Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Модифікований алгоритм розпізнавання двовимірних штрих-кодів»

Виконав: студент 2-ого курсу групи 1ІСТ-21м
спеціальності 126 – Інформаційні системи та технології_____
Звуздецький Є. О._____

Керівник: к.т.н., доц. каф. АІІТ_____
Іванов Ю. Ю._____

«____» _____ 2022 р.

Опонент: _____

_____

«____» _____ 2022 р.

Допущено до захисту
в.о. зав. кафедри АІІТ
д.т.н., проф. Бісікало О. В.
«____» _____ 2022 р.

Вінниця ВНТУ – 2022 рік

Вінницький національний технічний університет

Факультет інтелектуальних інформаційних технологій та автоматизації

Кафедра автоматизації та інтелектуальних інформаційних технологій

Рівень вищої освіти II-ий (магістерський)

Галузь знань – 12 – Інформаційні технології

Спеціальність – 126 – Інформаційні системи та технології

Освітньо-професійна програма – Інформаційні технології аналізу даних та зображень

ЗАТВЕРДЖУЮ

в. о. завідувача кафедри АІІТ

д.т.н., проф. Бісікало О. В.

«___»_____ 2022 р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Звуздецькому Єгору Олеговичу

1. Тема роботи: Модифікований алгоритм розпізнавання двовимірних штрих-кодів.

   Керівник роботи: к.т.н., доцент каф. АІІТ Іванов Ю. Ю.

   Затверджені наказом ВНТУ від «14» 09 2022 року № 203.

Строк подання роботи студентом: до «до 23» 12 2022 р.

2. Вихідні дані до роботи: база даних з образами двовимірних штрих-кодів; матриця пікселів образу $H \times B$; нейромережа Ліппмана; константа масштабування k_1; порогові параметри L, K та Q; кількість ітерацій роботи мережі *MaxNet*.
3. Зміст розрахунково-пояснювальної записки: вступ; аналіз предметної області; розробка модифікованого алгоритму розпізнавання двовимірних штрих-кодів; розробка програмного забезпечення та експериментальні дослідження; список використаних джерел; висновки.
4. Перелік графічного матеріалу: модель штучного нейрону; архітектура нейромережі Ліппмана; структура для навчання мережі; приклади кодування образів; обробка зображення; схема програми.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Іванов Ю. Ю., к.т.н., доцент каф. АІІТ	19.09.2022	07.12.2022
4	Лесько О. Й. к.е.н., проф. каф. ЕПтаВМ		

7. Дата видачі завдання: «19»_____09_____ 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області	10.10.22	
2	Розробка модифікованого алгоритму розпізнавання двовимірних штрих-кодів	25.10.22	
3	Розробка програмного забезпечення та експериментальні дослідження	10.11.22	
4	Підготовка економічного розділу	20.11.22	
5	Оформлення пояснювальної записки і графічного матеріалу	30.11.22	
6	Попередній захист роботи	до 07.12.22	
7	Остаточний захист роботи	до 23.12.22	

Студент _____ Звуздецький Є. О.

(підпис) (прізвище та ініціали)

Керівник роботи _____ Іванов Ю. Ю.

(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.032.26 + 004.942

Звуздецький Є. О. Модифікований алгоритм розпізнавання двовимірних штрих-кодів. Магістерська кваліфікаційна робота зі спеціальності 126 – Інформаційні системи та технології, освітньо-професійна програма – Інформаційні технології аналізу даних та зображень. Вінниця: ВНТУ, 2022. 90 с.

На укр. мові. Бібліогр.: 30 назв; рис.: 18; табл.: 17.

У даній роботі розглядаються питання, пов'язані з задачею розпізнавання двовимірних штрих-кодів за рахунок використання удосконаленої математичної моделі на основі нейромережі з базою еталонних образів та процедурою налаштування параметрів. Також розроблено програмне забезпечення та проведено низку експериментів для підтвердження ефективності запропонованої розробки.

Ключові слова: обробка образів, нейромережа, двовимірні штрих-коди, ковзне вікно, налаштування параметрів, корекція.

ABSTRACT

Zvuzdetskii E. O. Modified algorithm for recognition two-dimensional barcodes. Master's thesis in specialty 126 – Information systems and technologies, educational and professional program – Information technologies of data and image analysis. Vinnitsa: VNTU, 2022. 90 p.

In Ukrainian language. Bibliography: 30 titles; fig.: 18; tabl.: 17.

In this work have been considered some issues, which are connected with a task of recognizing barcodes through the use of a modification of mathematical model based on an artificial neural network with with a base of reference images and parameters optimization procedure. Software has been developed and a number of experiments have been conducted to confirm the effectiveness of the proposed.

Keywords: image processing, neural network, two-dimensional barcodes, sliding window, parameter setting, correction procedure.

ЗМІСТ

ВСТУП 4

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 7

1.1 Особливості розв'язання задачі розпізнавання образів 7

1.2 Принципи роботи штучної нейромережі 11

1.3 Огляд архітектур нейромереж 18

1.4 Висновки 25

2 РОЗРОБКА МОДИФІКОВАНОГО АЛГОРИТМУ РОЗПІЗНАВАННЯ ДВОВИМІРНИХ ШТРИХ-КОДІВ 26

2.1 Особливості роботи з двовимірними штрих-кодами 26

2.2 Алгоритм розпізнавання двовимірних штрих-кодів 30

2.3 Модифікація процедури обробки даних 36

2.4 Висновки 38

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ 39

3.1 Вибір мови програмування та опис програми 39

3.2 Експериментальні дослідження 40

3.3 Висновки 50

4 РОЗДІЛ ЕКОНОМІКИ 51

4.1 Технологічний аудит удосконаленого алгоритму розпізнавання образів 51

4.2 Розрахунок витрат на удосконалення алгоритму розпізнавання образів 55

4.3 Прогнозування комерційних ефектів від комерціалізації розробки 64

4.4 Висновки 68

ВИСНОВКИ 69

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 70

ДОДАТКИ 73

Додаток А (обов'язковий). Технічне завдання 74

Додаток Б (обов'язковий). Ілюстративна частина 78

Додаток В (обов'язковий). Лістинг програми 82

Додаток Г (обов'язковий). Довідка про можливість впровадження……

розробки 86

Додаток Д (обов'язковий). Протокол перевірки МКР 88

ВСТУП

Актуальність. Стрімкий розвиток комп'ютерної техніки, а відповідно і різноманітного програмного забезпечення – це одна з характерних рис сучасного періоду розвитку суспільства. Автоматизація проникає до усіх сфер життя людини з впровадженням принципово нових засобів, які реалізують інтелектуальні методи управління (експертні системи, нейромережі, еволюційні алгоритми) [1]. Активно розвиваються системи обміну даними, які використовують комбіноване стиснення, захист від пошкоджень та зберігання інформації. Такі системи включають двовимірні коди (Data Matrix, Aztec і QR-коди). Для коригування помилок у них застосовують вбудовані недвійкові циклічні коди Ріда-Соломона, які дають можливість відновити інформацію у випадку, якщо пошкоджено до $R \leq 30\%$ матриці коду [2,3].

Саме тому низку досліджень направлено на розв'язання задачі відновлення матриці для дуже пошкоджених двовимірних штрих-кодів ($R > 30\%$), коли коди Ріда-Соломона не можуть виправити потрібну частину помилок, тобто наявний високий рівень шумової компоненти. Для цього вчені застосовують каскадні конструкції кодів [2] та алгоритми пошуку ключових ознак [4, 5]. Подібні процедури вимагають використання трудомістких обчислювальних операцій у режимі реального часу, що може бути недоцільним у ситуаціях, коли на матриці відсутні ключові елементи-детектори або їхня структура значно пошкоджена, оскільки у цьому випадку декодуюче програмне забезпечення не зможе навіть зчитати код.

Дана задача досить актуальна, оскільки її рішення вимагає значних обчислювальних ресурсів і часових затрат. Сучасним підходом до її

розв'язання, у зв'язку зі зростанням обчислювальної потужності технічних засобів, є використання штучних нейромереж. Подібні дослідження розглядаються у наукових роботах закордонних та вітчизняних авторів, наприклад В.Д. Дмитрієнко [6-9], Т.В. Сазонової [10], С.Д. Катебі [11], А. Гупти [12] та інших. У даній роботі пропонується розв'язок задачі розпізнавання та коригування двовимірних штрих-кодів, який базується на використанні бази даних, штучної нейромережі та процедури корекції. Розроблений алгоритм можна використовувати у системах обміну даними у випадку, коли матриця двовимірного штрих-коду пошкоджена настільки сильно, що немає можливості зчитати / розпізнати даний код звичайним чином.

Мета і задачі дослідження. Метою роботи є підвищення ефективності розв'язання задачі розпізнавання пошкоджених двовимірних штрих-кодів за рахунок використання запропонованої моделі корекції.

Для досягнення мети необхідно розв'язати наступні *задачі*:

– проаналізувати алгоритми розпізнавання зображень;

– розробити математичний апарат для розпізнавання двовимірних штрих-кодів;

– розробити програмні засоби та виконати експериментальні дослідження для підтвердження ефективності розробки.

Об'єктом дослідження є процеси оброблення даних та розпізнавання образів.

Предметом дослідження є методи, моделі та інструментальні засоби оброблення даних та розпізнавання образів.

Методи дослідження. У роботі використано теорію цифрової обробки зображень для аналізу та перетворення образу до потрібного

вигляду; елементи теорії штучного інтелекту для розпізнавання образу з використанням простих обчислювальних операцій; експериментальне дослідження для перевірки достовірності отриманих результатів. Основним засобом для розробки програми є мова програмування Python.

Наукова новизна отриманих результатів полягає у наступному:

1. Запропоновано алгоритм розпізнавання пошкоджених двовимірних штрих-кодів, особливістю якого є удосконалена математична модель, що дозволяє уникнути трудомістких обчислювальних операцій та відновити матрицю даних, коли сканер не може зчитати та розпізнати матрицю коду.

Практичне значення результатів роботи:

1. Розроблене математичне, алгоритмічне та програмне забезпечення дозволяє застосувати запропонований алгоритм, як додаткову процедуру виправлення двовимірних штрих-кодів у системах обміну даними.

Апробація результатів та публікації. За результатами даної роботи опубліковано 2 тези доповіді на L науково-технічній конференції підрозділів ВНТУ (м. Вінниця, 2021) [13] та науково-практичній конференції з міжнародною участю "Актуальні задачі медичної, біологічної фізики та інформатики" (м. Вінниця, 2022) [14]. Основні результати можна використати на практиці, що підтверджено довідкою про можливість впровадження розробки.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Особливості розв'язання задачі розпізнавання образів

Теорія розпізнавання образів (об'єктів) – розділ комп'ютерних наук, який використовує основи та методи класифікації та ідентифікації предметів, явищ, процесів, сигналів, ситуацій, об'єктів, що характеризуються скінченним набором деяких властивостей та ознак. Розпізнавання образів представляє собою задачу ідентифікації об'єкта або визначення його властивостей по зображенню (оптичне розпізнавання), аудіозапису (акустичне розпізнавання) тощо (наприклад, задача розпізнавання літер, прийняття рішення про наявність дефекту у технічній деталі тощо). Замість терміну "розпізнавання" часто використовується інший термін – "класифікація" [4, 5].

Задача розпізнавання образів досить довго розглядалася людиною лише з боку біологічного та психологічного аспектів, тобто вивченню піддавалися лише якісні характеристики, які не дозволяли точно описати весь механізм функціонування. Отримання функціональних залежностей було, як правило, пов'язане з дослідженням рецепторів органів слуху, дотику або зору. Однак принципи формування рішення залишалися загадкою.

Заснована Н. Вінером на початку XX століття нова наука, яка отримала назву кібернетика (наука про загальні закономірності процесів управління і передавання інформації в машинах, живих організмах і суспільстві), дозволила застосувати кількісні методи у дослідженні

розпізнавання образів. Впродовж тривалого часу задача розпізнавання образів привертає увагу фахівців в області прикладної математики, а потім і інформатики [5].

В рамках кібернетики почав формуватися новий науковий напрямок "розпізнавання образів", пов'язаний з розробкою теоретичних основ і практичної реалізації систем, призначених для розпізнавання об'єктів, явищ та процесів. Завдяки цьому, значно розширилися можливості складних систем, які виконують різні інформаційні, логічні та аналітичні завдання.

Зазвичай, сприймаючи явища навколишнього світу (предмети, ситуації), людина здійснює їх класифікацію, розбиваючи їх на групи схожих явищ (але не тотожних). Інколи необхідно віднести в одну групу у чомусь "подібні" явища, які можуть відрізнятися один від одного. Наприклад, фігури, показані на рисунку 1.1, представляють "літеру А", незважаючи на їх значну відмінність у їх написанні.

Рисунок 1.1 – Прикладу образу з набором екземплярів (об'єктів)

Отримавши уявлення про дану літеру на основі деякої, зазвичай невеликої кількості її екземплярів, можна впізнати як завгодно велику кількість інших її екземплярів. Таким чином, існують множини деякого особливого типу. Їх називають образами. Образ – це класифікаційне угруповання, яке об'єднує (виділяє) певну групу об'єктів за певною ознакою.

Для оптичного розпізнавання образів можна застосувати структурні методи: метод перебору вигляду об'єкта під різними кутами, масштабами і зсувами з розрахунком метрики відстані; пошук контуру об'єкта і дослідження його властивостей (зв'язність, наявність кутів). Для букв потрібно перебирати шрифт, його властивості тощо. Статистичні методи розпізнавання використовують певну статистичну інформацію, визначаючи належність об'єкта до конкретного класу на основі апостеріорної ймовірності (байєсівське вирішувальне правило) [15-18].

Пошук з використанням математичних алгоритмів грунтується на властивостях зображення та виділенні на ньому глобальних або локальних дескрипторів. Глобальний дескриптор – це вектор ознак, отриманий при аналізі всього зображення, коли кожна точка вносить свій вклад у загальний результат. У даному випадку важлива схожість зображень у цілому, а не їх фрагментів. Локальні дескриптори формують вектори ознак, які описують найголовніші частини зображення, певні вибрані фрагменти або точки інтересу, що дозволяє виконувати більш точний пошук нечітких дублікатів серед зображень порівняно з використанням глобальних дескрипторів. Точкою інтересу називається така точка зображення, структура околу якої коваріантна заданим перетворенням зображення, тобто вона має певні структурні особливості, які суттєво відрізняють її від низки інших точок [4].

У контексті машинного навчання найбільш перспективним методом для розпізнавання, детектування і класифікації об'єктів є нейромережі, які побудовані за принципом моделювання взаємодії біологічних нейронів у корі головного мозку живих організмів. Однак, незважаючи на зростання

популярності нейронних мереж і значний прогрес в швидкодії та якості роботи, їх застосування в реальних задачах все ще обмежене. Це пов'язано як з тим, що робота алгоритмів вимагає істотних обчислювальних ресурсів, так і з тим, що навчання мереж – їх головна відмінність від класичних алгоритмів, не задається фіксованим чином, тобто наявний елемент випадковості в рішеннях поставлених задач. Крім того, складним є пошук оптимальної архітектури нейромережі. Відповідно класичні алгоритми не втратили свого значення [15].

У цілому задача розпізнавання образів складається з двох частин: навчання (learning) та розпізнавання (recognizing). Навчання – це процес, за допомогою якого система отримує здатність відповідати необхідними реакціями на низку зовнішніх впливів, а адаптація – це процедура налаштування її структури і параметрів для досягнення потрібної ефективності управління в умовах постійної зміни зовнішніх факторів.

Загалом виділяють два види навчання: самонавчання (self-organized learning) та контрольоване навчання (supervised learning). У першому випадку нейромережа самоорганізовується під дією зовнішніх факторів і вивчає навколишнє середовище самостійно, тобто без допомоги "учителя". Самонавчання використовується у задачах розпізнавання образів. Розглядають два різновиди контрольованого навчання: пряме (forward learning) і стимулююче навчання (reinforcement learning). Контрольоване навчання вимагає наявності "учителя", який змінює параметри нейромережі, спостерігаючи за її реакцією. Такий вид навчання зазвичай використовується при розв'язанні задач керування [19].

1.2 Принципи роботи штучної нейромережі

Мозок може організовувати свої структурні компоненти (нейрони) таким чином, що вони виконують конкретно задані задачі (розпізнавання образів, обробка сигналів від органів почуттів, моторні функції організму) набагато швидше, ніж сучасні суперкомп'ютери. Нейромережа представляє собою машинну інтерпретація мозку людини, в якому знаходяться мільйони нейронів (neurons) та передають інформацію у вигляді електричних імпульсів. Нейронні мережі здатні не тільки аналізувати вхідну інформацію, а й відтворювати її зі своєї пам'яті.

Структура нейронної мережі прийшла до галузі штучного інтелекту з біології. У багатоклітинних організмах мирно співіснують мільярди різноманітних клітин. У кожної з них є свої потреби, "запити" і "особливі побажання", але вони повинні нормально функціонувати. Відсутність контролю і не дотримання угод між клітинами призводить до порушень в організмі. Коли у клітин з'являється можливість вийти з-під контролю різних керуючих і контролюючих центрів, вони починають ділитися і розмножуватися до нескінченності, порушуючи апоптоз, не даючи нормально розвиватися і функціонувати іншим оточуючим клітинам і тканинам [19].

Важливу керівну і контролюючу роль в багатоклітинному організмі виконує нервова система. Вона об'єднує весь організм в єдине ціле, регулює, контролює і координує діяльність усіх органів і систем, тим самим забезпечуючи найкраще пристосування організму до змін навколишнього середовища і життя в цілому. Еволюція нервової системи

чимось схожа на еволюцію комп'ютерів. Наприклад, перші комп'ютери мали мінімальний набір функцій і можливостей.

Головними елементами нервової системи є нервові клітини – нейрони. Людський мозок, за оцінками вчених, складається приблизно з 10^{11} нейронів. Зовнішня поверхня мозку, на якій розташовуються нейрони, складається з звивин і борозен (складки і звивини мозку), які збільшують площу поверхні розташування нейронів. Кора головного мозку завтовшки 0,5-5 мм. Якщо розпрямити борозни і звивини мозку, то можна отримати поверхню площею приблизно 2200 $см^2$. Середня вага мозку складає 1,2 кг, причому мозок чоловіка на 130 г більше мозку жінки. Крім того, існують расові та національні відмінності. Наприклад "щасливими" володарями найлегшого мозку (1,185 кг) є австралійці (австралоїди), а найважчого (1,375 кг) – європейці (європеоїди).

Фактично нейрон представляє собою нервову клітину, яка є основним будівельним блоком для нервової системи. Ці вузькоспеціалізовані клітини здатні на передавання інформації як хімічним, так і електричним шляхом. Налічується багато видів нейронів, які виконують різні функції [18].

Нейрон складається з тіла клітини діаметром 3...120 мкм (в середньому 10…30 мкм) – соми і двох типів відгалужень: аксона (вісь) і дендритів (дерево). В тілі клітини розташоване ядро, яке містить інформацію про властивості нейрона. Ядро оточене плазмою, яка виробляє необхідні для нейрона матеріали. Тіло нервової клітини зовні обмежене мембраною з подвійного прошарку ліпідів (жирові сполуки).

Нейрон приймає сигнали (імпульси) від сусідніх нейронів через дендрити (приймачі) і передає сигнали, згенеровані сомою, вздовж аксона

(передавач), який складається з волокон (strands). На їх закінченнях існують невеликі потовщення, які називаються синаптичними закінченнями. Структура нейрона показана на рисунку 1.2.

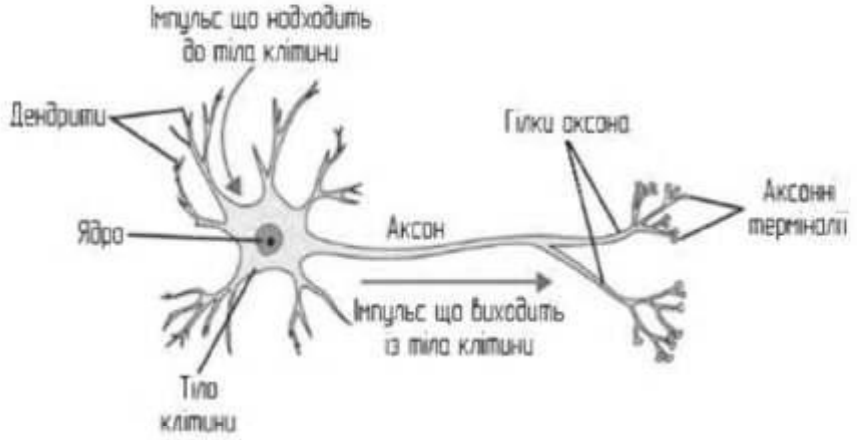


Рисунок 1.2 – Структура біологічного нейрона

Аксони і дендрити служать для зв'язку між різними нейронами. Нейрон може мати кілька дендритів і зазвичай тільки один аксон. Один нейрон може мати зв'язки з багатьма (до 20 тисяч) іншими нейронами. Різні нейрони мають різне співвідношення довжини аксона і дендритів.

Нейрони в нервовій системі оточені опорними і допоміжними клітинами, які називаються гліальними (клей). Їх призначенням є підтримка нервових зв'язків і забезпечення функціонування нервової мережі, яка об'єднує різні області кори головного мозку. Кількість гліальних клітин в центральній нервовій системі в 5-10 разів перевищує кількість нейронів. Хоча гліальні клітини мають мембранний потенціал, вони здебільшого не мають нервових закінчень і виконують підтримуючу роль для головного і спинного мозку.

Передача нервового імпульсу від нейрона до нейрона досить незвичайна і цікава. Нервовий імпульс в основному поширюється тільки в одному напрямку – з кількох дендритів до тіла клітини, і від тіла, по єдиному аксону до м'яза або якогось органу або дендритів наступного нейрона.

Синаптичне закінчення аксона не торкається іншого нейрона, до якого надходить збудження. Між синаптичним закінченням і дендритом сприймаючої клітини існує невеликий проміжок, який називається синаптичною щілиною, а саме з'єднання має назву синапс (synapse). У кожного з нейронів є від 1000 до 10 000 синапсів, через які він може встановити зв'язок з іншими нейронами [15-18].

Встановлюючи зв'язки між собою, нейрони утворюють цілі мережі. Ці групи інтегрованих, пов'язаних між собою нервових клітин називаються нейронними мережами. Можна сказати, що кожна нейромережа є думкою, спогадом, навиком, блоком інформації тощо.

Існує дві основні моделі навчання мозку. Перша модель полягає в засвоєнні і запам'ятовуванні фактів та інтелектуальних даних. Кожне ім'я, кожна дата, кожен логічний аргумент закарбовується в нейромережах мозку. Чим частіше повторюється матеріал, тим глибше він закарбовується у пам'яті – оскільки нейромережі стають міцнішими. Друга модель навчання мозку, яка є більш дієвою – це досвід (memory vs experience).

Оскільки можливих нейронних зв'язків у мозку більше, ніж атомів у Всесвіті, у мозку виникає проблема: як відшукувати спогади. Наприклад, коли на людину кидається злий собака, то яким чином мозок згадує потрібну реакцію так швидко? На допомогу приходять емоції (emotions).

Вони існують для того, щоб хімічно посилювати враження, перетворюючи їх у довгострокові спогади. Емоції, які самі частково є нейромережами, пов'язані зі всіма іншими нейромережами. Ці зв'язки допомагають мозку відшукувати в першу чергу найважливіші спогади. Вони гарантують, що важливий досвід (наприклад, дотик рукою до розпеченої плити) не буде швидко забутий.

Часто в популярних статтях мозок порівнюють з цифровим комп'ютером, зокрема ототожнюючи окремі ділянки мозку з роботою жорсткого диску або оперативної пам'яті. Однак, таке порівняння є недоречним, оскільки мозок не має ні центрального процесора, ні операційної системи, а також встановлених додатків. Очевидно, що архітектура головного мозку зовсім не схожа на архітектуру сучасних обчислювальних пристроїв. Головною відмінністю є те, що мозок постійно самонавчається, а з виконанням нового завдання формуються нові нейронні зв'язки. Комп'ютер під час свого функціонування залишається з постійною конфігурацією і логікою.

У моделюванні природних аналогів (мозку, генетичного спадкування, еволюції, роботи експертів) склалося два головних напрямки. Спадний (Top-Down AI) або прагматичний напрямок ґрунтується на тому, щоб спочатку створювати модель розумною, з набором багатьох програм. Тобто потрібно створити модуль пам'яті, в який були б записані всі “правила розумності”. Варто додати цей модуль пам'яті до моделі (робота, штучного інтелекту), як він негайно оживе і почне розумно мислити. Висхідний (Bottom-Up AI) або біонічний напрямок засновано на припущенні про те, що якщо відтворити природні структури і процеси, то й результати вирішення завдань такою системою будуть подібні до результатів роботи

природної системи. На базі цього підходу збудовані нейронні мережі. Замість того, щоб отримати всі правила розумності під час народження, мозок саморозвивається, натикаючись на все підряд і навчаючись на власному досвіді за допомогою спроб і помилок [13, 19].

Біологічні нейронні мережі створені у тривимірному просторі з різних типів нейронів, які з'єднані між собою величезною кількістю зв'язків. Різноманітні групування нейронів у мозку забезпечують оброблення інформації динамічним, інтерактивним та самоорганізуючим шляхом. Звісно, що для штучних аналогів мозку неможливо відтворити таку різноманітність через фізичні обмеження.

Об'єднуючись у мережі, штучні нейрони утворюють системи оброблення інформації, які забезпечують ефективну адаптацію моделі до безперервних змін факторів зовнішнього середовища. У процесі роботи нейромережі відбувається перетворення вхідного вектора сигналів у вихідний результат. Вид такого перетворення задається архітектурою мережі, характеристиками нейронних елементів, засобами управління та синхронізації потоків даних між нейронами. Важливим фактором ефективності нейромережі є вибір оптимальної кількості нейронів та типів зв'язків між ними, а також відповідних правил передавання даних.

Штучний нейрон (artificial neuron) – це обчислювальна одиниця, яка отримує інформацію, проводить над нею обчислення і передає її далі. Існує три основних типи нейронів: вхідний (*input*), прихований (*hidden*) і вихідний (*output*). Також є нейрон зміщення (*bias*). У тому випадку, коли нейромережа складається з великої кількості нейронів, вводять термін прошарку (*layer*). Відповідно, є вхідний прошарок, який отримує

інформацію, n прихованих прошарків, які її обробляють, і вихідний прошарок, який виводить результат.

У кожного з нейронів є 2 основні параметри: вхідні I (input) і вихідні O (output) дані. Синапс – це зв'язок між двома нейронами. У синапсів є 1 параметр – вага w (weight). Завдяки йому, вхідна інформація змінюється, коли передається від одного нейрона до іншого. Той нейрон, у якого вага на синапсах більша, буде передавати домінуючу інформацію наступному нейрону. Насправді сукупність ваг нейронної мережі або матриця ваг (weight matrix) – це своєрідний мозок всієї системи. Саме завдяки цим вагам вхідна інформація оброблюється і перетворюється в результат.

У разі вхідного нейрона: $I = O$, а для інших нейронів до поля I потрапляє сумарна інформація від усіх нейронів попереднього прошарку x, після чого вона нормалізується за допомогою функції активації f (або передавання) і надходить до поля O. Схема роботи штучного нейрона наведена на рисунку 1.3 [19].

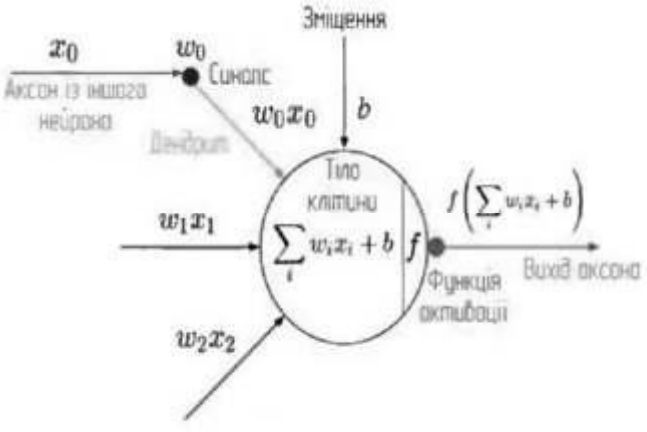


Рисунок 1.3 – Модель штучного нейрону

Шлях, яким нейрони з'єднуються між собою має значний вплив на роботу мережі. Більшість програмних моделей дозволяють користувачу додавати, вилучати та керувати з'єднаннями як завгодно. Коригуючи параметри, зв'язки можна робити як збуджуючими, так і гальмуючими.

Процес навчання (learning, training) нейронних мереж може розглядатися як визначення архітектури мережі й ітеративне налаштування вагових коефіцієнтів синаптичних зв'язків для ефективного виконання спеціально заданої задачі. Під час навчання нейронних мереж часто виникає проблема перенавчання (overfitting) або надмірно близької підгонки, тобто зайвої точної відповідності нейромережі певному набору навчальних прикладів, тобто мережа втрачає здатність до узагальнення.

1.3 Огляд архітектур нейромереж

За архітектурою зв'язків більшість відомих нейромереж, які знайшли практичне застосування, можна розділити на одношарові, багатошарові або з прямими зв’язками (перцептрони, карти Кохонена, глибокі мережі тощо) та з рекурентними зворотними зв'язками (мережі Хопфілда, Ліппмана, Коско) [18, 20, 21].

Хопфілд вперше представив свою автоасоціативну мережу (пам’ять) у 1982 р. На честь Хопфілда та нового підходу до моделювання, ця мережева парадигма згадується як мережа Хопфілда (Hopfield network). Вона базується на аналогії з фізикою динамічних систем, використовує три прошарки: вхідний, прошарок Хопфілда та вихідний прошарок. Кожен прошарок має однакову кількість нейронів.

Входи прошарку Хопфілда під'єднані до виходів відповідних нейронів вхідного прошарку через змінні ваги з'єднань. Виходи прошарку Хопфілда під'єднуються до входів всіх нейронів прошарку Хопфілда, за винятком самого себе. У режимі функціонування мережа скеровує дані з вхідного прошарку через фіксовані ваги з'єднань до прошарку Хопфілда, який коливається, поки не буде завершена певна кількість ітерацій, при цьому поточний стан прошарку передається на вихідний прошарок. Цей стан відповідає запрограмованому образу (рисунок 1.4).

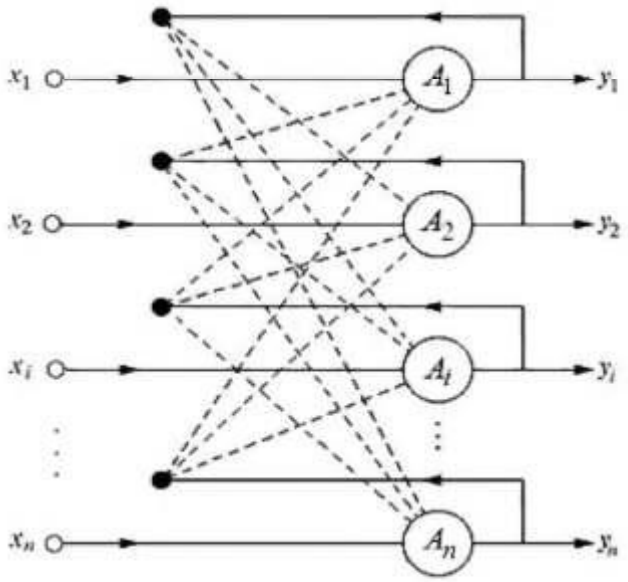


Рисунок 1.4 – Архітектура нейромережі Хопфілда

Для навчання мережі Хопфілда необхідно, щоб навчальний образ був представлений одночасно на вхідному та вихідному прошарках. Рекурсивний характер прошарку Хопфілда дозволяє виконувати корекцію всіх ваг з'єднань.

Гетероасоціативна мережа або двонаправлена асоціативна пам'ять була розроблена Коско (Kosko network), як модифікація моделі Хопфілда. Подібно до мережі Хопфілда, коли задано зашумлену версію одного

образу, також визначається асоційований з ним найближчий образ (видається його номер у списку). На рисунку 1.5 показаний приклад асоціативної пам'яті. Кількість входів мережі дорівнює кількості вихідних нейронів. Середні прошарки зв'язані один з одним. Вхідний та вихідний прошарки реалізують засоби введення та відновлення даних у мережі. Навчання подібне до мережі Хопфілда [15].

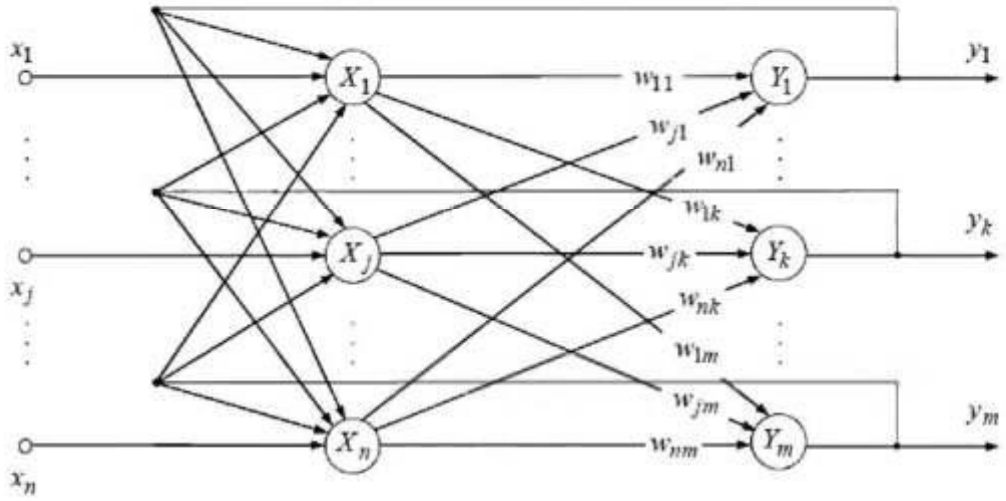


Рисунок 1.5 – Архітектура нейромережі Коско

Однією з простих та зручних штучних мереж для розпізнавання зображень є нейронна мережа Ліппмана (Lippmann network), яка розроблена в 1987 році Р. Ліппманом. Це релаксаційна багатошарова мережа зі зворотнім зв'язком між окремими шарами, яка розпізнає і класифікує зображення з використанням асоціативної пам'яті. По суті, вона складається з двох підмереж – мережі Хеммінга і MaxNet (стратегія "переможець забирає все") або мережі Хопфілда [20, 21].

Мережа представляє класифікатор, який базується на найменшій похибці для векторів двійкових входів (біполярний сигнал). Вектор виходів

навчальної множини є вектором класів, до яких належать образи. У режимі навчання вхідні вектори розподіляються до категорій, для яких відстань Хеммінга між зразковими вхідними векторами та тестовим вхідним вектором є мінімальною. Навчання мережі Хеммінга є подібним до мережі Хопфілда.

Дана мережа не вимагає виконання трудомістких обчислювальних операцій і може використовуватися для розпізнавання зображень, які складаються лише з чорних і білих пікселів, наприклад, текстові дані, підписи людей, штампи, штрих-коди тощо. За її допомогою можна організувати систему автоматичної корекції помилок. Саме дану мережу будемо досліджувати у цій роботі, тому її архітектура та математична основа буде представлена у розділі 2.

Мережа розроблена Кохоненом (Kohonen map) на початку 80-х рр. використовує для навчання метод послідовних наближень, а навчальна вибірка складається лише зі значень вхідних змінних. Вона може вирішувати задачі класифікації та кластеризації. Відмінність кластеризації від класифікації в тому, що перелік груп не заданий, а визначається в процесі роботи алгоритму. Мережа розпізнає кластери в навчальних даних і розподіляє дані до відповідних кластерів [15].

Якщо мережа зустрічається з набором даних, несхожим ні на один із відомих зразків, вона відносить його до нового кластеру. Мережу Кохонена можна використовувати і в задачах, де класи відомі – перевага буде у спроможності мережі виявляти подібність між різноманітними класами. Мережа Кохонена на рисунку 1.6 часто називається самоорганізованою картою (self-organized map). Її елементи розташовуються в певному просторі, зазвичай двовимірному.

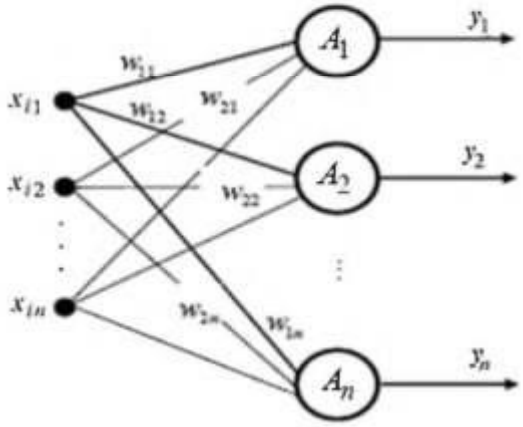


Рисунок 1.6 – Архітектура нейромережі Кохонена

Мережа з багатошаровими перцептронами (multilayer perceptron network) розроблена у 1970-х років декількома незалежними авторами (Вербор, Паркер, Румельгарт, Хінтон та Вільямс) [17]. Структура нейронної мережі з багатошаровими перцептронами наведена на рисунку 1.7, на якому нейрони організовані в пошарову структуру з прямим передаванням сигналу. Кожний нейрон мережі продукує зважену суму своїх входів, пропускає її через функцію активації і видає вихідне значення. Мережа може моделювати функцію практично будь якої складності, причому кількість прошарків і нейронів визначають складність функції.

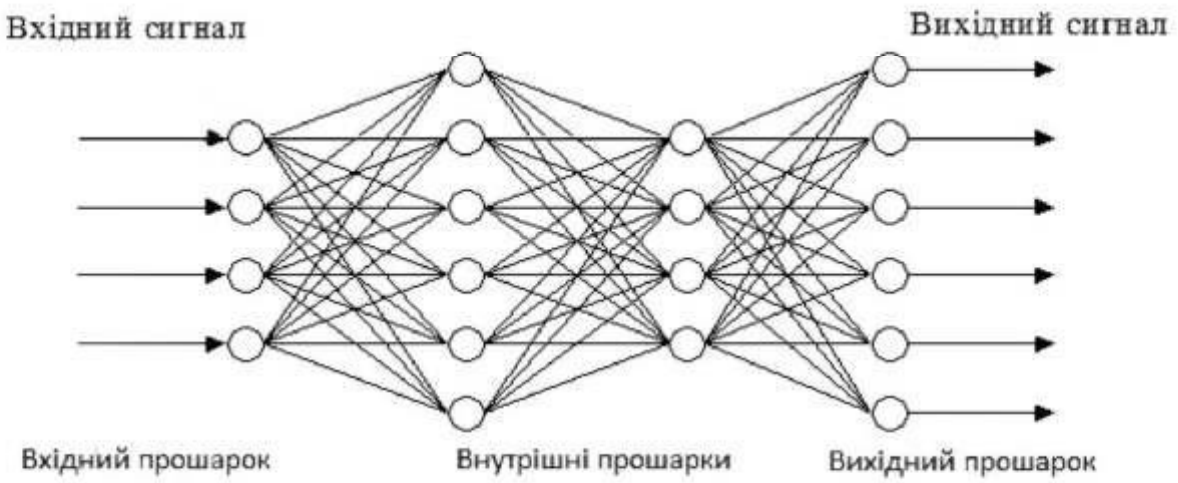


Рисунок 1.7 – Архітектура нейромережі з перцептронами

Визначення кількостей проміжних прошарків і нейронів в них є важливим при моделюванні даної мережі. Зазвичай кількість входів та виходів мережі визначаються кількістю вхідних та вихідних параметрів досліджуваного об'єкта, явища, процесу тощо. На відміну від зовнішніх прошарків, кількість нейронів прихованого прошарку обирається емпіричним шляхом. У більшості випадків достатня кількість нейронів становить $n_{прих} \approx n_{вх} + n_{вих}$, де $n_{вх}$, $n_{вих}$ – кількість нейронів у вхідному і, відповідно, у вихідному прошарках.

У ході роботи даної мережі потрібно знайти значення для синаптичних ваг, які спроможні мінімізувати похибку результату. Саме для цього існують алгоритми навчання, за допомогою яких відбувається оптимізація нейромережі до навчальних даних. Похибка визначається у ході проходження через мережу навчальних прикладів і подальшому порівнянню вихідних значень з реальними. Множина таких похибок задає функцію похибок, значення якої розглядається як похибка всієї нейромережі.

Для обмеження простору пошуку під час навчання ставиться задача мінімізації цільової функції похибки нейромережі. На даний час алгоритм, який засновано на зворотному поширенні похибки найбільш популярна, ефективна та легка модель для навчання даної нейромережі. Назва алгоритму пов’язана зі способом розрахунку похибок у конкретних шарах.

Згорткові нейронні мережі (convolutional neural networks) були запропоновані ЛеКуном та є підвидом глибоких мереж (deep networks). Їх основною особливістю є чергування згорткових (convolution) та субдискретизуючих (*subsampling*) прошарків і наявність повнозв'язних прошарків (fully layers) на виході.

Згорткові прошарки виконують операцію згортки над зображенням з деяким ядром (core), що дозволяє виділити певні ознаки зображення із збереженням його топології. Субдискретизуючі прошарки зменшують просторову розмірність зображення, що забезпечує інваріантність до масштабу. Чергування прошарків дозволяє складати карти ознак, що на практиці означає здатність до розпізнавання елементів (рисунок 1.8).

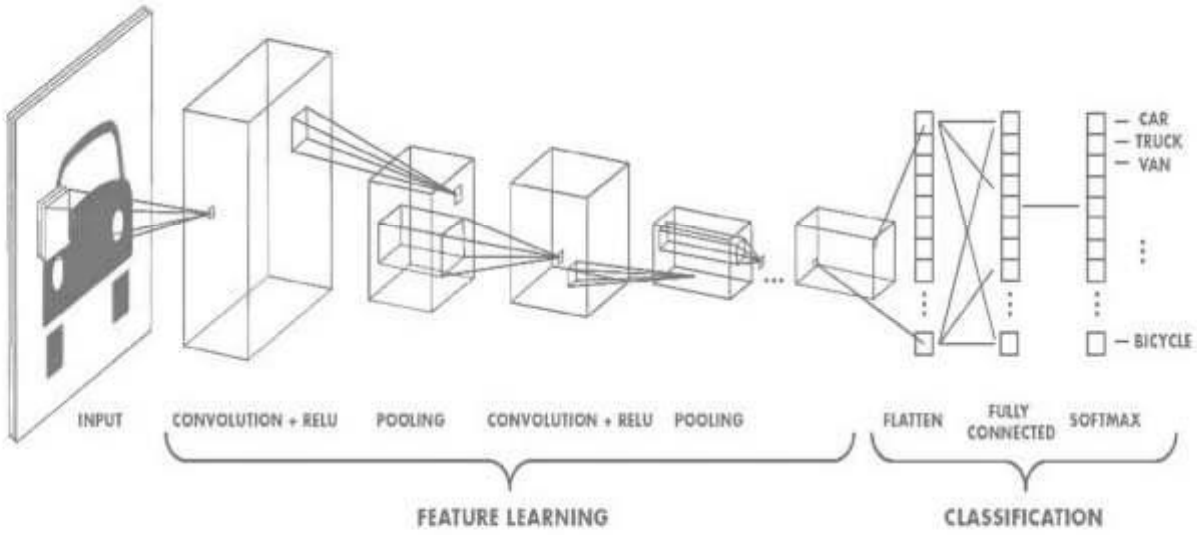


Рисунок 1.8 – Архітектура глибоких нейромереж

Основна ідея навчання базується на використанні алгоритму зворотного поширення похибки у комбінації з оптимізатором, наприклад Adam, Adagrad тощо [17]. Подібні мережі є найбільш сучасними і поширеними в багатьох практичних додатках для роботи з зображеннями – краулери-розпізнавачі у соцмережах, додатки Google (Arts, Artistic Networks) та Facebook (Masquerade), штучний зір тощо.

1.4 Висновки

У даному розділі виконано постановку задачі розпізнавання образів, наведено основну термінологію. Визначено, що перспективним підходом до розв'язання даної задачі є використання штучних нейромереж, математичний апарат яких пов'язаний з аналогіями обробки інформації людським мозком. Представлено моделі біологічного та штучного нейронів, описано ідею навчання нейромереж, розглянуто основні архітектури.

2 РОЗРОБКА МОДИФІКОВАНОГО АЛГОРИТМУ РОЗПІЗНАВАННЯ ДВОВИМІРНИХ ШТРИХ-КОДІВ

2.1 Особливості роботи з двовимірними штрих-кодами

Штрих-код – спосіб запису даних (найчастіше за допомогою смуг різної товщини), який містить певну інформацію та зручний для зчитування сканером. Використовується в торгівлі для ідентифікації товару, на квитках, документах, авто тощо. Присвоєнням штрих-кодів займається міжнародна некомерційна і неурядова організація – Асоціація *EAN*. Алгоритм перевірка підробки:

1. Додати всі цифри, які стоять на парних місцях.

2. Суму помножити на 3 та отримати результат *X*.

3. Скласти всі цифри, які стоять на непарних місцях (без контрольної цифри).

4. Додати до цієї суми *X*.

5. Від отриманої суми *YZ* залишити тільки одиниці *Z*, тобто виконати операцію *YZ* mod 10.

6. Виконати перевірку 10 – *Z*.

7. Якщо результат збігається з контрольною цифрою в штрих-коді – товар не підробка.

Подальший розвиток систем обміну інформацією призвів до розробки двовимірних штрих-кодів. Найбільш популярним є QR-код (рисунок 2.1). Даний код був розроблений відповідно до потреб автопромисловості у 1994 р. японською фірмою Denso Wave, яка належала

компанії Toyota. Основна задача даного коду була у збереженні великого об'єму даних на малій площі (не менше 2,5 $см^2$), причому скануванню не повинні завадити ні часткове забруднення, ні сильне пошкодження зображення [22-24].

Рисунок 2.1 – Приклад QR-коду

На практиці використовують стандартні коди від 1 (21 на 21 піксель) до 40 версії (177 на 177 пікселів). Але з застосуванням високоточної лазерної техніки на поверхні будь-яких елементів можна вигравіювати крихітні QR-коди. Також можна використовувати наночорнила (синій та зелений кольори), які можна побачити лише в інфрачервоному спектрі кольорів, що може застосовуватися для боротьби з фальшивомонетниками.

Зараз QR-код зустрічається майже всюди, наприклад за даними компанії comScore більше 20 млн. мешканців США використовують смартфони для сканування даного коду, в Японії він наноситься практично на всі товари, буклети, довідники, різну продукцію тощо.

Отже, його основна перевага – легке розпізнавання сканувальним обладнанням (наприклад, фотокамера смартфона) та велика інформаційна ємність (7089 цифр, 4296 літер), що дає можливість використовувати

даний код в торгівлі, на виробництві, в логістиці тощо. Тому саме його і будемо розглядати в даній роботі.

Щоб зрозуміти, як "витягти" інформацію з даного коду, потрібно розібратися в алгоритмах кодування та декодування. Існує кілька стандартів сімейства QR-кодів, основні принципи яких наведені в специфікаціях. Основним інструментом для кодування інформації в таких кодах є код Ріда-Соломона [25].

Коди Ріда-Соломона – це блокові коди з можливістю виправлення помилок, які вперше були описані в статті І. Ріда та Г. Соломона у 1960 році. Процес кодування для даних кодів є доволі простим та зазвичай не займає багато часу, на відміну від декодування – дуже часозатратного процесу. Даний недолік частково був усунутий Е. Берлекампом, але час декодування залишився доволі великим порівняно з іншими кодами. Лише за останні декілька років стало можливим надсилання даних з високою пропускною здатністю за допомогою даних кодів [2].

Декодування представляє собою зворотну до кодування операцію – ділення кодового слова c на поліном g. Основна задача синдромного декодера полягає в обчисленні синдрому помилки (залишок від ділення) з отриманого повідомлення. Якщо результат не нульовий, то це свідчить про наявність помилок в отриманому кодовому слові, в іншому випадку передавання повідомлення відбулось успішно.

Процес знаходження синдрому помилки виконується ітеративно за $2t$ ітерацій, оскільки кодове слово Ріда-Соломона має $2t$ синдромів. Отриманий синдром описує конфігурацію помилки, але не містить інформації про її позицію в повідомленні. Декодер Ріда-Соломона може виправити до t символів, які містять помилки, де $2t = n - k$.

Процедури алгебраїчного декодування кодів Ріда-Соломона можуть визначати будь-які комбінацій символів, які містять $2t$ помилок, або виправляти t помилок. Також існує можливість виправити t помилок стирання у відомих позиціях та відновити дані. Помилки стирання відбуваються тоді, коли відомо про позицію помилкового символу.

Структура декодера кодів Ріда-Соломона зображена на рисунку 2.2, де y – отримане повідомлення, S_i – синдром, L – поліном-локатор, v – кількість помилок, X_i – позиція помилки, Y_i – величини помилок, c – кодове слово [2].

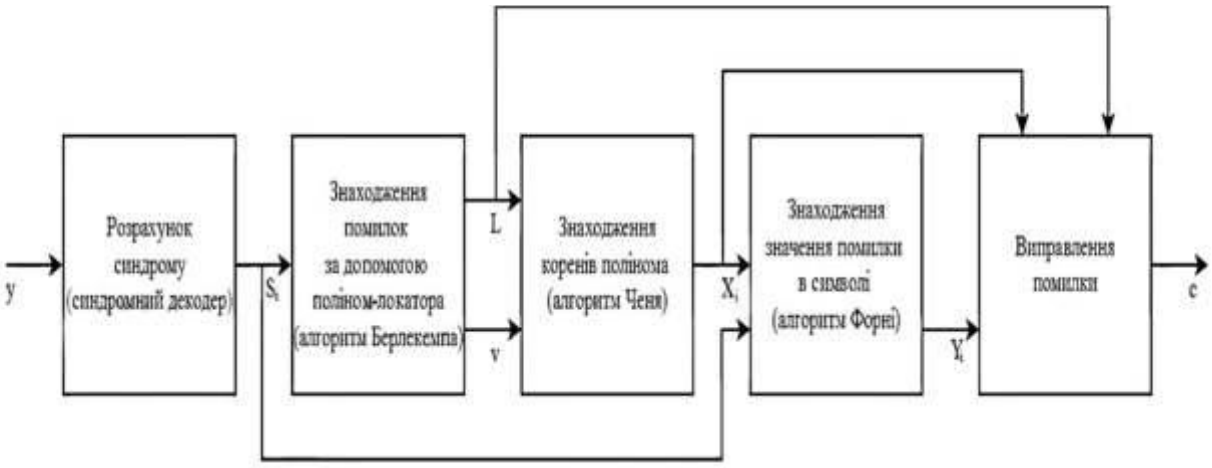


Рисунок 2.2 – Загальна структура декодера кодів Ріда-Соломона

Як показує практика, зазвичай більшу частину QR-матриці займають коригуючі коди. За стандартом дані з RS-кодами перед записом в картинку "перемішуються" та записуються в особливій послідовності на шаблонну картинку, куди додається технічна інформація для декодерів.

Для цих цілей використовують спеціальні динамічні маски та статична маска s з 15 біт виду 101010000010010. Існує 8 алгоритмів

формування динамічної маски, серед яких вибирається найкращий за певним критерієм. Критерії вибору засновані на системі штрафів.

Штраф призначається:

- за кожну групу з п'яти (3 бали) або більше (1 бал за кожний піксель після п'ятого) одноколірних пікселів в рядку / стовпці;
- за кожен одноколірний квадрат розміром 2x2 пікселя;
- якщо на матриці є області, які схожі на елементи орієнтації;
- якщо більше половини пікселів одного кольору.

Далі розраховується загальна сума штрафних балів E за чотирма правилами для кожної з 8-ми масок. У результаті для кодування вибирається маска A, яка має мінімальну кількість штрафних балів $N_A = \min_{i=1,8}(E_i)$.

2.2 Алгоритм розпізнавання двовимірних штрих-кодів

Таким чином, під час роботи з двовимірними штрих-кодами необхідно для початку проаналізувати матрицю P пікселів розміром $H \times B$, де H і B – висота і ширина заданого зображення, визначаючи кількість темних пікселів. Для визначення, що аналізований піксель є темним, потрібно виконати перетворення у відтінки сірого за виразом [3, 32-34]

$$Y = 0{,}299 \cdot R + 0{,}587 \cdot G + 0{,}114 \cdot B, \qquad (2.1)$$

де R, G, B – значення відповідних колірних складових.

Після виконання даних дій вибираємо певне граничне значення L та порівнюємо його з числом Y. Якщо $0 \le Y \le L$, тоді відповідний піксель вважається темним, інакше – білим. Тепер можна записати двійкову матрицю зображення розміром $m = H \times B$ (рисунок 2.3). Отже, для еталонних зображень n можна визначити відповідні бінарні вектори, які будуть використовуватися для навчання мережі і розпізнавання зразків.

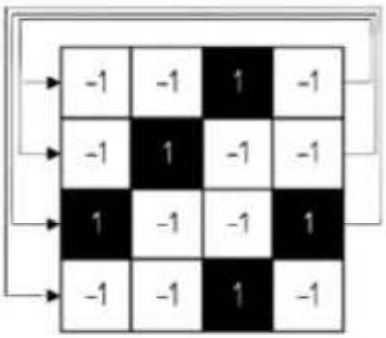


Рисунок 2.3 – Обробка зображення порядково

Потім необхідно виконати процес навчання мережі, використовуючи базу даних із еталонними зображеннями. Нейромережа Ліппмана (рисунок 2.4) задається m вхідними (S), n прихованими (Z, A), n вихідними (Y) нейронами та 4 прошарками. Значення n дорівнює кількості навчальних зображень, збережених мережею і представлених у формі m-мірних біполярних векторів $S = (s_1, \ldots, s_m)$. Також у цій мережі використовуються нейрон зміщення b. Матриця ваг W розмірністю $m \times n$ містить значення синаптичних зв'язків мережі прямого поширення, матриця E розмірності $n \times n$ зберігає значення синаптичних зв'язків нейронів мережі MaxNet (або Хопфілда). Функція активації нейронів f є пороговою [13, 14, 20, 21, 26].

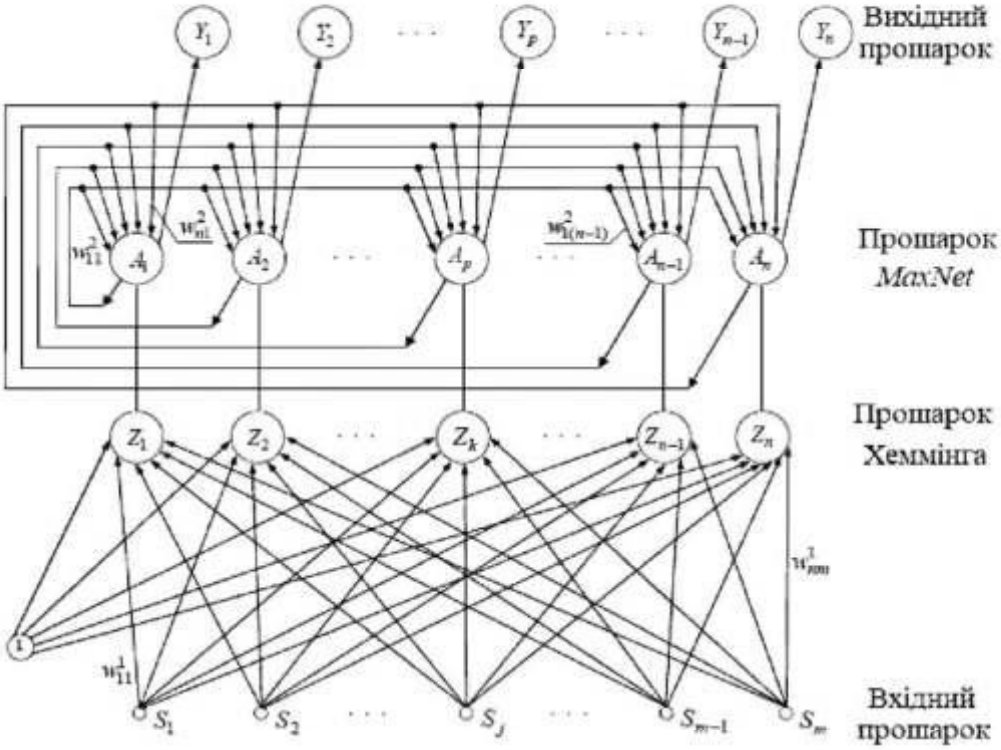


Рисунок 2.4 – Архітектура нейромережі Ліппмана

Значення ваг і порогові значення даної мережі визначаються з умов задачі, тому така мережа є штучною нейронною мережею з фіксованими з'єднаннями. Ключовою особливістю є той факт, що виходом мережі є не значення еталонного вектора, який найбільше схожий на вхід, а номер відповідної навчальної вибірки з підмережі *MaxNet* за принципом "переможець отримує все". Таким чином, тільки один "нейрон-переможець" залишається активним на виході всієї мережі [27].

Дана мережа представляє класифікатор, заснований на відстані Хеммінга, яка використовується, як міра подібності R вхідних і еталонних або навчальних зображень. Її можна визначити за співвідношенням

$$R = m - d(X, Y) = a, \tag{2.2}$$

де m – довжина еталонного або вхідного вектора; $d(X, Y)$ – відстань Хеммінга між векторами $X = (x_1, \ldots, x_m)$ та $Y = (y_1, \ldots, y_m)$; a – число однакових складових векторів.

Відстань Хеммінга визначається за виразом [26]

$$d(X,Y) = W(X \oplus Y) = \sum_{i=1}^{m} |x_i - y_i|, \tag{2.3}$$

де W – вагова функція.

Вихідний сигнал із входу S нейронів дорівнює

$$U_{output\ s_i} = f_s\left(U_{input\ s_i}\right) = s_i. \tag{2.4}$$

Запишемо для біполярних векторів X та Y їхній скалярний добуток, враховуючи однакові та відмінні складові, тобто

$$XY = \sum_{i=1}^{m} x_i \cdot y_i = a - d(X,Y). \tag{2.5}$$

Оскільки $m = a + d(X,Y)$, то формулу (2.5) можна записати у вигляді

$$XY = a - (m - a) = 2a - m. \tag{2.6}$$

На основі формул (2.5) та (2.6) можна отримати

$$a = \frac{m+XY}{2} = \frac{m}{2} + \frac{\sum_{i=1}^{m} x_i \cdot y_i}{2}. \tag{2.7}$$

Тоді кожний із Z нейронів (від 1 до n) розраховує свій вхідний сигнал відповідно до виразу

$$U_{input\ Z_k} = b + \sum_{i=1}^{m} w_{ik} \cdot U_{output\ S_i} = \frac{m}{2} + \frac{1}{2}\sum_{i=1}^{m} w_{ik} \cdot s_i, \quad (2.8)$$

де b – значення зміщення; w_{ik} – вага від i вхідного нейрону до k нейрону Z (значення +1 або -1).

Далі обчислюється значення функції активації для прихованих Z нейронів за таким правилом [20]

$$U_{output\ Z_k} = f_z\left(U_{input\ Z_k}\right) = \begin{cases} 0, & \text{якщо } U_{input\ Z_k} \leq 0; \\ v \cdot U_{input\ Z_k}, & \text{якщо } 0 < U_{input\ Z_k} \leq U; \\ U, & \text{якщо } U_{input\ Z_k} \geq U, \end{cases} \quad (2.9)$$

де v – параметр нормалізації; $U = \frac{1}{v}$ – константа.

Вихідний сигнал Z нейронів формує вхідний сигнал підмережі *MaxNet*, що можна записати у такому вигляді

$$U_{input\ A_k} = U_{output\ Z_k}. \quad (2.10)$$

Мережа MaxNet схожа за структурою на мережу Хопфілда, вона використовує вузли з пороговою логікою замість вузлів з жорстким обмеженням і передає вихідні дані кожного вузла назад, як вхідні дані, замість того, щоб забороняти шлях зворотного зв'язку. Дана підмережа використовує ітераційний процес отримання максимального вихідного сигналу в наступному вигляді [21]

$$U_{output\ A_k}(0) = U_{output\ Z_k}, \tag{2.11}$$

$$U_{input\ A_k}(t+1) = \sum_{i=1}^{n} w_{ik} \cdot U_{output\ A_i}(t) =$$

$$= U_{output\ A_k}(t) - \varepsilon \cdot \sum_{i=1, i \neq k}^{n} U_{output\ A_i}(t), \tag{2.12}$$

де t і $(t+1)$ – попередня та наступна ітерації, $0 < \varepsilon \leq 1$; $w_{ik} = 1$ для мережі MaxNet, якщо $i = k$ (для мережі Хопфілда $w_{ik} = 0$) та $-\varepsilon$ – в іншому випадку.

Далі розраховуємо значення функції активації для прихованих A нейронів за таким правилом

$$U_{output\ A_k}(t) = f_A\left(U_{input\ A_k}(t)\right) = \begin{cases} U_{input\ A_k}(t), & \text{якщо } U_{input\ A_k}(t) > 0; \\ 0, & \text{якщо } U_{input\ A_k}(t) \leq 0. \end{cases} \tag{2.13}$$

Ітераційний процес зупиняється, коли норма двох векторів стане меншою за значення eps_1. Отже, умова стабілізації задається у такому вигляді [26]

$$\left\| U_{output\ A_k}(t+1) - U_{output\ A_k}(t) \right\| = \sqrt{\sum_{k=1}^{n} (U_{output\ A_k}(t+1) - U_{output\ A_k}(t))^2} \leq eps_1. \tag{2.14}$$

Вихідні сигнали нейронів A формують вхідні сигнали нейронів Y, що можна показати у такому вигляді

$$U_{input\ Y_k} = U_{output\ A_k}. \tag{2.15}$$

Потім обчислюється значення функції активації для вихідних Y нейронів з використанням відхилення $eps_2 = 10^{-5}$-$10^{-7} \rightarrow 0$. Це можна зобразити в такому вигляді

$$U_{output\,Y_k} = f_Y\left(U_{input\,Y_k}\right) = \begin{cases} 1, & \text{якщо } U_{input\,Y_k} > eps_2; \\ 0, & \text{якщо } U_{input\,Y_k} \leq eps_2, \end{cases} \quad (2.16)$$

або можна вибрати максимальне значення $U_{input\,Y}$, тобто

$$U_{output\,Y_k} = \max_{\overline{1,n}}(\{U_{input\,Y_k}\}). \quad (2.17)$$

Індекс позитивного елемента $k = \overline{1,n}$ вихідного сигналу нейронів $U_{output\,Y}$ буде вказувати номер еталонного зображення, яке найкраще підходить вхідному тестовому сигналу.

Виходом мережі в ідеальному випадку повинен бути вектор з одним позитивним і всіма іншими нульовими елементами. Але, як це часто буває, вхідна вибірка може бути дуже зашумленою, і на виході мережі може з'явитися кілька позитивних значень. У цьому випадку зазвичай вважається, що мережа не може однозначно класифікувати вхідне зображення.

2.3 Модифікація процедури обробки даних

Необхідно задати розмір ковзного вікна K, який дорівнює кількості рядків (або стовпців), які будуть оброблятися (рисунок 2.5). Змінюючи

значення K, можна впливати на точність аналізу зображення. Далі задається порогова величина Q.

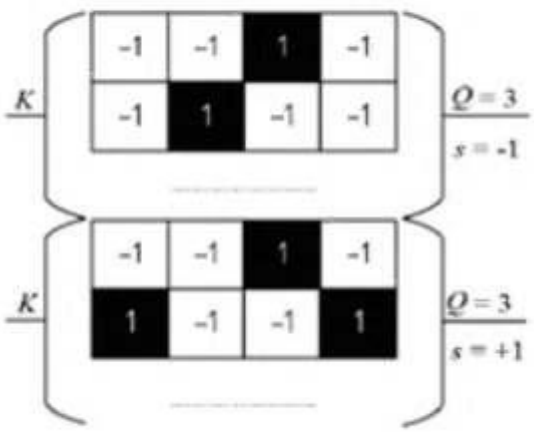


Рисунок 2.5 – Обробка зображення в режимі ковзного вікна

Обробляється K рядків та обчислюється N_i – кількість темних пікселів. Повторити для всіх рядків: якщо $N_i \geq Q$, тоді i-ий елемент вектору S_i дорівнює +1, інакше – -1. Отриманий вектор S із m елементами подати на вхідні нейрони. Тепер потрібно створити матрицю ваг W розмірністю:

$$M = m \times n = \lceil X / K \rceil \times n, \tag{2.18}$$

де X – висота H або ширина B даного зображення; $\lceil . \rceil$ – округлення числа до верхнього цілого.

Модифікація значно зменшить довжину вхідного вектору (особливо для зображень великих розмірів), що, в свою чергу, спростить обчислювальну складність навчання мережі.

2.4 Висновки

У даному розділі розглянуто особливості обробки двовимірних штрих-кодів, представлено модифікований алгоритм їх розпізнавання на основі нейромережі Ліппмана. Навчання нейромережі складається з двох частин: обробка зображень, розпізнавання і корекція зразка.

Описано математичний апарат, який застосовується для навчання та розпізнавання, наведено архітектуру мережі та показано приклад її роботи. Істотний недолік даної нейромережі полягає в тому, що вона не виділяє два або більше еталонних зображення, які мають однакові максимальні міри схожості. Перевагами є простота та достатня ефективність для вирішення певних задач аналізу даних.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

3.1 Вибір мови програмування та опис програми

Однією з найбільш популярних мов програмування для розв'язання інженерних задач є Python (рисунок 3.1). Це об'єктно-орієнтовна мова програмування з автоматичним управлінням ресурсів пам'яті. До переваг можна віднести простоту використання та різноманіття ефективних спеціалізованих бібліотек, наприклад SciPy та NumPy, які містять готові реалізації математичних функцій. Наприклад, бібліотека NumPy є досить ефективною для всіх видів математичних обчислень. Із недоліків можна виділити динамічну типізацію, яка в загальному випадку призводить до зменшення ефективності та часу виконання програми [28].

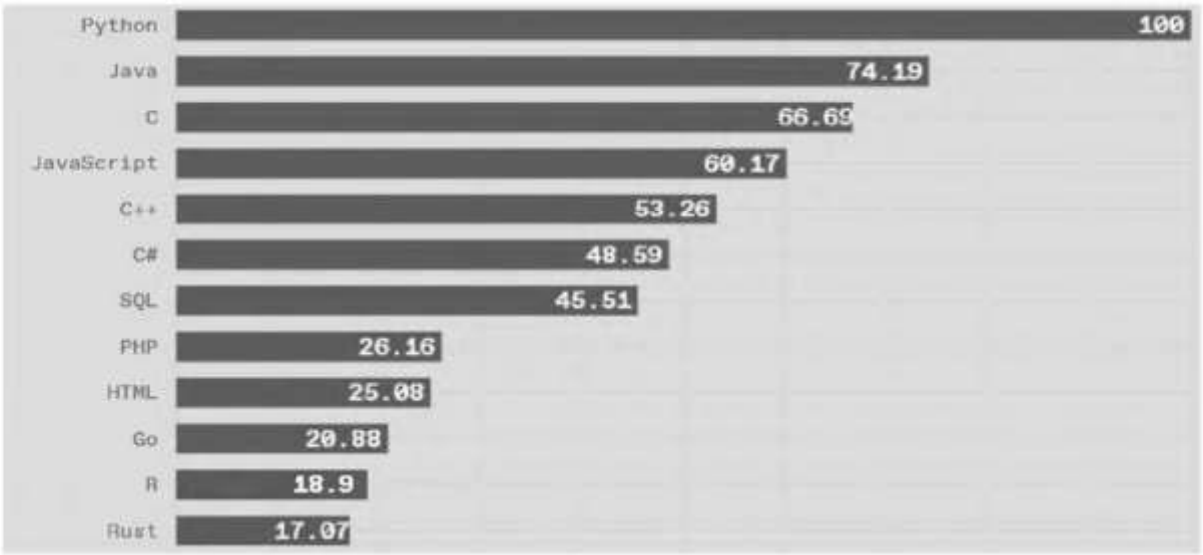


Рисунок 3.1 – Рейтинг мов програмування IEEE

Дана мова програмування поєднує потужність і гнучкість з високою ефективністю виконавчого коду й можливістю доступу до апаратних ресурсів комп'ютера. Тому програмне забезпечення для розв'язання поставленої задачі буде розроблятися саме на цій мові.

Основні технічні та мінімальні системні вимоги до розробленої програми: операційна система *Windows* 7 або новіша; наявність клавіатури та миші; процесор *Core* i5 (*Ryzen* 5) або кращий; оперативна пам'ять – 8 Гб; відеопам'ять – 1 Гб; жорсткий диск – 500 Гб і більше. Програмне забезпечення не тестувалось на іншому обладнанні, але можна вважати, що воно буде коректно працювати і на обладнанні нижчого рівня, оскільки не вимагає великих обчислювальних потужностей.

Схема програми представлена в додатках. Вхідні дані: база даних з образами; матриця пікселів $H \times B$; нейромережа Ліппмана; константа масштабування k_1; порогові параметри K та Q; кількість ітерацій роботи мережі MaxNet. Результати роботи: класифікований образ із бази даних або висновок про те, що зразок не можна розпізнати точно.

3.2 Експериментальні дослідження

Спочатку відбувається аналіз коду сканером (розпізнавання чорних та білих областей). Поглянувши на зображення QR-кода (рисунок 3.2), можна помітити низку виділених областей (не представляють інтересу з точки зору записаної інформації), які використовуються для детектування QR-коду та фокусування зчитуючого пристрою на ньому [3, 22].

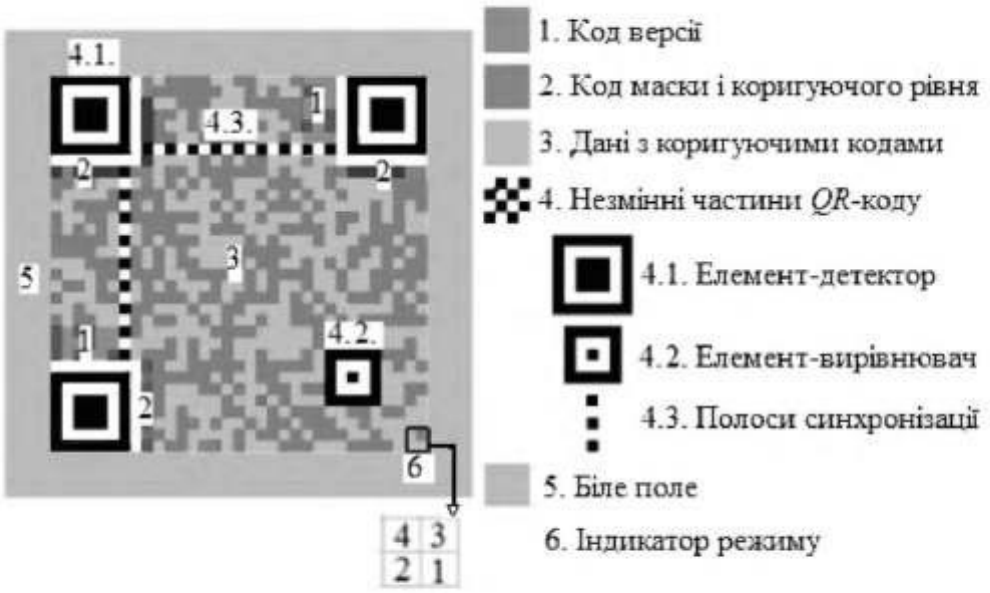


Рисунок 3.2 – Структура матриці QR-коду

Все інше поле коду несе корисну інформацію (системна інформація, дані, версія коду). Важливим елементом QR-коду є системна інформація *I*, яка дублюється, що дозволяє значно знизити ймовірність виникнення помилок під час зчитування коду. Вона складається з 15 бінарних символів: перших 5 біт корисної інформації *I* та 10 біт коригуючого коду, декодування якого дозволить скоригувати помилки у *I*. Для визначення рівня корекції помилок та номеру шаблона динамічної маски отримані 5 біт побітово додають за модулем два (*XOR*) до статичної маски:

$$[R_{1,2}, A_{3,4,5}] = \sum_{i=1}^{5} I_i \oplus s_i . \qquad (3.1)$$

Від інформації про версію коду залежить максимальний обсяг даних, які можуть бути записані в матрицю QR-коду. Визначити версію кода можна декількома способами [3, 22]:

– розрахувати кількість пікселів (модулів) матриці (таблиця 3.1);

Таблиця 3.1 – Визначення версії коду за кількістю пікселів

Версія	Кількість модулів
1	21x21
2	25x25
3	29x29
4	33x33
5	37x37
6	41x41
…	…
40	177x177

– визначити системну інформацію про версію кода, такі області є на матрицях старше 6-ої версії (таблиця 3.2);

– визначити координати елементів-вирівнювачів (номера стрічок та стовпців наведено в специфікаціях).

Аналізуючи таблицю 3.1, легко помітити лінійну закономірність: чергова версія QR-коду відрізняється від попередньої версії тим, що сторона матриці збільшена на 4 пікселі.

Далі необхідно застосування динамічної маски *A*. Даний етап відбувається з використанням шаблонів з таблиці 3.3. Початок координат (0, 0) у лівому верхньому куті матриці коду. Якщо вираз для генерації матриці зводиться до *true* для біта *b* з координатами (*i*, *j*), то даний біт інвертується (1, чорний модуль), інакше (*false*) – все залишається без змін (0, білий модуль), тобто

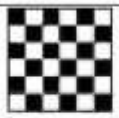
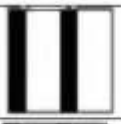
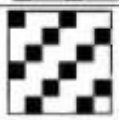
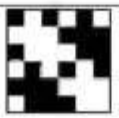
$$b_{i,j} = \begin{cases} 1, \ if \ T_{i,j} = true; \\ 0, \ if \ T_{i,j} = false. \end{cases} \quad (3.2)$$

Відбувається перетворення зображення в бінарну форму, причому чорний піксель в матриці коду означає 1, а білий – 0.

Таблиця 3.2 – Визначення версії коду за системною інформацією

Версія	Бінарна послідовність	Версія	Бінарна послідовність
7	001010010011111000	24	001000110111000110
8	000111101101000100	25	000100001111100110
9	100110010101100100	26	110101011111010110
10	011001011001010100	27	000001110001110110
11	011011111101110100	28	010110000011001110
12	001000110111001100	29	001111110011101110
13	111000100001101100	30	101011101011011110
14	010110000011011100	31	000000101001111110
15	000101001001111100	32	101010111001000001
16	000111101101000010	33	000001111011100001
17	010111010001100010	34	010111010001010001
18	111010000101010010	35	011111001111110001
19	001001100101110010	36	110100001101001001
20	011001011001001010	37	001110100001101001
21	011000001011101010	38	001001100101011001
22	100100110001011010	39	010000010101111001
23	000110111111111010	40	100101100011000101

Таблиця 3.3 – Шаблони динамічних масок

Номер маски, A	Алгоритм генерації шаблона, T	Вигляд маски
000	$(i + j) \bmod 2 = 0$	
001	$i \bmod 2 = 0$	
010	$j \bmod 3 = 0$	
011	$(i + j) \bmod 3 = 0$	
100	$(\text{floor}(i/2) + \text{floor}(j/3)) \bmod 2 = 0$	
101	$(i \cdot j) \bmod 2 + (i \cdot j) \bmod 3 = 0$	
110	$((i \cdot j) \bmod 2 + (i \cdot j) \bmod 3) \bmod 2 = 0$	
111	$((i + j) \bmod 2 + (i \cdot j) \bmod 3) \bmod 2 = 0$	

На наступному кроці відбувається декодування коригуючих RS-кодів та проводиться виправлення знайдених помилок [22].

Для декодування режиму місткості коду (формату інформації) необхідно зчитати заголовок d (4-х біти) у правому нижньому куті кода, який описує режим (таблиця 3.4). Потім виконати побітове додавання за модулем два (*XOR*) бінарних символів заголовка та динамічної маски та за отриманим кодом визначити тип даних, які записані у матрицю:

$$Ind_i = \sum_{i=1}^{4} d_i \oplus Mask_i . \quad (3.3)$$

Таблиця 3.4 – Список режимів місткості коду

Режим	Код	Місткість
Числовий	0001	до 7089 цифр
Алфавітно-числовий	0010	до 4296 символів
Байтовий	0100	до 2953 байт (букв кирилиці за *win*1251)
Кандзі	1000	до 1817 ієроглифів

Читання та декодування інформації з *QR*-коду відбувається зигзагом. Перший блок даних (пакет) після ідентифікатора режиму відповідає кількості символів, інші представляють корисні дані. Аналогічно використовується динамічна маска та операція додавання за модулем два (*XOR*) для *N* символів у наступному вигляді

$$D_i = \sum_{i=5}^{N} d_i \oplus Mask_i . \quad (3.4)$$

Декодування виконується відповідно до правил двійкового числення, системи ASCII або іншого методу перетворення символів [3, 22].

Найбільш просто робота з QR-кодами виконується за допомогою мобільного телефону з вбудованою фотокамерою і встановленою програмою зчитування (для Android: QR Scanner, Barcode Scanner, QR Droid; для iPhone і iPad: RedLaser, QR Reader, Bakodo; для Windows Phone

та BlackBerry OS: додаток вбудовано). При відсутності у користувача пристроїв розпізнавання QR-коду кодування та декодування можна виконати за допомогою скриптів з Інтернет-ресурсів або розроблених додатків.

Але, якщо виникли помилки, то необхідно застосувати подальшу процедуру. Для реалізації досліджень було розроблено і налагоджено програмне забезпечення, схема роботи якого наведена на рисунку 3.3.

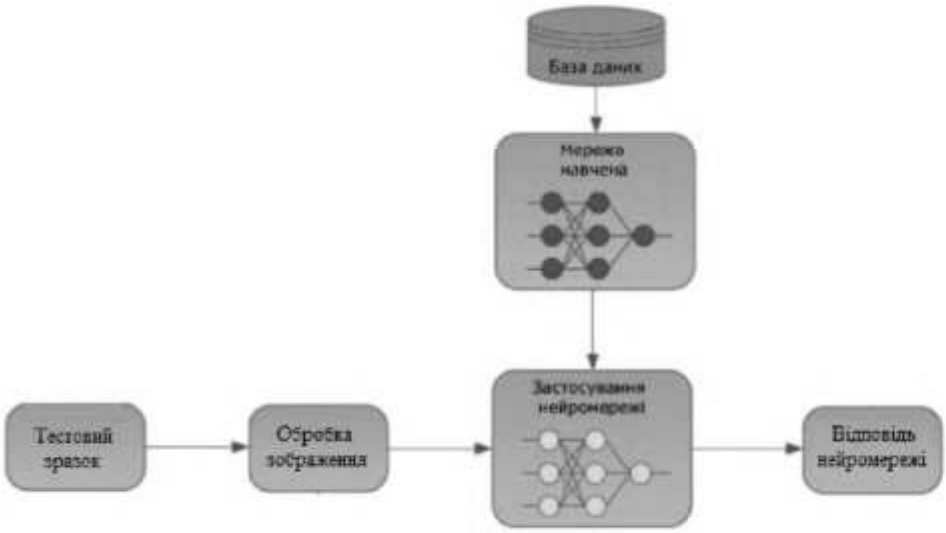


Рисунок 3.3 – Схема експерименту

На рисунках 3.4 і 3.5 показано 3 навчальних і тестових зображення, які не розпізнає декодуюче програмне забезпечення, оскільки структура детекторів пошкоджена, а коригуюча здатність кодів Ріда-Соломона не дає можливості виправити ключові помилки.

Застосуємо додаткову процедуру виправлення помилок, використовуючи запропонований алгоритм та базу еталонних QR-кодів. Для визначення оптимальних значень розміру ковзного вікна K та порогової величини Q виконано низку експериментів, в яких застосовано

пошкоджені (шум, відсутність деталей, злиття кольорів) зображення QR-кодів (розміри 512×512 та 1024×1024).

Рисунок 3.4 – Навчальні зображення

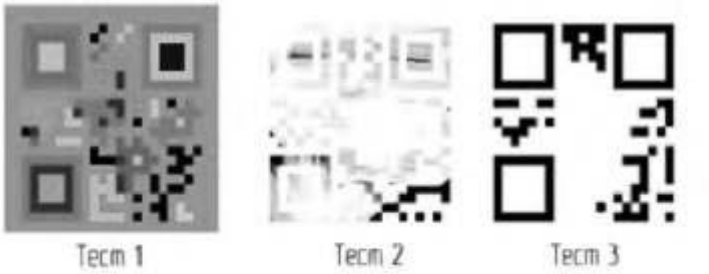


Рисунок 3.5 – Тестові зображення

При використанні одних комбінацій параметрів результати розпізнавання повністю збігалися з очікуваними, в разі використання інших комбінацій на виході мережі формувалося декілька позитивних значень, що не дозволяло виконати класифікацію вхідного образу. Це можна пояснити тим, що при використанні тієї або іншої пари параметрів K і Q формувалася матриця еталонних образів, в якій всі рядки досить мало відрізнялися один від одного, тому поданий на вхід зразок не міг бути віднесений до одного класу. Щоб домогтися коректного розпізнавання, необхідно експериментально виявляти оптимальні параметри навчання, завдяки яким в матриці еталонних образів рядки будуть відрізнятися.

Експериментальні результати розпізнавання зображень QR-кодів різних розмірів представлені в таблицях 3.5 та 3.6. Використовувалась лінійна залежність $Q = \alpha \cdot K + \beta$ для порогової величини Q і розміру ковзного вікна K, для якої експериментально вибрані значення констант $[\alpha; \beta] = [(200; 0), (400; 100)]$.

Таблиця 3.5 – Результати експерименту 1

K	Q	Тест 1	Тест 2	Тест 3	Кількість правильних відповідей	Результат
9-15	1800-3000	+	+	+	3	GOOD
16	3200	+	+	+	3	GOOD
17	3400	–	+	+	2	MEDIUM
19	3800	–	+	+	2	MEDIUM
21	4200	–	–	+	1	BAD
24	4800	+	–	+	2	MEDIUM
25	5000	–	+	–	1	BAD

Таблиця 3.6 – Результати експерименту 2

K	Q	Тест 1	Тест 2	Тест 3	Кількість правильних відповідей	Результат
1-25	500-10100	+	+	+	3	GOOD
26	10500	+	+	+	3	GOOD
27	10900	+	+	–	2	MEDIUM
28	11300	+	+	–	2	MEDIUM

Якщо порогова величина $Q \leq K$, то нейромережа не може правильно виконати розпізнавання зображень. Можна зробити висновок, що оптимальними значеннями (алгоритм правильно розпізнає всі тестові QR-коди) параметрів K і Q для тестових зображень є $[H \times B;\ K;\ Q] = [(512 \times 512;\ 16;\ 3200), (1024 \times 1024;\ 26;\ 10500)]$.

Для досягнення меншої обчислювальної складності навчання і більшої точності розпізнавання застосовано процедуру налаштування параметра K_t, коли (n-1) тестових зображень були розпізнані, наприклад, $[H \times B;\ \{K_t\}] = [(512 \times 512;\ \{17\text{-}20;\ 22\text{-}24;\ 26\}), (1024 \times 1024;\ \{27,\ 28\})]$. При цьому необхідно налаштовувати параметр Q. У таблиці 3.7 показано приклад процедури налаштування для розміру ковзного вікна $K = 9$ (розмір зображення 200×200), яке більше за оптимальне значення. Можна зробити висновок, що краще вибрати $K_t = 8$.

Таблиця 3.7 – Результати процедури налаштування

Q	Тест 1	Тест 2	Тест 3	Кількість правильних відповідей	Результат
455	–	+	–	1	BAD
465	–	+	–	1	BAD
480	+	+	–	2	BAD
490	+	–	–	1	BAD
500	+	+	–	2	BAD

3.3 Висновки

У даному розділі обгрунтовано вибір мови програмування, наведено основні вимоги до розробленого програмного забезпечення, яке розпізнає двовимірні штрих-коди, не потребуючи використання трудомістких обчислювальних операцій. У ході експериментальних досліджень визначено оптимальні параметри K та Q запропонованого алгоритму для зображень різного розміру.

4 РОЗДІЛ ЕКОНОМІКИ

4.1 Технологічний аудит удосконаленого алгоритму розпізнавання образів

Метою проведення комерційного і технологічного аудиту дослідження за темою «Модифікований алгоритм розпізнавання двовимірних штрих-кодів» є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності. Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями (таблиці 4.1) [29].

Таблиця 4.1 – Критерії оцінювання рівня комерційного потенціалу розробки та їх бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів

Продовження таблиці 4.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти на навчання	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій > 10-ти рр.	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій > 5-ти рр.	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти рр.	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років

Продовження таблиці 4.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Для проведення технологічного аудиту залучено 3 незалежних експертів:

– кандидат технічних наук, доцент Папінов В.М. Займається автоматизованим проектуванням комп'ютерних систем управління;

– кандидат технічних наук, доцент Кабачій В.В. Викладач дисциплін, які пов'язані з математичними методами оптимізації, моделюванням складних систем та мереж;

– кандидат технічних наук, доцент Коцюбинський В.Ю. Автор наукових робіт з комп'ютерної графіки та розпізнавання зображень.

За результатами оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки, наведених у таблиці 4.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в таблиці 4.3 [30]. Згідно проведених досліджень рівень комерційного потенціалу розробки становить 39,7 балів, що, відповідно до таблиці 4.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

Таблиця 4.2 – Результати оцінювання комерційного потенціалу розробки

Критерії	Експерт		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	4	5	4
2. Ринкові переваги (наявність аналогів)	1	2	1
3. Ринкові переваги (ціна продукту)	2	2	2
4. Ринкові переваги (технічні властивості)	2	2	3
5. Ринкові переваги (експлуатаційні витрати)	2	2	2
6. Ринкові перспективи (розмір ринку)	3	2	3
7. Ринкові перспективи (конкуренція)	3	2	3
8. Практична здійсненність (наявність фахівців)	5	5	5
9. Практична здійсненність (наявність фінансів)	3	4	3
10. Практична здійсненність (необхідність нових матеріалів)	5	5	5
11. Практична здійсненність (термін реалізації)	4	4	4
12. Практична здійсненність (розробка документів)	4	4	4
Сума балів	38	39	39
Середньоарифметична сума балів $СБ_с$	38,7		

Таблиця 4.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів, розрахована на основі висновків експертів	Рівень наукового та комерційного потенціалу розробки
0 – 10	Низький
11 – 20	Нижче середнього
21 – 30	Середній
31 – 40	Вище середнього
41 – 48	Високий

4.2 Розрахунок витрат на удосконалення алгоритму розпізнавання образів

При виконанні магістерської кваліфікаційної роботи було зроблено такі витрати [30]:

1. Витрати на основну заробітну плату дослідників розраховуємо у відповідності до посадових окладів працівників, за формулою:

$$З_о = \sum_{i=1}^{k} \frac{M_{mi} \cdot t_i}{T_p} = 14110{,}00 \cdot 24 / 24 = 14110{,}00 \; грн. , \qquad (4.1)$$

де k – кількість посад дослідників залучених до процесу досліджень; M_{mi} – місячний посадовий оклад конкретного дослідника, грн; t_i – кількість днів роботи конкретного дослідника, дні; T_p – середня кількість робочих днів в місяці, T_p = 24 дні.

Проведені розрахунки зведемо до таблиці 4.4.

Таблиця 4.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн.	Кількість днів роботи	Витрати на заробітну плату, грн.
Науковий керівник	14110,00	587,92	24	14110,00
Інженер-програміст 1-ї категорії	14100,00	587,50	24	14100,00
Консультант	14100,00	587,50	7	4112,50
Технік	7050,00	293,75	15	4406,25
Всього				36728,75

2. Витрати на основну заробітну плату робітників за відповідними найменуваннями робіт НДР розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i, \qquad (4.2)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн./год.; t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}}, \qquad (4.3)$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо $M_м$ = 6700,00 грн.; K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду; K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати; T_p – середнє число робочих днів в місяці, приблизно T_p = 24 дні; $t_{зм}$ – тривалість зміни, год.

Результати запишемо в таблицю 4.5. Тоді маємо:

$$C_1 = 6700{,}00 \cdot 1{,}10 \cdot 1{,}65 / (24 \cdot 8) = 63{,}34 \text{ грн.}$$

$$З_{р1} = 63{,}34 \cdot 6{,}00 = 380{,}02 \text{ грн.}$$

3. Додаткову заробітну плату розраховуємо як 10…12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%} = (36728{,}75 + 2406{,}77) \cdot 11 / 100\% = 4304{,}91 \text{ грн.}, \quad (4.4)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати; приймемо 11%.

Таблиця 4.5 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год.	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн.	Величина оплати на робітника грн.
Підготовка робочого місця розробника програмного забезпечення	6,00	2	1,10	63,34	380,02
Інсталяція програмного забезпечення моделювання	5,00	3	1,35	77,73	388,65
Введення програмних блоків розпізнавання	6,50	4	1,50	86,37	561,39
Налагодження програмних блоків розпізнавання	4,00	5	1,70	97,88	391,53
Тестування програмного забезпечення	7,00	5	1,70	97,88	685,18
Всього					2406,77

4. Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{доб}) \cdot \frac{Н_{зп}}{100\%} = (36728{,}75 + 2406{,}77 + 4304{,}91) \cdot \frac{22}{100} = 9556{,}89 \text{ грн.}, \quad (4.5)$$

де $Н_{зп}$ – норма нарахування на заробітну плату; приймаємо 22%.

5. Витрати на матеріали розраховуються окремо по кожному виду матеріалів за формулою:

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{вj} = 3{,}0 \cdot 270{,}00 \cdot 1{,}11 - 0 \cdot 0 = 899{,}10 \text{ грн.}, \quad (4.6)$$

де $Н_j$ – норма витрат матеріалу j-го найменування, кг; n – кількість видів матеріалів; $Ц_j$ – вартість матеріалу j-го найменування, грн./кг; $К_j$ – коефіцієнт транспортних витрат, ($К_j$ = 1,1 … 1,15); $В_j$ – маса відходів j-го найменування, кг; $Ц_{вj}$ – вартість відходів j-го найменування, грн./кг.

Проведені розрахунки зведемо до таблиці 4.6.

6. Витрати на комплектуючі, які використовують при проведенні НДР відсутні.

7. Витрати на спецустаткування, яке додатково закуповується при проведенні НДР відсутні.

8. Балансову вартість програмного забезпечення розраховуємо за формулою:

$$В_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot С_{прг.i} \cdot К_i = 9860{,}00 \cdot 1 \cdot 1{,}11 = 10944{,}60 \text{ грн.}, \quad (4.7)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн.; $С_{прг.i}$ – кількість одиниць програмного забезпечення відповідного

найменування, які придбані для проведення досліджень, шт.; K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, K_i = 1,10…1,12; k – кількість найменувань програмних засобів.

Таблиця 4.6 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн.	Норма витрат, кг	Величина відходів, кг	Ціна відходів, грн./кг	Вартість витраченого матеріалу, грн.
Папір Офісний А4 500	270,00	3,0	0	0	899,10
Папір для записів А4 250 (75%)	144,00	5,0	0	0	799,20
Органайзер офісний	172,00	2,0	0	0	381,84
Канцелярське приладдя	199,00	3,0	0	0	662,67
Картридж для принтера	1025,00	1,0	0	0	1137,75
Диск оптичний CD-R	22,85	3,0	0	0	76,09
Flesh-пам'ять 32 GB	400,00	1,0	0	0	444,00
Всього					4400,65

Отримані результати зведемо до таблиці 4.7.

Таблиця 4.7 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн.	Вартість, грн.
Математичне середовище MatLab	1	9860,00	10944,60
Прикладне ПЗ Mathematica	1	7580,00	8413,80
Всього			19358,40

9. У спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12} = (27560{,}00 \cdot 1) / (2 \cdot 12) = 1148{,}33 \; грн., \qquad (4.8)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн.; $t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців; $T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

Проведені розрахунки зведемо до таблиці 4.8.

10. Витрати на силову електроенергію розраховуємо за формулою:

$$B_е = \sum_{i=1}^{n} \frac{W_{уі} \cdot t_і \cdot Ц_е \cdot K_{вп і}}{\eta_і} = 0{,}42 \cdot 190{,}0 \cdot 6{,}20 \cdot \frac{0.95}{0.97} = 494{,}76 \; грн., \quad (4.9)$$

де $W_{уі}$ – встановлена потужність обладнання на визначеному етапі розробки, кВт; $t_і$ – тривалість роботи обладнання на етапі дослідження, год.; $Ц_е$ – вартість 1 кВт-години електроенергії, грн.; (вартість

електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_е$ = 6,20 грн.; $К_{вні}$ – коефіцієнт, що враховує використання потужності, $К_{вні} < 1$; η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

Проведені розрахунки зведемо до таблиці 4.9.

Таблиця 4.8 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн.
Персональний комп'ютер розробника ПЗ	27560,00	2	1	1148,33
Робоче місце інженера-програміста	7950,00	5	1	132,50
Пристрої передачі даних	6600,00	4	1	137,50
Оргтехніка	9320,00	5	1	155,33
Приміщення лабораторії розробки	310000,00	25	1	1033,33
ОС Windows	8320,00	3	1	231,11
Microsoft Office	7460,00	3	1	207,22
Середовище програмування	6900,00	3	1	191,67
Комп'ютерне обладнання	41458,00	2	1	1727,42
Всього				4964,42

Таблиця 4.9 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Персональний комп'ютер розробника ПЗ	0,42	190,0	494,76
Робоче місце інженера-програміста	0,36	190,0	424,08
Пристрої передачі даних	0,12	100,0	74,40
Оргтехніка	0,10	10,0	6,20
Комп'ютерне обладнання дослідження розпізнавання штрих-кодів	0,56	150,0	520,80
Всього			1520,24

11. Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників:

$$B_{св} = (З_о + З_р) \cdot \frac{H_{св}}{100\%} = (36728{,}75 + 2406{,}77) \cdot 25 / 100\% = 9783{,}88 \ грн., \quad (4.10)$$

де $H_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $H_{св} = 25\%$.

12. Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{сп} = (З_о + З_р) \cdot \frac{H_{сп}}{100\%} = (36728{,}75 + 2406{,}77) \cdot 35 / 100\% = 13697{,}43 \ грн., \quad (4.11)$$

де $H_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $H_{сп} = 35\%$.

13. До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%} = (36728{,}75 + 2406{,}77) \cdot 70 / 100\% = 27394{,}86 \ грн., \quad (4.12)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів} = 70\%$.

14. Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%} = (36728{,}75 + 2406{,}77) \cdot 120 / 100\% = 46962{,}62 \ грн., \quad (4.13)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв} = 120\%$.

15. Витрати на проведення науково-дослідної роботи розраховуємо як суму всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{ов} + В_{оп} + І_в + В_{нзв}, \quad (4.14)$$

$$В_{заг} = 36728{,}75 + 2406{,}77 + 4304{,}91 + 9556{,}892916 + 4400{,}65 + 0{,}00 + 0{,}00 +$$
$$+ 19358{,}40 + 4964{,}42 + 1520{,}24 + 9783{,}88 + 13697{,}43 + 27394{,}86 + 46962{,}62 =$$
$$= 181079{,}81 \text{ грн.}$$

16. Загальні витрати на завершення науково-дослідної роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta} = 181079{,}81 / 0{,}9 = 201199{,}79 \text{ грн.}, \quad (4.15)$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо $\eta = 0{,}9$.

4.3 Прогнозування комерційних ефектів від комерціалізації розробки

У ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку. Результати дослідження проведені передбачають комерціалізацію впродовж 4-ох років реалізації на ринку. У цьому випадку майбутній економічний ефект буде формуватися на основі:

ΔN – збільшення кількості споживачів продукту, у періоди часу, що аналізуються, від покращення його певних характеристик (1-й рік – +50 споживачів; 2-й рік – +100 споживачів; 3-й – +100 споживачів; 4-й – +50 споживачів);

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки, приймемо 200 осіб;

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 96000,00 грн;

$\pm\Delta Ц_о$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо 10692,00 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 4-х років, впродовж яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [30]:

$$\Delta П_i = (\pm \Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \qquad (4.16)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість; у 2022 році ставка податку на додану вартість складає 20%, а коефіцієнт λ=0,8333; ρ – коефіцієнт, який враховує рентабельність інноваційного продукту; приймемо ρ = 37%; ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2022 році ϑ = 18%.

Збільшення чистого прибутку 1-го року:

$$\Delta П_1 = (10692{,}00 \cdot 200{,}00 + 106692{,}00 \cdot 50) \cdot 0{,}8333 \cdot 0{,}37 \cdot (1 - 0{,}18) =$$
$$= 1881865{,}81 \text{ грн.}$$

Збільшення чистого прибутку 2-го року:

$$\Delta П_2 = (10692{,}00 \cdot 200{,}00 + 106692{,}00 \cdot 150) \cdot 0{,}8333 \cdot 0{,}37 \cdot (1 - 0{,}18) =$$
$$= 4568605{,}09 \text{ грн.}$$

Збільшення чистого прибутку 3-го року:

$$\Delta П_3 = (10692{,}00 \cdot 200{,}00 + 106692{,}00 \cdot 250) \cdot 0{,}8333 \cdot 0{,}37 \cdot (1 - 0{,}18) =$$
$$= 7255344{,}37 \text{ грн.}$$

Збільшення чистого прибутку 4-го року:

$$\Delta П_4 = (10692{,}00 \cdot 200{,}00 + 106692{,}00 \cdot 300) \cdot 0{,}8333 \cdot 0{,}37 \cdot (1 - 0{,}18) =$$
$$= 8598714{,}01 \text{ грн.}$$

Приведена вартість збільшення всіх чистих прибутків, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{t=1}^{T} \frac{\Delta П_t}{(1+\tau)^t}, \quad (4.17)$$

де $\Delta П_t$ – збільшення чистого прибутку у кожному з років, впродовж яких виявляються результати впровадження науково-технічної розробки, грн.; T – період часу, впродовж якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки; τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ=0,12; t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

Тоді маємо:

$$ПП = 1881865,81 / (1 + 0,12)^1 + 4568605,09 / (1 + 0,12)^2 +$$
$$+ 7255344,37 / (1 + 0,12)^3 + + 8598714,01 / (1 + 0,12)^4 =$$
$$= 1680237,33 + 3642064,01 + 5164210,81 + 5464638,21 = 15951150,35 \text{ грн.}$$

Величина початкових інвестицій, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \quad (4.18)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=3,5; $ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 201199,79 грн.

Тоді:

$$PV = k_{інв} \cdot ЗВ = 3,5 \cdot 201199,79 = 704199,27 \text{ грн.}$$

Абсолютний економічний ефект для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV = 15951150,35 - 704199,27 = 15246951,09 \text{ грн.} \quad (4.19)$$

Внутрішня економічна дохідність інвестицій, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки ($T_{ж}$ = 4 роки):

$$E_в = \sqrt[T_{ж}]{1 + \frac{E_{абс}}{PV}} - 1 = (1 + 15246951,09 / 704199,27)^{1/4} = 1,18. \quad (4.20)$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій:

$$\tau_{мін} = d + f, \quad (4.21)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2022 році в Україні d = 0,11; f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,35.

Тоді:

$$\tau_{мін} = 0,11 + 0,35 = 0,46 < E_в = 1,18,$$

тобто інвестувати в науково-дослідну роботу доцільно.

Період окупності інвестицій, які можуть бути вкладені у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в} = 1 / 1{,}18 = 0{,}85 \text{ років.} \quad (4.22)$$

Оскільки $T_{ок}$ < 3-ох років, що свідчить про комерційну привабливість науково-технічної розробки, то це може спонукати потенційного інвестора профінансувати впровадження та виведення її на ринок.

4.4 Висновки

Аналіз комерційного потенціалу розробки показав, що вона за своїми характеристиками випереджає аналоги, що підтверджує її перспективність. Результати економічної частини магістерської кваліфікаційної роботи зведено у таблицю 4.10. Отже, основні техніко-економічні показники удосконаленого алгоритму, визначені у технічному завданні, виконані.

Таблиця 4.10 – Результати економічної частини

Показники	Задані у технічному завданні	Досягнуті у магістерській кваліфікаційній роботі	Висновок
1. Витрати на розробку удосконаленого алгоритму	не більше 250 000 грн.	201 199,79 грн.	Виконано
2. Абсолютний ефект від впровадження	не менше 15 000 грн. за копію	19 058,69 грн. за копію	Виконано
3. Внутрішня дохідність інвестицій	не менше 40 %	46 %	Виконано
4. Термін окупності потенційних інвестицій	до 3-ох років	0,85 років	Виконано

ВИСНОВКИ

Проведений аналіз алгоритмів розпізнавання образів дозволив визначити, що перспективним є використання штучних нейромереж, процес навчання яких розглядається як визначення архітектури мережі та налаштування ваг синаптичних зв'язків між нейронами.

У роботі описано модифіковану математичну модель для розв'язання поставленої задачі на основі додаткової обробки даних та мережі Ліппмана, яка є класифікатором чорно-білих зображень. Вона потребує менше пам'яті та не вимагає трудомістких обчислювальних процедур для навчання, що є істотною перевагою. За її допомогою можна організувати систему автоматичної корекції помилок. Для проведення досліджень було розроблено та налагоджено програмне забезпечення, в якому виконано експерименти по розпізнаванню двовимірних штрих-кодів. Визначено умови коректної роботи нейромережі, а також випадки, коли можливі помилки розпізнавання і нестабільності її виходів. Розроблений алгоритм можна використовувати як додаткову процедуру виправлення двовимірних штрих-кодів у системах обміну даними.

Також у роботі проведено технологічний аудит запропонованого алгоритму. Аналіз комерційного потенціалу програмного забезпечення показав, що його розробка є перспективною.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Нікольський Ю.В., Пасічник В.В., Щербина Ю.М. Системи штучного інтелекту. Л.: Магнолія, 2013. 279 с.

2. Moon T.K. Error Correction Coding: Mathematical Methods & Algorithms. John Wiley & Sons, 2005. 800 p.

3. ISO/IEC 18004:2015. Estonian Centre for Standardization. URL: https://www.evs.ee/products/iso-iec-18004-2015 (дата звернення 10.09.2022).

4. Шапиро Л., Стокман Дж. Компьютерное зрение. Лаборатория знаний, 2006. 752 с.

5. Gonzalez R.C., Woods R.E. Digital Image Processing. Prentice Hall, 2008. P. 206-253.

6. Дмитрієнко В.Д., Заковоротний О.Ю. Основи нейрокомп'ютингу. Х.: НТМТ, 2012. 128 с.

7. Dmitrienko V., Zakovorotnyi A., Leonov S. and Khavina I. Neural Networks Art: Solving Problems with Multiple Solutions and New Teaching Algorithm. *The open neurology journal*. Vol. 8. 2014. P. 15-21.

8. Dmitrienko V.D., Zakovorotniy A.Yu., Leonov S.Yu. Neural Networks for Determining Affinity Functions. *International Congress on Human-Computer Interaction, Optimization and Robotic Applications*. 2020. P. 1-5.

9. Dmitrienko V.D., Zakovorotniy A.Yu., Leonov S.Yu. The Neural Network Art which uses the Hamming Distance to Measure an Image Similarity Score. *Journal of Engineering and Applied Sciences*. Vol. 14 (21). 2019. P. 8048-8054.

10. Sazonova T.V., Khristodulo O.I., Makhmutov A.A., Use Algorithm Based at Hamming Neural Network Method for Natural Objects Classification. *International Symposium "Intelligent Systems"*. 2017. P. 388-395.

11. Katebi S.D. Recognition of Cursive Texts Using Hamming Neural Nets. *Engineering Journal of Qatar University*. Vol. 7. 1994. P. 37-51.

12. Gupta A., Singh Y. Analysis of Hamming Network and MAXNET of Neural Network Method in the String Recognition. *International Conference on Communication Systems and Network Technologies*. 2011. P. 38-42.

13. Є.О. Звуздецький, А.Р. Дзюба, С.Г. Кривогубченко та Ю.Ю. Іванов. Огляд алгоритмів навчання нейронних мереж. *Науково-технічна конференція підрозділів ВНТУ*: матер. L науково-технічної конференції. Вінниця: ВНТУ, 2021. С. 1126-1127.

14. Звуздецький Є.О., Трофимчук А.О., Іванов Ю.Ю. та інші. Модифікований алгоритм оптимізації функції втрат нейромережі. *Актуальні задачі медичної, біологічної фізики та інформатики*: матер. науково-практичної конференції з міжнародною участю. Вінниця: ВНМУ ім. М.І. Пирогова, 2022. С. 73-74.

15. Fausett L. Fundamentals of Neural Networks: Architectures, Algorithms, and Applications. USA: Prentice-Hall Inc, 1994. 476 p.

16. Hagan M.T., Demuth H.B., Beale M.H., De Jesus O. Neural Network Design. 1012 p. URL: http://hagan.okstate.edu/ nnd.html (дата звернення 20.09.2022).

17. Heaton J. Introduction to Neural Networks. St. Louis: Heaton Research Inc., 2008. 241 p.

18. Субботін С.О. Нейронні мережі: теорія та практика. Житомир: Вид. О.О. Євенок, 2020. 184 с.

19. Новотарський М.А., Нестеренко Б.Б. Штучні нейронні мережі: обчислення. К., 2004. 408 с.

20. Lippmann R. An Introduction to Computing with Neural Nets. *IEEE ASSP Magazine*. V. 4 (2). 1987. P. 4-22.

21. Lippmann R.P., Gold B., Malpass M.L. A Comparison of Hamming and Hopfield Neural Nets for Pattern Classification. Fort Belvoir: Defense Technical Information Center, 1987. P. 1-35.

22. QR Code Tutorial from Thonky. URL: https://www.thonky.com/qr-code-tutorial/ (дата звернення 12.09.2022).

23. Barland G. Error Correction and QR Codes. URL: https://www.stthomas.edu/media/collegeofartsandsciences/mathematics/pdf/2017AndersonGroupBarlandCAMReport.pdf (дата звернення 12.09.2022).

24. Suran K. QR Code Image Correction based on Corner Detection and Convex Hull Algorithm. *Journal of Multimedia*. Vol. 8 (6). 2013. P. 662-668.

25. Szentandrasi I., Herout A., Dubska M. Fast Detection and Recognition of QR codes in High-Resolution Images. *Published in SCCG*. 2012. P. 129-136.

26. Надригайло Т.Ж., Молчанова К.А. Аналіз нейронних алгоритмів. Математичне моделювання. № 2 (25). Дніпро, 2011. С. 46-51.

27. Koutroumbas K., Kalouptsidis N. Generalized Hamming Networks and Applications. V. 18. 2005. P. 896-913.

28. Мізюк О. Путівник мовою програмування Python. URL: https://pythonguide.rozh2sch.org.ua (дата звернення 02.10.2022).

29. Кавецький В.В., Козловський В.О., Причепа І.В. Економічне обґрунтування інноваційних рішень. Вінниця: ВНТУ, 2016. 113 с.

30. Козловський В.О., Лесько О.Й., Кавецький В.В. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт. Вінниця: ВНТУ, 2021. 42 с.

ДОДАТКИ

Додаток А

(обов'язковий)

Технічне завдання

ЗАТВЕРДЖУЮ

в. о. завідувача кафедри АІІТ

д.т.н., проф. Бісікало О. В.

«___»_____ 2022 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на магістерську кваліфікаційну роботу

«Модифікований алгоритм розпізнавання двовимірних штрих-кодів»

08-31.МКР.003.02.000 ТЗ

Керівник роботи:

к.т.н., доц. каф. АІІТ

Іванов Ю. Ю.

«__» _____ 20__ р.

Виконавець:

ст. гр. 1ІСТ-21м

Звуздецький Є. О.

«__» _____ 20__ р.

Вінниця ВНТУ – 2022

1. Назва та галузь застосування

Розвиток інформаційних технологій призвів до створення систем обміну даними, які застосовують комбіноване стиснення, захист від пошкоджень та зберігання інформації. У ході роботи можуть виникати труднощі, коли на матриці коду відсутні ключові елементи-детектори або їхня структура значно пошкоджена. У цьому випадку декодуюче програмне забезпечення не зможе навіть зчитати код. У роботі пропонується підхід до розв'язання задачі розпізнавання двовимірних штрих-кодів на основі нейромережі з базою даних та процедурою корекції, що спрошує обчислення.

2. Підстави для розробки

Розробку системи здійснювати на підставі наказу по університету №203 від 14.09.2022 та завдання до магістерської кваліфікаційної роботи, складеного та затвердженого кафедрою «Автоматизації та інтелектуальних інформаційних технологій».

3. Мета та призначення розробки

Метою роботи є підвищення ефективності розв'язання задачі розпізнавання пошкоджених двовимірних штрих-кодів за рахунок використання запропонованої моделі корекції.

4. Джерела розробки

1. Katebi S.D. Recognition of Cursive Texts Using Hamming Neural Nets. *Engineering Journal of Qatar University*. Vol. 7. 1994. P. 37-51.

2. Новотарський М.А., Нестеренко Б.Б. Штучні нейронні мережі: обчислення. К., 2004. 408 с.

3. Gupta A., Singh Y. Analysis of Hamming Network and MAXNET of Neural Network Method in the String Recognition. *International Conference on Communication Systems and Network Technologies*. 2011. P. 38-42.

4. Dmitrienko V., Zakovorotnyi A., Leonov S. and Khavina I. Neural Networks Art: Solving Problems with Multiple Solutions and New Teaching Algorithm. *The open neurology journal*. Vol. 8. 2014. P. 15-21.

3. Показники призначення

Розроблено програмне забезпечення дозволяє виконати розпізнавання та відновлення двовимірних штрих-кодів.

Основні технічні вимоги та мінімальні системні вимоги до програми: операційна система *Windows* 7 або новіша; наявність клавіатури та миші; процесор *Core* i5 (*Ryzen* 5) або кращий; оперативна пам'ять – 8 ГБ; відеопам'ять – 512 МБ; жорсткий диск – 500 ГБ і більше.

Вхідні дані: база даних; матриця пікселів $H \times B$; нейромережа Ліппмана; константа масштабування k_1; порогові параметри L, K та Q; кількість ітерацій роботи мережі *MaxNet*.

Результати роботи програми: відновлений двовимірний штрих-код із бази даних або висновок про те, що зразок не можна розпізнати точно.

5. Економічні показники

До економічних показників входять:

- витрати на розробку – не менше 250 тис. грн.;
- вартість копії продукту – не менше 15 тис. грн.;
- мінімальна дохідність – не менше 40 %;
- термін окупності – до 3-ох років;
- інші економічні переваги у порівнянні з аналогами.

6. Стадії розробки

1. Розділ 1 «Аналіз предметної області» має бути виконаний до 10.10.22.

2. Розділ 2 «Розробка модифікованого алгоритму розпізнавання двовимірних штрих-кодів» має бути виконаний до 25.10.22.

3. Розділ 3 «Розробка програмного забезпечення та експериментальні дослідження» має бути виконаний до 10.11.22.
4. Економічний розділ має бути виконаний до 20.11.22.

7. Порядок контролю та приймання

1. Рубіжний контроль провести до 07.12.22.
2. Попередній захист магістерської кваліфікаційної роботи провести до 07.12.22.
3. Захист магістерської кваліфікаційної роботи провести до 23.12.22.

Додаток Б

(обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

МОДИФІКОВАНИЙ АЛГОРИТМ РОЗПІЗНАВАННЯ ДВОВИМІРНИХ ШТРИХ-КОДІВ

В.о. зав. кафедри АІІТ _____ д-р техн. наук, професор каф. АІІТ Бісікало О. В.

(підпис) (науковий ступінь, вчене звання, ініціали та прізвище)

Керівник роботи _____ канд. техн. наук, доцент каф. АІІТ Іванов Ю. Ю.

(підпис) (науковий ступінь, вчене звання, ініціали та прізвище)

Тех. контроль _____ канд. техн. наук, доцент каф. АІІТ Іванов Ю. Ю.

(підпис) (науковий ступінь, вчене звання, ініціали та прізвище)

Нормоконтроль _____ канд. техн. наук, доцент каф. АІІТ Іванов Ю. Ю.

(підпис) (науковий ступінь, вчене звання, ініціали та прізвище)

Опонент _____ _____

_____

(підпис) (науковий ступінь, вчене звання, ініціали та прізвище)

Студент гр. 1ІСТ-21м _____ Звуздецький Є. О.

(підпис) (ініціали та прізвище)

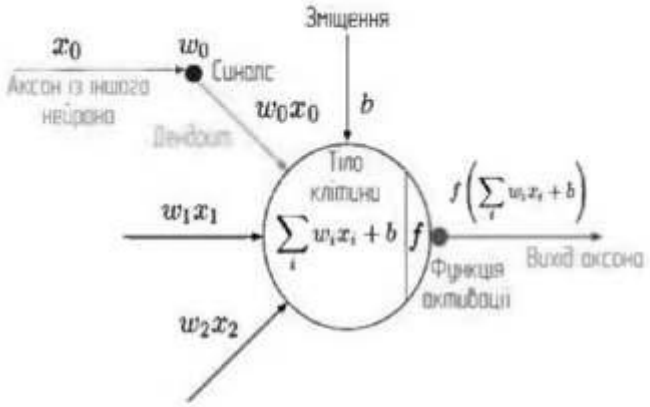


Рисунок Б.1 – Модель штучного нейрону

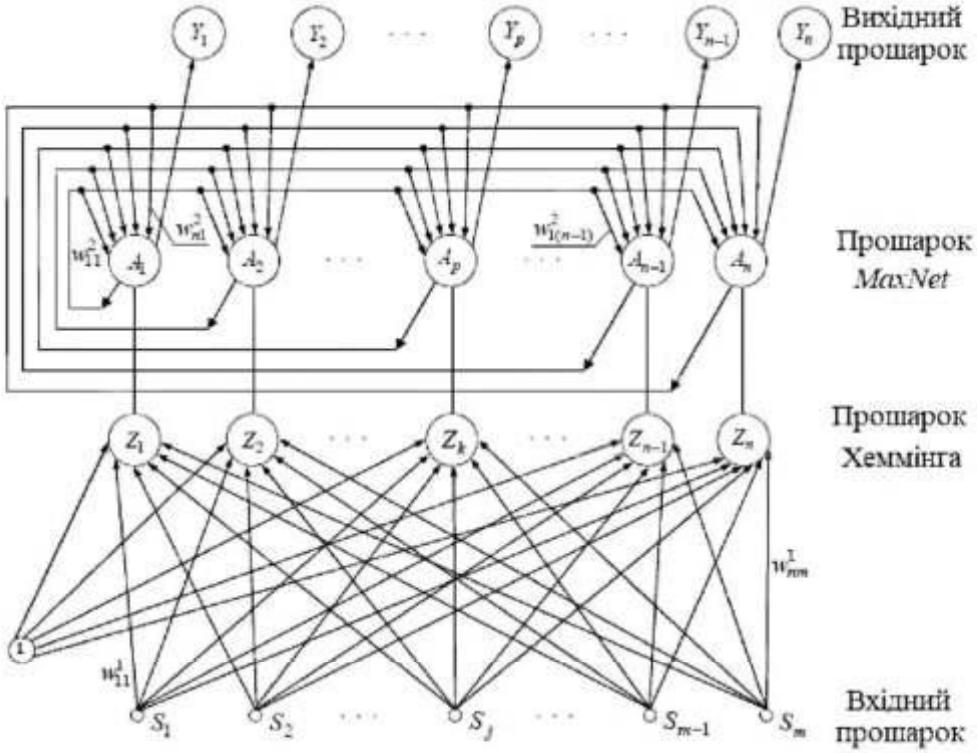


Рисунок Б.2 – Архітектура нейромережі Ліппмана

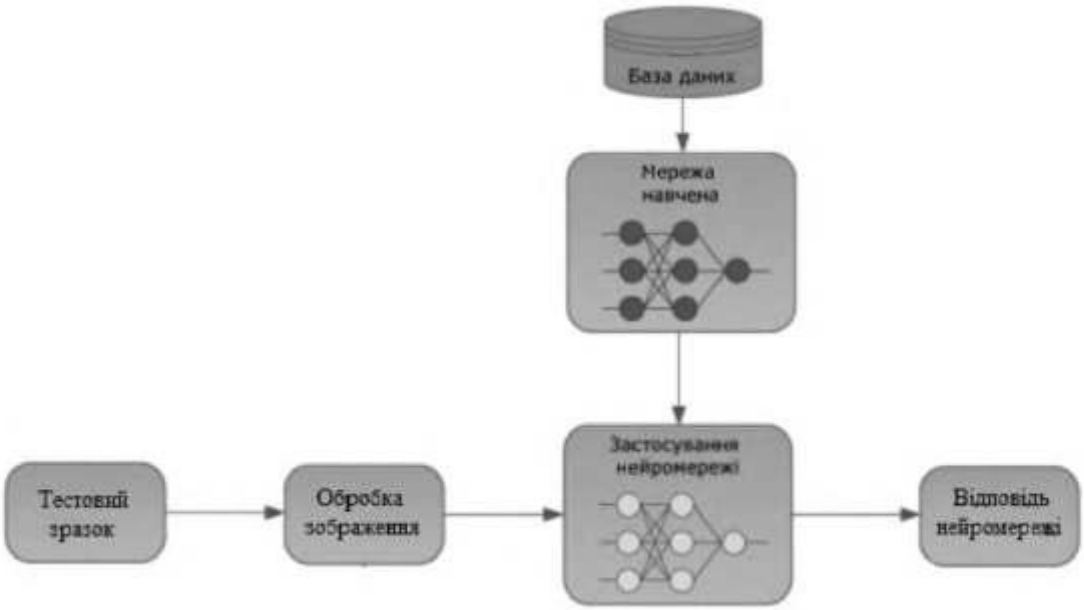


Рисунок Б.3 – Структура для навчання мережі

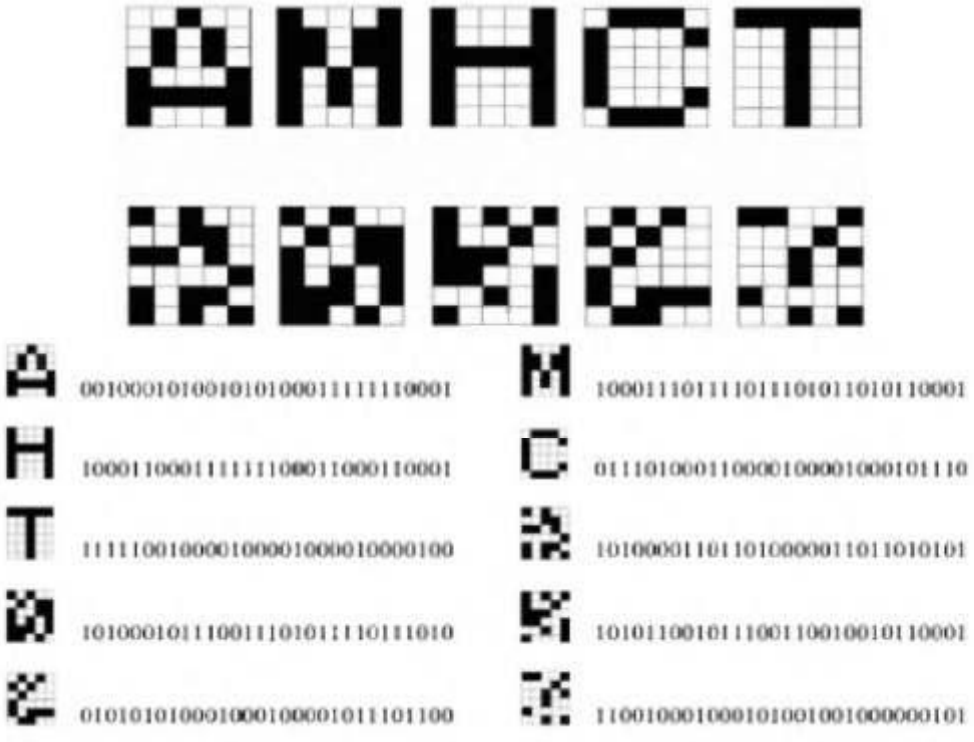


Рисунок Б.4 – Приклади кодування образів

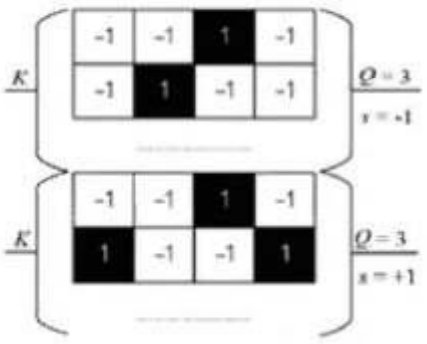


Рисунок Б.5 – Обробка зображення

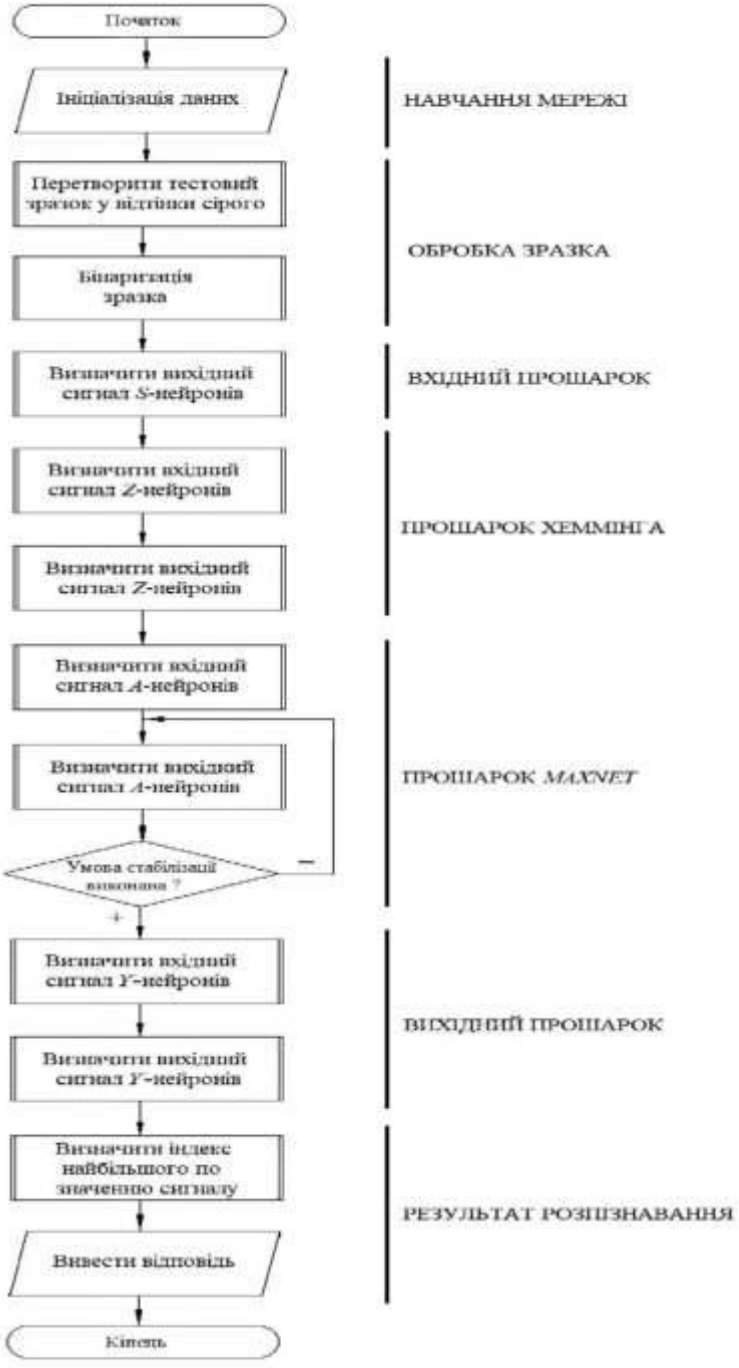


Рисунок Б.6 – Схема програми

Додаток В

(обов'язковий)

Лістинг програми

```
#########################################################################
import random
import imageio
#########################################################################

#########################################################################
class HammingNeuron:
    def __init__(self, weights, next_neuron=None):
        self.weights = list()
        self.inputs = list()
        self.next_neuron = None
        for w in weights:
            self.weights.append(w)
            self.inputs.append(0)
            self.next_neuron = next_neuron
    def change_weight(self, ind_of_weight, new_value):
        self.weights[ind_of_weight] = new_value
    def set_input(self, ind_of_input, value):
        self.inputs[ind_of_input] = value
    def set_next_neuron(self, next_neuron):
        self.next_neuron = next_neuron
    def count_output(self):
        res = 1/2 + sum(self.inputs[i] * self.weights[i] for i in range(0, len(self.weights)))/(2 *
len(self.weights))
        return res
    def get_output(self):
        self.next_neuron.set_value(self.count_output())
#########################################################################

#########################################################################
class MaxNetNeuron:
    def __init__(self, index, weights, next_neuron):
        self.value = 0
        self.reinitial_value = 0
        self.inputs = list()
        self.weights = list()
        self.layer_neurons = list()
        self.index = index
        self.next_neuron = next_neuron
        for w in weights:
            self.weights.append(w)
            self.inputs.append(None)
    def set_layer_neurons(self, layer_neurons):
        for n in layer_neurons:
            self.layer_neurons.append(n)
```

```
    def set_value(self, value):
        self.value = value
        self.reinitial_value = value
    def set_only_current_value(self, value):
        self.value = value
    def set_input(self, ind_of_neuron, value):
        self.inputs[ind_of_neuron] = value
    def count_output(self):
        # if first time
        if self.inputs[self.index] is None:
            return self.value
        # if not first time
        else:
            return self.inputs[self.index] - \
                sum(self.inputs[i] * self.weights[i] for i in range(0, len(self.weights)) if i !=
self.index)
    def recount_value(self):
        self.value = self.count_output()
    def get_output_inside_layer(self):
        for n in self.layer_neurons:
            n.set_input(self.index, self.value)
    def get_output(self):
        self.next_neuron.set_value(self.value)
    def reinitialize_neuron(self):
        self.value = self.reinitial_value
        for i in range(0, len(self.inputs)):
            self.inputs[i] = None
################################################################################

################################################################################
class ThresholdNeuron:
    def __init__(self, index, next_neurons):
        self.value = None
        self.next_neurons = list()
        self.index = index
        for n in next_neurons:
            self.next_neurons.append(n)
    def set_value(self, value):
        self.value = value
    def count_output(self):
        if self.value > 0:
            return 1
        else:
            return 0
    def get_output(self):
        for n in self.next_neurons:
            n.set_input(self.index, self.count_output())
################################################################################
```

```
#############################################################################
class HammingLayer:
    def __init__(self, weights, next_neurons):
        self.neurons = list()
        for i in range(0, len(weights)):
            new_neuron = HammingNeuron(weights[i], next_neurons[i])
            self.neurons.append(new_neuron)
    def run(self, inputs):
        for n in self.neurons:
            for i in range(0, len(inputs)):
                n.set_input(i, inputs[i])
        for n in self.neurons:
            n.get_output()
#############################################################################

#############################################################################
class MaxNetLayer:
    def __init__(self, next_neurons):
        self.neurons = list()
        k = len(next_neurons)
        for i in range(0, k):
            weights = [random.random() * 1/(k - 1) for j in range(0, i)] + [1] +\
                    [random.random() * 1/(k  - 1) for j in range(i + 1, k)]
            new_neuron = MaxNetNeuron(i, weights, next_neurons[i])
            self.neurons.append(new_neuron)
        for n in self.neurons:
            n.set_layer_neurons(self.neurons)
    def run(self):
        for n in self.neurons:
            n.get_output_inside_layer()
        for n in self.neurons:
            n.recount_value()
        for n in self.neurons:
            n.get_output()
    def reinitialize_layer(self, num_of_not_null_neuron, eps):
        self.neurons[num_of_not_null_neuron].reinitial_value -= eps
        for n in self.neurons:
            n.reinitialize_neuron()
#############################################################################

#############################################################################
class ThresholdLayer:
    def __init__(self, count, next_neurons):
        self.neurons = list()
        for i in range(0, count):
            new_neuron = ThresholdNeuron(i, next_neurons)
            self.neurons.append(new_neuron)
    def run(self):
        for n in self.neurons:
            n.get_output()
    def get_first_not_null_element(self):
        for n in self.neurons:
```

```
            if n.count_output() == 1:
                return self.neurons.index(n)
########################################################################################

########################################################################################
class HammingNetwork:
    def __init__(self, learning_examples, eps, max_count_of_outputs):
        self.max_count_of_outputs = max_count_of_outputs
        self.eps = eps
        self.output_layer = OutputLayer(learning_examples)
        self.threshold_layer = ThresholdLayer(len(learning_examples),
self.output_layer.neurons)
        self.max_net_layer = MaxNetLayer(self.threshold_layer.neurons)
        self.hamming_layer = HammingLayer(learning_examples, self.max_net_layer.neurons)
    def classification(self, example_inputs):
        res = []
        self.hamming_layer.run(example_inputs)
        first_time = True
        while(True):
            while(True):
                self.max_net_layer.run()
                if sum(n.count_output() for n in self.threshold_layer.neurons) == 1:
                    break
                for n in self.max_net_layer.neurons:
                    if n.next_neuron.count_output() == 0:
                        n.set_only_current_value(0)
            self.threshold_layer.run()
            res.append(self.output_layer.get_result())
            if first_time:
                first_time = False

self.max_net_layer.reinitialize_layer(self.threshold_layer.get_first_not_null_element(),
self.eps)
                continue
            else:
                if res[len(res) - 1] in res[0:len(res) - 1] or len(res) > self.max_count_of_outputs:
                    res = res[0:len(res) - 1]
                    return res
                else:

self.max_net_layer.reinitialize_layer(self.threshold_layer.get_first_not_null_element(),
self.eps)
                    continue
########################################################################################
```

Додаток Г

(обов'язковий)

Довідка про можливість впровадження розробки

Додаток Д

(обов'язковий)

Протокол перевірки МКР

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Модифікований алгоритм розпізнавання двовимірних штрих-кодів.

Тип роботи: магістерська кваліфікаційна робота

Підрозділ: кафедра автоматизації та інтелектуальних інформаційних технологій, факультет інтелектуальних інформаційних технологій та автоматизації

Показники звіту подібності Plagiat.pl (StrikePlagiarism)

Оригінальність 84,3% Схожість 15,7%

Аналіз звіту подібності (відмітити потрібне):

■ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату

☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на розгляд експертної комісії кафедри.

☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень

Особа, відповідальна за перевірку _____ Маслій Р. В.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Plagiat.pl (StrikePlagiarism) щодо роботи.

Автор роботи _____ Звуздецький Є. О.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Іванов Ю. Ю.
(підпис) (прізвище, ініціали)