

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ НЕЙРОМЕРЕЖНИХ
МЕТОДІВ РОЗПІЗНАННЯ РУКОПИСНИХ СИМВОЛІВ
RESEARCH AND SOFTWARE IMPLEMENTATION OF NEURAL
NETWORK METHODS FOR HANDWRITTEN CHARACTER RECOGNITION

Виконав: студент(ка) 2 курсу, групи КНТ-129м
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Калоян К.А.

(прізвище та ініціали)

Керівник Гофман Є.О.

(прізвище та ініціали)

Рецензент Скрупський С.Ю.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 20 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Калояна Карена Арменовича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація нейромережних методів розпізнання рукописних символів. Research and Software Implementation of Neural Network Methods for Handwritten Character Recognition

керівник проєкту (роботи) Гофман Євген Олександрович, к.т.н.,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “03” листопада 2020 року № 301

2. Строк подання студентом проєкту (роботи) 14 грудня 2020 року

3. Вихідні дані до проєкту (роботи) технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Вибір та обґрунтування структури системи, яка проєктується 3. Основні рішення щодо реалізації компонентів системи. 4. Керівництво програміста. 5. Керівництво оператора. 6. Економіко-організаційна частина. 7. Охорона праці та безпека в надзвичайних ситуаціях.

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

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5	Гофман Є.О., доцент		
6	Пожуєва Т.О., професор		
7	Коробко О.В., ст. викладач		
Нормоконтролер	Калініна М.В., асистент		

7. Дата видачі завдання “02” жовтня 2020 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз предметної області	1 тиждень	
2	Розробка та затвердження технічного завдання	2 тиждень	Технічне завдання
3	Проектування структури системи	3-4 тиждень	Структурна схема
4	Розробка схеми функціонування системи та модулів програми	5-6 тиждень	Функціональна схема. Модулі
5	Створення програмного коду системи	7–8 тижні	Текст програми
6	Тестування та відлагодження програми	9 тиждень	Працездатна система
7	Розробка розділів пояснювальної записки	10 тиждень	ПЗ
8	Розробка слайдів презентації	11-12 тиждень	Слайди презентації
9	Захист роботи	13 тиждень	

Студент(ка)

(підпис)

Калоян К.А.

(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис)

Гофман Є.О.

(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
127 с., 18 рис., 8 табл., 4 додатки, 43 джерела.

ЕВОЛЮЦІЙНИЙ МЕТОД, НЕЙРОМЕРЕЖА, ЗГОРТКОВА
НЕЙРОМЕРЕЖА, РОЗПІЗНАННЯ, РУКОПИСНИЙ СИМВОЛ,
КЛАСИФІКАЦІЯ.

Об'єкт дослідження – процес розпізнавання образів.

Предмет дослідження – нейромережні методи розпізнавання
рукописних символів.

Мета роботи – дослідження та розробка програмних засобів на основі
нейромережних технологій для розпізнавання рукописних символів.

Матеріали, методи та технічні засоби: структурне та об'єктно-
орієнтоване програмування, мова програмування C#, персональний
комп'ютер з монітором, клавіатурою та маніпулятором типу «миша», що
функціонує під управлінням операційної системи.

Результати. Створено програмні засоби для розпізнавання рукописних
символів на основі нейромережних технологій. Розроблена програмна система
може успішно експлуатуватися на персональних комп'ютерах під управлінням
операційних систем та використовуватися для розпізнавання рукописних
символів.

Висновки. Розроблено програмне забезпечення для розпізнавання
рукописних символів. Проаналізовано предметну область, розроблено
технічне завдання для реалізації програми, обрано середовище розробки,
розроблено структурну схему програми, виконано конструювання
програмного забезпечення для розпізнавання рукописних символів на основі
нейромережних технологій, здійснено тестування розробленої програми.

Галузь використання – розпізнавання образів.

ABSTRACT

Explanatory note to the final qualifying work of the magister: 127 pages, 18 figures, 8 tables, 4 appendixes, 43 sources.

EVOLUTIONARY METHOD, NEURAL NETWORK, ROLLED NEURAL NETWORK, RECOGNITION, MANUSCRIPT SYMBOL, CLASSIFICATION.

Object of study is a process of pattern recognition.

Subject of study are neural network methods of handwriting character recognition.

The purpose of the work is to study and develop a software based on neural network technologies for handwriting character recognition.

Materials, methods, and tools: structured and object-oriented programming, programming language C#, computer with a monitor, keyboard and mouse manipulator operating under the control of the operating system.

Results. The software for handwriting recognition based on neural network technologies has been developed. The developed software system can successfully operate on personal computers running operating system and can be used for handwriting recognition.

Conclusions. Software for handwriting recognition has been developed. The subject area is analyzed, the technical task for program realization is developed, the development environment is chosen, the structural scheme of the program is developed, the software writing recognition based on neural network technologies is executed, the developed software is tested.

The field of use is pattern recognition.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ.....	9
1 Аналіз предметної області.....	10
1.1 Постановка задачі розпізнавання образів на основі нейромережних моделей.....	10
1.2 Аналіз нейромережних моделей для розпізнавання образів	11
1.3 Використання згорткових нейронних мереж для розпізнавання образів.....	13
1.4 Аналіз методів навчання нейромереж для розпізнавання образів.....	14
1.5 Обґрунтування вибору генетичного алгоритму для навчання НМ нейронних мереж	18
1.6 Порівняльний аналіз програмних засобів синтезу розпізнавальних в моделей на основі нейронних мереж глибокого навчання	19
1.7 Висновки за розділом 1	21
2 Вибір та обґрунтування структури системи, яка проєктується.....	23
2.1 Дослідження та аналіз вимог до програмного продукту	23
2.2 Вибір мови програмування	24
2.3 Вибір середовища розробки.....	25
2.3 Структурна схема програми.....	27
2.4 Висновки за розділом 2	28
3 Основні рішення щодо реалізації компонентів системи	29
3.1 Реалізація згорткової нейронної мережі	29
3.2 Розробка гібридного методу синтезу нейромережних моделей на основі генетичного пошуку та зворотного поширення помилки.....	30
3.3 Тестування програмного забезпечення.....	32
3.4 Висновки за розділом 3	35
4 Керівництво програміста	36
4.1 Призначення й умови застосування програми.....	36

	7
4.2 Характеристики програми.....	36
4.3 Звернення до програми.....	37
4.4 Вхідні і вихідні дані	37
4.5 Повідомлення	37
5 Керівництво оператора	38
5.1 Призначення програми	38
5.2 Умови виконання програми	38
5.3 Виконання програми.....	39
5.4 Приклад роботи з програмою	39
5.5 Повідомлення оператору	40
6 Економіко-організаційна частина.....	41
6.1 Планування розробки програмного продукту.....	41
6.2 Визначення витрат на розробку програми	42
6.3 Розрахунок економічної ефективності програмного продукту.....	49
7 Охорона праці та безпека в надзвичайних ситуаціях	52
7.1 Аналіз потенційних небезпек.....	52
7.2 Заходи із забезпечення безпеки	53
7.3 Заходи з виробничої санітарії та гігієни праці.....	55
7.4 Заходи безпеки у надзвичайних ситуаціях	64
Висновки	69
Перелік джерел посилання	71
Додаток А Технічне завдання	76
Додаток Б Опис програми	80
Додаток В Текст програми	83
Додаток Д Слайди презентації.....	120

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

ГА – генетичний алгоритм;

ЕМ – еволюційний метод;

ЗНМ – згорткова нейронна мережа;

ЗПП – зворотне поширення помилки;

НМ – нейронна мережа;

ПЗ – програмне забезпечення;

ПК – персональний комп'ютер;

ПП – програмний продукт;

ШІ – штучний інтелект.

ВСТУП

У наш час для створення сучасних систем розпізнавання образів необхідно виконувати аналіз та оброблення наявної інформації, зокрема інформації у вигляді графічних образів [1]–[22]. Для побудови систем розпізнавання образів потрібно синтезувати розпізнавальні моделі в залежності від розв’язуваного завдання.

Метою роботи є дослідження та розробка програмних засобів на основі нейромережних технологій для розпізнавання рукописних символів. Оглянувши можливості за засоби створення розпізнавання образів, було вирішено як базис для створення відповідного програмного забезпечення використовувати згорткові нейронні мережі.

Для досягнення поставленої мети необхідно було розв’язати такі завдання:

- аналіз методів та моделей для розпізнавання образів;
- розробка технічного завдання для реалізації програми;
- вибір середовища розробки програмного забезпечення;
- розробка гібридного методу синтезу нейромережних моделей на основі генетичного пошуку та зворотного поширення помилки;
- конструювання програмного забезпечення для розпізнавання рукописних символів на основі нейромережних технологій;
- тестування розробленого програмного забезпечення.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

У наш час для розпізнавання графічних образів ефективно застосовуються нейромережні моделі, які мають механізми адаптації до природи вхідних даних та є нелінійними моделями, що дозволяє їм в певних межах моделювати складні залежності для побудови ефективних систем розпізнавання образів.

1.1 Постановка задачі розпізнавання образів на основі нейромережних моделей

Нехай задано X – множина об'єктів і їх ознак, Y – множина класів, до яких слід віднести об'єкти множини X . Існує невідома залежність $X \rightarrow Y$, реалізації якої відомі тільки на об'єктах кінцевої навчальної вибірки $X^n = \{(x_1, y_1), \dots, (x_n, y_n)\}$ [1]–[22].

Тоді завдання побудови розпізнавальної моделі в загальному вигляді може бути сформульовано таким чином: необхідно визначити функцію $F(X)$, яка буде здатна визначити приналежність довільного об'єкта $x \in X$ до відповідного йому класу $y \in Y$ з помилкою E , що не перевищує заданий рівень ξ .

Стосовно до НМ завдання побудови класифікатора можна уявити як задачу знаходження такої структури нейромоделі залежності $F(X, W)$ і значень її параметрів W , при якій критерій якості моделі (функція помилки E) не перевищить задане значення ξ .

Для випадку НМ-класифікатора, модель залежності в якості параметрів має набір значень вагових коефіцієнтів зв'язків (ваг) і порогових зсувів нейронів (порогів) W , а структурно складається з вхідного шару $V = \{v_i\}$, де v_i – вузол видимого шару, множини прихованих шарів мережі, $k = 1, \dots, n$, де n – кількість прихованих шарів мережі, і вихідного шару мережі H^n . На вхідний

шар мережі подаються значення з множини X , послідовно по шарам передаються від вхідного шару до вихідного шару і далі сигнали з вихідного шару нейронів належать до множини класів Y .

1.2 Аналіз нейромережних моделей для розпізнавання образів

Як відзначено вище, як базис для синтезу моделей розпізнавання образів, обрано нейронні мережі. У наш час для розпізнавання графічних образів відомі різні методи навчання розпізнавальних моделей [1]–[22], характеризуються чутливістю до шуму та схильністю до перенавчання, що пов'язане з недостатнім узагальненням даних.

Для вирішення таких проблем використовується технологія глибокого навчання (Deep Learning), яка є різновидом машинного навчання, здатним представляти внутрішню кореляцію, автоматично виявляти абстракції високого рівня і краще здійснювати узагальнення даних. Глибоке навчання реалізується за допомогою нейронних мереж із глибокою архітектурою, що відрізняються підходом до виявлення схованих особливостей у даних. Проаналізуємо різні типи глибоких нейронних мереж архітектури та методи їх синтезу.

Згортальна нейромережа (Convolutional neural network, CNN) чергує згортальні шари, субдискретизувальні шари і використовує на виході повнозв'язний шар нейронів. Структура мережі – прямоспрямована без зворотних зв'язків, багат шарова. Для навчання використовуються метод зворотного поширення помилки. Послідовне застосування згортки і підвибірки дозволяє складати карти ознак з карт ознак попереднього шару [5].

Автоасоціатор (Autoencoder) – мережа прямого поширення без зворотних зв'язків, що містить вхідний шар, один чи більш проміжних (схованих) шарів і вихідний шар, що містить стільки ж нейронів, скільки і вхідний шар. Автоасоціатор навчається з використанням зворотного поширення помилки. Цільовим виходом автоасоціатора є його вхід [1]–[4].

Глибока мережа довіри (Deep Belief Network, DBN) – ймовірнісна модель, що складається з декількох шарів стохастичних схованих змінних, кожен з яких захоплює кореляції високого порядку між діяльністю схованих особливостей у розташованому нижче шарі [1]–[4].

Основним будівельним блоком DBN є обмежена машина Больцмана (RBM), що містить шар видимих елементів (дані) і шар схованих елементів, що навчаються подавати особливості даних, захоплюючи зв'язок більш високого порядку. Процедура навчання DBN складається з попередньої підготовки (жадібне навчання стека RBM) і етапу тонкого настроювання на основі зворотного поширення помилки [1]–[4].

Розглянуті мережі мають наступні переваги. CNN має меншу кількість ваг, що налаштовуються, у порівнянні з повнозв'язною мережею, має стійкість до завад; універсальність у задачах розпізнавання; інваріантність до шуму. Автоасоціатор більш просто реалізується, ніж RBM, автоасоціатор вчиться «стискати» і «реконструювати» вхідні дані, тобто за необхідності може їх відновити, має широку сферу застосування. DBN здатна забезпечити більш абстрактні простори властивостей для класифікації, має стійкість до перекручувань вхідних даних.

Розглянуті мережі характеризуються такими недоліками. Архітектура CNN придатна тільки для розпізнавання зображень; містить занадто багато віріативних параметрів; багато параметрів істотно впливають на результат, але обираються емпірично. Автоасоціатор має проблему перенавчання. RBM має низьку ефективність через дуже велику ємність.

Для більш ефективної роботи CNN доцільно використовувати генетичний алгоритм для навчання ваг у фільтрах мережі, а в якості початкових значень вагових коефіцієнтів використовувати значення, отримання за допомогою методу зворотного поширення помилки, що дозволить уникнути зациклення в локальних мінімумах і підвищити середню ймовірність успішного розв'язання задачі розпізнавання образів.

1.3 Використання згорткових нейронних мереж для розпізнавання образів

Згорткова нейронна мережа (ЗНМ) – це такий тип штучної нейронної мережі прямого поширення, окремі нейрони якої являють собою ядра згортки. Згорткові мережі є варіаціями багатошарових персептронів. Вони мають широке застосування в розпізнаванні зображень та відео, рекомендаційних системах та обробці природної мови. Приклад архітектури ЗНМ зображено на рис. 1.1 [5].

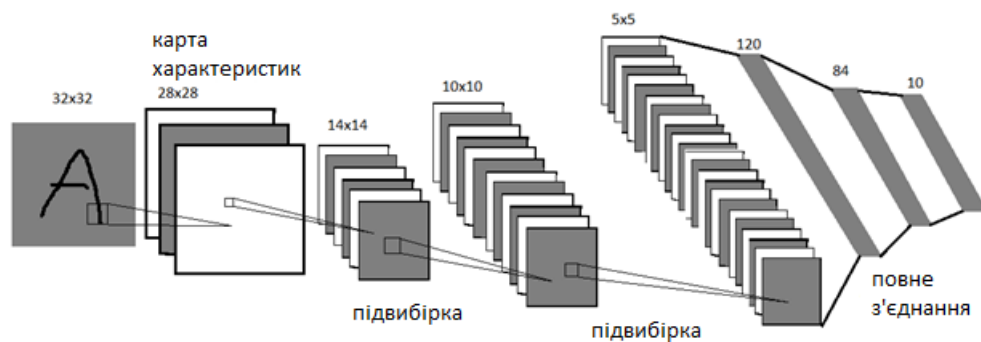


Рисунок 1.1 – Приклад архітектури ЗНМ [5]

У ЗНМ зазвичай чергуються згорткові шари і шари субдискретизації. На виході також може бути декілька шарів багатошарового персептрона.

Згорткові шари виконують операцію згортки: нейроном є матриця ваг, яку ще називають ядром згортки. Ядро поступово пересувається по вхідному зображенню, на кожному кроці значення ваг ядра помножуються на значення відповідних пікселів зображення. Після цього отримані значення проходять через функцію активації нейрона. Отримане значення є значенням відповідного пікселя карти ознак наступного шару. Операція субдискретизації використовується для зменшення розміру карти ознак.

1.4 Аналіз методів навчання нейромереж для розпізнавання образів

Існує декілька методів навчання нейронних мереж. Серед них можна виділити метод зворотного поширення, метод пружного поширення, а також генетичний метод [1]–[22].

Метод зворотного поширення є основним і має ще одну назву – Backpropagation, так як використовує алгоритм градієнтного спуску. Тобто за допомогою руху уздовж градієнта розраховується локальний мінімум і максимум функції.

Якщо уявити функцію у вигляді графіка, що визначає залежність помилки від ваги синапсу, то на цьому графіку необхідно знайти точку з мінімальним та максимальним значенням помилки. При цьому, кожна вага повинна мати своє графічне представлення і до кожної з них необхідно обчислити глобальний мінімум [1]–[22].

Сам градієнт являє собою векторне значення, що визначає напрямок і крутизну схилу. Градієнт знаходиться за допомогою похідної від функції в потрібній точці. Опинившись в певній точці із визначенням ваги, що розподіляється у випадковому порядку, обчислюється градієнт і визначається напрямок руху спуску, і так в кожній наступній точці, поки не досягається локальний мінімум, який дозволяє подальший спуск.

Щоб впоратися з цією неприємністю необхідно встановити потрібне значення моменту, яке дозволить подолати частину графіка і досягти потрібної точки. Якщо це значення буде недостатнім, то подолати опуклість не вийде, у випадку з установкою більш високого значення, ніж потрібно, є шанси «проскочити» глобальний мінімум.

Крім моменту прискорення, є ще поняття, що визначає загальну швидкість навчання мережі. Це значення, як і попереднє, являє собою гіперпараметр та підбирається методом проб і помилок. Оптимальний варіант

заздалегідь ніколи не відомий, дізнатися його можна тільки провівши кілька навчань та коригуючи кожен раз значення в потрібному напрямку.

Тепер можна перейти безпосередньо до методу навчання і визначити, що він із себе представляє. У процесі надходження інформації нейронна мережа послідовно передає її від одного нейрона до іншого за допомогою синапсів, до того моменту, поки інформація не опиниться на вихідному шарі і не буде видана як результат. Такий спосіб називається передачею вперед.

Після того, як результат отримано, обчислюється помилка та на її підставі виконується зворотний передачу. Суть цієї помилки - послідовно змінити вагу синапсів починаючи з вихідного та просуваючись до вхідного шару. При цьому значення ваги змінюється в бік кращого результату.

Для використання такого методу навчання підходять тільки ті функції активації, які можна диференціювати. Найпоширенішою функцією активації являється сигмоїд, також її ще називають логістичною функцією. У випадку, коли потрібно використовувати діапазон з від'ємними значеннями, використовують гіперболічний тангенс в якості функції активації.

Так як зворотне поширення обчислюється за допомогою розрахунків різниці результатів та множення його на похідну функції від вхідного значення.

Для того щоб успішно провести навчання, необхідно поширити отриману помилку на всю вагу мережі. Розрахувавши помилку на вихідному рівні можна обчислити різницю, яка буде послідовно передаватися між нейронами. Потім необхідно провести розрахунок градієнта для кожного вихідного зв'язку.

Далі, маючи всі необхідні дані, потрібно виконати оновлення ваг та розрахувати за допомогою функції активації значення, яке стане величиною зміни. При цьому не варто забувати про момент і швидкість навчання.

Одна ітерація методу зворотного поширення дає невеликий відсоток зменшення помилки, тому повторювати їх необхідно знову і знову, поки показник помилки не буде наближений до 0.

Метод пружного поширення (Resilient propagation або Rprop). Попередній спосіб навчання, представлений вище, має недолік у вигляді великих тимчасових витрат під час процесу навчання, недоречних у випадку, коли необхідно отримати швидкий результат. Для прискорення процесу було запропоновано різні алгоритми, що прискорюють процес синтезу нейромережі. Даний алгоритм використовує в якості основи навчання по епохах та застосовує тільки знаки часткових похідних для коригування вагових коефіцієнтів. Використовується певне правило, за яким здійснюється розрахунок величини корекції вагового коефіцієнта [4], [8]–[17].

Якщо на цьому етапі розрахунків похідна змінює свій знак, значить, зміна була надто великою та локальний мінімум був упущений і потрібно зробити відкат, тобто вагу повернути в зворотну позицію, а величину зміни зменшити.

Якщо знак похідної не змінився, то величина зміни ваги, навпаки, збільшується для більшої збіжності.

Якщо основні параметри корекції ваги зафіксувати, то налаштування глобальних параметрів можна уникнути. І це стане ще однією перевагою поточного методу над попереднім. Для цих параметрів є рекомендовані значення, однак, ніяких обмежень на їх вибір не накладається.

Щоб вага не приймала занадто великі або малі значення використовуються встановлені обмеження величини корекції. Величина компенсації також обчислюється за певним правилом.

Тобто якщо похідна функції в конкретній точці змінює знак з плюса на мінус, означає, що помилка зростає та вага вимагає корекції - відбувається її зменшення, в іншому випадку - збільшення.

Послідовність дій у цьому випадку наступна: ініціалізується величина корекції, обчислюються приватні похідні, підраховується нове значення корекції значень ваг, коригуються ваги. Якщо умова зупинки алгоритму не виконуються процес переходить до обчислення похідних і заново повторює цикл. Такий підхід дозволяє досягнути збіжності нейромережі швидше в кілька разів на відміну від попереднього варіанту навчання.

Також для синтезу нейромережних моделей використовуються генетичні методи (Genetic Algorithm). Генетичний метод являє собою спрощену інтерпретацію природного алгоритму, заснованого на схрещуванні результатів. Тобто відбувається схрещування результатів, вибір найкращих і формування на їх основі нового покоління.

У разі, якщо результат не влаштовує, алгоритм повторюється поки покоління не стане ідеальним. Алгоритм може завершитися без досягнення потрібного результату якщо кількість спроб буде вичерпано, або ж буде вичерпано час на мутацію. Цей алгоритм застосовується до процесу оптимізації ваги нейронної мережі при заданій топології «за замовчуванням» [1]–[22].

При цьому вага кодується двійковим кодом і кожен результат визначається повним набором ваги. Оцінка якості відбувається методом обчислення помилки на виході.

Також варто згадати такі методи навчання нейронних мереж:

- стохастичний метод, який коротко можна описати так: якщо знайдена величина оновлення - потрібно відразу оновити відповідну їй вагу;
- пакетний метод підсумовує значення всіх величин в конкретному циклі і тільки після закінчення проводить оновлення, що значно економить час, проте, при цьому страждає точність;
- міні-пакетний метод поєднує в собі плюси вищеописаних методів, ваги у вільному порядку розподіляються по обраних групах і змінюються на суму коефіцієнта корекції всіх ваг групи.

У процесі будь-якого навчання доводиться застосовувати гіперпараметри, підбір яких виконується вручну, при цьому вони не є змінними в певному рівнянні. Це вже згадані вище момент і швидкість навчання.

Також до цих параметрів можна віднести кількість прихованих шарів, число нейронів в шарі, присутність нейронів зміщення або, навпаки, їх відсутність. Наявність гіперпараметрів, в першу чергу, визначається типом штучної нейронної мережі. Правильний підбір їх значень безпосередньо впливає на збіжність мережі.

У випадку, коли процес навчання затягується, є ризик виникнення ситуації, в якій потрібно перенавчання мережі. Як правило, це відбувається в момент, коли мережа перенасичена даними та починає не навчатися, а просто запам'ятовувати результати для певного набору вхідних параметрів. При подачі нових параметрів виникають шуми, що впливають на результат.

Щоб уникнути такої необхідності, навчання необхідно проводити на різних вхідних даних мають значні відмінності між собою.

1.5 Обґрунтування вибору генетичного алгоритму для навчання НМ нейронних мереж

Для навчання ЗНМ було обрано генетичний алгоритм. При цьому кожний елемент популяції є набором вагових коефіцієнтів нейронної мережі. До популяції постійно застосовуються оператори селекції, схрещування та мутації.

Зазвичай ЗНМ навчають за допомогою методу зворотного поширення помилки (ЗПП). Це дозволяє відносно швидко навчити нейронну мережу, але знайдені вагові коефіцієнти не обов'язково будуть найкращими. Навчання НМ за допомогою ГА займає значно більше часу, але дозволяє знайти більш прийнятне рішення. Порівняння методів навчання наведено у табл. 1.1.

Таблиця 1.1 – Порівняння методів навчання нейронної мережі

Метод	Генетичний алгоритм	Зворотного поширення помилки
Час навчання (годин)	34,3	3,8
Помилка розпізнання (%)	0,75	2,5
Залежність від початкових значень ваг	низька	велика
Час навчання до помилки розпізнання 5% (годин)	11,4	0,2

Для наведення числових прикладів було використано вибірку з 2000 елементів, з яких 80% використано для навчання, а 20% – для тестування.

З таблиці видно, що навчання за допомогою ГА потребує значно більше часу, але це дозволяє знайти глобальний оптимум, що, в свою чергу, призводить до зменшення помилки розпізнання. При цьому збільшення часу навчання методом ЗПП не зменшить кількість помилок, тому що знайдено локальний оптимум.

1.6 Порівняльний аналіз програмних засобів синтезу розпізнавальних моделей на основі нейронних мереж глибокого навчання

У задачах побудови моделей розпізнавання рукописних символів проблема обробки великих обсягів даних у режимі реального часу. Для вирішення цієї проблеми доцільно використовувати нейронні мережі

глибокого навчання. В даний час розроблений ряд програмних бібліотек для побудови глибоких мереж, що представляється доцільним експериментально досліджувати.

Для дослідження були обрані активно використовувані на практиці бібліотеки для різних мов програмування. Серед них: TensorFlow (розробка Google, мова Python), Chainer (мова Python), Torch (мова Lua) і Synaptic (мова JavaScript).

Як конфігурацію нейронних мереж у дослідженні було обрано багат шаровий персептрон, як найбільш проста і розповсюджена мережа.

Вихідними даними були обрані два набори для класифікації з публічного репозитора Каліфорнійського університету в Ірині.

Перший набір – дані про тварин у зоопарку (101 спостереження, що описується 16 атрибутами). Завдання полягає в класифікації тварин на 7 різних класів.

Другий набір – дані про класифікацію людей за кольором шкіри (45057 спостереження, що характеризуються 3 атрибутами). Завдання полягає у визначенні, чи є представлений колір кольором людської шкіри.

У тестах використовувалися однакові конфігурації нейронних мереж для всіх бібліотек.

За результатами аналізу середнього навантаження на центральний процесор (ЦП) 4-ядерного комп'ютера при навчанні нейронної мережі на основі кожної з розглянутих бібліотек, визначено, що більшість бібліотек здатна задіяти лише частину наданих їм системних ресурсів і робить у середньому однакове навантаження на комп'ютер, незалежно від задачі.

У результаті проведення досліджень бібліотек по інших параметрах (використання пам'яті, швидкість навчання) і розгляду підтримуваного ними функціонала були отримані такі висновки.

Перевагами бібліотеки TensorFlow є оптимальна швидкість навчання і можливість роботи в кластері. До недоліків можна віднести більш складний (у

порівнянні з іншими розглянутими бібліотеками) процес опису архітектур нейронних мереж.

Фреймворк Chainer має достатній набір функцій для роботи з нейронними мережами і простий підхід до опису структур нейронних мереж, а також затрачає мінімальна кількість пам'яті серед розглянутих бібліотек, але не використовує всі ресурси ЦП і працює в однопотоковому режимі.

Фреймворк Torch містить багатий набір можливостей для наукових розрахунків і має велике співтовариство. Недолік – повільна робота на ЦП, у порівнянні з іншими розглянутими бібліотеками.

Бібліотека Synaptic дозволяє простий опис різних архітектур нейронних мереж і має високу швидкість роботи внаслідок простоти своєї реалізації. Серед недоліків – наявність лише базового набору функцій для побудови нейронних мереж, однопотокова архітектура (обмеження node.js) і відсутність підтримки роботи на графічному процесорі (ГП).

Таким чином, наявність зазначених недоліків обумовлює доцільність розробки власного програмного забезпечення для синтезу нейромережних моделей для розпізнавання рукописних символів.

1.7 Висновки за розділом 1

Проаналізовано процес розпізнавання образів. Надано постановку задачі розпізнавання образів на основі нейромережних моделей. Проаналізовано нейромережні моделі для розпізнавання образів. Визначено, що для розпізнавання рукописних символів доцільно використовувати згорткові нейронні мережі.

Проаналізовано методи навчання нейромереж для розпізнавання образів. Визначено, що для синтезу нейронних мереж широкого поширення набули метод зворотного поширення помилки та генетичні алгоритми.

Проте при використанні зворотного поширення помилки існує суттєва залежність результату (побудованої нейромережної розпізнавальної моделі)

від початкових значень ваг. Крім того, існує певна ймовірність зациклювання цього методу в області локального екстремуму. Генетичні методи мають механізми виходу із локальних екстремальних областей, проте вони є дуже повільними (див. табл. 1.1).

Тому для більш ефективної роботи CNN доцільно використовувати генетичний алгоритм для навчання ваг у фільтрах мережі, а в якості початкових значень вагових коефіцієнтів використовувати значення, отримання за допомогою методу зворотного поширення помилки, що дозволить уникнути зациклення в локальних мінімумах і підвищити середню ймовірність успішного розв'язання задачі розпізнавання образів.

Досліджено програмні засоби синтезу розпізнавальних моделей на основі нейронних мереж глибокого навчання. Визначено, що наявність недоліків існуючого програмного забезпечення обумовлює доцільність розробки власного програмного забезпечення для синтезу нейромережних моделей для розпізнавання рукописних символів.

2 ВИБІР ТА ОБҐРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ, ЯКА ПРОЄКТУЄТЬСЯ

2.1 Дослідження та аналіз вимог до програмного продукту

Метою розробки є створення програмного забезпечення для реалізації нейромережних методів розпізнавання рукописних символів.

Діловий контекст розробки полягає в тому, що по завершенню реалізації проєкту власник матиме готовий програмний продукт, з якого зможе отримати матеріальну вигоду, автоматизувавши процес розпізнавання рукописних символів.

Функціональними вимогами до програмного продукту (ПП) є такі:

- зчитування вибірок даних;
- додавання нових значень у вибірку;
- навчання системи за вибіркою (синтез нейромережної моделі для розпізнавання образів на основі даних навчальної вибірки);
- збереження і завантаження результатів навчання;
- розпізнавання введених користувачем цифр.

Розроблений ПП має зберігатися на DVD диску розміром 8ГБ. Дані повинні зчитуватися на платформі Windows.

Інтерфейс має бути інтуїтивно зрозумілий користувачу. Колір меню не повинен дратувати око користувача.

Для запуску розроблюваного ПП треба:

- процесор частотою 1 ГГц;
- місце на жорсткому диску – 1 Гб;
- відеокарта з оперативною пам'яттю 128 Мб;
- оперативна пам'ять – 1 Гб;
- маніпулятор миша.

Розповсюдження ПП повинно проводитись згідно з законам України.

2.2 Вибір мови програмування

Для розробки програмного забезпечення обрано мову C#, яка є об'єктно-орієнтованою мовою програмування з безпечною системою типізації для платформи .NET [23]–[25].

Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато що від своїх попередників – мов C++, Delphi та ін. – C#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад множинне спадкування класів (на відміну від C++).

Об'єктно-орієнтоване програмування – це підхід до розробки програмного забезпечення, заснований на об'єктах, а не на процедурах. Цей підхід дозволяє максимізувати принципи модульності і "приховування інформації".

Об'єктно-орієнтований принцип розробки дає багато переваг. Наприклад, кожен об'єкт інкапсулює його структуру даних з процедурою, використовуваною для роботи з екземплярами структури даних. Це дозволяє усунути в коді програми внутрішні залежності, які можуть швидко привести до того, що цей код буде важко обслуговувати.

Ключові риси об'єктно-орієнтованого програмування добре відомі:

- інкапсуляція – це визначення класів - користувача типів даних, які об'єднують свій вміст в єдиний тип і реалізують деякі операції або методи над ним. Класи зазвичай є основою модульності, інкапсуляції та абстракції даних в мовах ООП;

- спадкування – спосіб визначення нового типу, наслідуючи елементи (властивості і методи) існуючого і модифікуючи або розширюючи їх. Це сприяє вираженню спеціалізації і генералізації;

– поліморфізм, дозволяє одноманітно посилатися на об'єкти різних класів (зазвичай всередині деякої ієрархії). Це робить класи ще більш зручними і робить програми, засновані на них, легше для розширення і підтримки.

2.3 Вибір середовища розробки

Середовище розробки Microsoft Visual Studio – серія продуктів фірми Майкрософт, які включають інтегроване середовище розробки програмного забезпечення та ряд інших інструментальних засобів. Ці продукти дозволяють розробляти як консольні програми, так і програми з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms [23]–[25].

Microsoft Visual Studio є відкритим мовним середовищем. Це означає, що поряд з мовами програмування, включеними в середовище фірмою Microsoft Visual C++ .NET, Visual C# .NET, J# .NET, Visual Basic .NET, в нього можуть добавлятися будь-які мови програмування, при дотриманні певних умов. Головне обмеження, яке можна вважати і головною достоїнністю, полягає в тому, що всі мови, що включаються в середовище розробки Visual Studio .NET, повинні використовувати єдиний каркас – .NET Framework. Завдяки цьому досягаються багато властивостей:

- легкість використання компонентів, розроблених на різних мовах;
- можливість розробки декількох частин однієї програми на різних мовах;
- можливість безшовного налагодження такого застосування;
- можливість писати клас на одній мові, а його нащадків – на інших мовах.

Єдиний каркас призводить до зближення мов програмування, дозволяючи разом з тим зберігати їх індивідуальність і наявні в них достоїнсті.

Платформа Microsoft .NET забезпечує середовище виконання програм і визначає особливості розробки та виконання програмного коду на ПК. Вона має засоби організації взаємодії з операційною системою і прикладними програмами, методи доступу до баз даних, засоби підтримки мережних додатків, мови програмування і велику кількість базових класів [23]–[25].

Порівняльний аналіз середовищ розробки наведено у таблиці 2.1.

Таблиця 2.1 – Порівняльний аналіз середовищ розробки

Критерії порівняння	Середовище розробки		
	NetBeans	Eclipse	Microsoft Visual Studio
Швидкість модернізації	±	±	++
Підтримка репозиторіїв	++	+	++
Підтримка серверів	++	+	++
Розроблення графічного інтерфейсу користувача	+	–	++
Доступність (вимоги до розповсюдження)	+	+	+

Таким чином, в якості інструментарію розробки обрано Microsoft Visual Studio, що включає в себе інтегроване середовище розробки програмного забезпечення та ряд інших інструментальних засобів, та є досить зручним для розробки програмного забезпечення та його модифікації.

2.3 Структурна схема програми

Інтелектуальна система (ІС) – програмна система, яка здатна вирішувати завдання, що традиційно вважаються творчими, які належать конкретній предметній галузі, зокрема галузі розпізнавання рукописних символів. Основними модулями розробленого програмного забезпечення є такі:

- модуль роботи з даними – забезпечує роботу з вибірками даних, а також збереження та завантаження результатів роботи програми;
- модуль управління – забезпечує можливість управління системою;
- модуль НМ – реалізує згорткову нейронну мережу та можливість її тестування;
- модуль ГА – забезпечує можливість навчання НМ за допомогою генетичного алгоритму;
- модуль ЗПП – забезпечує можливість навчання НМ за допомогою метода зворотного поширення помилки.

Структурну схему програми зображено на рис. 2.1.

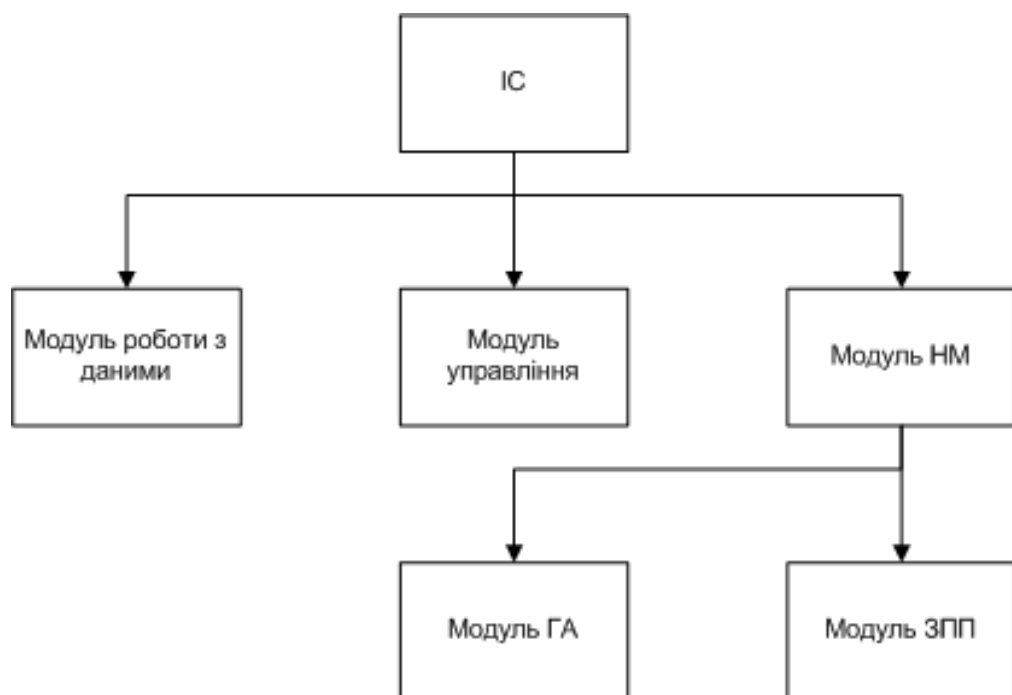


Рисунок 2.1 – Структурна схема програми

2.4 Висновки за розділом 2

Досліджено та проаналізовано вимоги до програмного продукту.

Для розробки програмного забезпечення обрано мову C#, яка є об'єктно-орієнтованою мовою програмування з безпечною системою типізації для платформи .NET.

В якості інструментарію розробки обрано Microsoft Visual Studio, що включає в себе інтегроване середовище розробки програмного забезпечення та ряд інших інструментальних засобів, та є досить зручним для розробки програмного забезпечення та його модифікації.

Розроблено структурну схему програми. Визначено, що основними модулями розробленого програмного забезпечення є такі: модуль роботи з даними, модуль управління, модуль реалізації загорткової нейронної мережі; модуль навчання НМ за допомогою генетичного алгоритму, модуль навчання НМ за допомогою метода зворотного поширення помилки.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Реалізація згорткової нейронної мережі

Реалізація згорткової нейронної мережі у розробленому програмному забезпеченні здійснюється наступним чином. На вхід мережі подається зображення 28x28 пікселів. Далі до вхідного зображення застосовується операція згортки ядром 5x5. Кожний піксель вхідного зображення помножується на відповідну чарунку ядра, після чого до суми отриманих результатів застосовується функція активації. Отримане значення являє собою значення відповідної чарунки карти ознак. Після цього ядро переміщається на 1 піксель і операція повторюється. В результаті буде отримано 8 карт ознак розміром 24x24. Приклад операції згортки наведено на рис. 3.1.

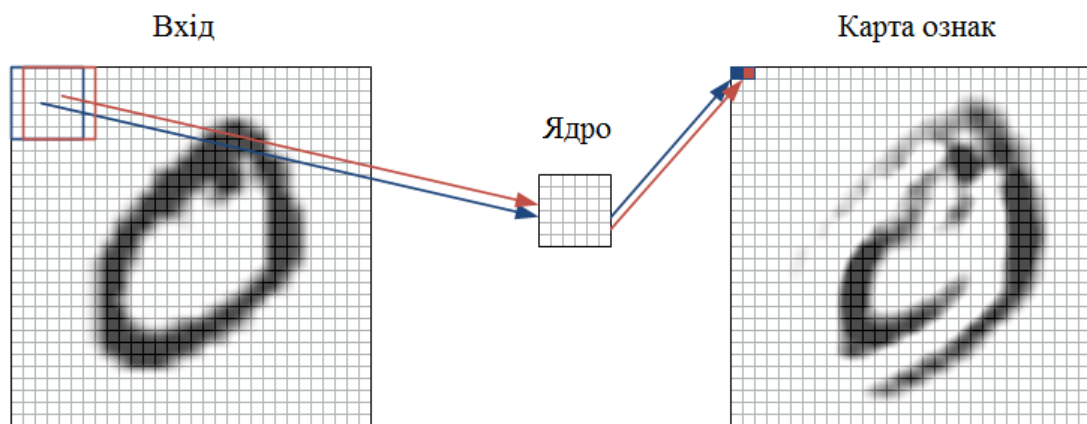


Рисунок 3.1 – Приклад операції згортки

Після цього до кожної з карт ознак застосовується операція субдискретизації. Обирається область карти ознак розміром 2x2, і до цих значень застосовується операція MaxPooling, що обирає максимальне значення. В результаті буде отримано 8 карт ознак розміром 12x12. Потім відбувається повторення попередніх етапів поки не буде отримано всі карти ознак відповідного розміру (4x4).

Далі за отриманими картами ознак обчислюються функції активації 15 нейронів наступного шару. Після цього обчислюються значення функцій активації 10 нейронів вихідного шару. Номер нейрона з найбільшим значенням i є розпізнаною цифрою, що є результатом процедури розпізнавання образів.

3.2 Розробка гібридного методу синтезу нейромережних моделей на основі генетичного пошуку та зворотного поширення помилки

У першому розділі визначено, що при використанні зворотного поширення помилки існує суттєва залежність результату (побудованої нейромережної розпізнавальної моделі) від початкових значень ваг. Крім того, існує певна ймовірність зациклювання цього методу в області локального екстремуму. Генетичні методи мають механізми виходу із локальних екстремальних областей, проте вони є дуже повільними.

Тому у цьому підрозділі для більш ефективної роботи CNN запропоновано використовувати генетичний алгоритм для навчання ваг у фільтрах мережі, а в якості початкових значень вагових коефіцієнтів використовувати значення, отримання за допомогою методу зворотного поширення помилки, що дозволить уникнути зациклення в локальних мінімумах і підвищити середню ймовірність успішного розв'язання задачі розпізнавання образів.

Спочатку задається вибірка вхідних даних, на основі якої за допомогою методу зворотного поширення помилки розв'язується завдання розпізнавання образів (синтезується розпізнавальна модель). Вагові коефіцієнти такої моделі використовуються в якості початкових значень вагових коефіцієнтів при подальшому генетичному пошуку.

Потім задаються: розмір популяції (PS), ймовірності мутації (PM) та схрещування (PC), кількість елітних хромосом (EC), критерій зупину (E).

Для кожного елемента популяції ваговим коефіцієнтам задаються випадкові значення.

Після цього для кожного елемента популяції обчислюється значення фітнес-функції за формулою (3.1):

$$F(H) = 1/N \sum_{i=0}^{10} \sqrt{\sum_{j=0}^{10} (X_{i,j} - Y_{i,j})^2}, \quad (3.1)$$

де $F(H)$ – значення фітнес-функції;

N – кількість елементів в навчальній вибірці;

$X_{i,j}$ – реальний вхід нейронної мережі;

$Y_{i,j}$ – бажаний вихід нейронної мережі.

Далі обирається ЕС елементів популяції з найменшим значенням фітнес-функції і додаються до наступної популяції. Обчислюється середнє значення фітнес-функції для всіх хромосом і помножується на 1,15.

Серед всіх хромосом, значення фітнес-функції яких менше значення, що обчисленого раніше, обирається PS-ЕС хромосом для наступної популяції з рівною ймовірністю.

Для хромосоми з найменшим значенням фітнес-функції обчислюється помилка розпізнання на тестовій вибірці. Якщо значення менше E , пошук завершується.

Після цього популяція ділиться на PS/2 пар. З ймовірністю PC для кожної пари за формулою (3.2) обчислюються вагові коефіцієнти:

$$w = w_1 k + w_2 (1 - k), \quad (3.2)$$

де w – ваговий коефіцієнт першого нащадка;

w_1 – ваговий коефіцієнт першого батька;

w_2 – ваговий коефіцієнт другого батька;

k – коефіцієнт, значення якого від 0 до 1, який визначає значущість кожного з батьків.

Для кожного елемента популяції з ймовірністю $PM \cdot 10$ всі вагові коефіцієнти помножуються на випадкове число в діапазоні $1-k - 1+k$, де k – мале число, значення якого залежить від ітерації.

Після цього для кожного елемента популяції з ймовірністю PM міняються місцями 2 випадкові ядра згортки, а також деякі вагові коефіцієнти помножуються на випадкове число від -1 до 1. Далі відбувається перехід до етапу оцінювання хромосом поточної популяції.

Таким чином, розроблено гібридний метод синтезу нейромережних моделей на основі генетичного пошуку та зворотного поширення помилки, в якому генетичний алгоритм використовується для навчання ваг у фільтрах мережі, а в якості початкових значень вагових коефіцієнтів використовуються значення, отримання за допомогою методу зворотного поширення помилки, що дозволяє уникнути зациклення в локальних мінімумах і підвищити точність розпізнавання за допомогою синтезованих нейромоделей.

3.3 Тестування програмного забезпечення

Для проведення експериментів використано такі набори параметрів еволюційного пошуку:

- розмір популяції – 20, ймовірність схрещування – 80%, ймовірність мутації – 3%, кількість елітних хромосом – 4 (A0);
- розмір популяції – 40, ймовірність схрещування – 80%, ймовірність мутації – 3%, кількість елітних хромосом – 6 (A1);
- розмір популяції – 80, ймовірність схрещування – 80%, ймовірність мутації – 3%, кількість елітних хромосом – 8 (A2).

Для кожного з набору параметрів використано такі методи навчання:

- навчання «з нуля» – без використання інформації, отриманої методом зворотного поширення (B0);

– навчання методом ЗПП до 10% помилок, після чого навчання за допомогою ГА (Б1);

– навчання методом ЗПП до 5% помилок, після чого навчання за допомогою ГА (Б2);

Також було проведено навчання методом ЗПП без використання ГА (В).

Для проведення експериментів було використано вибірку з 1250 елементів, 20% з яких використано для тестування.

Для порівняння методів було використано такі параметри:

– розпізнаваність після визначеного часу навчання – дозволяє визначити найкращий метод при умові обмеженого часу на навчання;

– час навчання, за який досягнуто визначену розпізнаваність – дозволяє визначити найкращий метод при умові необмеженого часу на навчання (якщо визначену розпізнаваність не досягнуто за 25 годин, вважається, що за допомогою методу неможливо досягнути цього).

Кожен з методів було запущено по 4 рази, після чого знайдено середнє арифметичне значення.

Результати експериментів було представлено в таблиці 3.1, при цьому, якщо в чарунці таблиці знаходяться 2 значення через кому, друге з них показує кількість врахованих експериментів замість 4. При цьому під час проведення неврахованих експериментів не було досягнуто необхідної точності за максимальний час.

Таблиця 3.1 – Результати експериментів

Метод	A0, Б0	A0, Б1	A0, Б2	A1, Б0	A1, Б1	A1, Б2	A2, Б0	A2, Б1	A2, Б2	В
Розп. через 1 хв.	34,2%	92,1%	95,2%	38,6%	94%	95,8%	33,7%	92,5%	95,3%	94,3%
Розп. через 3 хв.	59,1%	94,7%	96,5%	63%	95,9%	96,4%	59,1%	95,4%	95,8%	95,5%

Продовження таблиці 3.1

Метод	A0, Б0	A0, Б1	A0, Б2	A1, Б0	A1, Б1	A1, Б2	A2, Б0	A2, Б1	A2, Б2	В
Розп. через 10 хв.	82%	96,8%	96,7%	82,1%	98,2%	96,6%	76,3%	96%	95,9%	96,6%
Час до 95% розп.	-	4,2	0,1	-	1,9	0,1	-	3,1	0,1	2,9
Час до 97% розп.	-	13,4	13,3	-	7,2	11,1	-	18,6, 2	22,5, 1	12,3
Час до 99% розп.	-	-	24,1	-	21,8	-	-	-	-	-

На рисунку 3.2 зображено графік залежності якості навчання (точності розпізнавання) від часу навчання, де суцільною лінією показано метод зворотного поширення помилки, а пунктирною – модифікований генетичний метод.

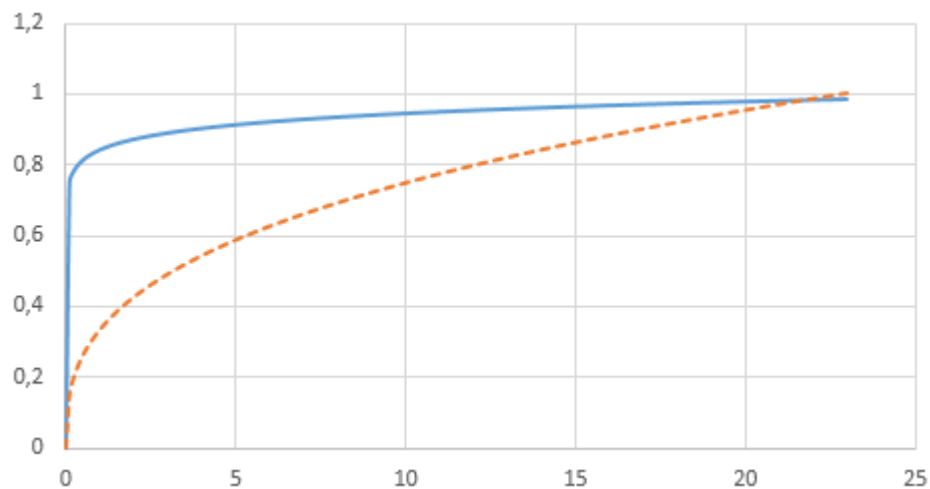


Рисунок 3.2 – Графік залежності якості навчання (точності розпізнавання) від часу навчання

Таким чином, результати експериментів показали, що генетичні алгоритми для навчання НМ доцільно використовувати, коли необхідно досягнути максимальної точності розпізнавання незалежно від затраченого на це часу. Якщо ж необхідно швидко досягнути великої, але не максимальної

точності, доцільніше використовувати метод зворотного поширення помилки. При цьому можна комбінувати ці методи для досягнення кращих результатів.

3.4 Висновки за розділом 3

Запропоновано основні рішення щодо реалізації програми. Описано реалізацію згорткової нейронної мережі.

Розроблено гібридний метод синтезу нейромережних моделей на основі генетичного пошуку та зворотного поширення помилки, в якому генетичний алгоритм використовується для навчання ваг у фільтрах мережі, а в якості початкових значень вагових коефіцієнтів використовуються значення, отримання за допомогою методу зворотного поширення помилки, що дозволяє уникнути зациклення в локальних мінімумах і підвищити точність розпізнавання за допомогою синтезованих нейромоделей.

Виконано тестування розробленого програмного забезпечення, що реалізує нейромережні методи розпізнавання рукописних символів. Результати тестування підтвердили, що програма може використовуватися за призначенням.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Призначення й умови застосування програми

Програмний продукт призначений для розпізнавання рукописних символів.

Для коректної роботи програму потрібно запускати в операційній системі Windows.

Програма не вимоглива до процесора та оперативної пам'яті, тому запуститься і швидко працюватиме на будь-якому комп'ютері з операційною системою Windows з використанням мінімуму ресурсів.

Для експлуатації програмного комплексу необхідні такі програмно-технічні засоби:

- ПК під керівництвом операційної системи та встановленим .Net Framework версії 4.5 та вище;
- тактова частота процесора 1 ГГц або більше;
- обсяг оперативної пам'яті не менше 1 Гб;
- обсяг вільного простору на диску – не менше 1 Гб;
- дозвіл екрана монітора не менше 1024x768;
- маніпулятор миша.

4.2 Характеристики програми

Програма була створена у середі програмування Visual Studio на мові C#. Для роботи з програмою користувач повинен мати ПК з операційною системою Windows і .Net Framework 4.5 та вище.

В результаті роботи програми, при запиті користувача, виводиться найбільш ймовірний результат розпізнавання введенного символу.

4.3 Звернення до програми

Запуск програми відбувається за допомогою файлу DigitRecognize.exe. Час завантаження програми – менше однієї секунди. Інших характеристик не визначено.

4.4 Вхідні і вихідні дані

Вхідними даними для програми є вибірка даних з прикладами рукописних цифр, вагові коефіцієнти нейронної мережі, а також введені користувачем цифри для розпізнання.

Вихідними даними для програми є вагові коефіцієнти нейронної мережі, результати тестування та результати розпізнання введеної користувачем цифри.

4.5 Повідомлення

Реалізована програмна система гарантує її стійке функціонування. У випадку виникнення помилок в програмі передбачено відображення стандартних діалогових вікон. Під час навчання у заголовку вікна виводиться інформація про процес навчання: кількість ітерацій, помилка розпізнання. При розпізнанні введеного символу виводиться повідомлення з найбільш ймовірним результатом.

5 КЕРІВНИЦТВО ОПЕРАТОРА

5.1 Призначення програми

Програмний продукт призначений для розпізнавання рукописних символів.

Для коректної роботи програму потрібно запускати в операційній системі Windows 7 або більш пізнішій версії операційної системи. Також повинен бути встановлений фреймворк .Net Framework версії 4.5 та вище.

Інтерфейс користувача забезпечує просту і ефективну взаємодію користувача з програмним продуктом.

Програма дозволяє користувачу виконувати такі функції:

- зчитування вибірок даних;
- додавання нових значень у вибірку;
- навчання системи за вибіркою (синтез нейромережної моделі для розпізнавання образів на основі даних навчальної вибірки);
- збереження і завантаження результатів навчання;
- розпізнання введених користувачем символів.

5.2 Умови виконання програми

Для експлуатації програмного комплексу необхідні такі програмно-технічні засоби:

- ПК під керівництвом операційної системи та встановленим .Net Framework версії 4.5 та вище;
- тактова частота процесора 1 ГГц або більше;
- обсяг оперативної пам'яті не менше 1 Гб;
- обсяг вільного простору на диску – не менше 1 Гб;
- дозвіл екрана монітора не менше 1024x768;
- маніпулятор миша.

Час роботи програми залежить від конфігурації комп'ютера, на якому вона виконується.

5.3 Виконання програми

Запуск програми відбувається за допомогою файлу DigitRecognize.exe. Час завантаження програми – менше однієї секунди. Інших характеристик не визначено.

5.4 Приклад роботи з програмою

Під час навчання у заголовку вікна виводиться інформація про процес навчання: кількість ітерацій, помилка розпізнання. При розпізнанні введеного символу виводиться повідомлення з найбільш ймовірним результатом. Приклад повідомлень зображено на рис. 5.1.

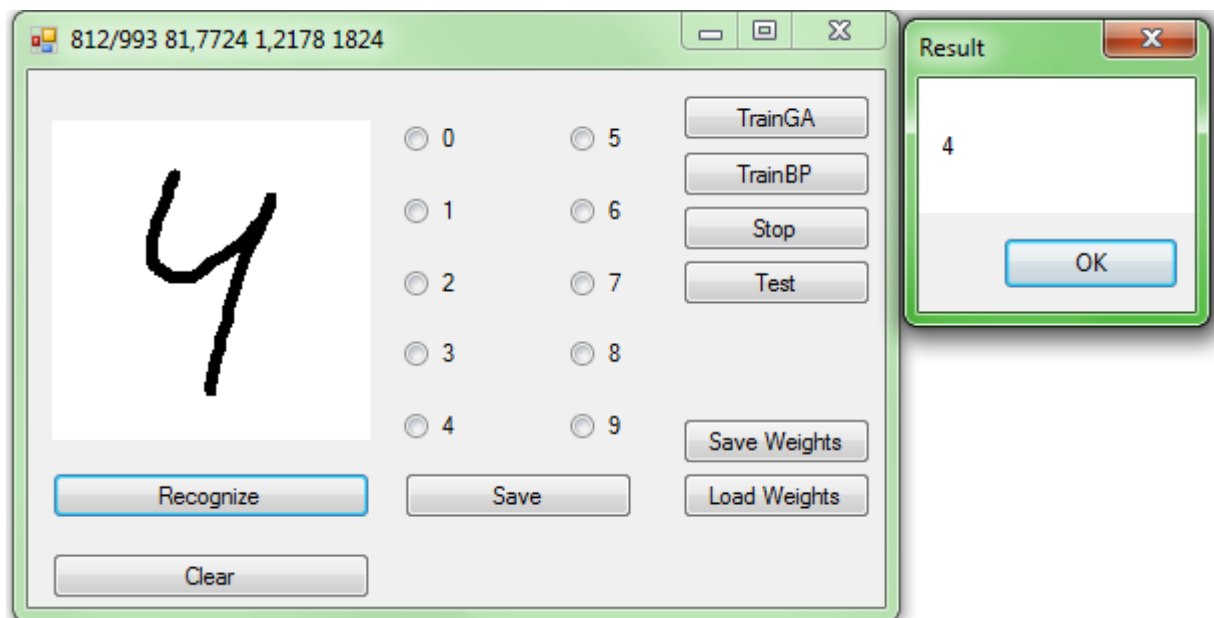


Рисунок 5.1 – Приклад роботи з програмою

5.5 Повідомлення оператору

Реалізована програмна система гарантує її стійке функціонування. У випадку виникнення помилок в програмі передбачено відображення стандартних діалогових вікон. Під час навчання у заголовку вікна виводиться інформація про процес навчання: кількість ітерацій, помилка розпізнання. При розпізнанні введеного символу виводиться повідомлення з найбільш ймовірним результатом.

6 ЕКОНОМІКО-ОРГАНІЗАЦІЙНА ЧАСТИНА

У дипломній кваліфікаційній роботі магістра досліджено та реалізовано програму для розпізнавання рукописних символів.

Очікується, що впровадження в експлуатацію розробленого програмного забезпечення на підприємстві дозволить скоротити одного співробітника, що відповідає за обробку рукописних символів, з заробітною платою 8 тис. гривень на місяць.

У розробці беруть участь програміст із заробітною платою 15000 грн. в місяць і консультант з заробітною платою 10000 грн. у місяць. На розробку проєкту було витрачено 2 місяці. Розробка програми розпочалася першого вересня і повинна бути виконана до кінця жовтня 2020 року.

6.1 Планування розробки програмного продукту

Весь комплекс розробки програмного продукту можна розділити на етапи [26]. Для кожного етапу вказуються трудомісткість, кількість виконавців і тривалість робіт. Інформація з характеристики робіт по розробці програмного забезпечення зведена у таблицю 6.1.

Таблиця 6.1 – Характеристика робіт по розробці програми

Найменування робіт	Трудомісткість		Виконавці	Тривалість, днів
	люди-дні.	%		
1	2	3	4	5
Аналіз ПЗ	8	11,9	Програміст, консультант	4
				4
Визначення вимог до ПП	12	17,9	Програміст, консультант	6
				6

Продовження таблиці 6.1

1	2	3	4	5
Проектування структури	24	35,8	Програміст, консультант	12 12
Розробка схеми функціонування програми	4	5,97	Програміст	4
Створення програмного коду	9	13,4	Програміст	9
Тестування і налагодження програмного продукту	7	10,4	Програміст	7
Складання програмної документації	3	4,5	Програміст	3
Проектування структури	24	35,8	Програміст, консультант	12 12

6.2 Визначення витрат на розробку програми

Для визначення витрат на розробку програми складають калькуляцію кошторисної вартості робіт, що включає статті:

- основна заробітна плата;
- додаткова заробітна плата;
- відрахування на єдиний соціальний внесок (ЄСВ);
- матеріали та комплектуючі вироби;
- витрати на спеціальне обладнання (на оплату машинного часу ЕОМ для написання і налагодження програмних засобів);

– накладні витрати.

Витрати за статтею «Основна заробітна плата» складаються з планового фонду зарплати всіх категорій працівників, зайнятих в розробці програми. Розрахунок зарплати на підставі даних про трудомісткість, представлений в таблиці 6.2.

Таблиця 6.2 – Основна заробітна плата

Посада виконавця	Чисельність, чол	місячний оклад	Кількість місяців роботи	Сума ЗП, грн
Програміст	1	15000	2	30000
Консультант	1	10000	1	10000
Разом	2			40000

Додаткову заробітну плату приймають рівною 10% від основної зарплати. Вона розраховується за формулою (6.1):

$$ЗП_{\text{дод}} = ЗП_{\text{осн}} \cdot 10\%. \quad (6.1)$$

Підставивши величину основної заробітної плати в дану формулу отримаємо: $ЗП_{\text{дод}} = 40000 \cdot 0.1 = 4000$ грн.

Відрахування на єдиний соціальний внесок (ЄСВ) визначають у відсотковому відношенні від сум основної та додаткової зарплати з урахуванням премій і доплат. Відрахування вираховуються за формулою (6.2):

$$ОТ = (ЗП_{\text{осн}} + ЗП_{\text{дод}}) \cdot 0,22. \quad (6.2)$$

Обчислюємо:

$$ОТ = (40000 + 4000) \cdot 0,22 = 9680 \text{ грн.}$$

Використовується два найменування матеріалів: картридж - 800 грн, папір - 200 грн. Витрати на матеріали розраховують за формулою (6.3):

$$M = \sum_{i=1}^n (\text{Ц}_i \cdot N_i \cdot (1 + K_{\text{т.з.}}) - \text{Ц}_{io} \cdot N_{io}), \quad (6.3)$$

де M – витрати на матеріали, покупні напівфабрикати і комплектуючі вироби, грн.;

$K_{\text{т.з.}}$ – коефіцієнт, що враховує транспортно-заготівельні витрати;

Ц_i – ціна i -го найменування матеріалу, напівфабрикату і комплектуючого, грн.;

N_i – потреба в i -му матеріалі, напівфабрикаті і комплектуючому;

Ц_{io} – ціна зворотних відходів i -го найменування матеріалу, грн;

N_{io} – кількість зворотних відходів i -го найменування;

n – кількість найменувань матеріалів, напівфабрикатів і комплектуючих.

Знаходимо:

$$\text{Ц}_{io} = 0; N_{io} = 0; K_{\text{т.з.}} = 0,05;$$

$$M = (1 + 0,05) \cdot (800 + 200) = 1050 \text{ грн.}$$

Таким чином, разом витрати на матеріали та комплектуючі вироби становлять 1050 гривень.

У статтю «Витрати на спеціальне обладнання» входять витрати на придбання, транспортування, монтаж і налагодження нестандартного обладнання. Практично, в даному випадку, в цій статті враховуються витрати на оплату машинного часу ЕОМ для написання і налагодження програмних засобів. Для цього необхідно скласти кошторис «витрат на утримання і експлуатацію устаткування», виходячи з якого визначиться вартість одного машино-години роботи ПК, після множення якої на машинний час пішло на

написання і налагодження програми отримаємо витрати на оплату машинного часу.

Амортизаційні відрахування визначають за формулою (6.4):

$$A = БВ \cdot \frac{H_a}{100}, \quad (6.4)$$

де БВ – балансова вартість обчислювальної техніки, грн;

H_a – норма амортизаційних відрахувань на повне оновлення обчислювальної техніки, %. Приймаємо, що техніка повинна працювати 4 роки, тому $H_a = 25\%$.

Балансова вартість обчислювальної техніки становить 30000 грн. Тому амортизаційні відрахування становлять:

$$A = 30000 \cdot 0,25 = 7500 \text{ грн.}$$

Статтю "Експлуатація обладнання" розраховують підсумовуванням витрат на силову електроенергію і допоміжні матеріали. Витрати на електроенергію визначають за формулою (6.5):

$$C_{\text{э}} = N_{\text{н}} \cdot \Phi_{\text{эф}} \cdot K_{\text{зв}} \cdot K_{\text{зм}} \cdot \text{Ц}_{\text{э}}, \quad (6.5)$$

де $N_{\text{н}}$ – номінальна потужність ЕОМ, кВт;

$\Phi_{\text{эф}}$ – річний ефективний фонд часу роботи ЕОМ, машино-год;

$K_{\text{зв}}$ – середній коефіцієнт завантаження за часом;

$K_{\text{зм}}$ – коефіцієнт завантаження по потужності;

$\text{Ц}_{\text{э}}$ – ціна одного кВт · год електроенергії, грн./ (кВт · год).

Номінальна Потужність ЕОМ – 0,2 кВт. Річний ефективний фонд часу роботи ЕОМ ставити 1800 годин. Середні коефіцієнти завантаження за часом

та за потужністю рівні відповідно 0,9 та 0,6. Ціна однієї кіловат-години електроенергії ставить 2,11 грн. Отримуємо:

$$C_e = 0,2 \cdot 1800 \cdot 0,9 \cdot 0,6 \cdot 2,11 = 410,18 \text{ грн.}$$

Приймаємо, що $C_e = 410$ грн.

Витрати за статтею "Заробітна плата обслуговуючих робітників та відрахування на соціальне страхування в інші фонди" визначають за формулою (6.6):

$$ЗП_{\text{обсл}} = ФЗП_{\Gamma} \cdot (1 + K_{\text{відрах.}}) \cdot t_{\text{обсл}} / \Phi_{\text{еф.обсл}}, \quad (6.6)$$

де $ФЗП_{\Gamma}$ – річний фонд заробітної плати (основної та додаткової) обслуговуючих робітників, грн.;

$K_{\text{відрах}}$ – коефіцієнт, що враховує відрахування на соціальне страхування і в інші фонди;

$t_{\text{обсл}}$ – час протягом року, необхідне технічне обслуговування ЕОМ, годин/рік. Приймаємо $t_{\text{обсл}} = 24$ год.;

$\Phi_{\text{еф.оюсл}}$ – річний ефективний фонд часу обслуговуючого персоналу, годин/рік. Приймаємо, що річний ефективний фонд часу обслуговуючого персоналу становить 1800 годин.

Заробітна плата обслуговуючого персоналу = 8000 грн/місяць.

Розрахуємо:

$$ФЗП_{\Gamma} = 8000 \cdot 12 = 96000 \text{ грн.,}$$

$$ЗП_{\text{обсл}} = 96000 \cdot (1 + 0,22) \cdot 24 / 1800 = 1562 \text{ грн.}$$

Суму витрат за статтею "Поточний ремонт обладнання" обчислимо як 3% від балансової вартості обладнання, оскільки неможливо заздалегідь

визначити, скільки коштів необхідно витратити на придбання запасних частин наявного обладнання. Балансова вартість обчислювальної техніки становить 30000 грн. Тому сума витрат за статтею "Поточний ремонт обладнання" становить: $30000 \cdot 0,03 = 900$ грн.

Інших витрат за проектом не передбачено, тому сума витрат за цією статтею дорівнює нуль гривень.

Кошторис витрат на утримання і експлуатацію устаткування наведений в таблиці 6.3.

Таблиця 6.3 – Кошторис витрат на утримання і експлуатацію устаткування

Найменування статей витрат	Сума, грн.
Амортизація обладнання	7500
Заробітна плата основна і додаткова обслуговуючих робітників з відрахуваннями на соціальні заходи	1562
Експлуатація обладнання (крім витрат на поточний ремонт)	410
Поточний ремонт обладнання	900
Інші витрати	0
Разом	10372

Витрати на оплату машинного часу ЕОМ для написання і налагодження програмних засобів визначаються за формулою (6.7):

$$C_{\text{мо}} = P_{\text{екс}} \cdot t_{\text{мо}}, \quad (6.7)$$

де $C_{\text{мо}}$ – витрати на оплату машинного часу, грн;

$P_{\text{екс}}$ – експлуатаційні витрати на одну годину машинного часу цієї цифрової ЕОМ, грн. / машино–год.;

$t_{\text{мо}}$ – машинний час цифрової ЕОМ для написання і налагодження даного програмного продукту, машино–год.

Розраховуємо:

$$P_{\text{екс}} = 10372 / 1800 = 5,76 \text{ грн. / машино–год.}$$

Для написання і налагодження даного програмного продукту ЕОМ експлуатується протягом 45 днів в одну зміну по 8 годин, що складає в сумі 360 годин. Таким чином отримуємо суму за статтею «Витрати на спеціальне обладнання»:

$$C_{\text{мо}} = 5,76 \cdot 360 = 2074 \text{ грн.}$$

До накладних витрат відносяться витрати на загальне управління і загальногосподарські потреби (заробітна плата апарату управління, канцелярські витрати і т.п.), утримання та експлуатацію будівель. Накладні витрати включаються до вартості розробки програми непрямым шляхом – у відсотках до основної заробітної плати розробників. Для обчислення цієї статті видатків у дипломній кваліфікаційній роботі магістра приймемо, що накладні видатки становлять 25% від основної заробітної плати розробників, що складає 40000 грн. Тому сума накладних видатків становить:

$$40000 \cdot 0,25 = 10000 \text{ грн.}$$

Калькуляція кошторисної вартості робіт з розробки програми наведена в таблиці 6.4.

Таблиця 6.4 – Калькуляція кошторисної вартості робіт з розробки програми

Найменування статей витрат	Сума, грн.
Основна заробітна плата	40000
Додаткова заробітна плата	4000
ЄСВ	9680
Матеріали і комплектуючі	1050
Витрати на спеціальне обладнання	2074
Накладні витрати	10000
Разом	66804

6.3 Розрахунок економічної ефективності програмного продукту

Річну економію від впровадження програми, що дозволяє знизити витрати машинного часу ЕОМ для рішення певного завдання в порівнянні із програмою, що застосовувалася раніше, визначають за формулою (6.8):

$$E = (T_{M_1} - T_{M_2}) \cdot B_{\text{екс.КТЗ}} + E_H^a \cdot \Phi_{\text{б.КТЗ}} \cdot (T_{M_1} - T_{M_2}) / \Phi_{\text{еф.КТЗ}} - S_{\text{рп}} / T_c + E_{\text{зп}}, \quad (6.8)$$

де T_{M_1} , T_{M_2} – машинний час, необхідний для розв’язання поставлених завдань, відповідно в старому та у новому варіантах, машино–год./рік;

$B_{\text{екс.КТЗ}}$ – експлуатаційні витрати, що доводяться на 1 год машинного часу ЕОМ, грн/ машино–год.;

E_H^a – нормативний коефіцієнт ефективності капітальних вкладень у засоби автоматизації;

$\Phi_{\text{б.КТЗ}}$ – балансова вартість ЕОМ (КТЗ), грн.;

$\Phi_{\text{еф.КТЗ}}$ – річний ефективний фонд часу роботи ЕОМ, машино–год.;

$S_{\text{рп}}$ – сумарні витрати на розробку програми, грн.;

T_c – термін служби впроваджуваної програми до її морального зношування, років. Приймаємо цей термін 5 років;

$E_{\text{зп}}$ – економія на заробітній платі.

До впровадження даного ПЗ дані функції виконувались співробітником, що відповідає за обробку рукописних символів, 1800 годин на рік, а після 0 годин на рік (економія складає 1,0 ставки одного співробітника, що відповідає за обробку рукописних символів, з заробітною платою 8 тис. гривень на місяць).

Обчислюємо:

$$E = (1800 - 0) \cdot 5,76 + 0,2 \cdot (30000 \cdot (1800 - 0)) / 1800 - 66804/5 + \\ + 8000 \cdot 1,0(1 + 0,22) \cdot 12 = 120127 \text{ грн.}$$

До Коефіцієнт ефективності капітальних вкладень наведений нижче Е (6.9):

$$E\phi = E / S_{\text{рп}}. \quad (6.9)$$

Строк окупності $T_{\text{ок}}$ обчислюється за формулою (6.10):

$$T_{\text{ок}} = S_{\text{рп}} / E. \quad (6.10)$$

Обчислюємо:

$$E\phi = 120127 / 66804 = 1,8.$$

$$T_{\text{ок}} = 66804 / 120127 = 0,56 \text{ року.}$$

Зведемо отримані техніко–економічні показники в таблицю 6.5.

Таблиця 6.5 – Техніко–економічні показники

Показники	Сума	Одиниця виміру
Витрати на розробку програми	66804	грн.
Річна економія	120127	грн.
Строк окупності програми	0,56	рік

Таким чином, впровадження в експлуатацію розробленого програмного забезпечення на виробничому підприємстві дозволить скоротити 1,0 ставки співробітника, що відповідає за обробку рукописних символів з заробітною платою 8 тис. гривень на місяць. Економічні розрахунки показали, що строк окупності програми 0,56 року, що значно менше часу її нормативного строку окупності (5 років). Крім того, річна економія при використанні програми становить 120127 грн. Виходячи з виконаних розрахунків, можна зробити висновок, що розробка даного програмного продукту є економічно доцільною.

7 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

7.1 Аналіз потенційних небезпек

У дипломній кваліфікаційній роботі магістра проводиться дослідження та програмна реалізація нейромережних методів розпізнавання рукописних символів. Під час роботи за комп'ютером, на дослідника можливий вплив наступних потенційних небезпек:

- підвищення рівню шуму, за рахунок роботи вентиляційних компонентів комп'ютерної техніки, пристроїв введення/виведення, ксероксів, принтерів, кондиціонерів та інших електричних пристроїв;
- погано організоване, захаращене робоче місце, нераціональне розміщення обладнання;
- небезпека ураження електричним струмом внаслідок порушення правил експлуатації електроприладів або їх несправності;
- підвищення рівнів електромагнітних полів;
- підвищення рівню іонізуючого випромінювання;
- підвищений рівень статичної електрики;
- підвищення або зниження температури або вологості повітря, а також його надмірна загазованість або запиленість;
- погане або недостатнє освітлення робочого місця природнім або штучним джерелом світла;
- надмірно великий об'єм інформації, що необхідно обробити за одиницю часу;
- тривале сидяче положення;
- навантаження на органи зору;
- інтелектуальне та нервово-емоційним навантаженням;
- кістково-м'язове напруження, що виникає через необхідність працівників тривалий час знаходитись в позах, що вимагають статичного напруження м'язів спини, шиї, рук, ніг, що призводить до їх стомлення [27].

7.2 Заходи із забезпечення безпеки

В офісних приміщеннях для роботи використовується велика кількість різноманітних електроприладів: принтери, ксерокси, сканери, комп'ютери, проектори, освітлювальні прилади, кондиціонери, обігрівачі, тощо, які живляться від мережі напругою 220 В. Саме тому небезпека ураження електричним струмом є однією з найбільших небезпек, яка може виникнути в разі недотримання правил користування ними.

Щоб запобігти випадкам ураження електричним струмом відповідно до Правил улаштування електроустановок (ПУЕ) [28], частини обладнання, що проводять електричний струм та можуть стати джерелами небезпеки, надійно ізольовані, а обладнання заземлене відповідно до ДСТУ 7237:2011.

Відповідно до НПАОП 0.00-4.12-05, всі працівники проходять навчання та перевірку знань з питань охорони праці, як під час прийняття на роботу так і періодично, один раз на три роки.

До роботи допускаються тільки особи, які пройшли затверджений на підприємстві курс охорони праці для роботи з комп'ютером [29].

Для захисту працівників від ураження електричним струмом проводяться організаційні та технічні заходи захисту.

Організаційні заходи включають в себе:

- призначення осіб, що будуть відповідати за організацію безпеки;
- навчання персоналу правилам безпеки та експлуатації обладнання;
- проведення інструктажів;
- використання попереджувальних плакатів і знаків (ДСТУ ISO 7010:2009).

До технічних заходів забезпечення безпеки працівників належать:

- захисне заземлення, занулення, відмикання;
- електричний поділ мережі;
- вирівнювання потенціалів (ДСТУ 2815-94);

- обмеження сили струму;
- забезпечення недоступності неізованих частин [30].

Для забезпечення безпеки працівників на підприємствах, де використовують комп'ютери забороняється:

- проводити ремонт та технічне обслуговування комп'ютера на робочому місці працівника;
- самостійно ремонтувати або намагатись виконати технічне налагодження комп'ютера без залучення компетентних спеціалістів;
- розміщувати на робочому місці зайві документи, деталі та предмети, що не потрібні для роботи;
- використовувати монітори з нечітким зображенням та монітори, у яких наявні поламки екрану;
- працювати з матричним принтером без антивібраційного покриття та зі знятою кришкою;
- експлуатація кабелів та проводів з пошкодженою ізоляцією або такою, що втратила захисні властивості в ході експлуатації;
- використання саморобних подовжувачів або нестандартного (саморобного) електронагрівального обладнання для опалення приміщення;
- користуватись пошкодженими розетками.

Також експлуатація електроустановок і електроустаткування проводиться згідно з НПАОП 40.1-1.01-97 «Правила безпечної експлуатації електроустановок» та НПАОП 40.1-1.21-98 «Правила безпечної експлуатації електроустановок споживачів», а також рекомендацій, що надавав виробник [29].

7.3 Заходи з виробничої санітарії та гігієни праці

Робочі місця офісних працівників, обладнані персональними комп'ютерами, відповідають вимогам «Правил охорони праці під час експлуатації електронно – обчислювальних машин» та «Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно – обчислювальних машин». Дані правила поширюються на всіх суб'єктів господарювання незалежно від форм власності, які у своїй діяльності здійснюють роботу, пов'язану з персональними комп'ютерами [31].

Площу приміщень, в яких розташовуються персональні комп'ютери визначають у відповідності до чинних нормативних документів. Згідно з Вимогами щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями розмір одного робочого місця обладнаного персональним комп'ютером становить не менше 6 м². Робочі місця розташовані на відстані не менше ніж 1м від стіни з вікном, а також 1,4 м від стіни, відстань між бічними поверхнями комп'ютера є не меншою за 1,2 м. Суміжні робочі місця працівників, у зв'язку з потребою особливої концентрації уваги, розділені перегородками висотою 1,5 – 2 м [32].

Робочі місця заборонено створювати у підвальних або цокольних приміщеннях будинків. У приміщеннях, де працюють з комп'ютерами, щодня здійснюється вологе прибирання, з метою запобігання запиленості підлоги та меблів. Покриття підлоги є матовим з антистатичними властивостями, а сама поверхня рівною та неслизькою.

Правильна кольорова гама також забезпечує підвищення продуктивності та психологічному комфорту. Найбільш сприятливими для нервової системи є світлі та пастельні тони – зеленувато-блакитний, ясно-сірий, золотавий. У той час як яскраві, контрастні поєднання (синій і жовтогарячий, червоний і фіолетовий) будуть викликати втому та роздратування у працівників.

За потреби та з урахуванням площі приміщення в ньому можуть бути встановлені додатково шафи, полиці, тумби чи інші системами для зберігання документів або особистих речей співробітників.

Конструкція робочого місця користувача комп'ютером повинна забезпечити йому підтримку оптимальної пози для роботи та відповідати сучасним вимогам до ергономіка.

Стіл, за яким працює спеціаліст забезпечує оптимальне розміщення всього необхідного обладнання та документів на робочій поверхні. Для цього він має висоту поверхні 680 – 800 мм, ширину 600 – 1400 мм і глибину 800 – 1000 мм. Такі параметри забезпечують можливість виконання операцій в зоні досяжності працівника.

Робочий стілець працівника відповідно до п. 4.8 ДСанПіН 3.3.2.007-98 є підйомно-поворотним, з можливістю регулювання висоти та куту нахилу сидіння та спинки. Регулювання параметрів стільця здійснюватися легко та надійно фіксується. Поверхня для сидіння та спинка крісла вироблені з напівм'якого, нековзкого та повітронепроникного покриття, що легко чиститься та не електризується. Поверхня для сидіння є плоскою з заокругленим краєм [33].

Для того, щоб знизити навантаження на органи зору між екраном комп'ютера та користувачем зберігається оптимальна відстань, що становить 600 – 700 мм, але не менше 600 мм з урахуванням літерно – цифрових знаків і символів. Також контролюється, щоб персональний комп'ютер та його комплектуючі не потрапляли під дію джерел тепла (батареї опалення, тощо) або під прямі сонячні промені.

Тривала робота за комп'ютером може негативно вплинути на організм людини. Тому щоб мінімізувати ризики важливо мати правильно спроектоване робоче місце та виконувати ряд рекомендацій, що стосуються режиму праці та відпочинку. Саме тому працівникам рекомендується кожну годину робити перерву мінімум на 15 хвилин, під час якої виходити на свіже повітря, виконувати не складні гімнастичні вправи. Під час же самої роботи

кожні 10-15 хвилин намагатись змінювати позу, рухати плечами, ногами та руками [31].

Всі приміщення обладнанні системами опалення та кондиціонування повітря, або припливно-витяжною вентиляцією, відповідно до норм та правил ДБН В.2.5-67:2013 «Опалення, вентиляція та кондиціонування». У приміщеннях забезпечується контроль параметрів мікроклімату та виконується його нормування, відповідно до встановлених санітарно-гігієнічних вимог державних санітарних норми мікроклімату виробничих приміщень

ДСН 3.3.6.042-99 Санітарні норми мікроклімату виробничих приміщень [34].

В усіх на робочих місцях працівників повинні забезпечуються оптимальні параметри мікроклімату:

- температура повітря в теплий період року становить 22–24°C, відносна вологість повітря – 40–60%, швидкість руху повітря – не вище за 0,2 м/с;
- температура повітря в холодний період року становить – 21-23°C, відносна вологість повітря – 40–60%, швидкість руху повітря – не більше 0,1 м/с.

Виконується регулярне очищення повітря, шляхом видалення забрудненого, нагрітого, іонізованого повітря з приміщення. Оскільки здатність обчислювальної техніки виділяти суттєву кількість тепла в середовище, призводить до підвищення температури та до зниження рівня вологості в приміщенні. Щоб мати змогу підтримувати необхідний об'єм чистого повітря в робочому просторі передбачено кондиціонування повітря. При недотриманні вказаних показників мікроклімату в офісних приміщеннях робочий день для робітників скорочується мінімум на 10% [35].

Для комфортної роботи співробітників кожне приміщення забезпечено елементами природного та штучного освітлення, відповідно до ДБН В. 2.5-28-2018. Природне і штучне освітлення» [36], оскільки тривала

робота за комп'ютером чи документами при поганому рівні освітленості може призвести до перенапруження органів зору

Зазвичай використовують природнє бокове одностороннє освітлення в день, а у ввечері – загальне рівномірне штучне освітлення. Робоче місце розміщене так, щоб природнє світло було з лівого боку) та уникати потрапляння прямого світла в очі. Також у працівників є можливість для корегування рівня природнього освітлення за допомогою жалюзі або штор. Оскільки яскраве світло в районі периферійного зору може збільшити навантаження на органи зору та призвести до швидкої втоми, освітлення є рівномірним, тобто ступінь освітленості приміщення та яскравість монітору комп'ютера повинні бути приблизно однаковими.

Коефіцієнт пульсації не перевищує 5 %. Рівень освітлення робочого столу в зоні розташування документів знаходиться в межах 300 – 500 лк. Освітлювальні прилади приміщення не повинні створювати відблисків на поверхні екрана, а освітлення екрана має не перевищувати 300 лк.

Підвищення рівню шуму погіршує умови праці та негативно впливає на організм людини. В результаті у робітників спостерігаються підвищений рівень стомленості, роздратованість, сильні головні болі, зниження концентрації уваги, нервово-психічне навантаження. Тривала дія інтенсивного шуму (вище 80 дБА) на слух людини призводить до його часткової або повної втрати [29].

Основними джерелами шуму на робочих місцях в офісах є принтери, кондиціонери, системи вентиляції та комп'ютери, в яких джерелами шуму є вентилятори, що охолоджують систему. Рівень шуму та вібрацій на робочих місцях осіб, що працюють за персональним комп'ютером, визначають відповідно до ДСН 3.3.6.037-99. Санитарные нормы производственного шума, ультразвука и инфразвука. Для забезпечення дотримання допустимих рівнів шуму на робочих місцях застосовуються засоби звукопоглинання, вибір яких обґрунтовується спеціальними інженерно-акустичними розрахунками [37].

Також в офісних приміщеннях дотримуються вимог, що прописані у ДСанПіН 3.3.2.007-98 та ДСанПіН 3.3.6.096-2002 щодо рівня неіонізуючих електромагнітних випромінювань, електростатичних і магнітних полів, а також підтримують в нормі рівень інтенсивності потоків інфрачервоного та ультрафіолетового випромінювання [38].

Розрахунок освітленості приміщення. Приміщення знаходиться на четвертому поверсі п'ятиповерхового цегляного будинку, що відповідає нормам ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин».

Розміри приміщення становлять: довжина $A = 9$ м, ширина $B = 6$ м, висота $H = 3,3$ м. Площа дорівнює: $S = 54$ м², а об'єм $V = 178,2$ м³.

В приміщенні наявне як природнє так і штучне освітлення, що відповідає нормам ДБН В.2.5-28:2018 «Природне і штучне освітлення». Природнє освітлення використовується в день, а у ввечері використовувати загальне рівномірне штучне освітлення.

Робоче місце розміщене таким чином, що природнє світло знаходиться з лівого боку (п. 4.3 ДСанПіН 3.3.2.007-98), що дозволяє уникнути потрапляння прямого світла в очі. Джерелом природнього освітлення в приміщенні є вікна, на яких для корегування рівня природнього світла наявні жалюзі. Світильники загального і місцевого освітлення дозволяють створюють нормальні умови освітленості і відповідний контраст між екраном і навколишнім оточенням з урахуванням виду роботи та вимог видимості з боку працівника.

Щоб перевірити чи відповідає рівень освітленості в приміщенні нормативним вимогам зазначених у ДБН В.2.5-28:2018 виконаємо розрахунок освітлення в приміщенні. Методом коефіцієнта використання світлового потоку, з урахуванням відбитих від стін, підлоги і стелі світлових потоків.

Згідно з ДБН В.2.5-28:2018 «Природне і штучне освітлення» освітлення в приміщенні нормується як $E_n = 300$ лк.

Висота робочої поверхні $h_p = 0,8$ м;

Коефіцієнти відбиття поверхонь приміщення (від: стелі – ρ_c ; стін – $\rho_{ст}$; підлоги – $\rho_{п}$) у світлому приміщенні $\rho_c=70\%$; $\rho_{ст}=50\%$; $\rho_{п}=30\%$;

Обираємо світильник типу ЛВП (IP=54 – ступінь захисту світильника, $[L/h] = 1,3$ – числове значення коефіцієнта світильника).

Коефіцієнт запасу k_z враховує зниження рівня освітленості з часом в результаті забруднення та старіння ламп, світильників і поверхонь приміщення, приймається в залежності від виробничих умов. Приймаємо $k_z = 1,4$.

Коефіцієнт нерівномірності (мінімального) освітлення z (відношення середньої освітленості до мінімальної освітленості), для люмінесцентних ламп низького тиску, як правило дорівнює $z = 1,1$.

Розрахуємо кількість рядів світильників у приміщенні за формулою (7.1):

$$N_p = B / ((H - h_p) \cdot [L/h]) = 6 / (3,3 - 0,8) \cdot 1,3 = 1,84 \approx 2. \quad (7.1)$$

Визначимо максимально припустиму відстань між рядами світильників за формулою (7.2):

$$L_{max} = B / N_p, \quad (7.2)$$

де B – ширина приміщення;

N_p – кількість рядів.

Обчислюємо:

$$L_{max} = 6 / 2 = 3 \text{ м.}$$

Розраховуємо висоту підвісу світильника над робочою поверхнею за формолою (7.3):

$$h = L_{max} / [L/h]. \quad (7.3)$$

Обчислюємо:

$$h = 3 / 1,3 = 2,3 \text{ м.}$$

Знайдемо висоту звисання світильника від стелі, формула (7.4):

$$h_3 = H - h_p - h, \quad (7.4)$$

де H – висота приміщення;

h_p – висота робочої поверхні;

h – висота підвісу світильника над робочою поверхнею.

Обчислюємо:

$$h_3 = 3,3 - 0,8 - 2,3 = 0,2 \text{ м.}$$

Чисельне значення індексу приміщення визначаємо за формулою (7.5):

$$i = A \cdot B / h(A + B), \quad (7.5)$$

де A – довжина приміщення, м;

B – ширина приміщення, м;

h – висота розміщення світильників над робочою поверхнею, м.

Обчислюємо:

$$i = 9 \cdot 6 / 2,3 \cdot (9 + 6) = 54 / 34,5 = 1,57.$$

Значення коефіцієнта використання світлового потоку η вибирається в залежності від виду джерела світла, типу обраного світильника, коефіцієнтів відбиття поверхонь приміщення ($\rho_c, \rho_{ст}, \rho_{п}$) та індексу приміщення. В нашому випадку $\eta = 0,5$.

Визначаємо сумарний світловий потік освітлювальної установки у даному виробничому приміщенні, формула (7.6):

$$\Phi_{\Sigma} = E_n \cdot S \cdot k_z \cdot z / \eta, \quad (7.6)$$

де Φ_{Σ} – розрахункове значення сумарного світлового потоку у приміщенні, лм;

E_n – нормоване значення освітленості, лк;

S – площа освітлюваної поверхні, м²;

k_z – коефіцієнт запасу;

z – коефіцієнт нерівномірності (мінімальної) освітленості;

η – коефіцієнт використання світлового потоку

Обчислюємо:

$$\Phi_{\Sigma} = 300 \cdot 9 \cdot 6 \cdot 1,4 \cdot 1,1 / 0,5 = 24948 / 0,5 = 49896 \text{ лм.}$$

Визначаємо умовну загальну кількість світильників у приміщенні, формула (7.7):

$$N^* = AB / L_{max}^2. \quad (7.7)$$

Обчислюємо:

$$N^* = 9 \cdot 6 / 9 = 6 \text{ шт.}$$

Розрахуємо світловий потік умовного джерела світла за формулою (7.8):

$$\Phi_{\text{л}}^* = \Phi_{\Sigma} / N_{\text{л}}, \text{ лм}, \quad (7.8)$$

де Φ_{Σ} – розрахункове значення сумарного світлового потоку у приміщенні, лм;

$N_{\text{л}}$ – загальна кількість ламп у приміщенні, шт;

n – кількість ламп у світильнику (в ЛВП світильнику знаходиться 2 лампи).

Обчислюємо:

$$\Phi_{\text{л}}^* = 49896 / 12 = 4158 \text{ лм}.$$

Обираємо тип стандартної лампи з найближчим значенням фактичного світлового потоку лампи $\Phi_{\text{л}}$ і знаходимо коефіцієнт m (співвідношення між розрахунковим світловим потоком лампи $\Phi_{\text{л}}^*$ та фактичним світловим потоком вибраної стандартної лампи $\Phi_{\text{л}}$):

Обираємо лампу ЛБ 58, $\Phi_{\text{л}} = 4300$ лм:

$$m = 4158 / 4300 = 0,96.$$

Визначаємо оптимальну (фактичну) кількість світильників у приміщенні, за формулою (7.9):

$$N = N_{\text{л}} \cdot m. \quad (7.9)$$

Обчислюємо:

$$N = 6 \cdot 0,96 = 6 \text{ шт}.$$

Фактична кількість ламп у приміщенні, формула (7.10):

$$N_{\text{фл}} = N_{\text{ф}} \cdot n. \quad (7.10)$$

Обчислюємо:

$$N_{\text{фл}} = 6 \cdot 2 \text{ шт} = 12 \text{ шт.}$$

Визначаємо загальну розрахункову освітленість E_p у приміщенні, що створюється при застосуванні стандартних ламп, за формулою (7.11):

$$E_p = \frac{\Phi_{\text{л}} \cdot N_{\text{фл}} \cdot \eta}{Sk_z}, \text{ лк.} \quad (7.11)$$

Обчислюємо:

$$E_p = 4300 \cdot 12 \cdot 0,5 / 9 \cdot 6 \cdot 1,4 \cdot 1,1 = 25800 / 83,16 = 310,2 \text{ лк.}$$

Тобто виконується умова $E_p = (-10\% \dots +20\%) \cdot E_n$, лк.

Отриманий результат порівнюємо з ДБН В.2.5-28:2018 й робимо висновок, що рівень штучного освітлення у приміщенні є достатнім.

7.4 Заходи безпеки у надзвичайних ситуаціях

7.4.1 Заходи з пожежної безпеки

Заходи з протипожежної безпеки розроблені для виробничого приміщення обладнаного ПК з ВДТ розроблені згідно вимог НАПБ А.01.001-2014 «Правила пожежної безпеки в Україні» [39].

Виходячи з аналізу речовин та матеріалів, які використовуються при роботі у приміщенні, обладнаному ПК з ВДТ:

- згідно ДБН В.1.1-7:2016 «Пожежна безпека об'єктів будівництва. Загальні вимоги» [40] у приміщенні, обладнаному ПК з ВДТ можлива пожежа класу – А та Е; воно належить до категорії «Д» з пожежної небезпеки – простір у приміщенні, у якому перебувають тверді горючі речовини та матеріали.

Оскільки приміщення належить до категорії «Д» з пожежної небезпеки, тому відповідно до вимог ДБН В.1.1-7:2016 «Пожежна безпека об'єктів будівництва. Загальні вимоги» [40] воно має II ступінь вогнестійкості.

У разі виникнення пожежі у приміщенні обладнаному ПК з ВДТ для евакуації персоналу відповідно до вимог ДБН В.1.1-7:2016 «Пожежна безпека об'єктів будівництва. Загальні вимоги» передбачені виходи, по обидві сторони приміщення, з одного боку вікно (на пожежні сходи), а з іншого – вхідні двері. Згідно СНиП 2.09.02-85* «Производственные здания», відстань від найбільш віддаленого робочого місця до найближчого евакуаційного виходу не обмежується.

Обладнання, силові та освітленні мережі приміщення, обладнаного ПК з ВДТ відповідають вимогам пожежної безпеки, оскільки виконані відповідно до вимог НПАОП 40.1-1.32-01 «Правила будови електроустановок. Електрообладнання спеціальних установок» [41], та мають ступінь захисту ізоляції обладнання IP44 яка відповідає класу пожежанебезпечної зони П-Па до якої належить приміщення.

Для виявлення початкової стадії загоряння, швидкого й точного оповіщення служби пожежної охорони про час і місце виникнення пожежі встановлені системи автоматичної пожежної сигналізації (АПС). Вони можуть самостійно пускати в хід установки пожежогасіння, коли пожежа не досягла великих розмірів. Системи АПС складаються з пожежних сповіщувачів, ліній зв'язку й прийомних пультів (станцій). Згідно вимог ДБН В.2.5-56:2014 «Системи протипожежного захисту», в приміщенні обладнаному ПК з ВДТ встановлена система пожежної й охоронної сигналізації «Фотон-ОФІС», яка забезпечує виявлення теплових і димових ознак пожежі і місця виникнення пожежі з точністю до місця розміщення датчика [29].

Оскільки приміщення що обладнане ПК з ВДТ має площу 54 м², тому відповідно до вимог п. 3.8 розділу «Типові норми належності вогнегасників» ДСТУ 4297:2004 «Пожежна техніка. Технічне обслуговування вогнегасників.

Загальні технічні вимоги» [42] для гасіння електроустановок, що знаходяться під напругою, передбачені вуглекислотні вогнегасники типу ВВК-3,5 у кількості 3 штуки (з розрахунку один вогнегасник с величиною заряду вогнегасної речовини 3 кг і більше, на 20 м² площі приміщення). Відстань між вогнегасниками та місцями можливих загорянь не перевищує 10 м.

7.4.2 Заходи з цивільного захисту

Укриття населення в захисних спорудах є комплексом заходів із їх завчасним будівництвом, обладнанням приміщень та підтримкою їх у готовності до використання. Вони здатні захистити населення від наслідків дії ядерної, хімічної, бактеріологічної та звичайної зброї, від аварій та катастроф, деяких стихійних лих, військових дій або терористичних актів.

За своїм призначенням та захисними властивостями захисні споруди поділяються на сховища, протирадіаційні укриття та найпростіші укриття.

Сховища є надійним захистом від небезпечних чинників, що виникають в результаті надзвичайних ситуацій. Вони представляють собою комплекс інженерних споруд, що забезпечують необхідні умови життєдіяльності протягом певного часу.

За захисними властивостями від дії вибухової сили сховища поділяються на п'ять класів. За часом побудови на: побудовані завчасно і швидко споруджені. За наявністю фільтровентиляційного обладнання бувають промислового виготовлення та спрощеного (виготовлені з підручних матеріалів). За місцем заходження: вбудовані, ті що знаходяться в підвальних приміщеннях будинків, та відокремлені, які знаходяться поза будинками. Зазвичай їх будують заздалегідь та підтримують у належному стані, однак вони також можуть будуватись і в період загрози нападу або під час воєнних дій (швидко зведені). Сховища також класифікують за місткістю: малі здатні вміщувати 150-300 чоловік, середні – 300-600 чол. та великі – понад 600 чол.

Сховища мають забезпечувати надійний захист людей від усіх уражаючих факторів та їх безперервне перебування в них не менше двох діб. Вони розташовуються на місцевості, що знаходиться на відстані від ліній водостоку та каналізації, щоб уникнути їх затоплення. Всі входи та виходи мають бути з тим же ступенем захисту, що й основні приміщення, а на випадок їх завалу, повинні бути передбачені аварійні виходи. Висота основних приміщень укриття не менше 2,2 м, а рівень підлоги розташований вище ґрунтових води не менш як на 20 см.

Протирадіаційними укриття дозволяють захистити людей від опромінення та впливу радіоактивних речовин в зонах радіаційного забруднення, від біологічних засобів у краплинно-рідинному вигляді та світлового випромінювання ядерного вибуху. Вони можуть створюватись у підвальних поверхах будинків та споруд.

При їх виборі та підготовці обов'язково враховують захисні властивості будівельних матеріалів та окремих конструкцій, оскільки певна щільність та товщина шару матеріалу здатні ослаблювати та зменшити потік радіоактивних речовин. Що дозволяє людині, що знаходиться в приміщенні отримати меншу дозу опромінення. Так підвали в дерев'яних одноповерхових будинках здатні ослабити дозу радіації в 7 разів, а в житлових одноповерхових кам'яних (цегляних) будинках – у 40, у двоповерхових – у 100, середня частина підвалу кількоповерхового кам'яного будинку – у 800–1000 разів. У випадку коли рівень радіації не високий для захисту від бактеріальних засобів, парів отруйних і сильнодіючих ядучих речовин можна використовувати кам'яні (цегляні) або дерев'яні будівлі.

Сховища та протирадіаційні укриття складаються з основних та допоміжних приміщень. Основними є приміщення для розміщення людей, пункти управління, медичні пункти, а до допоміжних відносяться приміщення, де розташовані фільтровентиляційне обладнання, системами водопостачання, опалення, санітарні вузли, приміщення для зберігання продовольства, тамбур-шлюзи (тамбури).

Найпростіші укриття для захисту або щілини є захисними спорудами, які можуть бути створеними населенням у найкоротший проміжок часу. Такі укриття здатні певною мірою захистити від ударної хвилі, радіаційного ураження та світлового випромінювання. Для будівництва щілин вибирають сухі місця, що знаходяться на підвищенні.

Найпростіші укриття типу щілини бувають двох видів відкритими та перекритими. Відкриті щілини дозволяють зменшити в 1,2–2 рази ймовірність ураження ударною хвилею, проникаючою радіацією та світловим випромінюванням. Перекриті щілини є більш надійним захистом населення, оскільки вони здатні повністю захистити від світлового випромінювання, від ударної хвилі – у 1,5–3 рази, від проникаючої радіації і радіоактивного випромінювання – у 200–300 разів, а також надійно захищає від осколкових і кулькових бомб, від запалювальних засобів.

Будівництво укриття починається з розмітки та визначення її плану на місцевості. Щілини роблять у вигляді прямолінійних ділянок з такими розмірами: довжина до 15 м, глибина 1,8–2,0 м; ширина: зверху – 1,1–1,3 м, а на дні – 0,8 м. Стіни укриття укріплюють дошками, хмизом, жердинами або іншим підручним матеріалом. Перекриття щілини роблять з накату колод, за яким йде шар глини товщиною 10–15 см, що дозволяє захистити укриття від потрапляння дощових вод. Потім шар ґрунту 20–40 см, а зверху все вкривають дерном. Також такі укриття можна будувати зі збірних залізобетонних конструкцій. Входи до щілини роблять східчасті, під прямим кутом до осі укриття. У перекритій щілині вхід може бути з однієї або двох сторін з дверима та тамбуром. В нормі найпростіші укриття типу щілина повинна забезпечити захист від 10 до 15 чоловік [43].

ВИСНОВКИ

У ході виконання дипломної кваліфікаційної роботи магістра було розв'язано актуальне завдання нейромережних методів розпізнавання рукописних символів. Проаналізовано процес розпізнавання образів. Надано постановку задачі розпізнавання образів на основі нейромережних моделей. Проаналізовано нейромережні моделі для розпізнавання образів. Визначено, що для розпізнавання рукописних символів доцільно використовувати згорткові нейронні мережі.

Проаналізовано методи навчання нейромереж для розпізнавання образів. Визначено, що для синтезу нейронних мереж широкого поширення набули метод зворотного поширення помилки та генетичні алгоритми.

Досліджено програмні засоби синтезу розпізнавальних моделей на основі нейронних мереж глибокого навчання. Визначено, що наявність недоліків існуючого програмного забезпечення обумовлює доцільність розробки власного програмного забезпечення для синтезу нейромережних моделей для розпізнавання рукописних символів. Досліджено та проаналізовано вимоги до програмного продукту. Для розробки програмного забезпечення обрано мову C#, яка є об'єктно-орієнтованою мовою програмування з безпечною системою типізації для платформи .NET. В якості інструментарію розробки обрано Microsoft Visual Studio, що включає в себе інтегроване середовище розробки програмного забезпечення та ряд інших інструментальних засобів, та є досить зручним для розробки програмного забезпечення та його модифікації.

Розроблено структурну схему програми. Визначено, що основними модулями розробленого програмного забезпечення є такі: модуль роботи з даними, модуль управління, модуль реалізації загорткової нейронної мережі; модуль навчання НМ за допомогою генетичного алгоритму, модуль навчання НМ за допомогою метода зворотного поширення помилки.

Наукова новизна роботи полягає у тому, що розроблено гібридний метод синтезу нейромережних моделей на основі генетичного пошуку та зворотного поширення помилки, в якому генетичний алгоритм використовується для навчання ваг у фільтрах мережі, а в якості початкових значень вагових коефіцієнтів використовуються значення, отримання за допомогою методу зворотного поширення помилки, що дозволяє уникнути зациклення в локальних мінімумах і підвищити точність розпізнавання за допомогою синтезованих нейромоделей.

Практичне значення роботи полягає у тому, що розроблено програмне забезпечення, що реалізує нейромережні методи розпізнавання рукописних символів.

Виконано тестування розробленого програмного забезпечення, що реалізує нейромережні методи розпізнавання рукописних символів. Результати тестування підтвердили, що програма може використовуватися за призначенням.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Хайкин С. Нейронные сети: полный курс / С. Хайкин; пер. с англ. – М.: Вильямс, 2006. – 1104 с.
2. Нейронні мережі : навчальний посібник / С. О. Субботін, А. О. Олійник; під заг. ред. проф. С. О. Субботіна. – Запоріжжя : ЗНТУ, 2013. – 125 с.
3. Генетические алгоритмы: Учебное пособие / Л. А. Гладков, В. В. Курейчик, В. М. Курейчик – М: Физматлит, 2006. – 320 с.
4. Применение нейросетей в распознавании изображений [Электрон. ресурс]. – Режим доступа: <https://geektimes.ru/post/74326/>.
5. Что такое свёрточная нейронная сеть [Электрон. ресурс] – Режим доступа: <https://habrahabr.ru/post/309508/>.
6. Encyclopedia of artificial intelligence / Eds.: J. R. Dorico, J. D. de la Calle, A. P. Sierra. – New York : Information Science Reference, 2009. – Vol. 1-3. – 1677 p.
7. Субботін С. О. Подання й обробка знань у системах штучного інтелекту та підтримки прийняття рішень: Навчальний посібник. / С. О. Субботін. – Запоріжжя: ЗНТУ, 2008. – 341 с.
8. Субботін С. О. Інтелектуальні системи : навчальний посібник / С. О. Субботін, А. О. Олійник ; під заг. ред. проф. С. О. Субботіна. – Запоріжжя : ЗНТУ, 2013. – 216 с.
9. Субботін С. О. Ітеративні, еволюційні та мультиагентні методи синтезу нечіткологічних і нейромережних моделей: монографія / С. О. Субботін, А. О. Олійник, О. О. Олійник ; під заг. ред. С. О. Субботіна. – Запоріжжя : ЗНТУ, 2009. – 375 с.
10. Олійник А. О. Еволюційні обчислення та програмування: Навчальний посібник / А. О. Олійник, С. О. Субботін, О. О. Олійник. – Запоріжжя : ЗНТУ, 2010. – 324 с.

11. Прогрессивные технологии моделирования, оптимизации и интеллектуальной автоматизации этапов жизненного цикла авиадвигателей : Монография / А. В. Богуслаев, Ал. А. Олейник, Ан. А. Олейник и др. ; под ред. Д. В. Павленко, С. А. Субботина. – Запорожье : ОАО «Мотор Сич», 2009. – 468 с.
12. Рассел С. Искусственный интеллект: современный подход / С. Рассел, П. Норвиг. – М.: Вильямс, 2006. – 1408 с.
13. Субботин С. А. Метод формирования баз знаний для нейронечетких моделей / Субботін С. О. // Нейроинформатика и ее приложения: Материалы XIV Всероссийского семинара, 6–8 октября 2006 г. / Под ред. А. Н. Горбаня, Е. М. Миркеса. Отв. За выпуск Г. М. Садовская. – Красноярск: ИВМ СОРАН, 2006. – С. 116–118.
14. WC3 [Электрон. ресурс]. – Режим доступа: http://wc3-maps.ru/publ/sozдание_kart/3d_modelirovanie_uchebniki/23.
15. Зайченко Ю. П. Основы проектування інтелектуальних систем: навчальний посібник / Ю. П. Зайченко. – К.: Слово, 2004. – 352 с.
16. Люгер Дж. Ф. Искусственный интеллект: стратегии и методы решения сложных проблем / Дж. Ф. Люгер. – М. : Вильямс, 2005. – 864 с.
17. Олійник А. О. Інтелектуальний аналіз даних: навчальний посібник / А. О. Олійник, С. О. Субботін, О. О. Олійник. – Запоріжжя : ЗНТУ, 2011. – 271 с.
18. Барсегян А. А. Технологии анализа данных: Data Mining, Visual Mining, Text Mining, OLAP: Уч. пос. / А. А. Барсегян. – СПб : BHV, 2007. – 384 с.
19. Брянецев И. Н. Data Mining. Теория и практика / И. Н. Брянецев. – М. : БДЦ-Пресс, 2006. – 208 с.
20. Интеллектуальные информационные технологии проектирования автоматизированных систем диагностирования и распознавания образов : монография / [С. А. Субботин, Ан. А. Олейник, Е. А. Гофман, С. А. Зайцев,

Ал. А. Олейник]; под ред. С. А. Субботина. – Харьков: Компания СМІТ, 2012. – 318 с.

21. Bentley J. L. Writing Efficient Programs (Prentice-Hall Software Series) [Text] / J. L. Bentley. – Englewood Cliffs, NJ : Prentice Hall, 1982. – 170 p.

22. Рідкокаша А. А. Основи систем штучного інтелекту : . навчальний посібник / А. А. Рідкокаша, К. К. Голдер. – Черкаси, "Відлуння–Плюс", 2002. – 240 с.

23. Кнут Д. Искусство программирования [Текст] / Д. Кнут. – 3-е изд. – М.: Вильямс. – Т. 2, 2000 – 832 с.

24. C# мова програмування [Електрон. ресурс]. – 2020. – Режим доступу: https://uk.wikipedia.org/wiki/C_Sharp.

25. C# плюсы и минусы [Электрон. ресурс]. – 2020. – Режим доступа: <https://habrahabr.ru/post/262461/>.

26. Экономика и организация производства программных продуктов [Электрон. ресурс]. – 2019. – Режим доступа: <http://megaobzor.net/e-book/book.html>.

27. Депутат О. П. Цивільна оборона. Навчальний посібник. / О. П. Депутат, І. В. Коваленко, І. С. Мужик. – Львів.: Афіша, 2001. – 336 с.

28. Правила улаштування електроустановок [Текст]. Міненерговугілля України, 2017. – 617с.

29. Атаманчук П. Охорона праці в галузі. Навчальний посібник / П. Атаманчук, В. Мендерецький, О. Панчук, Р. Билык. К: Центр навчальної літератури, 2017. – 322 с.

30. Електробезпека в будівлях і спорудах. Вимоги до захисних заходів від ураження електричним струмом [Текст] : ДСТУ Б В.2.5-82:2016. – На заміну ДБН В.2.5-27-2006 ; чинний від 2017-04-01. – К. : ДП «УкрНДНЦ», 2016. – 109 с.

31. Вимоги охорони праці при виконанні робіт на персональному комп'ютері [Електрон. ресурс] – Режим доступу: <https://selidovo->

rada.gov.ua/novini/vimogi-okhoroni-pratsi-pri-vikonanni-robit-na-personalnomu-komp-yuteri.

32. Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроям [Електрон. ресурс] : НПАОП 0.00-7.15-18. – На заміну НПАОП 0.00-1.28-10 ; чинний від 2018-05-18. – К. : Мінсоцполітики України, 2018. – 6 с. – Режим доступу: <http://zakon.rada.gov.ua/laws/show/z0508-18>.

33. Курс лекцій по «Эргономике». Ю. М. Мадиевский. – Под ред. д-ра. техн. наук., проф. Березуцкого В. В. – Х.: НТУ «ХПИ» , 2015. – 294 с.

34. ДСН 3.3.6.042-99 Санітарні норми мікроклімату виробничих приміщень. / Харків: Форт, 2000 – 24 с.

35. Охорона праці офісних працівників: організація та обладнання робочого місця [Електрон. ресурс] – Режим доступу: <http://oppb.com.ua/news/ohorona-praci-ofisnyh-pracivnykiv-organizaciya-ta-obladnannya-robochogo-miscya>.

36. ДБН В.2.5-28:2018. Природне та штучне освітлення. Посібник: проблеми природного і штучного освітлення. / К: Етін, 2019. – 180 с.

37. ДСН 3.3.6.037-99. Санитарные нормы производственного шума, ультразвука и инфразвука. – К. : МОЗ Украины. – 1999. – 98 с.

38. Робота в офісі: основні санітарно-гігієнічні вимоги [Електрон. ресурс]. – 2018. – Режим доступу: <https://te.dsp.gov.ua/robota-v-ofisi-osnovni-sanitarno-gigiyenichni-vymogy/>.

39. Правила пожежної безпеки в Україні. 2015 / Харків : Форт, 2015 – 124 с.

40. ДБН В.1.1-7:2016 «Пожежна безпека об'єктів будівництва. Загальні вимоги». / Київ: Міністерство регіонального розвитку, будівництва та житлово-комунального господарства України, 2017. – 39 с.

41. НПАОП 40.1-1.32-01 «Правила будови електроустановок. Електрообладнання спеціальних установок». / Харків: Індустрія, 2010. – 118 с.

42. ДСТУ 4297:2004 «Пожежна техніка. Технічне обслуговування вогнегасників. Загальні технічні вимоги» / Харків: Форт, 2004 – 48 с.

43. Цивільний захист: Конспект лекцій для студентів всіх спеціальностей і форм навчання / В. О. Васійчук, В. Є. Гончарук, О. С. Дацько та др. / За ред. В. Є. Гончарука. – Л: ВЛп, 2011. – 208 с.

ДОДАТОК А
Технічне завдання

Вступ

Програма призначена для розпізнавання рукописних символів.

A.1 Підстави для розробки

Підставою для розробки є завдання на дипломну роботу на тему «Дослідження та програмна реалізація нейромережних методів розпізнавання рукописних символів», затверджене наказом по Національному університету «Запорізька політехніка».

A.2 Призначення розробки

Програма призначена для розпізнавання рукописних символів на основі нейромережних методів.

A.3 Вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Програма має виконувати такі функції:

- зчитування вибірок даних;
- додавання нових значень у вибірку;
- навчання системи за вибіркою (синтез нейромережної моделі для розпізнавання образів на основі даних навчальної вибірки);
- збереження і завантаження результатів навчання;
- розпізнавання введених користувачем символів.

Інтерфейс користувача повинен забезпечити легке орієнтування у навігаційному меню, а також легке отримання знань о кількості зорів'я та набоїв у головного персонажу.

А.3.2 Вимоги до надійності

Програма повинна коректно працювати, перевіряти чи відповідають вхідні дані потрібному формату. У разі невідповідності вхідних даних видавати повідомлення про це.

А.3.3 Вимоги до експлуатації

Програма може бути записана на диск, та на флеш носії. Експлуатація програмного продукту здійснюється відповідно до експлуатаційної документації на розроблену програму, що містить інформацію, необхідну для освоєння та експлуатації програмного продукту.

А.3.4 Вимоги до складу й параметрів технічних засобів

Для роботи з програмним продуктом рекомендується використовувати наступне апаратне забезпечення:

- ПК під керівництвом операційної системи та встановленим .Net Framework версії 4.5 та вище;
- тактова частота процесора 1 ГГц або більше;
- обсяг оперативної пам'яті не менше 1 Гб;
- обсяг вільного простору на диску – не менше 1 Гб;
- дозвіл екрана монітора не менше 1024x768;
- маніпулятор миша.

А.3.5 Вимоги до інформаційної й програмної сумісності

Для роботи програми необхідна операційна система.

A.3.6 Вимоги до транспортування і зберігання

Програма повинна зберігатися на диску або на флеш носії і транспортуватися в спеціальних боксах.

A.3.7 Вимоги до програмної документації

Програма повинна поставлятися з технічним завданням, керівництвом оператора, керівництвом програміста.

A.4 Стадії та етапи розробки

Програмний продукт повинен бути виконаний у 16-ти тижневий строк з моменту отримання завдання.

ДОДАТОК Б
Опис програми

Б.1 Загальні відомості

Програма розроблена за допомогою мови програмування C#.

Б.2 Функціональне призначення

Програма призначена для розпізнавання рукописних символів.

Б.3 Опис логічної структури

Програму розроблено на основі модульної структури. Основними модулями розробленого програмного забезпечення є такі: модуль роботи з даними, модуль управління, модуль реалізації загорткової нейронної мережі; модуль навчання НМ за допомогою генетичного алгоритму, модуль навчання НМ за допомогою метода зворотного поширення помилки.

Б.4 Використані технічні засоби

Для роботи з програмою рекомендується використовувати наступне апаратне забезпечення:

- ПК під керівництвом операційної системи та встановленим .Net Framework версії 4.5 та вище;
- тактова частота процесора 1 ГГц або більше;
- обсяг оперативної пам'яті не менше 1 Гб;
- обсяг вільного простору на диску – не менше 1 Гб;
- дозвіл екрана монітора не менше 1024x768;
- маніпулятор миша.

Б.5 Виклик і завантаження

Запуск програми відбувається за допомогою файлу DigitRecognize.exe. Час завантаження програми – менше однієї секунди. Інших характеристик не визначено.

Б.6 Вхідні дані

Вхідними даними для програми є вибірка даних з прикладами рукописних цифр, вагові коефіцієнти нейронної мережі, а також введені користувачем цифри для розпізнання.

Б.7 Вихідні дані

Вихідними даними для програми є вагові коефіцієнти нейронної мережі, результати тестування та результати розпізнання введеної користувачем цифри.

ДОДАТОК В
Текст програми

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace MNIST_CNN {
    public partial class Form1 : Form {

        private const int CONVOLUTION_SIZE = 5;
        private const int FEATURE_SIZE = 8;
        private const int POOL_SIZE = 2;

        private const int WIDTH = 28;
        private const int HEIGHT = 28;

        private const int CONV_WIDTH_1 = WIDTH - CONVOLUTION_SIZE + 1;
        private const int CONV_HEIGHT_1 = HEIGHT - CONVOLUTION_SIZE + 1;

        private const int POOLED_WIDTH_1 = CONV_WIDTH_1 / 2;
        private const int POOLED_HEIGHT_1 = CONV_HEIGHT_1 / 2;

        private const int CONV_WIDTH_2 = POOLED_WIDTH_1 - CONVOLUTION_SIZE + 1;
        private const int CONV_HEIGHT_2 = POOLED_HEIGHT_1 - CONVOLUTION_SIZE + 1;

        private const int POOLED_WIDTH_2 = CONV_WIDTH_2 / 2;
        private const int POOLED_HEIGHT_2 = CONV_HEIGHT_2 / 2;

        private const int HIDDEN_SIZE = 15;
        private const int OUTPUT_SIZE = 10;

        private Random rand = new Random();
        private double[, ,] cw1 = new double[FEATURE_SIZE, CONVOLUTION_SIZE,
CONVOLUTION_SIZE];
        private double[, , ,] cw2 = new double[FEATURE_SIZE, FEATURE_SIZE,
CONVOLUTION_SIZE, CONVOLUTION_SIZE];
        private double[] cb1 = new double[FEATURE_SIZE];

```

```

private double[] cb2 = new double[FEATURE_SIZE];

private double[,,,] w1 = new double[FEATURE_SIZE, POOLED_WIDTH_2,
POOLED_HEIGHT_2, HIDDEN_SIZE];
private double[,] w2 = new double[HIDDEN_SIZE, OUTPUT_SIZE];

private double[] b1 = new double[HIDDEN_SIZE];
private double[] b2 = new double[OUTPUT_SIZE];

private double[,] train;
private double[,] test;

private int[] trainAnswer;
private int[] testAnswer;

private double BestGA;
private double correct = 0;
private double total = 0;
private double cost;
private double iterations = 0;
private Queue<double> costList = new Queue<double>();
private Queue<bool> correctList = new Queue<bool>();

private bool training = false;
private bool IsInit = false;
private bool Isloaded = false;

public Form1 () {
    InitializeComponent();
}

double[,] BmpToArray(Bitmap bmp)
{
    double[,] temp = new double[bmp.Width, bmp.Height];
    for (int i = 0; i < bmp.Width; i++)
        for (int j = 0; j < bmp.Height; j++)
            temp[i, j] = 255 - bmp.GetPixel(i, j).B;
    int HS = -1;
    int HE = -1;
    int WS = -1;
    int WE = -1;
    for (int i = 0; i < bmp.Width; i++)

```

```

{
    for (int j = 0; j < bmp.Height; j++)
        if (temp[i, j] != 0) WS = i;
    if (WS != -1) break;
}
for (int i = bmp.Width - 1; i > -1; i--)
{
    for (int j = 0; j < bmp.Height; j++)
        if (temp[i, j] != 0) WE = i;
    if (WE != -1) break;
}
for (int i = 0; i < bmp.Height; i++)
{
    for (int j = 0; j < bmp.Width; j++)
        if (temp[j, i] != 0) HS = i;
    if (HS != -1) break;
}
for (int i = bmp.Height - 1; i > -1; i--)
{
    for (int j = 0; j < bmp.Width; j++)
        if (temp[j, i] != 0) HE = i;
    if (HE != -1) break;
}
int W = WE - WS;
int H = HE - HS;
int max;
if (W > H) { max = W; HS = ((HE + HS) / 2) - W / 2; HE = HS + W; } else
{ max = H; WS = ((WE + WS) / 2) - H / 2; WE = WS + H; }
if (WS < 0) WS = 0;
if (HS < 0) HS = 0;
if (HE > 159) { HS -= HE; HS += 159; HE = 159; }
if (WE > 159) { WS -= WE; WS += 159; WE = 159; }
double SCoeff = (double)max / 28.0;
int tmp1;
int tmp2;
int ltmp1;
int ltmp2;
int count;
double[, ] res = new double[28, 28];
double sum = 0;
for (int i = 0; i < 28; i++)
    for (int j = 0; j < 28; j++)
    {

```

```

        sum = 0;
        ltmp1 = (int)(SCoef * i);
        ltmp2 = (int)(SCoef * j);
        tmp1 = (int)(SCoef * (i + 1));
        tmp2 = (int)(SCoef * (j + 1));
        count = 0;
        for (int m = WS + ltmp1; m < WS + tmp1; m++)
            for (int n = HS + ltmp2; n < HS + tmp2; n++)
            {
                if (m < 159 && n < 159)
                {
                    count++;
                    sum += temp[m, n];
                }
            }

        {
            res[i, j] = (sum / (double)count) / 128.0 - 1.0;
        }
    }
    return res;
}

class Unit
{
    public double Value;
    public double[, ,] cw1;
    public double[, , ,] cw2;
    public double[] cb1;
    public double[] cb2;

    public double[, , ,] w1;
    public double[, ] w2;

    public double[] b1;
    public double[] b2;
    public Unit()
    {
        cw1 = new double[FEATURE_SIZE, CONVOLUTION_SIZE,
CONVOLUTION_SIZE];
        cw2 = new double[FEATURE_SIZE, FEATURE_SIZE, CONVOLUTION_SIZE,
CONVOLUTION_SIZE];
    }
}

```

```

        cb1 = new double[FEATURE_SIZE];
        cb2 = new double[FEATURE_SIZE];
        w1 = new double[FEATURE_SIZE, POOLED_WIDTH_2, POOLED_HEIGHT_2,
HIDDEN_SIZE];

        w2 = new double[HIDDEN_SIZE, OUTPUT_SIZE];
        b1 = new double[HIDDEN_SIZE];
        b2 = new double[OUTPUT_SIZE];

    }
    public Unit(Form1 f, bool b)
    {

        cw1 = new double[FEATURE_SIZE, CONVOLUTION_SIZE,
CONVOLUTION_SIZE];

        cw2 = new double[FEATURE_SIZE, FEATURE_SIZE, CONVOLUTION_SIZE,
CONVOLUTION_SIZE];

        cb1 = new double[FEATURE_SIZE];
        cb2 = new double[FEATURE_SIZE];
        w1 = new double[FEATURE_SIZE, POOLED_WIDTH_2, POOLED_HEIGHT_2,
HIDDEN_SIZE];

        w2 = new double[HIDDEN_SIZE, OUTPUT_SIZE];
        b1 = new double[HIDDEN_SIZE];
        b2 = new double[OUTPUT_SIZE];
        double range1 = 1 / Math.Sqrt(CONVOLUTION_SIZE * CONVOLUTION_SIZE);
        double range2 = 1 / Math.Sqrt(FEATURE_SIZE * CONVOLUTION_SIZE *
CONVOLUTION_SIZE);

        double range3 = 1 / Math.Sqrt(FEATURE_SIZE * POOLED_WIDTH_2 *
POOLED_HEIGHT_2);

        double range4 = 1 / Math.Sqrt(HIDDEN_SIZE);

        for (int i = 0; i < FEATURE_SIZE; i++)
            for (int j = 0; j < CONVOLUTION_SIZE; j++)
                for (int k = 0; k < CONVOLUTION_SIZE; k++)
                    cw1[i, j, k] = f.cw1[i, j, k];

        for (int i = 0; i < FEATURE_SIZE; i++)
            for (int j = 0; j < FEATURE_SIZE; j++)
                for (int k = 0; k < CONVOLUTION_SIZE; k++)
                    for (int l = 0; l < CONVOLUTION_SIZE; l++)
                        cw2[i, j, k, l] = f.cw2[i, j, k, l];

        for (int i = 0; i < FEATURE_SIZE; i++)

```

```

        for (int j = 0; j < POOLED_WIDTH_2; j++)
            for (int k = 0; k < POOLED_HEIGHT_2; k++)
                for (int l = 0; l < HIDDEN_SIZE; l++)
                    w1[i, j, k, l] = f.w1[i, j, k, l];

    for (int i = 0; i < HIDDEN_SIZE; i++)
        for (int j = 0; j < OUTPUT_SIZE; j++)
            w2[i, j] = f.w2[i, j];

    for (int i = 0; i < HIDDEN_SIZE; i++)
        b1[i] = f.b1[i];

    for (int i = 0; i < OUTPUT_SIZE; i++)
        b2[i] = f.b2[i];

    for (int i = 0; i < FEATURE_SIZE; i++)
    {
        cb1[i] = f.cb1[i];
        cb2[i] = f.cb2[i];
    }
}

public Unit(Form1 f)
{
    cw1      =      new      double[FEATURE_SIZE,      CONVOLUTION_SIZE,
CONVOLUTION_SIZE];

    cw2  =  new  double[FEATURE_SIZE,  FEATURE_SIZE,  CONVOLUTION_SIZE,
CONVOLUTION_SIZE];

    cb1 = new double[FEATURE_SIZE];
    cb2 = new double[FEATURE_SIZE];
    w1  = new double[FEATURE_SIZE,  POOLED_WIDTH_2,  POOLED_HEIGHT_2,
HIDDEN_SIZE];

    w2 = new double[HIDDEN_SIZE, OUTPUT_SIZE];
    b1 = new double[HIDDEN_SIZE];
    b2 = new double[OUTPUT_SIZE];
    double range1 = 1 / Math.Sqrt(CONVOLUTION_SIZE * CONVOLUTION_SIZE);
    double range2 = 1 / Math.Sqrt(FEATURE_SIZE * CONVOLUTION_SIZE *
CONVOLUTION_SIZE);

    double range3 = 1 / Math.Sqrt(FEATURE_SIZE * POOLED_WIDTH_2 *
POOLED_HEIGHT_2);

    double range4 = 1 / Math.Sqrt(HIDDEN_SIZE);

```

```

        for (int i = 0; i < FEATURE_SIZE; i++)
            for (int j = 0; j < CONVOLUTION_SIZE; j++)
                for (int k = 0; k < CONVOLUTION_SIZE; k++)
                    cw1[i, j, k] = f.GetRandomGaussian(range1);

        for (int i = 0; i < FEATURE_SIZE; i++)
            for (int j = 0; j < FEATURE_SIZE; j++)
                for (int k = 0; k < CONVOLUTION_SIZE; k++)
                    for (int l = 0; l < CONVOLUTION_SIZE; l++)
                        cw2[i, j, k, l] = f.GetRandomGaussian(range2);

        for (int i = 0; i < FEATURE_SIZE; i++)
            for (int j = 0; j < POOLED_WIDTH_2; j++)
                for (int k = 0; k < POOLED_HEIGHT_2; k++)
                    for (int l = 0; l < HIDDEN_SIZE; l++)
                        w1[i, j, k, l] = f.GetRandomGaussian(range3);

        for (int i = 0; i < HIDDEN_SIZE; i++)
            for (int j = 0; j < OUTPUT_SIZE; j++)
                w2[i, j] = f.GetRandomGaussian(range4);

        for (int i = 0; i < HIDDEN_SIZE; i++)
            b1[i] = 0;

        for (int i = 0; i < OUTPUT_SIZE; i++)
            b2[i] = 0;

        for (int i = 0; i < FEATURE_SIZE; i++)
            cb1[i] = cb2[i] = 0;
    }

    public Unit(Unit u)
    {
        cw1 = new double[FEATURE_SIZE, CONVOLUTION_SIZE,
CONVOLUTION_SIZE];

        cw2 = new double[FEATURE_SIZE, FEATURE_SIZE, CONVOLUTION_SIZE,
CONVOLUTION_SIZE];

        cb1 = new double[FEATURE_SIZE];
        cb2 = new double[FEATURE_SIZE];
        w1 = new double[FEATURE_SIZE, POOLED_WIDTH_2, POOLED_HEIGHT_2,
HIDDEN_SIZE];

        w2 = new double[HIDDEN_SIZE, OUTPUT_SIZE];
        b1 = new double[HIDDEN_SIZE];

```



```

    for (int i = 0; i < FEATURE_SIZE; i++)
    {
        f.cb1[i] = cb1[i];
        f.cb2[i] = cb2[i];
    }
    for (int m = 0; m < FEATURE_SIZE; m++)
        for (int i = 0; i < POOLED_WIDTH_2; i++)
            for (int j = 0; j < POOLED_HEIGHT_2; j++)
                for (int k = 0; k < HIDDEN_SIZE; k++)
                    f.w1[m, i, j, k] = w1[m, i, j, k];
    for (int i = 0; i < HIDDEN_SIZE; i++)
        for (int j = 0; j < OUTPUT_SIZE; j++)
            f.w2[i, j] = w2[i, j];
    for (int i = 0; i < HIDDEN_SIZE; i++)
        f.b1[i] = b1[i];
    for (int i = 0; i < OUTPUT_SIZE; i++)
        f.b2[i] = b2[i];
}
}

class GA
{
    int it;
    int EliteCount;
    Form1 f;
    Random rnd;
    int PopulationSize;
    double PCross;
    double PMut;
    Unit result;
    bool End;
    public GA(int PS, double PC, double PM, int EC, Form1 f1)
    {
        EliteCount = EC;
        f = f1;
        End = false;
        rnd = new Random();
        PopulationSize = PS;
        if (PS % 2 == 1) PopulationSize--;
        PCross = PC;
        PMut = PM;
        Unit[] Population;
        it = 0;

```

```

Population = Start();

while(f.training)
{
    it++;
    Population = Step(Population);

    if (End) break;
}

double V = double.MaxValue;
result = new Unit();
for (int i = 0; i < Population.Count(); i++)
{
    //Population[i].Value = FitnessFcn(Population[i]);
    if (V > Population[i].Value)
    {
        V = Population[i].Value;
        result=new Unit(Population[i]);
    }
}
f.Invoke(new Action(() =>
{

    result.Set(f);
    f.training = false;

})));
MessageBox.Show("Complete!");

}
Unit[] Step(Unit[] Population)
{
    Population = Crossover(Population);
    Population = Mutation(Population);
    double avg = 0;
    for (int i = 0; i < Population.Count(); i++)
    {
        Population[i].Value = FitnessFcn(Population[i]);
        avg += Population[i].Value;
        f.Invoke(new Action(() =>

```

```

        {
            f.Text = it.ToString() + " " + i.ToString() + " " +
String.Format("{0:0.000}", avg / ((double)i + 1.0)) + " " + String.Format("{0:0.000}",
Population[i].Value);
        }));
    }
    Population = Selection(Population);

    double V = double.MaxValue;
    int ind = 0;
    for (int i = 0; i < Population.Count(); i++)
    {

        if (V > Population[i].Value)
        {
            V = Population[i].Value;
            ind = i;
        }
    }
    Population[ind].Set(f);
    f.Invoke(new Action(() =>
    {
        f.Test();
        f.label1.Text = "Best value: " +
V.ToString()+"\n"+f.BestGA.ToString();
    }));

    return Population;

}
Unit[] Start()
{
    Unit[] Population = new Unit[PopulationSize];
    for (int i = 0; i < PopulationSize; i++)
        Population[i] = new Unit(f);
    if(f.IsInit)
        for (int i = 0; i < PopulationSize; i++)
        {
            Population[i] = new Unit(f,true);
        }
    return Population;
}

```

```

}
Unit[] Crossover(Unit[] pop)
{

    int PSize = pop.Count();
    if (PSize % 2 != 0) PSize--;
    int PairCount = PSize / 2;
    Unit[] Population = new Unit[PSize];
    Unit Parent1;
    Unit Parent2;
    Unit Kid1;
    Unit Kid2;

    for (int a = 0; a < PairCount; a++)
    {
        if (PCross < rnd.NextDouble())
        {
            Kid1=new Unit(pop[a * 2]);
            Kid2=new Unit(pop[a * 2 + 1]);

        }
        else
        {
            Parent1=new Unit(pop[a * 2]);
            Parent2=new Unit(pop[a * 2 + 1]);
            Kid1 = new Unit(Parent1);
            Kid2 = new Unit(Parent2);

            for (int i = 0; i < FEATURE_SIZE; i++)
                for (int j = 0; j < CONVOLUTION_SIZE; j++)
                    for (int k = 0; k < CONVOLUTION_SIZE; k++)
                    {
                        Kid1.cw1[i, j, k] = (Parent1.cw1[i, j,
k]*4.0+Parent2.cw1[i, j, k])/5.0;
                        Kid2.cw1[i, j, k] = (Parent2.cw1[i, j, k] *
4.0 + Parent1.cw1[i, j, k]) / 5.0;
                    }
            for (int m = 0; m < FEATURE_SIZE; m++)
                for (int i = 0; i < FEATURE_SIZE; i++)
                    for (int j = 0; j < CONVOLUTION_SIZE; j++)
                        for (int k = 0; k < CONVOLUTION_SIZE; k++)
                            {

```

```

Kid1.cw2[m,i, j, k] = (Parent1.cw2[m,i, j,
k] * 4.0 + Parent2.cw2[m,i, j, k]) / 5.0;
Kid2.cw2[m,i, j, k] = (Parent2.cw2[m,i, j,
k] * 4.0 + Parent1.cw2[m,i, j, k]) / 5.0;
    }
    for (int i = 0; i < FEATURE_SIZE; i++)
    {
        Kid1.cb1[i] = (Parent1.cb1[i] * 4.0 + Parent2.cb1[i])
/ 5.0;
        Kid2.cb1[i] = (Parent2.cb1[i] * 4.0 + Parent1.cb1[i])
/ 5.0;
        Kid1.cb2[i] = (Parent1.cb2[i] * 4.0 + Parent2.cb2[i])
/ 5.0;
        Kid2.cb2[i] = (Parent2.cb2[i] * 4.0 + Parent1.cb2[i])
/ 5.0;
    }
    for (int m = 0; m < FEATURE_SIZE; m++)
        for (int i = 0; i < POOLED_WIDTH_2; i++)
            for (int j = 0; j < POOLED_HEIGHT_2; j++)
                for (int k = 0; k < HIDDEN_SIZE; k++)
                {
                    Kid1.w1[m, i, j, k] = (Parent1.w1[m, i, j,
k] * 4.0 + Parent2.w1[m, i, j, k]) / 5.0;
                    Kid2.w1[m, i, j, k] = (Parent2.w1[m, i, j,
k] * 4.0 + Parent1.w1[m, i, j, k]) / 5.0;
                }
        for (int i = 0; i < HIDDEN_SIZE; i++)
            for (int j = 0; j < OUTPUT_SIZE; j++)
            {
                Kid1.w2[ i, j] = (Parent1.w2[i, j] * 4.0 +
Parent2.w2[ i, j]) / 5.0;
                Kid2.w2[i, j] = (Parent2.w2[ i, j] * 4.0 +
Parent1.w2[ i, j]) / 5.0;
            }
        for (int i = 0; i < HIDDEN_SIZE; i++)
        {
            Kid1.b1[i] = (Parent1.b1[i] * 4.0 + Parent2.b1[i]) /
5.0;
            Kid2.b1[i] = (Parent2.b1[i] * 4.0 + Parent1.b1[i]) /
5.0;
        }
        for (int i = 0; i < OUTPUT_SIZE; i++)
        {

```

```

Kid1.b2[i] = (Parent1.b2[i] * 4.0 + Parent2.b2[i]) /
5.0;

Kid2.b2[i] = (Parent2.b2[i] * 4.0 + Parent1.b2[i]) /
5.0;

    }

    }
    Population[a * 2] = new Unit(Kid1);
    Population[a * 2 + 1] = new Unit(Kid2);
}

return Population;
}
Unit[] Mutation(Unit[] pop)
{
    int m1;
    int m2;
    double temp;
    double koef=1.0/(100.0+2.0*(double)it);
    Unit[] Population = new Unit[pop.Count()];
    for (int a = 0; a < pop.Count(); a++)
    {
        Population[a]=new Unit(pop[a]);
        if (PMut > rnd.NextDouble())
        {
            m1 = rnd.Next(FEATURE_SIZE);
            m2 = rnd.Next(FEATURE_SIZE);
            for (int j = 0; j < CONVOLUTION_SIZE; j++)
                for (int k = 0; k < CONVOLUTION_SIZE; k++)
                {
                    temp = Population[a].cw1[m1, j, k];
                    Population[a].cw1[m1, j, k] =
Population[a].cw1[m2, j, k];

                    Population[a].cw1[m2, j, k] = temp;
                }
            m1 = rnd.Next(FEATURE_SIZE);
            m2 = rnd.Next(FEATURE_SIZE);
            for (int i = 0; i < FEATURE_SIZE; i++)
                for (int j = 0; j < CONVOLUTION_SIZE; j++)
                    for (int k = 0; k < CONVOLUTION_SIZE; k++)
                    {
                        temp = Population[a].cw2[m1,i, j, k];

```

```

Population[a].cw2[m1,i, j, k] =
Population[a].cw2[m2,i, j, k];

Population[a].cw2[m2, i,j, k] = temp;
}
for (int i = 0; i < FEATURE_SIZE; i++)
{
Population[a].cb1[i] *= (rnd.NextDouble() * 2.0 +
1.0);

Population[a].cb1[i] += rnd.NextDouble() * 0.4 - 0.2;
Population[a].cb2[i] *= (rnd.NextDouble() * 2.0 +
1.0);

Population[a].cb2[i] += rnd.NextDouble() * 0.4 - 0.2;
}
for (int m = 0; m < FEATURE_SIZE; m++)
for (int i = 0; i < POOLED_WIDTH_2; i++)
for (int j = 0; j < POOLED_HEIGHT_2; j++)
for (int k = 0; k < HIDDEN_SIZE; k++)
{
if (rnd.NextDouble() < 0.1)
{
Population[a].w1[m, i, j, k] *=
(rnd.NextDouble() * 2.0 + 1.0);

Population[a].w1[m, i, j, k] +=
rnd.NextDouble() * 0.4 - 0.2;
}
}
}
for (int i = 0; i < HIDDEN_SIZE; i++)
for (int j = 0; j < OUTPUT_SIZE; j++)
{
if (rnd.NextDouble() < 0.1)
{
Population[a].w2[i, j] *= (rnd.NextDouble() *
2.0 + 1.0);

Population[a].w2[i, j] += rnd.NextDouble() *
0.4 - 0.2;
}
}
}
for (int i = 0; i < HIDDEN_SIZE; i++)
{
Population[a].b1[i] *= (rnd.NextDouble() * 2.0 + 1.0);
Population[a].b1[i] += rnd.NextDouble() * 0.4 - 0.2;
}
for (int i = 0; i < OUTPUT_SIZE; i++)

```

```

{
    Population[a].b2[i] *= (rnd.NextDouble() * 2.0 + 1.0);
    Population[a].b2[i] += rnd.NextDouble() * 0.4 - 0.2;
}

}

if (PMut * 10.0 > rnd.NextDouble())
{
    for (int i = 0; i < FEATURE_SIZE; i++)
        for (int j = 0; j < CONVOLUTION_SIZE; j++)
            for (int k = 0; k < CONVOLUTION_SIZE; k++)
            {
                Population[a].cw1[i, j, k] *=
(rnd.NextDouble() * koef*2.0 + 1.0-koef);
            }
        for (int m = 0; m < FEATURE_SIZE; m++)
            for (int i = 0; i < FEATURE_SIZE; i++)
                for (int j = 0; j < CONVOLUTION_SIZE; j++)
                    for (int k = 0; k < CONVOLUTION_SIZE; k++)
                    {
                        Population[a].cw2[m, i, j, k] *=
(rnd.NextDouble() * koef * 2.0 + 1.0 - koef);
                    }
            for (int i = 0; i < FEATURE_SIZE; i++)
            {
                Population[a].cb1[i] *= (rnd.NextDouble() * koef * 2.0
+ 1.0 - koef);

                Population[a].cb2[i] *= (rnd.NextDouble() * koef * 2.0
+ 1.0 - koef);
            }
        for (int m = 0; m < FEATURE_SIZE; m++)
            for (int i = 0; i < POOLED_WIDTH_2; i++)
                for (int j = 0; j < POOLED_HEIGHT_2; j++)
                    for (int k = 0; k < HIDDEN_SIZE; k++)
                    {
                        Population[a].w1[m, i, j, k] *=
(rnd.NextDouble() * koef * 2.0 + 1.0 - koef);
                    }
            for (int i = 0; i < HIDDEN_SIZE; i++)
                for (int j = 0; j < OUTPUT_SIZE; j++)
                {
                    Population[a].w2[i, j] *= (rnd.NextDouble() * koef
* 2.0 + 1.0 - koef);

```

```

        }
        for (int i = 0; i < HIDDEN_SIZE; i++)
        {
            Population[a].b1[i] *= (rnd.NextDouble() * koef * 2.0
+ 1.0 - koef);

        }
        for (int i = 0; i < OUTPUT_SIZE; i++)
        {
            Population[a].b2[i] *= (rnd.NextDouble() * koef * 2.0
+ 1.0 - koef);

        }

    }

}

return Population;
}
Unit[] Selection(Unit[] pop)
{
    Unit[] Population;

    double avg = 0.0;
    List<Unit> list = new List<Unit>();
    List<Unit> list1 = new List<Unit>();
    double[] min = new double[EliteCount];
    int[] el = new int[EliteCount];
    for (int i = 0; i < EliteCount; i++)
    {
        min[i] = double.MaxValue;
    }
    int tempi;
    double tempd;
    for (int i = 0; i < pop.Count(); i++)
    {
        if (pop[i].Value <= min[0]) { min[0] = pop[i].Value; el[0] =
i; }

        for (int j = 1; j < EliteCount; j++)
        {
            if (min[j - 1] < min[j])
            {
                tempi = el[j];

```

```

        tempd = min[j];
        el[j] = el[j - 1];
        min[j] = min[j - 1];
        el[j - 1] = tempi;
        min[j - 1] = tempd;
    }
}
}
for (int i = 0; i < EliteCount; i++)
{
    list1.Add(pop[el[i]]);
}
for (int i = 0; i < pop.Count(); i++)
    avg += pop[i].Value;
avg /= pop.Count();
avg *= 1.15;
for (int i = 0; i < pop.Count(); i++)
{
    if (pop[i].Value <= avg) list.Add(pop[i]);
}
int tmp;
for (int i = 0; i < PopulationSize - EliteCount; i++)
{
    tmp = rnd.Next(list.Count());
    list1.Add(list[tmp]);
}

Population = list1.ToArray();
return Population;
}
double FitnessFcn(Unit u)
{
    u.Set(f);
    int All = 0;
    int Right = 0;
    double[] Wanted = new double[10];
    double[] Result = new double[10];
    double delta;
    double Delta=0;
    for (int i = 0; i < f.train.Count(); i++)
    {
        delta = 0;

```

```

        for (int j = 0; j < 10; j++) Wanted[j] = -1;
        Wanted[f.trainAnswer[i]] = 1;

        All++;
        Result = f.RunNetwork(f.train[i]);
        for (int j = 0; j < 10; j++)
            delta += (Result[j] - Wanted[j]) * (Result[j] - Wanted[j]);
        delta = Math.Sqrt(delta);
        Delta += delta;
        if (f.Predict(f.RunNetwork(f.train[i]), f.trainAnswer[i]))
            Right++;
    }

    return Delta / (double)All;
}

}

void Init()
{
    IsInit = true;
    double range1 = 1 / Math.Sqrt(CONVOLUTION_SIZE * CONVOLUTION_SIZE);
    double range2 = 1 / Math.Sqrt(FEATURE_SIZE * CONVOLUTION_SIZE *
CONVOLUTION_SIZE);
    double range3 = 1 / Math.Sqrt(FEATURE_SIZE * POOLED_WIDTH_2 *
POOLED_HEIGHT_2);
    double range4 = 1 / Math.Sqrt(HIDDEN_SIZE);

    for (int i = 0; i < FEATURE_SIZE; i++)
        for (int j = 0; j < CONVOLUTION_SIZE; j++)
            for (int k = 0; k < CONVOLUTION_SIZE; k++)
                cw1[i, j, k] = GetRandomGaussian(range1);

    for (int i = 0; i < FEATURE_SIZE; i++)
        for (int j = 0; j < FEATURE_SIZE; j++)
            for (int k = 0; k < CONVOLUTION_SIZE; k++)
                for (int l = 0; l < CONVOLUTION_SIZE; l++)
                    cw2[i, j, k, l] = GetRandomGaussian(range2);

    for (int i = 0; i < FEATURE_SIZE; i++)
        for (int j = 0; j < POOLED_WIDTH_2; j++)
            for (int k = 0; k < POOLED_HEIGHT_2; k++)

```

```

        for (int l = 0; l < HIDDEN_SIZE; l++)
            w1[i, j, k, l] = GetRandomGaussian(range3);

    for (int i = 0; i < HIDDEN_SIZE; i++)
        for (int j = 0; j < OUTPUT_SIZE; j++)
            w2[i, j] = GetRandomGaussian(range4);

    for (int i = 0; i < HIDDEN_SIZE; i++)
        b1[i] = 0;

    for (int i = 0; i < OUTPUT_SIZE; i++)
        b2[i] = 0;

    for (int i = 0; i < FEATURE_SIZE; i++)
        cb1[i] = cb2[i] = 0;
}

void TrainBP()
{
    LoadImages();
    Init();
    total = 0;
    correct = 0;
    cost = 0;
    costList.Clear();
    correctList.Clear();
    training = true;

    while (training)
    {
        ShuffleTraining();
        for (int j = 0; j < train.Count() && training; j++)
        {
            Application.DoEvents();
            double alpha = 1000.0 / (iterations + 20000);

            if (Predict(RunNetwork(train[j]), trainAnswer[j]))
            {
                correct++;
                correctList.Enqueue(true);
            }
            else
            {

```

```

        correctList.Enqueue(false);
    }

    double currCost = BackPropagate(train[j], trainAnswer[j],
alpha);

    cost += currCost;
    costList.Enqueue(currCost);

    total++;
    iterations++;

    if (total > train.Count())
    {
        total--;
        if (correctList.Dequeue())
            correct--;
        cost -= costList.Dequeue();
    }
    Invoke(new Action(() =>
    {
        this.Text = correct + "/" + total + " " +
String.Format("{0:0.0000} {1:0.0000} {2}", correct / total * 100.0, cost / total,
iterations);

    }));
    }
}

void LoadImages()
{
    IsLoaded = true;
    DirectoryInfo di;
    int ImgCount;
    List<double[,]> tr=new List<double[,]>();
    List<double[,]> te = new List<double[,]>();
    List<int> trA = new List<int>();
    List<int> teA = new List<int>();
    for (int i = 0; i < 10; i++)
    {

        di = new DirectoryInfo("D:/Train/" + i.ToString());
        ImgCount = di.GetFiles().Count();
    }
}

```

```

// ImgCount = 1;
for (int j = 0; j < ImgCount; j++)
{
    Invoke(new Action(() =>
    {

        this.Text = i + " " + j;

    }));
    if (j % 5 != 1)
    {
        tr.Add(BmpToArray(new Bitmap("D:/Train/" + i.ToString() +
"/" + j.ToString() + ".bmp")));
        trA.Add(i);
    }
    else
    {
        te.Add(BmpToArray(new Bitmap("D:/Train/" + i.ToString() +
"/" + j.ToString() + ".bmp")));
        teA.Add(i);
    }
}

train = tr.ToArray();
test = te.ToArray();
trainAnswer = trA.ToArray();
testAnswer = teA.ToArray();

}

void Test()
{
    if (!IsLoaded)
        LoadImages();
    total = 0;
    correct = 0;
    correctList.Clear();
    for (int i = 0; i < test.Count(); i++)
    {
        if (Predict(RunNetwork(test[i]), testAnswer[i]))
            correct++;
    }
}

```

```

        total++;
        Invoke(new Action(() =>
        {
            this.Text = correct + "/" + total + " " +
String.Format("{0:0.0000}", correct / total * 100.0);

        }));

    }
    BestGA = correct / total * 100.0;
}

private void Stop_Click (object sender, EventArgs e) {
    training = false;
}

private void Train_Click (object sender, EventArgs e) {
    System.Threading.Thread t = new System.Threading.Thread(TrainBP);
    t.Start();
}

private void TestButton_Click (object sender, EventArgs e) {
    System.Threading.Thread t = new System.Threading.Thread(Test);
    t.Start();
}

private void ShuffleTraining () {
    for (int i = 0; i < train.Count(); i++)
    {
        int swapIndex = rand.Next(train.Count() - i) + i;
        double[,] tempArray = train[i];
        train[i] = train[swapIndex];
        train[swapIndex] = tempArray;

        int temp = trainAnswer[i];
        trainAnswer[i] = trainAnswer[swapIndex];
        trainAnswer[swapIndex] = temp;
    }
}

private bool Predict (double[] output, int ans) {
    int max = 0;

```

```

        for (int i = 1; i < OUTPUT_SIZE; i++)
            if (output[i] > output[max])
                max = i;

        return max == ans;
    }

    private int Recognize(double[] input)
    {
        int max = 0;
        for (int i = 1; i < OUTPUT_SIZE; i++)
            if (input[i] > input[max])
                max = i;

        return max;
    }

    private double[] RunNetwork (double[,] input) {
        double[, ,] a1 = new double[FEATURE_SIZE, CONV_WIDTH_1, CONV_HEIGHT_1];
        double[, ,] p1 = new double[FEATURE_SIZE, POOLED_WIDTH_1,
POOLED_HEIGHT_1];

        for (int i = 0; i < FEATURE_SIZE; i++)
            for (int j = 0; j < CONV_WIDTH_1; j++)
                for (int k = 0; k < CONV_HEIGHT_1; k++)
                    a1[i, j, k] = cb1[i];

        for (int i = 0; i < FEATURE_SIZE; i++) {
            for (int m = 0; m < CONV_WIDTH_1; m++)
                for (int n = 0; n < CONV_HEIGHT_1; n++) {
                    for (int j = 0; j < CONVOLUTION_SIZE; j++)
                        for (int k = 0; k < CONVOLUTION_SIZE; k++)
                            a1[i, m, n] += cw1[i, j, k] * input[m + j, n + k];
                    a1[i, m, n] = Math.Tanh(a1[i, m, n]);
                }

            for (int m = 0; m < POOLED_WIDTH_1; m++)
                for (int n = 0; n < POOLED_HEIGHT_1; n++) {
                    p1[i, m, n] = Math.Max(Math.Max(a1[i, m * 2, n * 2], a1[i,
m * 2, n * 2 + 1]),
Math.Max(a1[i, m * 2 + 1, n * 2],
a1[i, m * 2 + 1, n * 2 + 1]));
                }
            }
    }

```

```

    }

    }

    double[, ,] a2 = new double[FEATURE_SIZE, CONV_WIDTH_2, CONV_HEIGHT_2];
    double[, ,] p2 = new double[FEATURE_SIZE, POOLED_WIDTH_2,
    POOLED_HEIGHT_2];

    for (int i = 0; i < FEATURE_SIZE; i++)
        for (int j = 0; j < CONV_WIDTH_2; j++)
            for (int k = 0; k < CONV_HEIGHT_2; k++)
                a2[i, j, k] = cb2[i];

    for (int map = 0; map < FEATURE_SIZE; map++) {
        for (int i = 0; i < CONV_WIDTH_2; i++) {
            for (int j = 0; j < CONV_HEIGHT_2; j++) {
                for (int pmap = 0; pmap < FEATURE_SIZE; pmap++) {
                    for (int m = 0; m < CONVOLUTION_SIZE; m++) {
                        for (int n = 0; n < CONVOLUTION_SIZE; n++) {
                            a2[map, i, j] += cw2[map, pmap, m, n] *
    p1[pmap, i + m, j + n];

                                }
                            }
                        }
                    a2[map, i, j] = Math.Tanh(a2[map, i, j]);
                }
            }
        }

    }

    for (int map = 0; map < FEATURE_SIZE; map++)
        for (int i = 0; i < POOLED_WIDTH_2; i++)
            for (int j = 0; j < POOLED_HEIGHT_2; j++)
                p2[map, i, j] = Math.Max(Math.Max(a2[map, i * 2, j * 2],
    a2[map, i * 2, j * 2 + 1]),
    Math.Max(a2[map, i * 2 + 1, j *
    2], a2[map, i * 2 + 1, j * 2 + 1]));

    double[] a3 = new double[HIDDEN_SIZE];

    for (int i = 0; i < HIDDEN_SIZE; i++)
        a3[i] = b1[i];

    for (int pmap = 0; pmap < FEATURE_SIZE; pmap++)
        for (int i = 0; i < POOLED_WIDTH_2; i++)

```

```

        for (int j = 0; j < POOLED_HEIGHT_2; j++)
            for (int k = 0; k < HIDDEN_SIZE; k++)
                a3[k] += p2[pmap, i, j] * w1[pmap, i, j, k];

    for (int i = 0; i < HIDDEN_SIZE; i++)
        a3[i] = Math.Tanh(a3[i]);

    double[] a4 = new double[OUTPUT_SIZE];

    for (int i = 0; i < OUTPUT_SIZE; i++)
        a4[i] = b2[i];

    for (int i = 0; i < HIDDEN_SIZE; i++)
        for (int j = 0; j < OUTPUT_SIZE; j++)
            a4[j] += w2[i, j] * a3[i];
    for (int i = 0; i < OUTPUT_SIZE; i++)
    {
        a4[i] = Math.Tanh(a4[i]);
    }

    return a4;
}

private double BackPropagate (double[,] input, int answer, double
learningRate) {

    double[, ,] a1 = new double[FEATURE_SIZE, CONV_WIDTH_1, CONV_HEIGHT_1];
    double[, ,] p1    =    new    double[FEATURE_SIZE,    POOLED_WIDTH_1,
POOLED_HEIGHT_1];

    for (int i = 0; i < FEATURE_SIZE; i++)
        for (int j = 0; j < CONV_WIDTH_1; j++)
            for (int k = 0; k < CONV_HEIGHT_1; k++)
                a1[i, j, k] = cb1[i];

    for (int i = 0; i < FEATURE_SIZE; i++) {
        for (int m = 0; m < CONV_WIDTH_1; m++)
            for (int n = 0; n < CONV_HEIGHT_1; n++) {
                for (int j = 0; j < CONVOLUTION_SIZE; j++)
                    for (int k = 0; k < CONVOLUTION_SIZE; k++)
                        a1[i, m, n] += cw1[i, j, k] * input[m + j, n + k];
                a1[i, m, n] = Math.Tanh(a1[i, m, n]);
            }
    }
}

```

```

        for (int m = 0; m < POOLED_WIDTH_1; m++)
            for (int n = 0; n < POOLED_HEIGHT_1; n++) {
                p1[i, m, n] = Math.Max(Math.Max(a1[i, m * 2, n * 2], a1[i,
m * 2, n * 2 + 1]),
Math.Max(a1[i, m * 2 + 1, n * 2],
a1[i, m * 2 + 1, n * 2 + 1]));
            }
    }

    double[, ,] a2 = new double[FEATURE_SIZE, CONV_WIDTH_2, CONV_HEIGHT_2];
    double[, ,] p2 = new double[FEATURE_SIZE, POOLED_WIDTH_2,
POOLED_HEIGHT_2];

    for (int i = 0; i < FEATURE_SIZE; i++)
        for (int j = 0; j < CONV_WIDTH_2; j++)
            for (int k = 0; k < CONV_HEIGHT_2; k++)
                a2[i, j, k] = cb2[i];

    for (int map = 0; map < FEATURE_SIZE; map++) {
        for (int i = 0; i < CONV_WIDTH_2; i++) {
            for (int j = 0; j < CONV_HEIGHT_2; j++) {
                for (int pmap = 0; pmap < FEATURE_SIZE; pmap++) {
                    for (int m = 0; m < CONVOLUTION_SIZE; m++) {
                        for (int n = 0; n < CONVOLUTION_SIZE; n++) {
                            a2[map, i, j] += cw2[map, pmap, m, n] *
p1[pmap, i + m, j + n];
                        }
                    }
                }
                a2[map, i, j] = Math.Tanh(a2[map, i, j]);
            }
        }
    }

    for (int map = 0; map < FEATURE_SIZE; map++)
        for (int i = 0; i < POOLED_WIDTH_2; i++)
            for (int j = 0; j < POOLED_HEIGHT_2; j++)
                p2[map, i, j] = Math.Max(Math.Max(a2[map, i * 2, j * 2],
a2[map, i * 2, j * 2 + 1]),
Math.Max(a2[map, i * 2 + 1, j *
2], a2[map, i * 2 + 1, j * 2 + 1]));

```

```

double[] a3 = new double[HIDDEN_SIZE];

for (int i = 0; i < HIDDEN_SIZE; i++)
    a3[i] = b1[i];

for (int pmap = 0; pmap < FEATURE_SIZE; pmap++)
    for (int i = 0; i < POOLED_WIDTH_2; i++)
        for (int j = 0; j < POOLED_HEIGHT_2; j++)
            for (int k = 0; k < HIDDEN_SIZE; k++)
                a3[k] += p2[pmap, i, j] * w1[pmap, i, j, k];

for (int i = 0; i < HIDDEN_SIZE; i++)
    a3[i] = Math.Tanh(a3[i]);

double[] a4 = new double[OUTPUT_SIZE];

for (int i = 0; i < OUTPUT_SIZE; i++)
    a4[i] = b2[i];

for (int i = 0; i < HIDDEN_SIZE; i++)
    for (int j = 0; j < OUTPUT_SIZE; j++)
        a4[j] += w2[i, j] * a3[i];

for (int i = 0; i < OUTPUT_SIZE; i++)
    a4[i] = Math.Tanh(a4[i]);

// CALCULATING ERROR

double[] e4 = new double[OUTPUT_SIZE];

double cost = 0;

for (int i = 0; i < OUTPUT_SIZE; i++) {
    e4[i] = (a4[i] - (i == answer ? 1 : -1)) * (1 - a4[i] * a4[i]);
    cost += (a4[i] - (i == answer ? 1 : -1)) * (a4[i] - (i == answer ?
1 : -1));
}

double[] e3 = new double[HIDDEN_SIZE];
for (int i = 0; i < HIDDEN_SIZE; i++)

```

```

        for (int j = 0; j < OUTPUT_SIZE; j++)
            e3[i] += e4[j] * w2[i, j] * (1 - a3[i] * a3[i]);

double[:,] pe2 = new double[FEATURE_SIZE, POOLED_WIDTH_2,
POOLED_HEIGHT_2];

for (int i = 0; i < FEATURE_SIZE; i++)
    for (int j = 0; j < POOLED_WIDTH_2; j++)
        for (int k = 0; k < POOLED_HEIGHT_2; k++)
            for (int l = 0; l < HIDDEN_SIZE; l++)
                pe2[i, j, k] += e3[l] * w1[i, j, k, l];

double[:,] ce2 = new double[FEATURE_SIZE, CONV_WIDTH_2,
CONV_HEIGHT_2];

for (int i = 0; i < FEATURE_SIZE; i++)
    for (int j = 0; j < POOLED_WIDTH_2; j++)
        for (int k = 0; k < POOLED_HEIGHT_2; k++) {

            if (a2[i, j * 2, k * 2] == p2[i, j, k])
                ce2[i, j * 2, k * 2] = pe2[i, j, k];

            else if (a2[i, j * 2 + 1, k * 2] == p2[i, j, k])
                ce2[i, j * 2 + 1, k * 2] = pe2[i, j, k];

            else if (a2[i, j * 2, k * 2 + 1] == p2[i, j, k])
                ce2[i, j * 2, k * 2 + 1] = pe2[i, j, k];

            else if (a2[i, j * 2 + 1, k * 2 + 1] == p2[i, j, k])
                ce2[i, j * 2 + 1, k * 2 + 1] = pe2[i, j, k];

        }

for (int i = 0; i < FEATURE_SIZE; i++)
    for (int j = 0; j < CONV_WIDTH_2; j++)
        for (int k = 0; k < CONV_HEIGHT_2; k++)
            ce2[i, j, k] *= (1 - a2[i, j, k] * a2[i, j, k]);

double[:,] pe1 = new double[FEATURE_SIZE, POOLED_WIDTH_1,
POOLED_HEIGHT_1];

for (int map = 0; map < FEATURE_SIZE; map++)
    for (int i = 0; i < CONV_WIDTH_2; i++)
        for (int j = 0; j < CONV_HEIGHT_2; j++) {
            for (int pmap = 0; pmap < FEATURE_SIZE; pmap++) {
                for (int m = 0; m < CONVOLUTION_SIZE; m++)

```

```

        for (int n = 0; n < CONVOLUTION_SIZE; n++)
            pe1[pmap, i + m, j + n] += ce2[map, i, j] *
cw2[map, pmap, m, n];
    }
}

double[,,] ce1 = new double[FEATURE_SIZE, CONV_WIDTH_1,
CONV_HEIGHT_1];

for (int i = 0; i < FEATURE_SIZE; i++)
    for (int j = 0; j < POOLED_WIDTH_1; j++)
        for (int k = 0; k < POOLED_HEIGHT_1; k++) {

            if (a1[i, j * 2, k * 2] == p1[i, j, k])
                ce1[i, j * 2, k * 2] = pe1[i, j, k];

            else if (a1[i, j * 2 + 1, k * 2] == p1[i, j, k])
                ce1[i, j * 2 + 1, k * 2] = pe1[i, j, k];

            else if (a1[i, j * 2, k * 2 + 1] == p1[i, j, k])
                ce1[i, j * 2, k * 2 + 1] = pe1[i, j, k];

            else if (a1[i, j * 2 + 1, k * 2 + 1] == p1[i, j, k])
                ce1[i, j * 2 + 1, k * 2 + 1] = pe1[i, j, k];

        }

for (int i = 0; i < FEATURE_SIZE; i++)
    for (int j = 0; j < CONV_WIDTH_1; j++)
        for (int k = 0; k < CONV_HEIGHT_1; k++)
            ce1[i, j, k] *= (1 - a1[i, j, k] * a1[i, j, k]);

// GRADIENT DESCENT
for (int i = 0; i < HIDDEN_SIZE; i++)
    for (int j = 0; j < OUTPUT_SIZE; j++)
        w2[i, j] -= a3[i] * e4[j] * learningRate;

for (int map = 0; map < FEATURE_SIZE; map++)
    for (int i = 0; i < POOLED_WIDTH_2; i++)
        for (int j = 0; j < POOLED_HEIGHT_2; j++)
            for (int k = 0; k < HIDDEN_SIZE; k++)
                w1[map, i, j, k] -= p2[map, i, j] * e3[k] *
learningRate;

```

```

    for (int map = 0; map < FEATURE_SIZE; map++)
        for (int i = 0; i < CONV_WIDTH_2; i++)
            for (int j = 0; j < CONV_HEIGHT_2; j++)
                for (int pmap = 0; pmap < FEATURE_SIZE; pmap++)
                    for (int m = 0; m < CONVOLUTION_SIZE; m++)
                        for (int n = 0; n < CONVOLUTION_SIZE; n++)
                            cw2[map, pmap, m, n] -= p1[pmap, i + m, j + n]
                                * ce2[map, i, j] * learningRate;

    for (int map = 0; map < FEATURE_SIZE; map++)
        for (int i = 0; i < CONV_WIDTH_1; i++)
            for (int j = 0; j < CONV_HEIGHT_1; j++)
                for (int m = 0; m < CONVOLUTION_SIZE; m++)
                    for (int n = 0; n < CONVOLUTION_SIZE; n++)
                        cw1[map, m, n] -= input[i + m, j + n] * ce1[map,
i, j] * learningRate;

    for (int i = 0; i < FEATURE_SIZE; i++) {
        for (int j = 0; j < CONV_WIDTH_2; j++) {
            for (int k = 0; k < CONV_HEIGHT_2; k++) {
                cb2[i] -= ce2[i, j, k] * learningRate;
            }
        }

        for (int j = 0; j < CONV_WIDTH_1; j++) {
            for (int k = 0; k < CONV_HEIGHT_1; k++) {
                cb1[i] -= ce1[i, j, k] * learningRate;
            }
        }
    }

    for (int i = 0; i < HIDDEN_SIZE; i++)
        b1[i] -= e3[i] * learningRate;

    for (int i = 0; i < OUTPUT_SIZE; i++)
        b2[i] -= e4[i] * learningRate;

    return cost;
}

private double GetRandomGaussian (double stdDev) {
    double u1 = rand.NextDouble();
    double u2 = rand.NextDouble();

```

```

        double randStdNormal = Math.Sqrt(-2.0 * Math.Log(u1)) * Math.Sin(2.0 *
Math.PI * u2);

        return stdDev * randStdNormal;
    }

    private void SaveWeights_Click (object sender, EventArgs e) {
        saveFileDialog1.ShowDialog();

        using (System.IO.StreamWriter file = new
System.IO.StreamWriter(saveFileDialog1.FileName)) {
            file.WriteLine(iterations);
            for (int i = 0; i < FEATURE_SIZE; i++)
                for (int j = 0; j < CONVOLUTION_SIZE; j++)
                    for (int k = 0; k < CONVOLUTION_SIZE; k++)
                        file.WriteLine(cw1[i, j, k]);

            for (int i = 0; i < FEATURE_SIZE; i++)
                for (int j = 0; j < FEATURE_SIZE; j++)
                    for (int k = 0; k < CONVOLUTION_SIZE; k++)
                        for (int l = 0; l < CONVOLUTION_SIZE; l++)
                            file.WriteLine(cw2[i, j, k, l]);

            for (int i = 0; i < FEATURE_SIZE; i++)
                for (int j = 0; j < POOLED_WIDTH_2; j++)
                    for (int k = 0; k < POOLED_HEIGHT_2; k++)
                        for (int l = 0; l < HIDDEN_SIZE; l++)
                            file.WriteLine(w1[i, j, k, l]);

            for (int i = 0; i < HIDDEN_SIZE; i++)
                for (int j = 0; j < OUTPUT_SIZE; j++)
                    file.WriteLine(w2[i, j]);

            for (int i = 0; i < HIDDEN_SIZE; i++)
                file.WriteLine(b1[i]);

            for (int i = 0; i < OUTPUT_SIZE; i++)
                file.WriteLine(b2[i]);

            for (int i = 0; i < FEATURE_SIZE; i++)
                file.WriteLine(cb1[i]);
            for (int i = 0; i < FEATURE_SIZE; i++)
                file.WriteLine(cb2[i]);
        }
    }
}

```

```

private void LoadWeights_Click (object sender, EventArgs e) {
    IsInit = true;
    total = 0;
    correct = 0;
    correctList.Clear();
    openFileDialog1.ShowDialog();
    using (StreamReader reader = new
StreamReader(openFileDialog1.FileName)) {
        iterations = double.Parse(reader.ReadLine());
        for (int i = 0; i < FEATURE_SIZE; i++)
            for (int j = 0; j < CONVOLUTION_SIZE; j++)
                for (int k = 0; k < CONVOLUTION_SIZE; k++)
                    cw1[i, j, k] = double.Parse(reader.ReadLine());

        for (int i = 0; i < FEATURE_SIZE; i++)
            for (int j = 0; j < FEATURE_SIZE; j++)
                for (int k = 0; k < CONVOLUTION_SIZE; k++)
                    for (int l = 0; l < CONVOLUTION_SIZE; l++)
                        cw2[i, j, k, l] = double.Parse(reader.ReadLine());

        for (int i = 0; i < FEATURE_SIZE; i++)
            for (int j = 0; j < POOLED_WIDTH_2; j++)
                for (int k = 0; k < POOLED_HEIGHT_2; k++)
                    for (int l = 0; l < HIDDEN_SIZE; l++)
                        w1[i, j, k, l] = double.Parse(reader.ReadLine());

        for (int i = 0; i < HIDDEN_SIZE; i++)
            for (int j = 0; j < OUTPUT_SIZE; j++)
                w2[i, j] = double.Parse(reader.ReadLine());

        for (int i = 0; i < HIDDEN_SIZE; i++) {
            b1[i] = double.Parse(reader.ReadLine());
        }

        for (int i = 0; i < OUTPUT_SIZE; i++)
            b2[i] = double.Parse(reader.ReadLine());

        for (int i = 0; i < FEATURE_SIZE; i++)
            cb1[i] = double.Parse(reader.ReadLine());
        for (int i = 0; i < FEATURE_SIZE; i++)
            cb2[i] = double.Parse(reader.ReadLine());
    }
}

```

```

    }
    Bitmap bmp;
    private void SaveImage_Click(object sender, EventArgs e)
    {
        int n = 9;
        DirectoryInfo di;
        int ImgCount;

        foreach (RadioButton rb in this.Controls.OfType<RadioButton>())
        {
            if (rb.Checked)
            {
                di = new DirectoryInfo("D:/Train/" + n.ToString());
                ImgCount = di.GetFiles().Count();
                pictureBox1.Image.Save("D:/Train/" + n.ToString() + "/" +
ImgCount.ToString() + ".bmp");
            }
            n--;
        }
        bmp = new Bitmap(159, 159);
        for (int i = 0; i < 159; i++)
            for (int j = 0; j < 159; j++)
                bmp.SetPixel(i, j, Color.White);
        pictureBox1.Image = bmp;
    }

    private void ClearImage_Click(object sender, EventArgs e)
    {
        bmp = new Bitmap(159, 159);
        for (int i = 0; i < 159; i++)
            for (int j = 0; j < 159; j++)
                bmp.SetPixel(i, j, Color.White);
        pictureBox1.Image = bmp;
    }

    private void RecognizeDigit_Click(object sender, EventArgs e)
    {
        MessageBox.Show(Recognize(RunNetwork(BmpToArray(new
Bitmap(pictureBox1.Image))))).ToString(), "Result");
    }

    private void Form1_Load(object sender, EventArgs e)
    {

```

```

        bmp = new Bitmap(159, 159);
        for (int i = 0; i < 159; i++)
            for (int j = 0; j < 159; j++)
                bmp.SetPixel(i, j, Color.White);
        pictureBox1.Image = bmp;
    }
    bool IsPressed = false;
    int X;
    int Y;
    private void pictureBox1_MouseDown(object sender, MouseEventArgs e)
    {
        IsPressed = true;
        X = e.X;
        Y = e.Y;
    }

    private void pictureBox1_MouseUp(object sender, MouseEventArgs e)
    {
        IsPressed = false;
    }

    private void pictureBox1_MouseMove(object sender, MouseEventArgs e)
    {
        if (IsPressed)
        {
            Graphics g = Graphics.FromImage(bmp);
            Pen p = new Pen(Color.Black);
            p.Width = 6;
            g.DrawLine(p, X, Y, e.X, e.Y);
            g.FillEllipse(new SolidBrush(Color.Black), e.X - 3, e.Y - 3, 6,
6);

            X = e.X;
            Y = e.Y;
            pictureBox1.Image = bmp;
        }
    }

    void TrainGA() {
        if(!IsLoaded)
            LoadImages();
        training = true;
        GA ga = new GA(20, 0.8, 0.035, 4, this);
    }

```

```
private void button4_Click(object sender, EventArgs e)
{
    System.Threading.Thread t = new System.Threading.Thread(TrainGA);
    t.Start();
}
}
```

ДОДАТОК Д
Слайди презентації

Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Дослідження та програмна реалізація нейромережних методів розпізнавання рукописних символів

Виконав

ст. гр. КНТ-129м

К. А. Калоян

Керівник:

к.т.н.

Є. О. Гофман

Рисунок Д.1 – Слайд № 1

Об'єкт, предмет та мета роботи

Об'єкт дослідження – процес розпізнавання образів.

Предмет дослідження – нейромережеві методи розпізнавання рукописних символів.

Мета роботи – дослідження та розробка програмних засобів на основі нейромережових технологій для розпізнавання рукописних символів.

Рисунок Д.2 – Слайд № 2

Завдання роботи

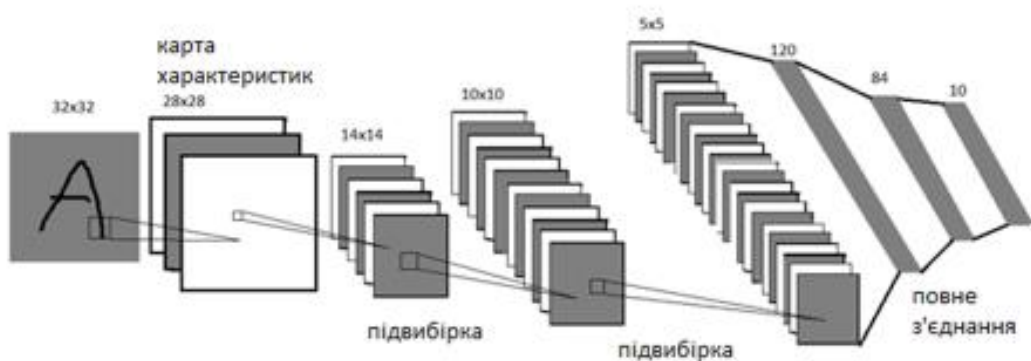
Для досягнення поставленої мети необхідно було розв'язати такі завдання:

- аналіз методів та моделей для розпізнавання образів;
- розробка технічного завдання для реалізації програми;
- вибір середовища розробки програмного забезпечення;
- розробка гібридного методу синтезу нейромережевих моделей на основі генетичного пошуку та зворотного поширення помилки;
- конструювання програмного забезпечення для розпізнавання рукописних символів на основі нейромережевих технологій;
- тестування розробленого програмного забезпечення.

3

Рисунок Д.3 – Слайд № 3

Згорткові нейронні мережі



4

Рисунок Д.4 – Слайд № 4

Гібридний метод синтезу нейромережевих моделей на основі генетичного пошуку та зворотного поширення помилки

На етапі ініціалізації генетичного пошуку в якості початкових значень вагових коефіцієнтів використовувати значення, отримання за допомогою методу зворотного поширення помилки.

Оцінювання хромосом популяції:

$$F(H) = \frac{1}{N} \sum_{i=0}^N \sqrt{\sum_{j=0}^{10} (X_{i,j} - Y_{i,j})^2}$$

Отримання нових хромосом за допомогою оператора схрещування:

$$w = w_1 k + w_2 (1 - k)$$

5

Рисунок Д.5 – Слайд № 5

Вибір середовища розробки

Критерій порівняння	Середовище розробки		
	NetBeans	Eclipse	Microsoft Visual Studio
Швидкість модернізації	±	±	++
Підтримка репозиторіїв	++	+	++
Підтримка серверів	++	+	++
Розроблення графічного інтерфейсу користувача	+	—	++
Доступність (вимоги до розповсюдження)	+	+	+

6

Рисунок Д.6 – Слайд № 6

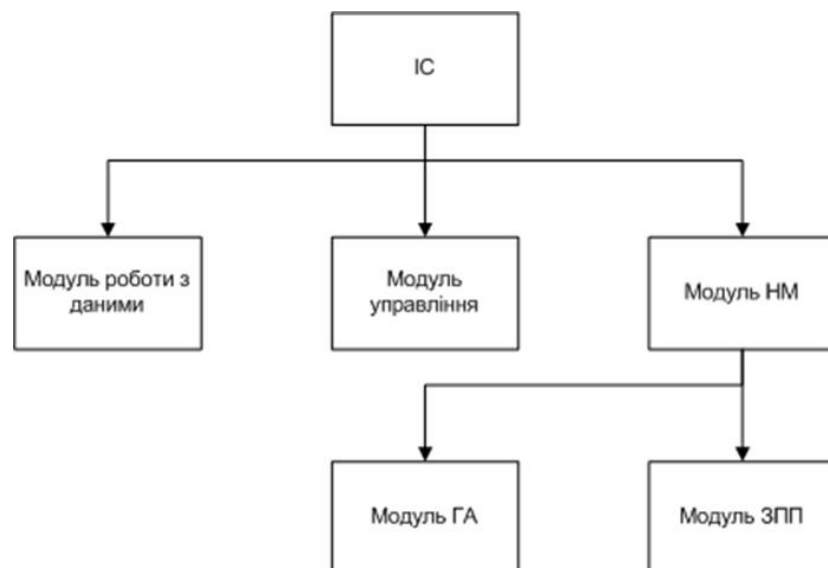
Вибір мови програмування

Критерій порівняння	C#	VB.NET	Java
Підтримка імперативної парадигми програмування	+	+	+
Підтримка об'єктно-орієнтованої парадигми програмування	+	+	+
Наявність open-source компілятора	+	+	+
Зручність використання для створення windows-застосунків	+	+/-	+/-
Можливість роботи з шаблонами (аналоги templates, generics, ...)	+	+	+
Значення параметрів за замовченням	+	+	-
Кросплатформність	+	-	+

7

Рисунок Д.7 – Слайд № 7

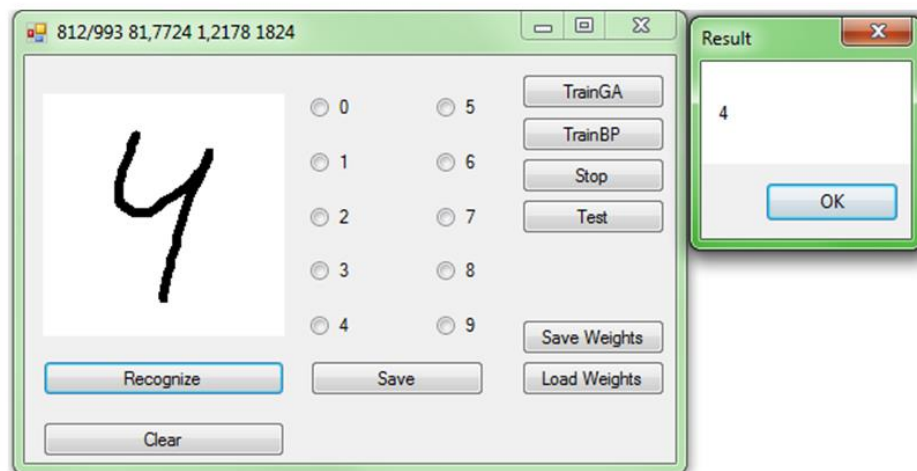
Структурна схема програми



8

Рисунок Д.8 – Слайд № 8

Приклад роботи з програмою



9

Рисунок Д.9 – Слайд № 9

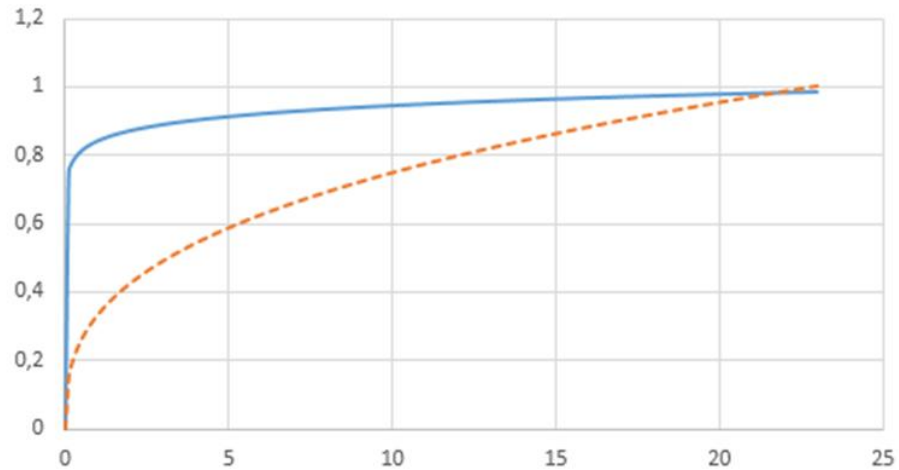
Експерименти та результати

Метод	A0, B0	A0, B1	A0, B2	A1, B0	A1, B1	A1, B2	A2, B0	A2, B1	A2, B2	B
Розп. через 1 хв.	34.2%	92.1%	95.2%	38.6%	94%	95.8%	33.7%	92.5%	95.3%	94.3%
Розп. через 3 хв.	59.1%	94.7%	96.5%	63%	95.9%	96.4%	59.1%	95.4%	95.8%	95.5%
Розп. через 10 хв.	82%	96.8%	96.7%	82.1%	98.2%	96.6%	76.3%	96%	95.9%	96.6%
Час до 95% розп.	-	4.2	0.1	-	1.9	0.1	-	3.1	0.1	2.9
Час до 97% розп.	-	13.4	13.3	-	7.2	11.1	-	18.6, 2	22.5, 1	12.3
Час до 99% розп.	-	-	24.1	-	21.8	-	-	-	-	-

10

Рисунок Д.10 – Слайд № 10

Експерименти та результати. Залежність точності розпізнавання від часу навчання



11

Рисунок Д.11 – Слайд № 11

ВИСНОВКИ

У ході виконання дипломної кваліфікаційної роботи магістра було розв'язано актуальне завдання нейромережових методів розпізнавання рукописних символів.

Розв'язано такі завдання:

- аналіз методів та моделей для розпізнавання образів;
- розробка технічного завдання для реалізації програми;
- вибір середовища розробки програмного забезпечення;
- розробка гібридного методу синтезу нейромережових моделей на основі генетичного пошуку та зворотного поширення помилки;
- конструювання програмного забезпечення для розпізнавання рукописних символів на основі нейромережових технологій;
- тестування розробленого програмного забезпечення.

12

Рисунок Д.12 – Слайд № 12

ВИСНОВКИ

Розроблено структурну схему програми. Визначено, що основними модулями розробленого програмного забезпечення є такі: модуль роботи з даними, модуль управління, модуль реалізації загорткової нейронної мережі; модуль навчання НМ за допомогою генетичного алгоритму, модуль навчання НМ за допомогою метода зворотного поширення помилки.

Наукова новизна роботи полягає у тому, що розроблено гібридний метод синтезу нейромережових моделей на основі генетичного пошуку та зворотного поширення помилки, в якому генетичний алгоритм використовується для навчання ваг у фільтрах мережі, а в якості початкових значень вагових коефіцієнтів використовуються значення, отримані за допомогою методу зворотного поширення помилки, що дозволяє уникнути заціклення в локальних мінімумах і підвищити точність розпізнавання за допомогою синтезованих нейромоделей..

Практичне значення роботи полягає у тому, що розроблено програмне забезпечення, що реалізує нейромережові методи розпізнавання рукописних символів.

Виконано тестування розробленого програмного забезпечення, що реалізує нейромережові методи розпізнавання рукописних символів. Результати тестування підтвердили, що програма може використовуватися за призначенням.

13

Рисунок Д.13 – Слайд № 13