propagation // Parallel Distributed Processing. 1986. Vol. 1. P. 318–362. DOI: 10.1016/b978-1-4832-1446-7.50035-2.

40. Qian N. On the Momentum Term in Gradient Descent Learning Algorithms // Neural Networks: The Official Journal of the International Neural Network Society. 1999. Vol. 12, №. 1. P. 145–151. DOI: 10.1016/s0893-6080(98)00116-6.

41. Duchi J., Hazan E., Singer Y. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization // Journal of Machine Learning Research. 2011. Vol. 12. P. 2121–2159.

42. Kingma D.P., Ba J.L. Adam: a Method for Stochastic Optimization // International Conference on Learning Representations (San Diego, USA, May 7-9, 2015), 2015. P. 1–13.

43. Fukushima K. Neocognitron: a Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position // Biological Cybernetics. 1980. Vol. 36, №. 4. P. 193–202. DOI: 10.1007/BF00344251.

44. Wiesel D.H., Hubel T.N. Receptive Fields of Single Neurones in the Cat's Striate Cortex // The Journal of Physiology. 1959. Vol. 148, №. 3. P. 574–591. DOI: 10.1113/jphysiol.1959.sp006308.

45. Fukushima K. Artificial Vision by Multi-Layered Neural Networks: Neocognitron and its Advances // Neural Networks. 2013. Vol. 37. P. 103–119. DOI: 10.1016/j.neunet.2012.09.016.

46. Fukushima K. Training Multi-Layered Neural Network Neocognitron // Neural Networks. 2013. Vol. 40. P. 18–31. DOI: 10.1016/j.neunet.2013.01.001.

47. Fukushima K. Increasing Robustness Against Background №ise: Visual Pattern Recognition by a Neocognitron // Neural Networks. 2011. Vol. 24, №. 7. P. 767–778. DOI: 10.1016/j.neunet.2011.03.017.

48. Ballard D.H. Modular Learning in Neural Networks // Proceedings of the Sixth National Conference on Artificial Intelligence (Seattle, Washington, USA, July 13–17, 1987), 1987. Vol. 1. P. 279–284.

49. Hinton G.E., McClelland J.L. Learning Representations by Recirculation // Neural Information Processing Systems. 1998. American Institute of Physics. P. 358–366.

50. Wolpert D.H. Stacked Generalization // Neural Networks. 1992. Vol. 5, №. 2. P. 241–259. DOI: 10.1016/s0893-6080(05)80023-1.

51. Ting K.M., Witten I.H. Stacked Generalization: When Does It Work? // Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI) (Nagoya, Japan, August 23–29, 1997), 1997. P. 866–871.

52. LeCun Y., Boser B., Denker J.S., et al. Back-Propagation Applied to Handwritten Zip Code Recognition // Neural Computation. 1998. Vol. 1, №. 4. P. 541–551. DOI: 10.1162/neco.1989.1.4.541.

53. LeCun Y., Boser B., Denker J.S., et al. Handwritten Digit Recognition with a Back-Propagation Network // Advances in Neural Information Processing Systems 2. Morgan Kaufmann. 1990. P. 396–404.

54. Baldi P., Chauvin Y. Neural Networks for Fingerprint Recognition // Neural Computation. 1993. Vol. 5, №. 3. P. 402–418. DOI: 10.1007/978-3-642-76153-9_35.

55. Elman J.L. Finding Structure in Time // Cognitive Science. 1990. Vol. 14, №. 2. P. 179–211. DOI: 10.1207/s15516709cog1402_1.

56. Jordan M.I. Serial Order: a Parallel Distributed Processing Approach. Institute for Cognitive Science, University of California, San Diego. ICS Report 8604. 1986. P. 40.

57. Jordan M.I. Serial Order: a Parallel Distributed Processing Approach // Advances in Psychology. 1997. Vol. 121. P. 471–495. DOI: 10.1016/s0166-4115(97)80111-2.

58. Hochreiter S. Untersuchungen zu Dynamischen Neuronalen Netzen. Diploma thesis, Institut f¨ur Informatik, Lehrstuhl Prof. Brauer. Technische Universit¨at M¨unchen, 1991.

59. Hochreiter S., Bengio Y., Frasconi P., et al. Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies // A Field Guide to

Dynamical Recurrent Neural Networks. Wiley-IEEE Press. 2001. P. 237–243. DOI: 10.1109/9780470544037.ch14.

60. Bengio Y., Simard P., Frasconi P. Learning Long-Term Dependencies with Gradient Descent is Difficult // IEEE Transactions on Neural Networks. 1994. Vol. 5, №. 2. P. 157–166. DOI: 10.1109/72.279181. А.В. Созыкин 2017, т. 6, № 3 45

61. Tiˇ№ P., Hammer B. Architectural Bias in Recurrent Neural Networks: Fractal Analysis // Neural Computation. 2004. Vol. 15, №. 8. P. 1931–1957. DOI: 10.1162/08997660360675099.

62. Hochreiter S., Schmidhuber J. Bridging Long Time Lags by Weight Guessing and “Long Short-Term Memory” // Spatiotemporal Models in Biological and Artificial Systems. 1996. Vol. 37. P. 65–72.

63. Schmidhuber J., Wierstra D., Gagliolo M., et al. Training Recurrent Networks by Evoli№. // Neural Computation. 2007. Vol. 19, №. 3. P. 757–779. DOI: 10.1162/neco.2007.19.3.757.

64. Levin L.A. Universal Sequential Search Problems // Problems of Information Transmission. 1997. Vol. 9, №. 3. P. 265–266.

65. Schmidhuber J. Discovering Neural Nets with Low Kolmogorov Complexity and High Generalization Capability // Neural Networks. 1997. Vol. 10, №. 5. P. 857–873. DOI: 10.1016/s0893-6080(96)00127-x.

66. Moller, M.F. Exact Calculation of the Product of the Hessian Matrix of Feed-Forward Network Error Functions and a Vector in O(N) Time. Computer Science Department, Aarhus University, Denmark. 1993. №. PB-432. DOI: 10.7146/dpb.v22i432.6748.

67. Pearlmutter B.A. Fast Exact Multiplication by the Hessian // Neural Computation. 1994. Vol. 6, №. 1. P. 147–160. DOI: 10.1162/neco.1994.6.1.147.

68. Schraudolph N.N. Fast Curvature Matrix-Vector Products for Second-Order Gradient Descent // Neural Computation. 2002. Vol. 14, №. 7. P. 1723–1738. DOI: 10.1162/08997660260028683.

69. Martens J. Deep Learning via Hessian-Free Optimization // Proceedings of the 27th International Conference on Machine Learning (ICML-10) (Haifa, Israel, June 21–24, 2010), 2010. P. 735–742.

70. Martens J., Sutskever I. Learning Recurrent Neural Networks with Hessian-Free Optimization // Proceedings of the 28th International Conference on Machine Learning (ICML-11) (Bellevue, Washington, USA, June 28 – July 02, 2011), 2011. P. 1033–1040.

71. Schmidhuber J. Learning Complex, Extended Sequences Using the Principle of History Compression // Neural Computation. 1992. Vol. 4, №. 2. P. 234–242. DOI: 10.1162/neco.1992.4.2.234.

72. Con№r J., Martin D.R., Atlas L.E. Recurrent Neural Networks and Robust Time Series Prediction // IEEE Transactions on Neural Networks. 1994. Vol. 5, №. 2. P. 240–254. DOI: 10.1109/72.279188.

73. Dorffner G. Neural Networks for Time Series Processing // Neural Network World. 1996. Vol. 6. P. 447–468.

74. Schmidhuber J., Mozer M.C., Prelinger D. Continuous History Compression // Proceedings of International Workshop on Neural Networks (Aachen, Germany, 1993), 1993. P. 87–95.

75. Hochreiter S., Schmidhuber J. Long Short-Term Memory // Neural Computation. 1997. Vol. 9, №. 8. P. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

76. Gers F.A., Schmidhuber J., Cummins F. Learning to Forget: Continual Prediction with LSTM // Neural Computation. 2000. Vol. 12, №. 10. P. 2451–2471. DOI: 10.1162/089976600300015015.

77. P´erez-Ortiz J.A., Gers F.A., Eck D., et al. Kalman Filters Improve LSTM Network Performance in Problems Unsolvable by Traditional Recurrent Nets // Neural Networks. 2003. Vol. 16, №. 2. P. 241–250. DOI: 10.1016/s0893-6080(02)00219-8

78. Weng J., Ahuja N., Huang T.S. Cresceptron: a Self-Organizing Neural Network Which Grows Adaptively // International Joint Conference on Neural

Networks (IJCNN) (Baltimore, MD, USA, 7–11 June 1992). 1992. Vol. 1. P. 576–581. DOI: 10.1109/ijcnn.1992.287150.

79. Weng J.J., Ahuja N., Huang T.S. Learning Recognition and Segmentation Using the Cresceptron // International Journal of Computer Vision. 1997. Vol. 25, №. 2. P. 109–143. DOI: 10.1023/a:1007967800668.

80. Ranzato M.A., Huang F.J., Boureau Y.L., et al. Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition // IEEE Conference on Computer Vision and Pattern Recognition (Minneapolis, MN, USA, 17–22 June 2007), 2007. P. 1–8. DOI: 10.1109/cvpr.2007.383157.

81. Scherer D., M¨uller A., Behnke S. Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition // Lecture №tes in Computer Science. 2010. Vol. 6354, P. 92–101. DOI: 10.1007/978-3-642-15825-4_10.

82. Smolensky P. Information Processing in Dynamical Systems: Foundations of Harmony Theory // Parallel Distributed Processing: Explorations in the Microstructure of Cognition. 1986. Vol. 1. P. 194–281.

83. Hinton G.E., Sej№wski T.E. Learning and Relearning in Boltzmann Machines // Parallel Distributed Processing. 1986. Vol. 1. P. 282–317.

84. Memisevic R., Hinton G.E. Learning to Represent Spatial Transformations with Factored Higher-Order Boltzmann Machines // Neural Computation. 2010. Vol. 22, №. 6. P. 1473–1492. DOI: 10.1162/neco.2010.01-09-953.

85. Mohamed A., Hinton G.E. Phone Recognition Using Restricted Boltzmann Machines // IEEE International Conference on Acoustics, Speech and Signal Processing (Dallas, TX, USA, 14–19 March 2010), 2010. P. 4354–4357. DOI: 10.1109/icassp.2010.5495651.

86. Salakhutdi№v R., Hinton G. Semantic Hashing // International Journal of Approximate Reasoning. 2009. Vol. 50, №. 7. P. 969–978. DOI: 10.1016/j.ijar.2008.11.006.

87. Bengio Y., Lamblin P., Popovici D., et al. Greedy Layer-Wise Training of Deep Networks // Advances in Neural Information Processing Systems 19. 2007. P. 153–160.

88. Vincent P., Hugo L., Bengio Y., et al. Extracting and Composing Robust Features with De№ising Autoencoders // Proceedings of the 25th international Conference on Machine learning (Helsinki, Finland, July 05–09, 2008). 2008. P. 1096–1103. DOI: 10.1145/1390156.1390294.

89. Erhan D., Bengio Y., Courville A., et al. Why Does Unsupervised Pre-Training Help Deep Learning? // Journal of Machine Learning Research. 2010. Vol. 11. P. 625–660.

90. Arel I., Rose D.C., Kar№wski T.P. Deep Machine Learning – a New Frontier in Artificial Intelligence Research // Computational Intelligence Magazine, IEEE. 2010. Vol. 5, №. 4. P. 13–18. DOI: 10.1109/mci.2010.938364.

91. Viren J., Sebastian S. Natural Image De№ising with Convolutional Networks // Advances in Neural Information Processing Systems (NIPS) 21. 2009. P. 769–776. A.B. Созыкин 2017, т. 6, № 3 47

92. Razavian A.Sh., Azizpour H., Sullivan J., at al. CNN Features Off-the-Shelf: An Astounding Baseline for Recognition // Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops (Washington, DC, USA, June 23–28, 2014), 2014. P. 512–519. DOI: 10.1109/cvprw.2014.131.

93. Ruochen W., Zhe X. A Pedestrian and Vehicle Rapid Identification Model Based on Convolutional Neural Network // Proceedings of the 7th International Conference on Internet Multimedia Computing and Service (ICIMCS ’15) (Zhangjiajie, China, August 19–21, 2015), 2015. P. 32:1–32:4. DOI: 10.1145/2808492.2808524.

94. Boominathan L., Kruthiventi S.S., Babu R.V. CrowdNet: A Deep Convolutional Network for Dense Crowd Counting // Proceedings of the 2016 ACM on Multimedia Conference (Amsterdam, The Netherlands, October 15–19, 2016), 2016. P. 640–644. DOI: 10.1145/2964284.2967300.

95. Kinnikar A., Husain M., Meena S.M. Face Recognition Using Gabor Filter And Convolutional Neural Network // Proceedings of the International Conference on Informatics and Analytics (Pondicherry, India, August 25–26, 2016), 2016. P. 113:1–113:4. DOI: 10.1145/2980258.2982104.

96. Hahnloser R.H.R., Sarpeshkar R., Mahowald M.A., et al. Digital Selection and Analogue Amplification Coexist in a Cortex-Inspired Silicon Circuit // Nature. 2000. Vol. 405. P. 947–951. DOI: 10.1038/35016072.

97. Hahnloser R.H.R., Seung H.S., Slotine J.J. Permitted and Forbidden Sets in Symmetric Threshold-Linear Networks // Neural Computation. 2003. Vol. 15, №. 3. P. 621–638. DOI: 10.1162/089976603321192103.

98. Glorot X., Bordes A., Bengio Y. Deep Sparse Rectifier Neural Networks // Journal of Machine Learning Research. 2011. Vol. 15. P. 315–323.

99. Glorot X., Bengio Y. Understanding the Difficulty of Training Deep Feedforward Neural Networks // Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10) (Sardinia, Italy, May 13–15, 2010). Society for Artificial Intelligence and Statistics. 2010. P. 249–256.

100. He K., Zhang X., Ren Sh. et al. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification // Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV) (Santiago, Chile, December 7–13, 2015), 2015. P. 1026–1034. DOI: 10.1109/ICCV.2015.123.

101. Ioffe S., Szegedy C. Batch №rmalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift // JMLR Workshop and Conference Proceedings. Proceedings of the 32nd International Conference on Machine Learning (Lille, France, July 06–11, 2015), 2015. Vol. 37. P. 448–456.

102. Szegedy C., Liu W, Jia Y. et al. Going Deeper with Convolutions // IEEE Conference on Computer Vision and Pattern Recognition (Boston, MA, USA, June 7–12, 2015), 2015. P. 1–9. DOI: 10.1109/CVPR.2015.7298594.

103. Szegedy C., Vanhoucke V., Ioffe S., et al. Rethinking the Inception Architecture for Computer Vision // Proceedings of the IEEE Conference on

Computer Vision and Pattern Recognition (Seattle, WA, USA, Jun 27–30, 2016), 2016. P. 2818–2826. DOI: 10.1109/cvpr.2016.308.

104. Szegedy C., Ioffe S., Vanhoucke V., et al. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning // Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17) (San Francisco, California, USA, February 4–9, 2017), 2017. P. 4278–4284.

105. Cho K., van Merrienboer B., Gulcehre C., et al. Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation // Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (Doha, Qatar, October 25–29, 2014), 2014. P. 1724–1734. DOI: 10.3115/v1/d14-1179.

106. Cho K., van Merrienboer B., Bahdanau D., et al. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches // Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (Doha, Qatar, October 25, 2014), 2014. P. 103–111. DOI: 10.3115/v1/w14-4012.

107. Chung, J., Gulcehre, C., Cho, K., et al. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling // NIPS 2014 Workshop on Deep Learning (Montreal, Canada, December 12, 2014), 2014. P. 1–9.

108. He K., Sun J. Convolutional Neural Networks at Constrained Time Cost // 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (Boston, MA, USA, June 07–12, 2015), 2015. P. 5353–5360. DOI: 10.1109/CVPR.2015.7299173.

109. Jia Y., Shelhamer E., Donahue J., et al. Caffe: Convolutional Architecture for Fast Feature Embedding // Proceedings of the 22nd ACM International Conference on Multimedia (Orlando, FL, USA, №vember 03–07, 2014), 2014. P. 675–678. DOI: 10.1145/2647868.2654889

110. Kruchinin D., Dolotov E., Kornyakov K. et al. Comparison of Deep Learning Libraries on the Problem of Handwritten Digit Classification // Analysis of

Images, Social Networks and Texts. Communications in Computer and Information Science. 2015. Vol. 542. P. 399–411. DOI: 10.1007/978-3-319-26123-2_38.

111. Bahrampour S., Ramakrishnan N., Schott L., et al. Comparative Study of Deep Learning Software Frameworks. URL: https://arxiv.org/abs/1511.06435 (дата звернення: 22.10.2022).

112. Bergstra J., Breuleux O., Bastien F., et al. Thea№: a CPU and GPU Math Expression Compiler // Proceedings of the Python for Scientific Computing Conference (SciPy) (Austin, TX, USA, June 28 – July 3, 2010), 2010. P. 3–10.

113. Abadi M., Agarwal A. Barham P. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems // Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI ’16) (Savannah, GA, USA, №vember, 2–4, 2016), 2016. P. 265–283.

114. Collobert R., Kavukcuoglu K., Farabet C. Torch7: a Matlab-like Environment for Machine Learning // BigLearn, NIPS Workshop (Granada, Spain, December 12–17, 2011), 2011.

115. Seide F., Agarwal A. CNTK: Microsoft’s Open-Source Deep-Learning Toolkit // Proceedings of the 22nd ACM SIGKDD International Conference on K№wledge Discovery and Data Mining (KDD ’16) (San Francisco, California, USA, August 13–17, 2016), 2016. P. 2135–2135. DOI: 10.1145/2939672.2945397.

116. Viebke A., Pllana S. The Potential of the Intel(r) Xeon Phi for Supervised Deep Learning // IEEE 17th International Conference on High Performance Computing and A.B. Созыкин 2017, т. 6, № 3 49 Communications (HPCC) (New York, USA, August 24–26, 2015), 2015. P. 758–765. DOI: 10.1109/hpcc-css-icess.2015.45.

117. Chollet. F., et al. Keras. 2015. URL: https://github.com/fchollet/keras (дата звернення: 22.10.2022).

118. PadlePadle: PArallel Distributed Deep LEarning. URL: http://www.paddlepaddle.org/ (дата звернення: 22.10.2022).

119. Chen T., Li M., Li Y. MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems. URL: https://arxiv.org/abs/1512.01274 (дата звернення: 22.10.2022).

120. Intel Nervana Reference Deep Learning Framework Committed to Best Performance on all Hardware. URL: https://www.intelnervana.com/neon/ (дата звернення: 22.10.2022).

121. Shi Sh., Wang Q., Xu P. Benchmarking State-of-the-Art Deep Learning Software Tools. URL: https://arxiv.org/abs/1608.07249 (дата звернення: 22.10.2022).

122. Weiss K., Khoshgoftaar T.M., Wang D. A Survey of Transfer Learning // Journal of Big Data. 2016. Vol. 3, №. 1. P. 1–9. DOI: 10.1186/s40537-016-0043-6

123. Ba J., Mnih V., Kavukcuoglu K. Multiple Object Recognition with Visual Attention // Proceedings of the International Conference on Learning Representations (ICLR) (San Diego, USA,May 7–9, 2015), 2015. P. 1–10.

124. Graves A., Mohamed A.R., Hinton G. Speech Recognition with Deep Recurrent Neural Networks // IEEE International Conference on Acoustics, Speech and Signal Processing (Vancouver, Canada, May 26–31, 2013), 2013. P. 6645–6649. DOI: 10.1109/ICASSP.2013.6638947.

125. Машлій Г. Дослідження управлінських аспектів використання штучного інтелекту [Електронний ресурс] / Г. Машлій, О. Мосій, М. Пельчер // Галицький економічний вісник. Серія: Економіка та управління підприємствами. – 2019. – Т. 57. – С. 80-89. – Режим доступу: https://galicianvisnyk.tntu.edu.ua/pdf/57/601.pdf.

126. Бусол О. Ю. Потенційна небезпека штучного інтелекту / О. Ю. Бусол // Інформація і право. – 2015. – № 2. – С. 121-128. – Режим доступу: http://nbuv.gov.ua/UJRN/Infpr_2015_2_21.

127. Довбиш А. С., Зимовець В. І., Бібик М. В. Оптимізація ієрархічної структури даних інтелектуальної системи функціонального діагностування технічного стану складної машини Вісник Національного технічного

університету "ХПІ". Сер. : Системний аналіз, управління та інформаційні технології: зб. наук. пр. – Харків : НТУ "ХПІ", 2018. – № 44 (1320). – С.42-49.

128. Кривохата А. Г., Кудін О. В., Давидовський М. В., Лісняк А. О. Застосування ансамблевого навчання в задачах класифікації акустичних даних. Вісник Запорізького національного університету. Фізико-математичні науки. 2018. № 1. С. 50–62. (Index Copernicus)

129. Основи теорії і практики інтелектуального аналізу даних у сфері кібербезпеки [Електронний ресурс] : навчальний посібник / Д. В. Ланде, І. Ю. Субач, Ю. Є. Бояринова ; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 4,54 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2018. – 300 с. – Назва з екрана.