Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних систем управління

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему:

Нейромережевий застосунок для розпізнавання текстових зображень із завадами

Виконав: студент 2 курсу, групи 2АКІТ-21м,
спеціальності 151 – Автоматизація та
комп'ютерно-інтегровані технології

Анатолій ПОЛІЩУК
Ім'я ПРІЗВИЩЕ

Керівник: д.т.н., доцент, професор каф. КСУ
степінь, звання, посада

В'ячеслав КОВТУН
Ім'я ПРІЗВИЩЕ

«12» грудня 2022 р.

Опонент: доцент каф.АІІТ
степінь, звання, посада

Юрій ІВАНОВ
Ім'я ПРІЗВИЩЕ

«14» грудня 2022 р.

Допущено до захисту
Зав. кафедри КСУ

В'ячеслав КОВТУН

«14» грудня 2022 р.

Вінниця ВНТУ – 2022 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних систем та автоматизації
Кафедра комп'ютерних систем управління
Рівень вищої освіти перший (магістерський)
Галузь знань – 15 – Автоматизація та приладобудування
Спеціальність – 151 – Автоматизація та комп'ютерно-інтегровані технології
Освітньо - професійна програма – Інтелектуальні комп'ютерні системи

ЗАТВЕРДЖУЮ
Завідувач кафедри КСУ


В'ячеслав КОВТУН

"03" жовтня 2022 року

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Поліщуку Анатолію Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Нейромережевий застосунок для розпізнавання текстових зображень із завадами керівник роботи д.т.н. Ковтун В. В. затверджені наказом ВНТУ від "14" вересня 2022 року №203

2. Термін подання студентом роботи "12" грудня 2022 року

3. Вихідні дані до роботи: експлуатаційні дані з об'єкту дослідження, сучасна програмна архітектура, мова програмування C#.

4. Зміст текстової частини: 1– аналіз сучасного розвитку технологій розпізнавання символів на тлі завад; 2 – розробка додатку для розпізнавання символів на тлі завад 3 – реалізація додатку для розпізнавання символів на тлі завад на основі нейромережі;

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень): архітектура додатку розпізнавання символів на тлі завад; пікселізація зображення; векторизація зображення; структура додатку розпізнавання символів на тлі завад; алгоритм безпосередньо розпізнавання символів на тлі завад; діаграма класів; основні класи програми; діаграма послідовності роботи програми; головне вікно програми; вікно налаштувань; редактор бази слів; налаштування модуля блочної обробки; навчання нейронної мережі; результат розпізнавання; тестове розпізнавання зображення із завадами; достовірність розпізнавання символів на тлі завад

1. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	виконаня прийняв
1-3	Ковтун В.В., доцент кафедри АІІТ		
4	Небава М.І., пофесор кафедри ЕПВМ		

2. Дата видачі завдання

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Огляд предметної області	04.09.22р	21.09.22р	
2	Розробка програмного забезпечення та експериментальні дослідження	22.09.22р	11.11.22р	
3	Підготовка економічної частини	12.11.22р	07.12.22р	
4	Оформлення пояснювальної записки і графічного матеріалу	08.12.22р	16.12.22р	
5	Попередній захист роботи	17.12.22р	17.12.22р	
6	Остаточний захист роботи	22.12.22р	22.12.22р	

Студент _____

(підпис) (*Ім'я ПРІЗВИЩЕ*)

Керівник роботи _____

(підпис) (*Ім'я ПРІЗВИЩЕ*)

АНОТАЦІЯ

УДК 004.9

Поліщук А. А Нейромережевий застосунок для розпізнавання текстових зображень із завадами. Магістерська кваліфікаційна робота зі спеціальності 151 – Автоматизація та комп’ютерно-інтегровані технології, освітня програма – Інтелектуальні комп’ютерні системи. Вінниця: ВНТУ, 2022. 105 с.

На укр. мові. Бібліогр.: 46 назв; рис.: 27; табл. 14.

У магістерскій кваліфікаційній роботі розроблено методи розпізнавання символів і програмних засобів та їх роботи на фоні перешкод на основі нейромережевої структури. У оглядово-аналітичній частині роботи досліджено сучасні методи та підходи по розпізнаванню символів на тлі завад, Проаналізовано існуючі програми по розпізнаванню тексту. У теоретично-методичній частині обгрунтувується вибір типу нейромережі. Розроблений продукт аналізується та тестується. У економічній частині проаналізований технічний рівень і розрахована собівартість реалізації розробки. Ілюстративна частина складається з 15 плакатів із результатами роботи.

Ключові слова: нейронна мережа, розпізнавання зображенб, алгоритм.

ABSTRACT

UDC 004.9

Polishchuk A. A Neural network application for recognition of text images with interference. Master's thesis on specialty 151 - Automation and computer-integrated technologies, educational program - Intelligent computer systems. Vinnytsia: VNTU, 2022. 105 p.

In Ukrainian speech Bibliography: 46 titles; Fig.: 27; table 14.

In the master's qualification thesis, the methods of character recognition and software tools and their operation against the background of obstacles based on the neural network structure were developed. In the review and analytical part of the work, modern methods and approaches for character recognition against the background of interference were investigated, and existing programs for text recognition were analyzed. In the theoretical and methodological part, the choice of the type of neural network is justified. The developed product is analyzed and tested. In the economic part, the technical level is analyzed and the cost of development implementation is calculated. The illustrative part consists of 15 posters with the results of the work.

Keywords: neural network, image recognition, algorithm

Відгук
керівника магістерської кваліфікаційної роботи

студента Поліщука Анатолія Андрійовича група 2КІТ-21м
на тему: Нейромережевий застосунок для розпізнавання текстових зображень із завадами.

Актуальність роботи в контексті спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» доведена результатами інформаційного пошуку та аналізу літературних джерел. Додатковим підтвердженням актуальності роботи слугують опубліковані тези на Науково-технічній конференції факультету інтелектуальних інформаційних технологій та автоматизації.

Представлені в магістерській кваліфікаційній роботі рішення обгрунтовані послідовним ланцюгом дій, які включають пошук інформації про проблему, її узагальнення, формулювання прикладних задач для її вирішення, проектування відповідного програмного засобу для вирішення поставлених задач, його тестування і формулювання висновків. Раціональність та ефективність прийнятих рішень доведені результатами тестування.

Дипломник показав хороший рівень спеціальних знань і «м'яких» навичок. Дипломник продемонстрував вміння: - вирішувати поставлені керівником завдання самостійно, згідно власноруч розробленої схеми заходів; - здійснювати пошук і узагальнення інформації; - комунікативні навички. Доведенням ерудиції та креативності дипломника є вчасно представлена магістерська кваліфікаційна робота.

Основні результати, представлені в роботі отримані дипломником самостійно. Матеріалу роботи властивий високий ступінь оригінальності, що доведено результатами перевірники на наявність запозичень.

Дипломник працював ритмічно, без суттєвих відхилень від затвердженого графіку. Втрати зв'язку з керівником не було.

Недоліки: Бажано було аналізувати свіжіші літературні джерела. Результати багатоваріантного аналізу бажано було звести в таблицю. Не всі рисунки достатньо якісні. В тексті дипломної роботи зустрічаються поодинокі вади форматування. Автор не акцентує увагу на тім, які саме з отриманих результатів опубліковані у вигляді тез.

Загалом магістерська кваліфікаційна робота ***відповідає*** спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології», заслуговує на оцінку В, а її автор ***заслуговує*** присудження кваліфікації: ступінь вищої освіти магістр, спеціальність «Автоматизація та комп'ютерно-інтегровані технології», освітня програма «Інтелектуальні комп'ютерні системи».

Керівник магістерської кваліфікаційної роботи

д.т.н., доцент каф. КСУ Ковтун В.В.

Відгук
опонента на магістерську кваліфікаційну роботу

студента Поліщука Анатолія Андрійовича група 2КІТ-21м
на тему: Нейромережевий застосунок для розпізнавання текстових зображень із завадами.

Актуальність роботи в контексті спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» доведена результатами інформаційного пошуку та аналізу літературних джерел. Додатковим підтвердженням актуальності роботи слугують опубліковані тези на Науково-технічній конференції факультету інтелектуальних інформаційних технологій та автоматизації.

Перший розділ магістерської кваліфікаційної роботи повністю присвячений огляду літературних та інформаційних джерел за обраною темою. Проаналізовано не менш ніж три аналоги створюваної системи. Функціональність та форм-фактор створеної системи цілком визначені на основі результатів критичного огляду літератури.

Прийняті рішення обґрунтовані результатами огляду літератури, результатами проектування та втілені в функціонуючу програмну систему. Результати її тестування доводять правильність прийнятих рішень.

Експериментальні дослідження продумані і повні. Тести охоплюють як функції інтерфейсу створеної системи, так і доводять якість та повноту виконання нею функціонального призначення, обґрунтованого на етапі проектування.

Вміст графічної частина повною мірою репрезентує всі отримані в магістерській кваліфікаційній роботі результати. Якість рисунків в графічній частині прийнятна.

Недоліки: Для тестування функцій системи можна було застосувати спеціалізовані автоматизовані середовища. Питанню вибору системи позиціонування, яку використовує авторський додаток, варто було приділити більше уваги.

Загалом магістерська кваліфікаційна робота ***відповідає*** спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології», заслуговує на оцінку _____, а її автор ***заслуговує*** присудження кваліфікації: ступінь вищої освіти магістр, спеціальність «Автоматизація та комп'ютерно-інтегровані технології», освітня програма «Інтелектуальні комп'ютерні системи».

Опонент на магістерську кваліфікаційну роботу

Доцент каф. АІІТ _____ Іванов Ю.Ю.

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІНТЕЛЕКТУАЛЬНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА АВТОМАТИЗАЦІЇ

ЗМІСТ

ВСТУП

Актуальність теми дослідження. За останні роки комп’ютерні технології змінюються та вдосконалюються з неймовірною швидкістю. Особливо швидко розвиваються портативні пристрої, які наздогнали своїх настільних попередників. Така швидкість розробки обладнання не може не впливати на попит ринку програмного забезпечення. Призначення будь-якого персонального портативного пристрою — автоматизувати певний тип людської діяльності, будь то планування зустрічі чи розрахунок банківської операції.

Окремою сферою діяльності для портативних пристроїв, яка поки що відстає від людини, є природне завдання категоризації та інтелектуальної діяльності. Великих успіхів досягнуто в нових напрямках програмного забезпечення. Голосове керування пристроями вже давно стало стандартною частиною основного програмного забезпечення.

Одним із таких видів діяльності пристрою є завдання ідентифікації. Зі зменшенням розміру та через ергономічні потреби мобільні та портативні пристрої майже втратили свої звичні клавіатури. Однак в ритмі сучасного життя можливість швидкого введення даних в комп'ютер залишається важливою перевагою. Тому потрібні спеціальні програми, здатні швидко реагувати на дії користувача та оцифровувати зрозумілу візуальну інформацію для використання на комп’ютерах.

Процес розпізнавання тексту добре вивчений, з багатьма застосуваннями та робочими прикладами. Проте дослідникам і розробникам все ще ведеться боротьба за пошук нових можливостей щодо якості та швидкості.

Існує досить багато систем загального призначення, які можуть якісно розпізнавати текст і працювати досить швидко, особливо якщо на шляху розпізнавання є проблеми з якістю зображення або розпізнаного об'єкта.

Існує потреба в продукті, який міг би централізовано обробляти вхідну графічну інформацію, розпізнавати сканований текст і обробляти запити користувачів, а з іншого – працювати швидко й ефективно.

Нові розробки мають відповідати двом критеріям: бути повністю адаптивними, а отже універсальними, а також швидко та якісно діяти (чого бракує адаптивним підходам).

Таким чином, застосування сучасних методів штучного інтелекту для вирішення завдань забезпечить подальше застосування інформаційних технологій у сфері розпізнавання символів на тлі перешкод, а також є актуальною темою в дослідженнях нових методів штучного інтелекту.

Мета та мета дослідження. Метою кваліфікаційної роботи MSc є підвищення надійності розпізнавання символів у контексті перешкод для використання штучних нейронних мереж.

Для досягнення поставлених цілей необхідно вирішити такі завдання:

1) Проаналізувати предметну область розпізнавання символів у контексті інтерференції;

2) Розглянути існуючі методи вирішення задачі розпізнавання символів у розрізі перешкод та обґрунтувати вибір методів, Задовольняти мету даної магістерської кваліфікаційної роботи;

3) Структура нейронної мережі, яка використовується при розробці;

4) етап розробки інформаційної технології розпізнавання символів в умовах перешкод;

5) Розробити алгоритм функцій, реалізованих програмним забезпеченням розпізнавання символів у контексті перешкод на основі нейронних мереж;

6) Програмна реалізація інформаційної технології розпізнавання символів на фоні перешкод на основі нейронної мережі;

7) Тестувати та аналізувати програмну реалізацію розпізнавання символів на фоні перешкод на основі нейронних мереж;

Об'єктом дослідження є процес розпізнавання персонажа на фоні перешкод.

Тема дослідження – надійність методів розпізнавання символів і програмних засобів та їх роботи на фоні перешкод на основі нейромережевої структури.

Методи дослідження. У роботі використовуються такі наукові методи дослідження: системний аналіз для аналізу структури інформаційних систем, теорія нейронних мереж для реалізації інформаційних технологій розпізнавання символів у контексті перешкод, методи математичної статистики для розробки процесу з використанням програмних засобів пошуку та розрахувати результати експерименту, орієнтований Програмна реалізація об’єкта програмується.

Практичні наслідки отриманих результатів такі:

1. Розроблено алгоритм розпізнавання символів на основі нейронної мережі на фоні перешкод.

2. Для нейромережевих функцій розроблено алгоритм для підвищення надійності розпізнавання символів в контексті перешкод.

3. Розроблено програмний засіб розпізнавання символів на основі штучної нейронної мережі на фоні перешкод.

Розроблений алгоритм може бути реалізований у початковому процесі як лекція з дисципліни «Нейрономережевий підхід до розпізнавання символів на фоні перешкод з використанням нейронних мереж» предмет «Нейрономережевий підхід до обчислювального інтелекту».

Обґрунтованість теоретичних положень магістерської роботи визначається точністю постановки проблеми, правильним застосуванням математичних методів у процесі доведення наукових положень, строгим виведенням аналітичних співвідношень, порівнянням результатів з відомими результатами та результати математичного моделювання з тими, що знаходяться в розробці.Збіжність результатів, отриманих під час впровадження програмного засобу.

1 АНАЛІЗ СУЧАСНОГО РОЗВИТКУ ТЕХНОЛОГІЙ РОЗПІЗНАВАННЯ СИМВОЛІВ НА ТЛІ ЗАВАД

1.1 Аналіз існуючих методів розпізнавання символів на тлі завад

Найкращі методи розпізнавання зображень/символів використовують оптичні властивості [2, 3]. Тобто символ має декілька властивостей, і система перевіряє наявність цих властивостей у зображення. Цей пошук виконується дуже швидко і має високу метрику точності розпізнавання, але він має кілька явних недоліків [4, 5]. По-перше, для кожного символу та кожного шрифту потрібно знайти та формалізувати десятки або навіть сотні символів, перевіряючи їх на перетини. По-друге, виявлення перешкод або ідентифікація перешкод здійснюється лише за допомогою попередньої обробки, тому зникнення символів призводить до втрати цієї інформації. Загалом, цей спосіб не підходить для розпізнавання рукописних символів і може бути модифікований під поставлене завдання, але показник якості розпізнавання буде дуже низьким.

Для задач розпізнавання образів існує принципово інший підхід.

Для розпізнавання можна використовувати клітинні автомати [5]. Цей автомат є набором локальних правил, які діють на дискретний світ. Ці правила працюють однаково будь-де в дискретному світі та використовують локальне середовище для зміни свого стану. Час в автоматі клітки змінюється в окремих частинах, змінюючи зовнішній вигляд поля клітки. Поле — це N-вимірний простір, розділений на дискретні одиниці за деякими правилами.

Існує багато типів автоматів із кліткою: стандартні автомати з кліткою – засновані на класичному визначенні, наведеному вище; мобільні клітинні автомати – додатково містять активну одиницю, яка змінює свій стан; машина Тьюринга – активна одиниця має кілька станів; альтернативна система – коли клітина зміни блоку тощо.

Ці методи засновані на використанні бінарних зображень і обробці клітинних автоматів для виділення особливостей конкретних символів.

Цей метод вимагає більш ретельної підготовки, вимагає розробки набору систем правил клітинних автоматів і вимагає обробки всіх автоматів одночасно, що вимагає великих обчислень.

Адаптивні методи розпізнавання, засновані на методах інтелектуальної класифікації, є більш загальними.

Одним із найпоширеніших методів є використання нейронних мереж [6].

Нейронні мережі [7-9] – це системи однотипних елементів обробки, що формують вихідні сигнали з вхідних даних. Той факт, що характер мережі змінюється залежно від параметрів елементів, використовується для використання системи класифікації та завдань машинного навчання [10].

Існують методи побудови, методи визначення оптимальних параметрів мережі, таких як кількість шарів, розмір шару, топологія, основні методи представлення вхідного сигналу та їх переваги та недоліки. Одним із найкращих (на практиці) способів представити вхідний сигнал є вибір знімка векторного поля зображення. На відміну від точкових знімків, для цього методу потрібна менша мережа і, отже, менший обсяг Обчислювальний Однак цей метод вимагає більш складної попередньої обробки. Нейронна мережа виконує завдання класифікації самостійно. У багатовимірному просторі нейронна мережа вибирає області, які відповідають деяким екземплярам класу [10, 11].

Головною перевагою нейромережевих структур є їх здатність до навчання. Крім того, вони можуть дуже успішно звертатися до класифікацій об’єктів, які не очікуються під час навчання, оскільки вони наближають об’єкти відповідно до відомих критеріїв.

Недоліками нейромережевих структур є низька точність розпізнавання та високі обчислювальні вимоги (у випадку мережевого моделювання на основі архітектури фон Неймана, тому що зазвичай потреба в швидкості

процесора низька, але кількість процесорів важлива). Низька точність також є великим недоліком.

У статті представлено метод побудови та використання нейронних мереж, які ідентифікують не лише первинний вихід нейронної мережі, але й вторинні виходи, які можуть вказувати неправильний напрямок. Такі помилки можна обробляти на етапі постобробки, що зменшить ризик помилок.

Незважаючи на свої недоліки, нейронні мережі є одним із найпотужніших об’єктів для розпізнавання рукописних символів [6]. Структури нейронної мережі дуже лояльно ставляться до незнайомих зображень і намагаються класифікувати їх як найближче знайоме представлення. На практиці людина вводить символ, який дуже схожий на оригінал, але додає деякі помилки нелінійно. У разі розпізнавання друкованого тексту помилка може бути апроксимована як лінійна або нелінійна, якщо не враховувати оптичні перешкоди та шуми. В обох випадках нейронна мережа змогла знайти деякі схожі наближення, тоді як інші методи могли взагалі не розпізнати символи.

Враховуючи наведені методи та їх недоліки, можна зробити висновок, що проблема ідентифікації все ж потребує більш швидкого та точного вирішення. Найпопулярніші та практичні програми використовують регулярність у символах для ідентифікації, що дає високі показники ідентифікації, такі як швидкість або точність, але може стати безсилою у разі перешкод або втрати частини інформації.

Необхідно розробити програму, яка є повністю адаптивною та готовою працювати з будь-яким набором символів в автоматичному режимі. На практиці компанії отримують індивідуальні продукти, в яких програми оптимізовані та доопрацьовані під конкретні завдання. Однак, якщо озирнутися в історію, кожна технологічна революція містить нові можливості для автоматизації. Тому бажаний продукт повинен мати

можливість адаптуватися до потреб замовника, будучи при цьому повністю загальним програмним забезпеченням (тобто без втручання програміста).

1.2 Аналіз програмних засобів для розпізнавання символів на тлі завад

Першим аналогом є програма [12]. Це програма для оптичного розпізнавання тексту (OCR - Optical Character Recognition). Він призначений для перетворення сканованих документів, PDF-документів і файлів зображень документів (включно з цифровими фотографіями) у формат. Недоліком такої схеми є низька надійність розпізнавання та висока вартість.

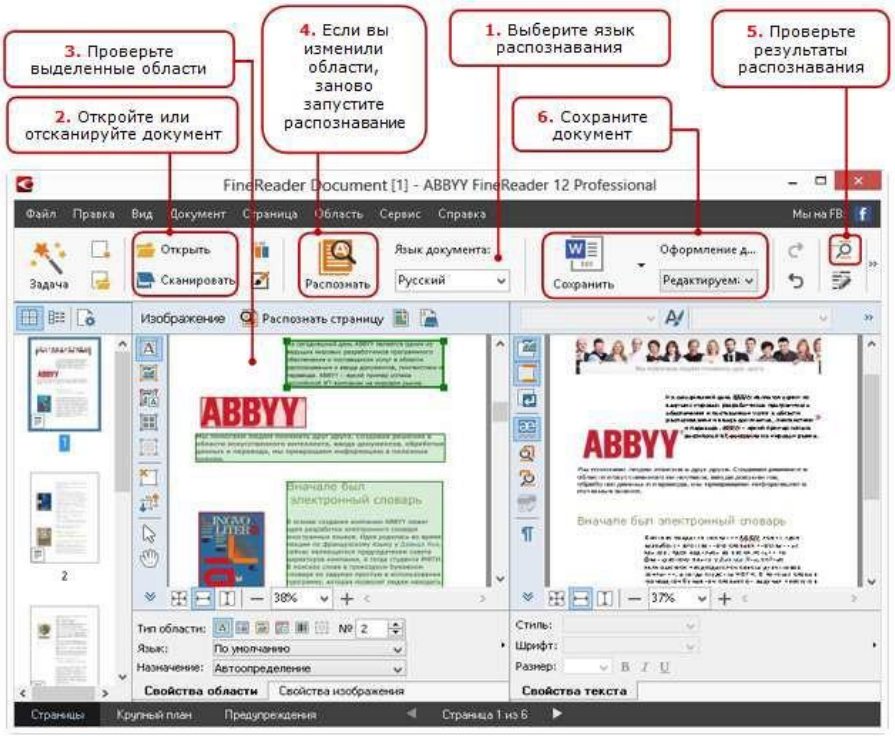


Рисунок 1.1 –Програма ABBYY FineReader

Другим аналогом є програма CueiForm (див. рис. 1.2) [13]. Ця програма була створена компанією Cognitive Technologies для швидкого та високоякісного сканування та розпізнавання документів (роздруківок, фотокопій) як тексту. Щоб розпізнати текст найвищої якості на роздруківках і відсканованому папері, програма використовує спеціальний алгоритм OCR

(оптичне розпізнавання символів), який дозволяє легко та швидко визначати мови, шрифти тощо. Недоліком цієї програми є низька надійність розпізнавання, вона може розпізнавати лише друкований текст, а не рукописний.

Рисунок 1.2 – Програма CuneiForm

Третьою програмою моделювання є програма MeOCR 1.0 (див. рис. 1.3) [14]. Програма використовує OCR для перетворення сканованих документів у текстові документи, які можна редагувати, і експорту їх у Microsoft Word одним клацанням миші. Його можна використовувати для економії часу, коли вам потрібно перекласти скановані документи в MS Word.

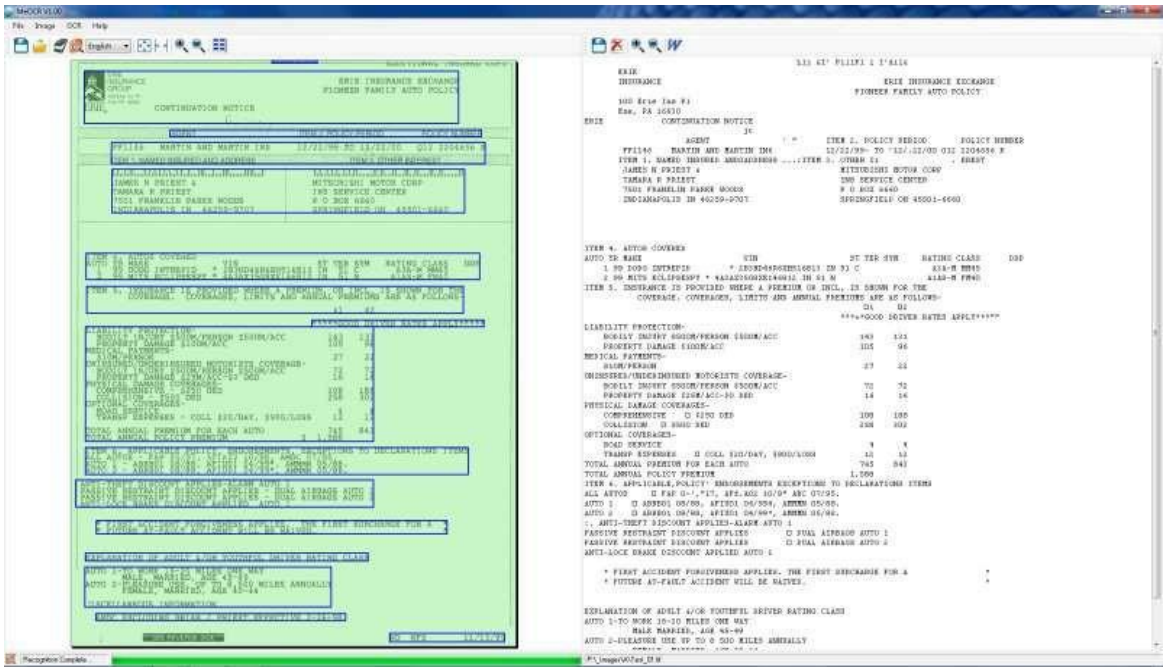


Рисунок 1.3 – Програма MeOCR 1.0

Програма дуже точна і економить час за рахунок зменшення кількості виправлень і необхідних правок. Програма також зберігає формат (більшість програм OCR цього не робить). Він підтримує кілька мов: болгарську, хорватську, чеську, датську, голландську, англійську, естонську, французьку, німецьку, угорську, італійську, латвійську, литовську, польську, португальську, румунську, російську, сербську, словенську, іспанську, шведську, турецьку, українську. Відскануйте зображення або завантажте зображення з файлу. Підтримуються всі основні формати файлів. BMP, TIFF, JPEG, GIF, PNG. Крім того, програма підтримує колір і може розпізнавати текст на більшості зображень, навіть кольорових. Дозволяє визначити лише частину зображення, вибравши область за допомогою миші. Підтримка пристроїв Twain: зберігайте зображення з будь-якого пристрою Twain, включаючи камери. Також доступна підтримка MS Word: експорт документів MS Word для легкого редагування та форматування. І дозволяє зберігати зображення у форматах BMP, TIFF, JPEG, GIF, PNG. Вбудований текстовий редактор: дозволяє редагувати документи в Me OCR.

Тому в якості розробленої процедури моделювання було обрано процедуру Me OCR [14]. Крім певних переваг, моделювання має і недоліки.

В основному надійність розпізнавання рукописних символів невисока, особливо коли на зображенні є перешкоди, програма на диску велика, програма перевантажена «зайвими функціями», а інтерфейс незручний.

1.3 Постановка задачі дослідженя

Комп’ютери та комп’ютери розроблені для заміни людей, наприклад, у сферах, де люди виконують свої повсякденні завдання. Сьогодні більшість проблем вирішено, і повсякденні обчислення майже завжди виконуються на ПК. Комп’ютери можуть виконувати прості арифметичні операції швидко та без помилок. Але деякі процеси так і не вирішені до кінця. Особливо це стосується неоднозначних проблем, які вимагають розумних рішень.

Одним із таких завдань є розпізнавання тексту [4]. Розроблено сотні програм, здатних розпізнавати текст. Найкраща розробка використовує дуже прості методи, але завжди вузькоспеціалізована. Тому існують програми, які розробляють систему пошуку областей шаблону для кожного шрифту. Такий метод працює і діє дуже швидко і ефективно. Але за якістю розпізнавання звичайна людина легко перевершить такі системи.

Людина, використовуючи свій мозок і асоціативну пам'ять, бачить зображення в цілому, використовуючи знання про контекст. Ця додаткова (хоча й незначна) вартість дозволяє досить легко впоратися з розпізнаванням слів і речень, навіть якщо частина зображення пошкоджена. Тому комп'ютеру достатньо знищити 2-3 символи, і результат розпізнавання не задовольнить користувача. Хоча людині легко зрозуміти сенс речення: «Тут якісь дивні символи». Розпізнавання зображень або тексту є або може використовуватися для оптимізації роботи в багатьох сферах. Наприклад, для оцифрування бібліотечних документів, для сканування документів, для автоматичної перевірки. Хоча можна попередньо обробити та підготувати інформацію для сканування, є багато випадків, коли це важко або неможливо використовувати попередню обробку.

Наприклад, автоматизовані системи використовуються для автоматизації реєстрації пасажирів, сканування паспортів для пошуку квитків. Така система має зручний інтерфейс, який не ускладнює процедури для пасажирів, але дозволяє компаніям використовувати менше персоналу. Такі системи успішно працюють у багатьох авіакомпаніях.

Очевидно, що для ефективності дуже важливо, щоб якість системи була якомога вищою, оскільки інформацію неможливо ідентифікувати під час її читання, інакше можливі наслідки помилки можуть призвести до того, що клієнт отримає не той квиток. В інших випадках такі випуски можуть стосуватися цінних паперів і мати більш серйозні наслідки.

Однак ми зосередимося на загальних системах розпізнавання зображень, а також тексту, які виконуватимуть розпізнавання змістовного тексту, але також будуть готові розпізнавати окремі семантично нерелевантні символи.

Така система є більш загальною і може бути налаштована під конкретного замовника, крім того, вона має можливість навчитися розпізнавати друкований текст і Розпізнавання рукописного тексту.

Магістерська робота в цьому розділі стосується недостатньої надійності сучасних процесів розпізнавання символів у контексті перешкод. Розглянуто основні методи та програмні засоби розпізнавання символів у контексті перешкод та визначено наступні недоліки:

1) Надійність розпізнавання недостатня, тобто порівняно небагато символів правильно розпізнається за наявності перешкод у зображенні символу, 2) Низька швидкість розпізнавання, що пов’язано з послідовним характером порівняння символів із посилання.

Для усунення цих недоліків запропоновано метод розпізнавання символів на фоні перешкод на основі штучної нейронної мережі. На основі цього методу необхідно розробити інформаційну технологію, впровадити ІТ-програмне забезпечення, перевірити виконання вимог до програмних засобів та усунути встановлені недоліки.

2 РОЗРОБКА ДОДАТКУ ДЛЯ РОЗПІЗНАВАННЯ СИМВОЛІВ НА ТЛІ ЗАВАД

2.1 Розробка методу для розпізнавання символів на тлі завад

Розпізнавання символів є дуже складним завданням як в теоретичному, так і в практичному сенсі, хоча багато організмів, включаючи людину, можуть впоратися з нею швидко і досить точно. Для ефективного виконання цього процесу досить складно створити штучну систему і реалізувати її технічно. Розпізнавання образів — це порівняння властивостей об’єкта із зображенням об’єкта.

Прикладом розпізнавання зображень може бути система розпізнавання тексту та окремих символів, біометричних параметрів людей, штрих-кодів, номерів транспортних засобів тощо. Система розпізнавання тексту передбачає, що на вході є зображення з текстом (в заданому графічному форматі). Текст повинен генеруватися на виході системи. Загальна архітектура такої системи складається з наступних послідовних операцій [3, 5]:

1. Коли вводиться зображення, спочатку потрібно видалити шум і зменшити його до форми, яка дозволяє ефективний вибір символів та їх ідентифікацію.

2. Система повинна сегментувати зображення на текстові блоки відповідно до характеристик зображення.

3. Зазвичай система повинна розкладати блоки тексту на слова, а слова – на символи, оскільки розпізнавання символів простіше, ніж розпізнавання цілих слів.

4. Після цього система виконує певний алгоритм ідентифікації. Саме розпізнавання символів може виконуватися різними специфічними методами. Ви можете порівнювати з шаблонами, шукати фрагменти інформації,

очікувані для певного символу, використовувати нейронні мережі тощо. [15, 16].

Після ідентифікації вони отримують набір символів, з яких формується текст.

Існує кілька основних систем розпізнавання тексту [4, 16]. Принципи роботи таких систем базуються на різноманітних стратегіях, але в цілому алгоритми в загальному вигляді базуються на формулюванні гіпотез і їх подальшій перевірці. Крім того, швидкість алгоритму залежить від порядку припущень, тому часто важливо знайти найкращий порядок припущень на основі даних, отриманих із зображень.

OCR зазвичай застосовується до растрових зображень сторінки. Тим часом більшість систем створюють шаблони для різних шрифтів і контурів. Програма розпізнає шрифт і розпізнає його на основі шаблонів. У деяких випадках програмне забезпечення використовує числові значення деяких символів для визначення нових шрифтів [7].

Найпопулярніші програми розпізнавання тексту засновані на пошуку певних характеристик символів. Така система складається із сотень алгоритмів, кожен з яких шукає ідентифікаційний символ для певного символу. Такі системи іноді використовують принцип нечіткої множини для визначення ймовірності розпізнавання символу. Однак такі системи мають ряд недоліків. Відсутність одного пікселя або наявність регулярного чи нерегулярного шуму може значно погіршити якість розпізнавання.Нейронна мережа може бути класифікатором у системі розпізнавання тексту. Цей класифікатор можна навчити, регулюючи коефіцієнти на елементах мережі, щоб, залежно від вхідних даних, бажаний результат з’являвся на виході. Нейронні мережі можуть успішно використовуватися в системах розпізнавання образів. Суть мережі полягає в тому, що вона виділяє регіони в багатовимірному просторі всіх можливих вхідних даних. Наприклад, кожному регіону може відповідати символ. Якщо після навчання точка з цього простору прикладається до входу мережі (тобто будь-якого вхідного

сигналу), мережа намагатиметься визначити, до якого регіону належить сигнал. В результаті мережа здатна реагувати навіть на вхідні сигнали, з якими мережа ще не стикалася, а результатом роботи буде область, найближча до заданої точки. Тому точність мережі фактично залежить від якості навчання [10].

Одним з недоліків нейронних мереж є їх розмір. Зазвичай потрібна досить велика структура мережі. Однак ця складність зумовлена традиційними методами обробки інформації. Сьогодні більшість комп'ютерів складається з кількох дуже потужних процесорів, які можуть дуже швидко виконувати атомарні арифметичні операції. Нейронні мережі, по суті, складаються з великої кількості простих елементів, які просто виконують дію суматора з пороговою функцією. Однак таких елементів багато. Якщо порівняти біологічну нейронну мережу людини з сучасним процесором, людина виконує прості математичні операції, такі як множення або ділення, дуже повільно, хоча процесор виконує їх за мільярдні частки секунди. З іншого боку, достатньо 0,2 секунди, щоб зрозуміти, що картина написана в стилі кубізму, що для бінарного процесора є майже непосильним завданням.

Більш простим варіантом реалізації нейронної мережі є моделювання її на звичайному процесорі. Однак процесори обробляють інформацію послідовно, тому, хоча кожен елемент мережі незалежний і всі обчислення можуть виконуватися паралельно, час, необхідний для обробки одного вхідного сигналу, пропорційний розміру мережі [9, 11].

Існує багато типів нейронних мереж, які поділяються на одношарові та багатошарові.

Хоча один нейрон може виконувати найпростіші процедури розпізнавання, потужність нейронних обчислень походить від з'єднань нейронів у мережі. Найпростіша мережа складається з набору нейронів, які утворюють шар, як показано на правій стороні малюнка. 2.1. Вершини кола зліва використовуються лише для призначення вхідного сигналу. Вони не виконують обчислень і тому не вважаються шаром. З цієї причини вони

позначені маленькими кружками, щоб відрізнити їх від обчислювальних нейронів, позначених великими кружками. Кожен елемент у вхідному наборі X пов’язаний з кожним штучним нейроном за допомогою окремої ваги, і кожен нейрон виводить зважену суму своїх входів у мережу.

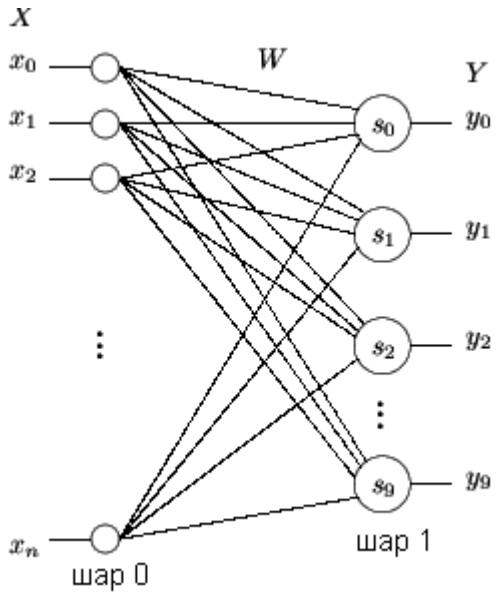


Рисунок 2.1 – Одношарова нейромережа

Великі та складні нейронні мережі зазвичай мають велику обчислювальну потужність. Хоча були створені мережі будь-якої мислимої конфігурації, ієрархічна організація нейронів імітує ієрархічну структуру певних областей мозку. Такі багаторівневі мережі мають більші можливості, ніж одношарові мережі.

Багатошарові мережі можуть бути сформовані з каскадних шарів. Вихід одного шару є входом наступного шару. Аналогічна мережа показана на малюнку. 2.2 Лише коли функція активації між рівнями є лінійною, багаторівнева мережа може підвищити обчислювальну потужність порівняно з однорівневою мережею. Обчислення результату шару складається з множення вхідного вектора на першу вагову матрицю, а потім множення результуючого вектора на другу вагову матрицю (якщо немає нелінійної функції активації) −

Оскільки множення матриці є асоціативним, то це показує, що двошарова лінійна мережа еквівалентна одному шару з ваговою матрицею, яка дорівнює добутку двох вагових матриць. Тому будь-яку багатошарову лінійну мережу можна замінити еквівалентною однорівневою мережею, але обчислювальна потужність одношарової мережі досить обмежена. Тому для розширення можливостей мережі в порівнянні з однорівневими мережами потрібні нелінійні функції активації.

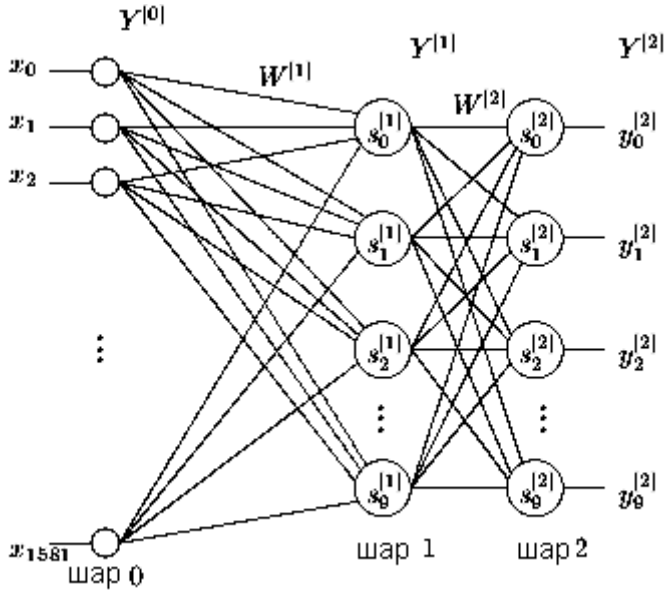


Рисунок 2.2 – Багатошарова нейромережа

2.2 Обгрунтування вибору типу нейромережі для розпізнавання символів на тлі завад

Нейронні мережі справді є найкращими кандидатами для реалізації адаптивних рішень проблем класифікації. Однак для ефективного вирішення поставленої задачі необхідно вибрати певний тип мережі та проаналізувати існуючі варіанти реалізації нейронних мереж.

Для огляду було обрано чотири популярні варіанти реалізації нейронних мереж: нейронні мережі із зворотним поширенням помилок, нейронні мережі Хебба, нейронні мережі Хопфілда та нейронні мережі Хеммінга.

Аналіз кількох варіантів дозволить більш детально розглянути математичну основу процесу навчання нейронної мережі та прийняти правильні рішення щодо використання тієї чи іншої реалізації.

2.2.1 Нейромережі із зворотним поширенням помилки

Коли в мережі є тільки один рівень, його та навчені викладачами алгоритми очевидні, оскільки відомий правильний початковий стан нейронів в одному шарі, а налаштування синаптичних зв’язків здійснюється в напрямку мінімізації помилок у виході. мережі. Наприклад, за таким принципом побудований алгоритм навчання одношарового персептрона.

У багатошаровій мережі оптимальне вихідне значення нейронів у всіх шарах, крім останнього, зазвичай невідоме, і більше неможливо навчити персептрон із двома або більше шарами, тільки вихід штучної нейронної мережі. У розглянутій вище мережі відсутні зворотні зв'язки, тобто зв'язки від виходу певного рівня до входу того ж рівня або попереднього рівня. Такі спеціальні мережі називаються мережами без зворотного зв'язку або мережами прямого поширення. У мережах без зворотного зв'язку пам'ять відсутня, а їх вихід повністю визначається поточним входом і значеннями ваги.

Одним із варіантів вирішення цієї проблеми є розробка набору вихідних сигналів, відповідних вхідним сигналам кожного рівня ШНМ, звичайно, це дуже трудомістка операція і не завжди здійсненна [10, 11]. Другий варіант - це динамічне коригування синаптичних ваг, під час якого зазвичай вибирається найслабший зв'язок і вносяться невеликі зміни в один або інший бік, і лише ті зміни, які призводять до зменшення помилки виведення, зберігаються по всій мережі. Третій варіант полягає в поширенні сигналу помилки від виходу ШНМ до його входу в напрямку, протилежному прямому поширенню сигналу в нормальному режимі роботи. Цей алгоритм навчання ШНМ називається процесом зворотного поширення.

Відповідно до методу найменших квадратів цільовою функцією мінімізації похибки ШНМ є:

$$E(w) = \frac{1}{2}\sum_{j,p}(y_{j,p}^{(N)} - d_{j,p})^2 \quad (2.1)$$

додати всі нейрони та всі зображення до вихідного рівня,

обробляється мережею. Він мінімізується методом градієнтного спуску, тобто ваговий коефіцієнт регулюється наступним чином:

$$\Delta w_{ij}^{(n)} = -\mu \frac{\partial E}{\partial w_{ij}} \quad (2.2)$$

Тут коефіцієнт швидкості навчання:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial y_j}\frac{dy_j}{ds_j}\frac{\partial s_j}{\partial w_{ij}} \quad (2.3)$$

Тут - вихід нейрона - зважена сума його вихідних сигналів - параметрів функції активації. Оскільки множник є похідною функції по її аргументам, то похідну функції активації необхідно визначати по всій осі абсцис. У зв'язку з цим унімодальні функції та інші неоднорідні функції активації не підходять для розглянутої штучної нейронної мережі. Вони використовують функції згладжування, такі як гіперболічний тангенс або класична сигмоподібна з експонентою. У випадку гіперболічного тангенса:

$$\frac{dy_j}{ds_j} = 1 - s^2 \quad (2.4)$$

Третій множник, очевидно, дорівнює нейрону попереднього шару.
Перший множник легко розкладається наступним чином:

$$\frac{\partial E}{\partial y_i} = \sum_k \frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial s_k}\frac{\partial s_k}{\partial y_i} = \sum_k \frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial s_k} w_{jk}^{(n+1)} \tag{2.5}$$

Тут сумування по виконується серед нейронів шару.
Ввівши нову змінну

$$\delta_j^{(n)} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial s_j} \tag{2.6}$$

ми отримаємо рекурсивну формулу

$$\delta_j^{(n)} = \left[\sum_k \delta_j^{(n+1)} w_{jk}^{(n+1)}\right]\frac{\partial y_j}{\partial s_j} \tag{2.7}$$

Для вихідного шару

$$\delta_l^{(N)} = \left(y_l^{(N)} - d_l\right)\frac{dy_l}{ds_l} \tag{2.8}$$

Тепер ми можемо записати (2.6) в розкритому вигляді:

$$\Delta w_{ij}^{(n)} = -\mu\delta_j^{(n)} y_i^{(n-1)} \tag{2.9}$$

Іноді, щоб надати процесу корекції ваги деяку інерцію для згладжування різких стрибків при русі по поверхні цільової функції, доповнюють (2.9) значенням зміни ваги на попередній ітерації

$$\Delta w_{ij}^{(n)}(t) = -\mu(\rho \Delta w_{ij}^{(n)}(t-1) + (1-\rho)\delta_j^{(n)} y_i^{(n-1)}) \qquad (2.10)$$

Таким чином, повний алгоритм навчання ШНМ з використанням процесу зворотного поширення помилок виглядає наступним чином:

1. Нанесіть одне з можливих зображень у нормальному режимі роботи нейронної мережі на вхід мережі та обчисліть значення останнього при поширенні сигналу від входу до виходу. ми знаємо

$$s_j^{(n)} = \sum_{i-0}^{M} y_i^{(n-1)} w_{ij}^{(n)} \qquad (2.11)$$

$$y_j^{(n)} = f\left(s_j^{(n)}\right), \text{де } f(x) - \text{ сигмоід} \qquad (2.12)$$

$$y_q^{(0)} = I_q \qquad (2.13)$$

2. Розрахувати для вихідного шару по формулі (2.8). Розрахувати по формулі (2.9) або (2.10) зміни ваг шару .

3. Розрахувати по формулах (2.6) і (2.9) (або (2.8) і (2.10)) відповідно.

4. Скоригувати всі ваги в ШНМ

$$w_{ij}^{(n)}(t) = w_{ij}^{(n)}(t-1) + \Delta w_{ij}^{(n)}(t) \qquad (2.14)$$

5. Якщо помилка мережі серйозна, перейдіть до кроку 1. В іншому випадку кінець.

На кроці 1 мережа чергує всі навчальні зображення у випадковому порядку, щоб мережа не забула деякі зображення, як запам’ятовує інші.

З рівняння (2.12) видно, що коли початкове значення наближається до нуля, ефект навчання значно зменшується. Використовуючи бінарні вхідні вектори, в середньому половина вагових коефіцієнтів не буде скоригована, тому бажано перемістити діапазон можливих значень виходу нейрона [0,1] до межі [-0,5,+0,5], тобто. шляхом застосування логістичної функції до простої модифікації для досягнення. Наприклад, сигма з експонентою перетворюється у форму

$$f(x) = -0.5 + \frac{1}{1 + e^{-ax}} \qquad (2.15)$$

У процесі навчання може виникнути ситуація, коли велике позитивне або негативне значення вагового коефіцієнта перемістить робочу точку на сигмовиді багатьох нейронів в область насичення. Відповідно до (2.7) і (2.8), мале значення похідної логістичної функції призведе до зупинки навчання, таким чином паралізуючи ШНМ. По-друге, застосування градієнтного спуску не гарантує, що буде знайдено глобальний мінімум цільової функції, а не локальний мінімум. Ця проблема пов'язана з іншою проблемою - вибором значення швидкості навчання. Доказ конвергенції навчання під час зворотного поширення базується на похідних, тобто збільшенні ваг, тому швидкість навчання має бути нескінченно малою, але в цьому випадку швидкість навчання буде неприйнятно низькою. З іншого боку, занадто велика корекція ваги призведе до постійної нестабільності процесу навчання. Тому в якості η зазвичай вибирають число менше 1, але не надто мале, наприклад 0,1, яке можна поступово зменшувати в процесі навчання. Крім того, щоб виключити випадкові попадання в локальні мінімуми, іноді, після стабілізації значення вагових коефіцієнтів, η короткочасно сильно

збільшується, щоб почати градієнтний спуск з нової точки. Якщо багаторазове повторення цього процесу переводить алгоритм у той самий стан ШНМ, ми можемо з більшою чи меншою впевненістю сказати, що знайдено глобальний максимум, і нічого більше.

Існує інший спосіб усунення локальних мінімумів, який також усуває параліч ШНМ, і він передбачає застосування випадкових ШНМ.

Алгоритм навчання нейронної мережі з використанням процесу зворотного поширення означає, що існують деякі зовнішні зв’язки, які надають мережі вхідні та цільові вихідні зображення. Алгоритми, які використовують схожі концепції, називаються алгоритмами навчання з наставником. Для його успішної роботи необхідно, щоб експерти на попередньому етапі створили еталонні виходи для кожного вхідного зображення. Наприклад, навчання людського мозку, на перший погляд, відбувається без вчителя: зорові, слухові, тактильні та інші рецептори сприймають інформацію ззовні і здійснюють своєрідну самоорганізацію всередині нервової системи. Проте беззаперечно, що в житті є чимало вчителів – як у прямому, так і в переносному значенні – які координують зовнішні впливи.

Основна характеристика навчання без вчителя – його «самостійність». Процес навчання з викладачем передбачав регулювання ваги синапсів. Деякі алгоритми змінюють структуру мережі, тобто взаємозв'язок нейронів і навіть їх кількість, це перетворення називається самоорганізацією. Зрозуміло, що коригування синапсу може здійснюватися лише на основі інформації, наявної в нейроні, а саме його стану та вагових коефіцієнтів, які вже існують. На основі цих міркувань і за аналогією з відомим принципом самоорганізації нейронів був створений алгоритм навчання Хебба.

2.2.2 Нейромережі Хебба

За Хеббом, нервові клітини головного мозку з'єднані між собою великою кількістю прямих і зворотних збудливих зв'язків, утворюючи нейронну мережу. Кожен нейрон просторово-часово підсумовує сигнали від пошкодженого нейрона, щоб визначити потенціал через його мембрану. Нейрони спрацьовують, коли потенціал мембрани перевищує порогове значення. Нейрони вогнестійкі та стійкі до втоми. Ефективність зв'язків може змінюватися під час роботи мережі, збільшуючись між порушеними нейронами. Це змушує нейрони об’єднуватися в групи клітин — ті, які найчастіше порушуються разом — і змушує групи клітин відокремлюватися одна від одної. Коли ансамбль порушується, він повністю збуджений. Різні ансамблі можуть перекриватися: той самий нейрон може входити до різних ансамблів. Електрична активність мозку зумовлена послідовним збудженням єдиного цілого [10, 15].

Метод вивчення сигналу Хебба полягає у зміні вагових коефіцієнтів відповідно до наступних правил:

$$w_{ij}(t) = w_{ij}(t-1) + \alpha y_i^{(n-1)} y_j^{(n)} \qquad (2.16)$$

Існує також і диференціальний метод навчання Хебба:

$$w_{ij}(t) = w_{ij}(t-1) + \alpha\left[y_i^{(n-1)}(t) - y_i^{(n-1)}(t-1)\right]\left[y_j^{(n)}(t) - y_j^{(n)}(t-1)\right] \qquad (2.17)$$

З рівняння (2.17) можна побачити, що синапси, що з’єднують ці нейрони, вихід яких найбільше змінюється в напрямку збільшення, навчаються найбільше.

Повний алгоритм навчання за наведеною вище формулою виглядатиме так:

1. Під час фази ініціалізації всім вагам присвоюються невеликі випадкові значення.

2. Вхідне зображення подається на вхід мережі, сигнал збудження поширюється по всіх шарах за принципом класичної прямої мережі, тобто для кожного нейрона обчислюється зважена сума його входів і потім застосовується функція активації (передачі) нейрона, результатом є її початкове значення.

3. Змінити ваговий коефіцієнт відповідно до початкового значення нейрона, отриманого за формулою (2.16) або (2.17).

4. Цикл - перехід до кроку 2, поки вихідне значення мережі не стабілізується до заданої точності.

Використання цього нового методу визначення завершення навчання відрізняється від того, що використовується для мереж зворотного поширення, оскільки значення налаштування синапсів практично необмежене. На другому кроці циклу чергуйте всі зображення у вхідному наборі.

Слід зазначити, що через випадковий розподіл ваг на фазі ініціалізації тип відповіді на кожен клас вхідних зображень невідомий заздалегідь і представлятиме будь-яку комбінацію станів нейрона на вихідному рівні. У той же час мережа здатна узагальнювати подібні зображення, відносячи їх до одного класу. Тестування навченої мережі дозволяє визначити топологію класів на вихідному рівні. Щоб відповіді навченої мережі було зручно представлено, мережу можна доповнити шаром, наприклад, за алгоритмом навчання одношарового персептрона, який необхідно змусити відображати початкові відповіді мережі в необхідних зображеннях. .

Інший алгоритм навчання без вчителя - алгоритм Кохонена - передбачає коригування синапсів на основі значень з попередніх ітерацій.

$$w_{ij}(t) = w_{ij}(t-1) + \alpha\left[y_i^{(n-1)} - w_{ij}(t-1)\right] \qquad (2.18)$$

Як видно з наведеної вище формули, навчання спрощено, щоб мінімізувати різницю між вхідним сигналом нейрона від виходу нейронів попереднього шару та ваговими коефіцієнтами його синапсів [8, 9].

Повний алгоритм навчання має приблизно таку саму структуру, що й метод Хебба, але на кроці 3 із усього шару вибирається нейрон, значення синапсу якого максимально подібне до вхідного зображення, а ваги регулюються відповідно до рівняння (2.18) лише для це. Це так зване схвалення може супроводжуватися гальмуванням усіх інших нейронів у шарі та введенням вибраних нейронів у насичений стан. Такі нейрони можна вибрати, наприклад, шляхом обчислення скалярного множення вектора вагових коефіцієнтів і вектора вхідних значень. Нейрон-переможець дає найбільший результат.

Іншим варіантом є обчислення відстані між цими векторами в p-вимірному просторі, де p — розмір вектора.

$$D_j = \sqrt{\sum_{i=0}^{p-1}(y_i^{(n-1)} - w_{ij})^2} \qquad (2.19)$$

де індекс нейрона n-го шару, i – індекс підсумовування нейрона (n-1)-го шару, вага синапсів, що з’єднують нейрони; вихід нейрона (n-1)-го шару – n-й шар вхідне значення. Немає необхідності знаходити корені в рівнянні (2.19), оскільки це просто інша відносна оцінка.

У цьому випадку «перемагає» нейрон з найменшою відстанню. Іноді нейрони примусово виключаються з розгляду через занадто часту сертифікацію, щоб «зрівняти» права всіх нейронів у шарі. Найпростіший варіант цього алгоритму полягає в придушенні тільки переможних нейронів.

При використанні навчання за алгоритмом Кохонена одним з підходів є нормалізація вхідного зображення та - на етапі ініціалізації - нормалізація початкових значень вагових коефіцієнтів.

$$x_i = x_i / \sqrt{\sum_{j=0}^{n-1} x_j^2} \qquad (2.20)$$

де — компоненти вектора вхідного зображення або вектора вагових коефіцієнтів, а n — його розмірність. Це дозволяє скоротити тривалість процесу навчання.

Ініціалізація вагових коефіцієнтів випадковими значеннями може призвести до злиття різних класів, що відповідають щільно розподіленим вхідним зображенням, або, навпаки, розбиття на додаткові підкласи у випадку близьких зображень одного класу. Щоб уникнути цього, використовуються опуклі методи комбінування. Його суть зводиться до того, що вхідне нормалізоване зображення трансформується:

$$x_i = \alpha(t) x_i + \big(1 - \alpha(t)\big) \frac{1}{\sqrt{n}} \qquad (2.21)$$

де коефіцієнти змінюються від нуля до одиниці під час процесу навчання, тому спочатку майже ідентичні зображення застосовуються до

входу мережі, а з часом вони все більше зближуються до оригінального зображення. Коефіцієнт ваги встановлюється на етапі ініціалізації як

$$w_o = \frac{1}{\sqrt{n}} \tag{2.22}$$

де n - розмірність вектора ваги для нейронів ініціалізованого шару.

На основі описаного вище методу будується особливий тип нейронної мережі — так звана самоорганізована структура із синаптичним налаштуванням — як самоорганізована карта ознак. Для них після вибору нейрона j з найменшою відстанню від n-го шару (2.19) за формулою (2.18) навчається не тільки цей нейрон, а й його сусіди по R. Значення R дуже велике на першій ітерації, щоб усі нейрони були навчені, але з часом воно зменшується до нуля. Тому чим ближче до кінця навчання, тим точніше можна визначити групу нейронів, що відповідає кожному класу зображень.

2.2.3 Нейромережі Хопфілда

Серед різноманітних конфігурацій штучних нейронних мереж деякі конфігурації класифікуються за принципом навчання. У такій мережі ваги синапсів обчислюються тільки один раз перед початком роботи мережі на основі інформації оброблених даних, і все навчання мережі зводиться до цього обчислення [11]. З одного боку, представлення попередньої інформації можна розглядати як допомогу вчителю, але з іншого боку, мережа насправді просто запам’ятовує зразки до того, як на її вхід надходять реальні дані, і не може змінити свою поведінку, тому ми можемо говорити про зв'язок зі "світом" (вчителем) зв'язки петлі зворотного зв'язку не доведені. Серед мереж зі схожою логікою роботи найбільш відомими є мережа Хопфілда і

мережа Хеммінга, які зазвичай використовуються для організації асоціативної пам'яті.

Структурна схема мережі Хопфілда показана на малюнку 1. 2.3 Він складається з одного шару нейронів, кількість яких є одночасно кількістю входів і виходів мережі. Кожен нейрон з’єднаний з усіма іншими нейронами синапсами, а також має вхідний синапс, через який надходять сигнали. Вихідний сигнал зазвичай генерується на аксоні.

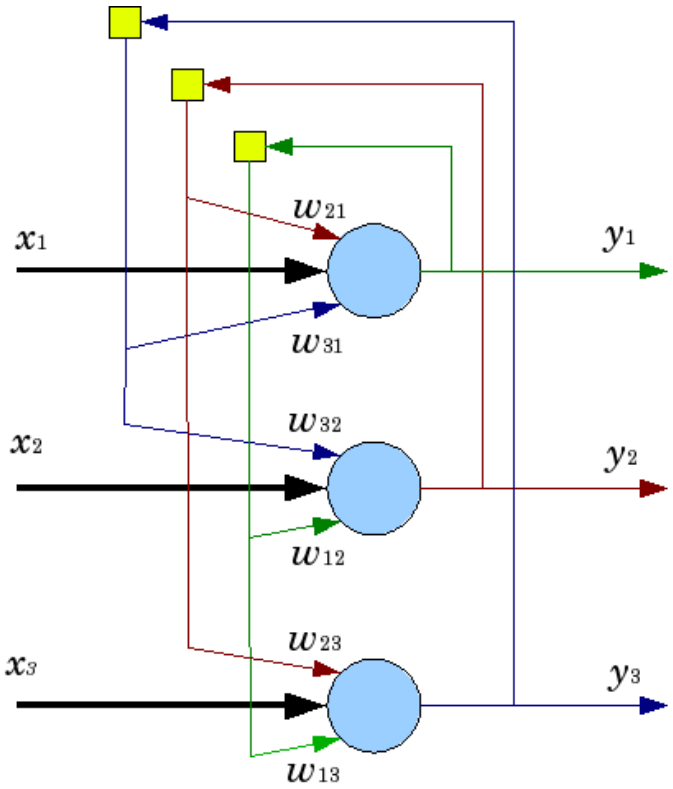


Рисунок 2.3 –Схема нейромережі Хопфілда

Наприклад, якщо сигнал являє собою якесь зображення, то, відобразивши дані з мережевого виходу в графічній формі, ви зможете побачити картинку, яка точно відповідає одному з прикладів (у разі успіху) або « вільна імпровізація» мережі (в невдачі).

Під час фази ініціалізації мережі синаптичні ваги встановлюються наступним чином:

$$w_{ij} = \begin{cases} \sum_{k=0}^{m-1} x_i^k x_j^k, i \neq j \\ 0, \qquad\qquad i = j \end{cases} \tag{2.23}$$

Алгоритм роботи мережі такий (p – кількість ітерацій):

1. На вхід мережі подається невідомий сигнал. Насправді його введення здійснюється шляхом безпосередньої установки величини аксона:

$$y_i(0) = x_i, i = 0, \dots, n-1 \tag{2.24}$$

Тому вказівка вхідної синаптичної мережі в явному вигляді є суто умовною. Нуль у правій дужці означає нульову ітерацію в мережевому циклі.

2. Розрахувати новий стан нейрона

$$s_i(p+1) = \sum_{i=0}^{n-1} w_{ij} y_i(p), j = 0, \dots, n-1 \tag{2.25}$$

і нові значення аксонів

$$y_i(p+1) = f\left[s_j(p+1)\right] \tag{2.26}$$

де активаційна функція у вигляді стрибка, наведена на рис. 2.4.

3. Перевірте, чи змінилося початкове значення аксона на останній ітерації. Якщо так - переходимо до пункту 2, інакше (якщо вихід

стабілізувався) - закінчуємо. Тим часом вихідний вектор є зразком, який найкраще відповідає вхідним даним.

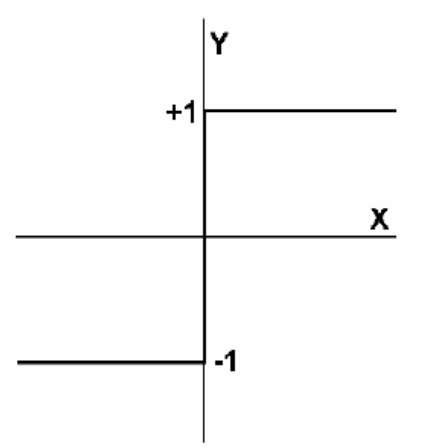


Рисунок 2.4 – Стрибкоподібна функція активації

Іноді мережа не розпізнає і виводить зображення, якого не існує. Це пов’язано з обмеженими можливостями мережі. Для мереж Хопфілда число m образів пам'яті не повинно перевищувати значення, приблизно рівне 0,15 * n. Крім того, якщо два зображення A і B дуже схожі, вони можуть викликати взаємну кореляцію в мережі, тобто поява вектора A на вході мережі спричинить появу вектора B на її виході, і навпаки.

2.2.4 Нейромережі Хеммінга

Коли мережі не потрібно відправляти вибірку явно, тобто достатньо отримати номер вибірки, і мережа Хеммінга успішно реалізує асоціативну пам'ять. Порівняно з мережею Хопфілда, мережа характеризується меншою вартістю пам'яті та обчислювальними зусиллями, що видно з її структури.

Мережа Хеммінга (рис. 2.5) складається з двох шарів. Перший і другий шари мають по m нейронів, де m — кількість вибірок. Кожен нейрон першого шару має n синапсів, підключених до входу мережі (утворюючи уявний нульовий шар). Нейрони другого шару з’єднані один з одним за допомогою гальмівних (негативний зворотний зв’язок) синаптичних зв’язків.

Кожен нейрон має унікальні синапси позитивного зворотного зв’язку, з’єднані з його аксонами.

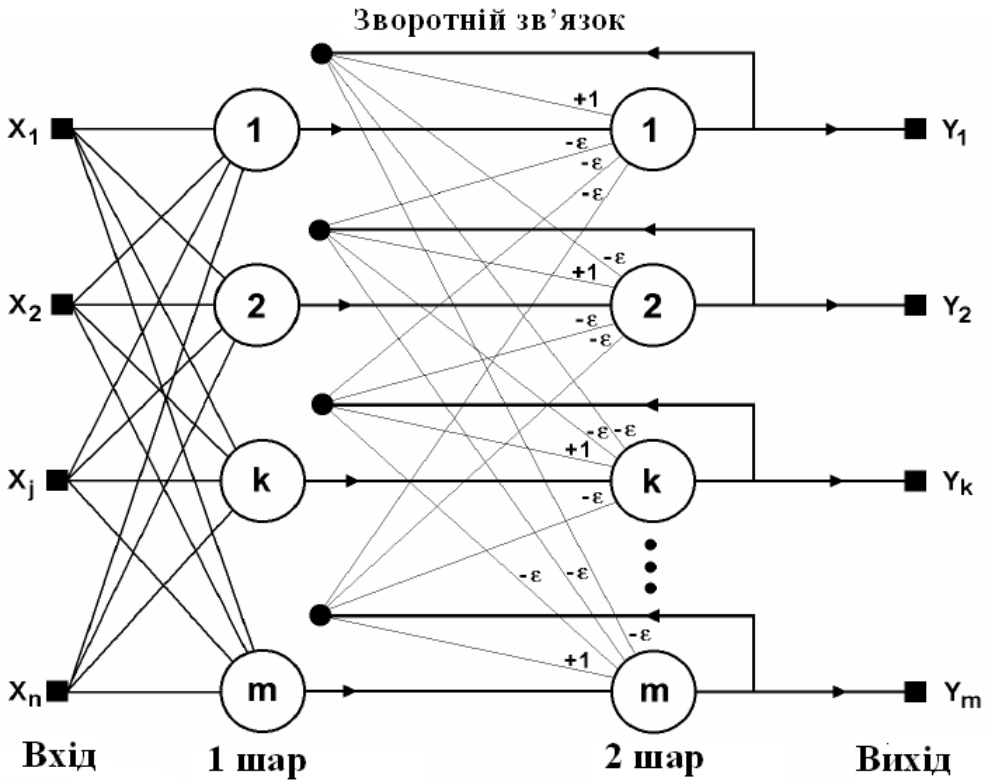


Рисунок 2.5 – Схема нейромережі Хеммінга

Ідея мережі полягає в тому, щоб знайти відстань Хеммінга від тестованого методу до всіх зразків. Відстань Хеммінга — це кількість бітів, які відрізняються у двох двійкових векторах. Мережа повинна вибрати вибірку з найменшою відстанню Хеммінга від невідомого вхідного сигналу, в результаті чого активується лише один вихід мережі, що відповідає цій вибірці.

Під час фази ініціалізації вагам першого рівня та порогу функції активації присвоюються наступні значення:

$$w_{ik} = \frac{x_i^k}{2}, i = 0, \dots, n-1, k = 0, \dots, m-1 \quad (2.27)$$

$$T_k = \frac{n}{2}, k = 0, \dots, m-1 \quad (2.28)$$

Ваговий коефіцієнт другого шару гальмівних синапсів дорівнює певній величині. Синапс нейрона, з'єднаний з його аксоном, має вагу +1.

Алгоритм роботи мережі Хеммінга виглядає наступним чином:

1. Невідомий вектор X подається на вхід мережі

, який обчислює стан нейронів першого шару (верхній індекс у дужках вказує кількість шарів):

$$y_j^{(1)} = s_j^{(1)} = \sum_{i=0}^{n-1} w_{ij} x_i + T_j \quad , j = 0, \dots, m-1 \quad (2.29)$$

Після цього отриманим значенням ініціалізуються значення аксонів другого шару:

$$y_j^{(2)} = y_j^{(1)}, j = 0, \dots, m-1 \quad (2.30)$$

2. Обчислити нові стани нейронів другого шару:

$$s_j^{(2)}(p+1) = y_j(p) - \varepsilon \sum_{k=0}^{m-1} y_k^{(2)}(p), k \neq j, j = 0, \dots, m-1 \quad (2.31)$$

і значення їх аксонів:

$$y_j^{(2)}(p+1) = f\left[s_j^{(2)}(p+1)\right], j = 0, \dots, m-1 \qquad (2.32)$$

Функція активації має вигляд порогу, а значення F має бути достатньо великим, щоб будь-яке можливе значення параметра не призводило до насичення.

3. Перевірте, чи змінилися вихідні дані нейронів другого шару під час останньої ітерації. Якщо так, перейдіть до кроку 2. В іншому випадку кінець.

З оцінки алгоритму видно, що роль першого рівня досить умовна: значення його вагового коефіцієнта використовується один раз на кроці 1, і мережа більше не посилається на нього, тому перший рівень може бути повністю виключено з мережі (замінено матрицею вагових коефіцієнтів).

2.2.5 Вибір нейромережі для створюваного додатку

Розглянувши чотири типи мереж, ми виберемо ту, яка оптимізована для реалізації в системі розпізнавання образів. Для цього ми розглянемо найважливіші для нас критерії.

Оскільки система має працювати якомога ефективніше, потрібен мінімальний розмір мережі, оскільки навчання мережі є процесом, що вимагає обчислень.

Перший критерій – швидкість. Навчання мережі має вимагати якомога менше операцій. Для підвищення продуктивності можна додати додаткову вимогу: можливість тренуватися на цілочисельній арифметиці. Однак такий підхід знижує точність рішення, що негативно позначається на другому критерії: якості розпізнавання. Отже, давайте просто зосередимося на ефективності з точки зору найменшої операції ділення, множення та додавання (віднімання, оскільки ці дві операції еквівалентні з точки зору швидкості з точки зору комп’ютера) [11].

За першим критерієм найкращим типом мережі є традиційна мережа зворотного поширення помилок. Це демонструє той факт, що навчання можна оптимізувати, вибравши функцію активації з простими похідними. Таким чином ми зменшуємо кількість арифметичних операцій на процесорі. Інші типи мереж мають надто складні структури, які потребують додаткових витрат процесорного часу для врахування топології. Тому в якості методу навчання обрано алгоритм зворотного поширення помилок, а в якості типу мережі – багатошаровий персептрон.

Однак використання лише одношарової мережі призводить до поганого розпізнавання та якості навчання. Оскільки однією з вимог системи є постійна адаптація та швидке реагування на зміни, необхідно використовувати багатошарову мережу [10].

Тому була обрана багатошарова мережа з алгоритмом навчання: зворотне поширення помилок для реалізації.

Для покращення результатів розпізнавання буде використано семантичний фільтр на основі спеціальної структури (бази даних), яка здатна розпізнавати ймовірність помилок введення.

2.3 Модифікація методу подання зображення до нейромережі

Існує кілька варіантів подачі зображення на вхід структури нейронної мережі. Керуючись критеріями швидкості, необхідно вибрати спосіб, який мінімізує розмір мережі.

Одним із варіантів обробки зображень є використання сіткової апроксимації. Для цього вихідне зображення розбивається на блоки. Далі кожен блок отримує певне значення, зазвичай 0 або 1, де 1 означає, що блок містить частину символу, а 0 означає, що блок порожній. Приклад наведено на рис. 2.6.

Чим вище роздільна здатність сітки, тим точніші результати дає цей метод. Однак навіть при розмірі 16X16 на вході нейронної мережі знаходиться 256 пікселів, або 256 сигналів.

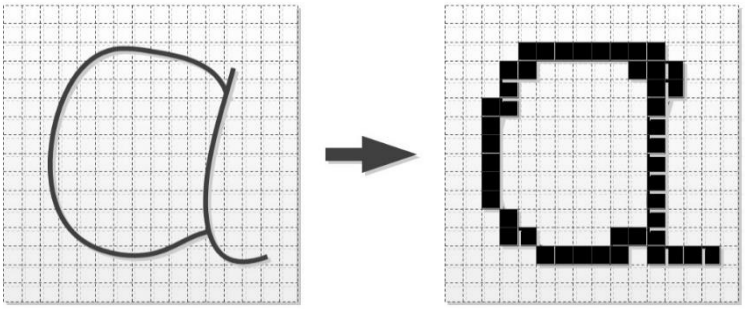


Рисунок 2.6 – Пікселізація зображення

Отже, такий підхід, очевидно, призводить до збільшення розміру мережі, тому слід розглянути інший підхід.

Другий варіант — використання векторизації зображення. У цьому випадку зображення символу перетворюється в еквівалентний вектор, а векторне сумарне перетворення подається в мережу. Приклад такого процесу показаний на Фіг.1. 2.7.

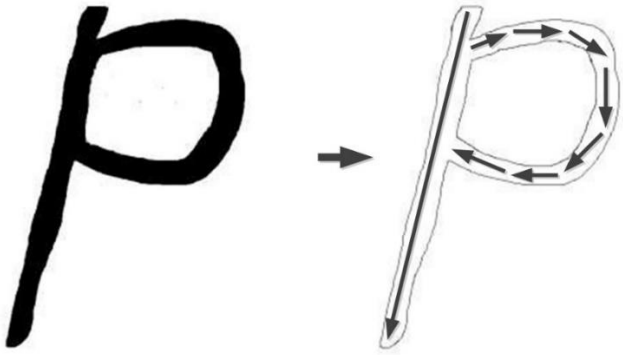


Рисунок 2.7 – Векторизація зображення

Такий підхід зменшує розмір мережі. Проте векторизація – досить складний процес, який вимагає тривалої обробки зображення. Тому в якості методу була обрана модифікація методу векторизації.

Для цього пікселізація буде виконана перед векторизацією, а векторизація буде виконана з використанням положення центру пікселя, максимально оптимізуючи кількість векторів. Такий підхід лише трохи збільшує кількість векторів порівняно з процесом гнучкої векторизації.

Але головне те, що, на відміну від пікселізації, для введення нейронної мережі не використовуються непотрібні блоки, тому розмір вхідного шару зберігається на належному рівні.

2.4 Архітектура та модель розпізнавання символів на тлі завад

Для успішного та швидкого впровадження інтелектуальних технологій необхідно розробляти інформаційні системи багаторівневим способом. На першому етапі ми розглянемо основні робочі компоненти та опишемо більш детально, як саме необхідні компоненти можуть бути реалізовані.

Розглядаючи блоковий поділ інформаційної системи, отримуємо її загальну архітектуру (рисунок 2.8)

Вхідні дані надходять безпосередньо в блок попередньої обробки. Цей блок розбиває зображення, вибирає символи та передає області на вхід блоку розділення завдань. Цього можна досягти за допомогою простого алгоритму обертання та вибору. Так, наприклад, достатньо знайти лінію поділу лінії і обробити лінію за досить простим принципом - поблочно виділяти символи/прогалини.

Блок розділення завдань вирішує, які символи належать до якого слова, і передає набори символів слово за словом, які далі обробляються нейронною мережею в блоки для розпізнавання. Блок розділення також повинен вирішити, які символи слід вважати спотвореними, надто шумними тощо.

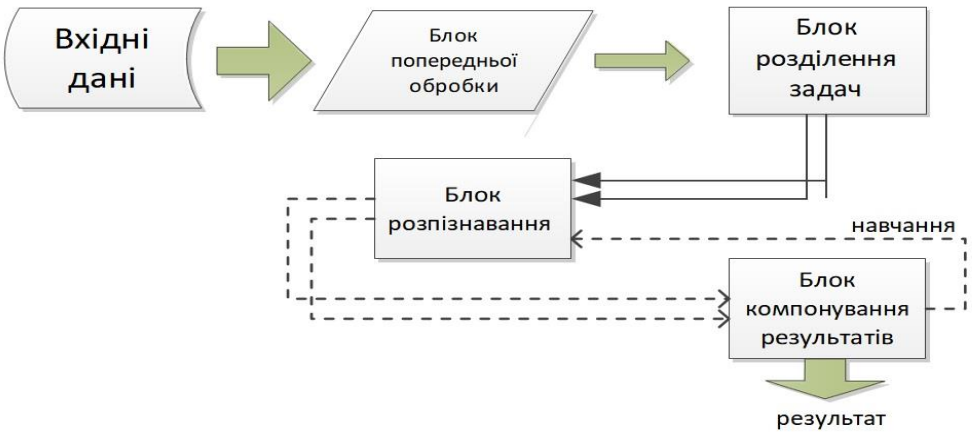


Рисунок 2.8 – Архітектура додатку розпізнавання символів на тлі завад

Наступний блок – блок ідентифікації. Його завдання — визначити нечітку множину відповідності між зображенням і певним символом на основі заданого символу [17]. Блок розпізнавання повинен бути реалізований за допомогою нейронної мережі, що дозволить його перенавчати під час роботи. Ідентифікаційні дані передаються до наступного блоку, блоку синтезу результату, у формі заданого вектора.

Блок шукає в базі даних споріднені слова за знайденими словами, формуючи результати. Пошук реалізований на основі рекурсивного алгоритму. Для ефективної реалізації вам потрібно додати кешування та відсікти непотрібні рішення. Результати повертаються користувачеві у вигляді розпізнаних слів. Однак блоки компонування виконують іншу мету.

Після розпізнавання слова блок передає цю інформацію блоку розпізнавання, щоб той, у свою чергу, перевірив себе та виправив свої помилки. Тут дуже важливо, щоб блок розпізнавання був реалізований у вигляді нейронної мережі з усіма необхідними властивостями.

Для навчання мережі буде використовуватися алгоритм зворотного поширення. Для цього використовується математичний інструментарій, описаний у підрозділі 2.2.1.

тобто визначити похибку обробки:

$$E(w) = \frac{1}{2}\sum_{j,p}(y_{j,p}^{(N)} - d_{j,p})^2 \quad (2.33)$$

На основі якої виконується градієнтний спуск для налаштування ваг мережі:

$$\Delta w_{ij}^{(n)} = -\mu \frac{\partial E}{\partial w_{ij}} \quad (2.33)$$

Для внутрішнього шару використовується обрахунок аналогічним чином:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial y_j}\frac{dy_j}{ds_j}\frac{\partial s_j}{\partial w_{ij}} \quad (2.34)$$

Очевидно, вам потрібно вибрати функцію активації.

Наступні типи сигмоїди найчастіше використовуються як функції активації:

функція Фермі (експоненціальна сигмоїда),

раціональна сигма,

Гіперболічний тангенс.

Час обробки для обчислення раціональної сигмоїдної функції активації є найкоротшим з усіх сигмоїд. Обчислення гіперболічного тангенса займає найбільше процесорних циклів. Сигмоїда дуже повільно обчислюється порівняно з функцією активації порогу. Якщо ви можете розпочати порівняння з деяким значенням (порогом) відразу після підсумовування в пороговій функції, то у випадку сигмоїдної функції активації вам потрібно обчислити сигмоїд (у кращому випадку для виконання трьох операцій

потрібен час: за модулем, додавання та ділення) перед порівнянням із порогом (наприклад, нуль). Якщо врахувати, що всі прості операції обчислюються процесором приблизно за однаковий проміжок часу, то робота сумованої сигмоїдальної функції активації (яка займе стільки ж часу) буде повільнішою за порогову функцію активації на пропорції 1:4.

Враховуючи вищесказане, розрахунок буде виконано за допомогою функції Фермі, оскільки вона має просту похідну, її легко обчислити та підходить для ідентифікації нейронів.

Розглянемо підсистему розпізнавання слів.

Як зазначалося вище, у цьому випадку програма розпізнавання символів повинна повернути вектор нечітких наборів [17] для слів.

Вектор нечіткої множини подібності символів — це впорядкований набір нормалізованих нечітких множин, що представляє ймовірність того, що зображення відповідає символу в наборі всіх символів:

$$v = \begin{matrix} < \{(a: p_{1,1}), (b: p_{1,2}), \dots, (z: p_{1,n})\}, \\ (a: p_{2,1}), (b: p_{2,2}), \dots, (z: p_{2,n}) \\ \vdots \\ (a: p_{m,1}), (b: p_{m,2}), \dots, (z: p_{m,n}) > \end{matrix} \quad (2.35)$$

де a,b,…,z – символи алфавіту, а px,i – відповідно ймовірності відповідності x-вого зображення символа i-тому символу алфавіту.

Знання про відповідність між векторами та словами, отримані з бази даних, далі використовуються для визначення результатів розпізнавання та подальшого вдосконалення системи оптичного розпізнавання.

Давайте детальніше розглянемо слово база даних. Ця база може повністю складатися з найбільш часто вживаних слів.

Вектор нечіткої множини отримується з підсистеми керування [17]. Для кожного слова в базі даних ми обчислимо ймовірність того, що даний вектор відповідає певному слову.

Припустимо, що відсутність або наявність деяких додаткових символів погіршить ймовірність у λ разів. потім можна обчислити за рекурсивною формулою:

$$f(s,v) = \max(f(s.tail, v.tail) \cdot v.head[s.head], \lambda f(s.tail, v), \lambda(s, v.tail)) \quad (2.36)$$

$$f(s,[]) = \lambda f(s.tail, []) \quad (2.37)$$

$$f([], v) = \lambda f([], v.tail) \quad (2.38)$$

$$f([],[]) = 1 \quad (2.39)$$

де f(s,v) – ймовірність, що слово s відповідає вектору v. X.head – перший символ вектора/слова, а X.tail – вектор/слово без першого символу. [] – відповідно позначає пустий вектор/стрічку. v.head[x] – описує ймовірність першого символа вектора відповідати деякому символу x.

Як бачите, останні три рівняння є основою для рекурсії.

Ви можете скористатися простішою обробкою, просто видаливши префікси та суфікси або перевіривши слова. Усі ці способи можуть прискорити роботу алгоритму, але можуть некоректно чи неточно вирішити задачу визначення відповідностей.

Основні 100 000 найбільш вживаних слів англійської мови. Взагалі стільки слів зайві. Тому необхідно розробити методи оптимізації, тобто розбити слова на вживані групи і спочатку перевірити збіги з найбільш часто вживаними словами, а потім переходити на інший рівень. Наприклад, якщо на першому рівні знайдено слово з рейтингом вище певного порогу, пошук можна зупинити.

Цю оптимізацію можна покращити, створивши графіки подібності. Якщо слово має високу ймовірність відповідності, тоді ви можете розглянути кілька схожих слів, можливо, вони матимуть вищу ймовірність відповідності.

Такий граф потрібно побудувати лише один раз, тому складність побудови нас зовсім не цікавить. Ви можете використовувати просту побудову графіка на основі відстані Левенштейна. А поріг подібності дуже низький, наприклад різниця в 1-2 символи.

Зібравши разом усе, що було сказано в цьому розділі, ми маємо вже досяжну систему, яка буде виконувати щось подібне до асоціативного пошуку в базі даних.

Загалом слід зазначити, що алгоритм подібності є рекурсивним і повільним у виконанні, при простому аналізі швидкість його роботи може досягати O(3N). Для оптимізації рекомендується використовувати кеш результатів алгоритму, щоб пошук можна було зберегти на позначці O(N2). Іншим прискоренням є використання алгоритму відсікання на основі результатів. Це не покращить оцінки найгіршого випадку, але значно пришвидшить роботу в середньому випадку.

2.5 Складові додатку розпізнавання символів на тлі завад

Інформаційна технологія розпізнавання символів у контексті інтерференції з нейронними мережами на основі зворотного поширення помилок має дві складові: навчання та безпосередню роботу нейронної мережі (рис. 2.9). Сформулюємо фази цих компонентів.

Навчання (проводиться при завантаженні файлу з еталонними зображеннями символів для розпізнавання (шрифту)), складається з наступних етапів:

1. Процес створення нейронної мережі, яка повертає помилки.

2. Ітераційний процес обчислення значень вагових коефіцієнтів нейронів у кожному шарі мережі (пошук вагової матриці W кожного шару).

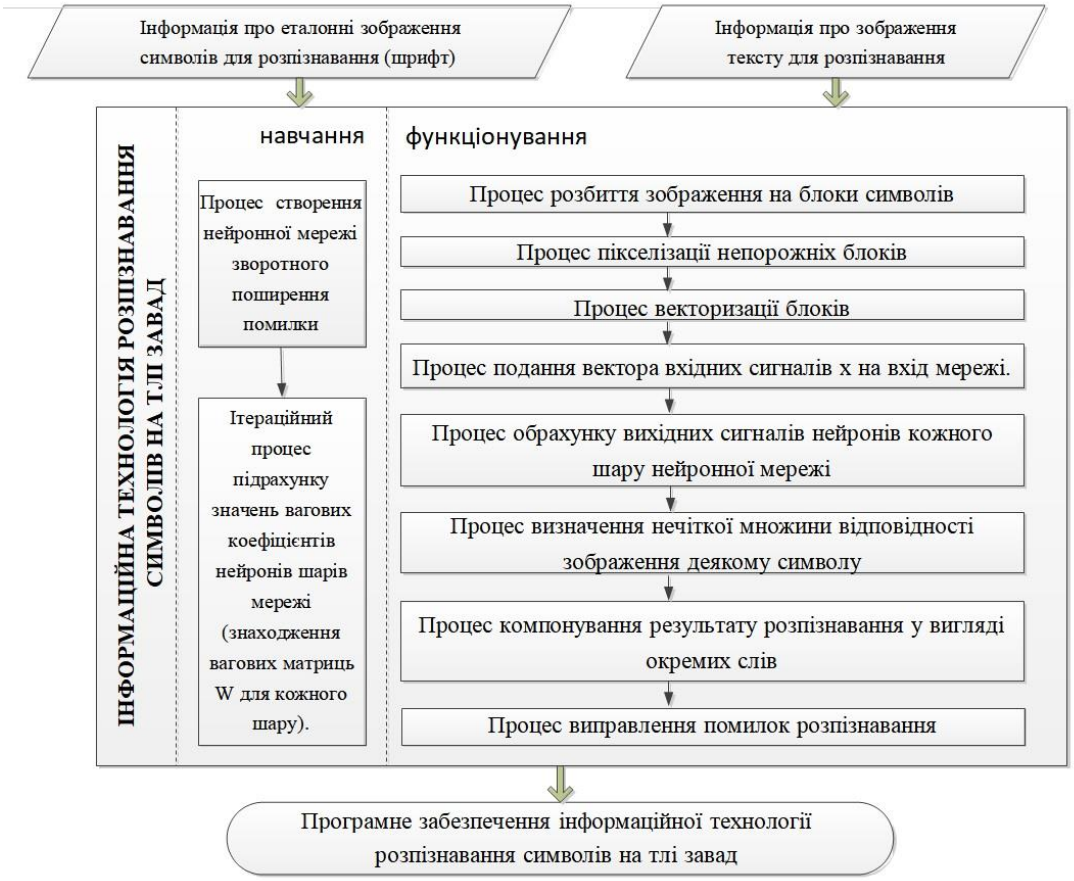


Рисунок 2.9 – Структура додатку розпізнавання символів на тлі завад

функція:

1. Процес поділу зображення на блоки символів.

2. Процес пікселізації непорожніх блоків.

3. Процес блокової векторизації.

4. Процес подачі вектора вхідного сигналу x на вхід мережі.

5. Процес розрахунку вихідних сигналів нейронів кожного шару нейронної мережі.

6. Процес визначення нечіткого набору, що зображення відповідає символу.

7. Процес об'єднання результатів розпізнавання в окремі слова.

8. Процес виправлення помилок ідентифікації.

У цій частині магістерської ідентифікаційної роботи здійснюється теоретична демонстрація розвитку інформаційної технології розпізнавання символів на фоні перешкод.

Теоретичною основою реалізації інформаційних технологій доцільно обрати штучну нейронну мережу, а нейронну мережу зворотного поширення помилок – як мережу, що забезпечує найвищу надійність і швидкість розпізнавання символів в умовах перешкод. Проаналізовано структуру та математичну модель нейронної мережі зворотного розповсюдження помилок, яка з метою підвищення надійності розпізнавання символів на тлі перешкод модифікована таким чином: Метод зворотного поширення помилок у навчанні нейронної мережі розраховано за допомогою Функція Фермі. Активація нейронів покращує метод, який покращує векторизацію зображень шляхом пікселізації перед векторизацією, що може спростити структуру нейронної мережі. Розроблено процес обробки даних для запропонованої інформаційної технології та його компоненти: послідовність навчання та операцій.

3 РЕАЛІЗАЦІЯ ДОДАТКУ ДЛЯ РОЗПІЗНАВАННЯ СИМВОЛІВ НА ТЛІ ЗАВАД НА ОСНОВІ НЕЙРОМЕРЕЖІ

3.1 Розробка алгоритму безпосередньо розпізнавання символів на тлі завад

Загальний алгоритм роботи програми розпізнавання символів на фоні перешкод наведено на рисунку 1.3.1.

Для коректного та швидкого проектування використовуються парадигми об'єктно-орієнтованого програмування та можливість автоматичного проектування систем за допомогою UML. Розроблена архітектура класу повністю відображає архітектуру програми, описану в розділі 2.4.

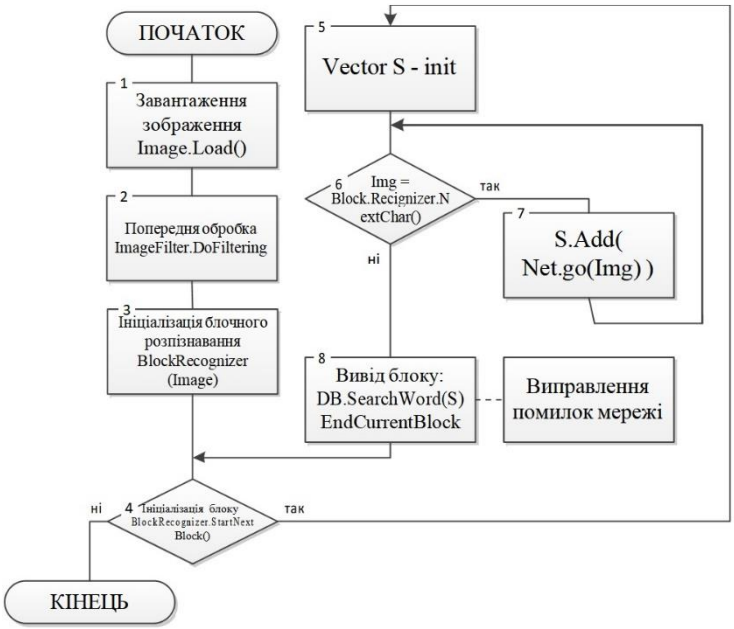


Рисунок 3.1 – Алгоритм безпосередньо розпізнавання символів на тлі завад

Програмне забезпечення складається з кількох модулів (рис. 3.2). Модуль NeuNet-Layer-Vertex виконує оптичне розпізнавання.

Загалом, три класи реалізують нейронну мережу. Навчання нейронної мережі реалізовано на основі алгоритму зворотного поширення. Нейронна мережа має трирівневу структуру:

1. Вхідний – має розмір 256 – 1024 нейронів;

2. Внутрішні - розміром 200-500 нейронів;

3. Оригінальний розмір 26.

Іншим важливим компонентом є клас Searcher - він реалізує роботу над базою слів і виконує необхідні алгоритми для пошуку підходящих слів. Реалізовано на основі динамічного програмування. Таке впровадження скорочує витрати часу.

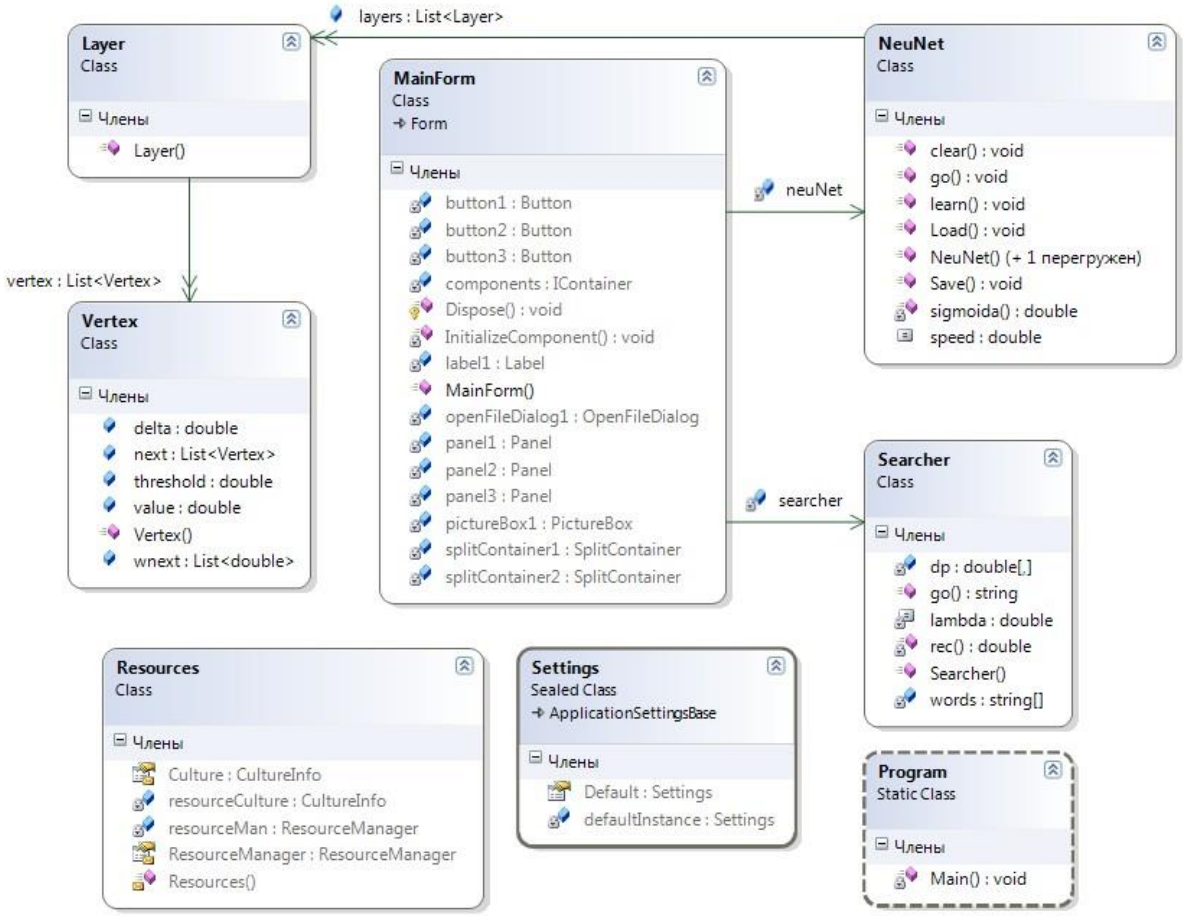


Рисунок 3.2 – Діаграма класів

Алгоритм пошуку: function F(S,V,a,b)

Вхідні дані: S – слово, V – вектор, a – поточна позиція в слові, b-поточна позиція в векторі;

Вихідні дані: Ймовірність відповідності вектора слову;

1. Якщо a вказує на кінець слова AND b вказує на кінець вектора – повернути 1.0 інакше перейти до кроку 2;

2. Якщо a вказує на кінець слова - повернути λF(S,V,a,b+1) інакше перейти до кроку 3;

3. Якщо b вказує на кінець вектора – повернути λF(S,V,a+1,b) інакше перейти до кроку 4;

4. Якщо функція вже була визвана з цими параметрами – повернути записане в кеш значення результату, інакше перейти до кроку 5;

5. Обрати як результат максимальне з трьох значеннь: a. F(S,V,a+1,b+1) * V[b][S[a]];

b. λF(S,V,a+1,b);

c. λF(S,V,a,b+1);

Записати значення до кешу та повернути як результат.

Інші класи використовуються як допоміжні для розробки графічних інтерфейсів. Основний клас форми виконує функції попередньої обробки та призначення завдань. Також прийнято рішення щодо продовження онлайн-навчання.

3.2 Вибір середовища програмування для реалізації додатку розпізнавання символів на тлі завад

Основними критеріями вибору мови програмування є:

- мова програмування скомпільована;
- Кросплатформна сумісність;
- Підтримує принципи ООП.

Для розробки цього продукту була обрана мова програмування C#, яка скомпільована з MIL. Це дозволяє програмам працювати в будь-якому середовищі, де встановлено віртуальну машину .NET. Середовищем розробки обрано пакет Microsoft Visual Studio. Це пояснюється тим, що середовище розробки Visual Studio є дуже поширеним і простим у

використанні. Безкоштовна версія «урізаної» версії сімейства Visual Studio Express також вступає в дію.

Microsoft Visual C# [18, 19] — це інтегроване середовище розробки програм для мови C#, розроблене Microsoft і доступне окремо як частина набору Microsoft Visual Studio або як безкоштовний пакет Visual C# Express Edition з обмеженим набором функцій [19].

3.3 Програні елементи розпізнавання символів на тлі завад

Програма складається з двох збірок: iTell.exe і iTellUtils.dll. У першій збірці, крім інтерфейсу редактора для налаштування виділення блоку символів, присутній графічний інтерфейс користувача.

Збірка iTell містить такі основні класи:

- Console і ConsoleForm - класи, які інкапсулюють роботу консолі та графічний інтерфейс консолі відповідно;

- DBForm - редактор тезаурусів;

- LearningForm - редактор для роботи з нейронними мережами;

- MainForm - головне вікно, що містить інтерфейс користувача і контролює всі інші процеси комплексу;

- Програма - стандартний клас для точок входу програми;

- Recognizer - помічник, який з'єднує інтерфейс користувача з системою розпізнавання;

- SettingsForm і UserSettings - вікно налаштувань користувача та клас, який інкапсулює налаштування користувача.

Збірку iTellUtils можна використовувати незалежно від системи.

Ця збірка містить клас для роботи з нейронними мережами: NeuralNetwork. Це дозволяє використовувати структуру мережі, навчати її та запускати на вхідних даних. Цей клас може використовувати будь-яку кількість шарів. Для запису файлів використовується звичайний текстовий формат (для можливості ручного редагування).

Для виконання обчислень через мережу на вхід подається вектор значень від рецепторів - дробові значення. Після запуску функції Run() ви можете використовувати функцію Output(), щоб отримати значення на виході нейронної мережі. Для навчання функція Learn(a,b) використовується з двома аргументами: перший

є вхідними даними, а другий – вихідним рівнем, необхідним для мережі.

Клас WordDB інкапсулює роботу тезауруса. Цей клас дозволяє створювати порожню базу даних, завантажувати базу даних із файлу (формат файлу — це список слів, по одному на стрічку, виберіть простий формат для легкого ручного редагування). Цей клас містить функціональні можливості для збереження бази даних у файлі, що дозволяє додавати нові слова, виконувати пошукові запити за префіксами. Однак основна функція — знайти відповідне слово.

Це виконується функцією findBestWords(iTellUtils.FuzzyVector), яка отримує вектор нечіткого набору як параметр і таким чином створює масив слів, які найкраще відповідають вектору нечіткого набору.

Іншим важливим класом є клас ImageFilter, який можна використовувати для попередньої обробки зображень. Цей клас виконує фільтрацію на основі фільтрів ранжирування.

Клас BlockRecognizer ділить зображення на блоки. Функція StartImage використовується для завантаження зображення. Потім виконується функція Next*Char, доки не буде повернено порожній об’єкт, що вказує на завершення зображення. Основні класи показані на малюнку. 3.3.

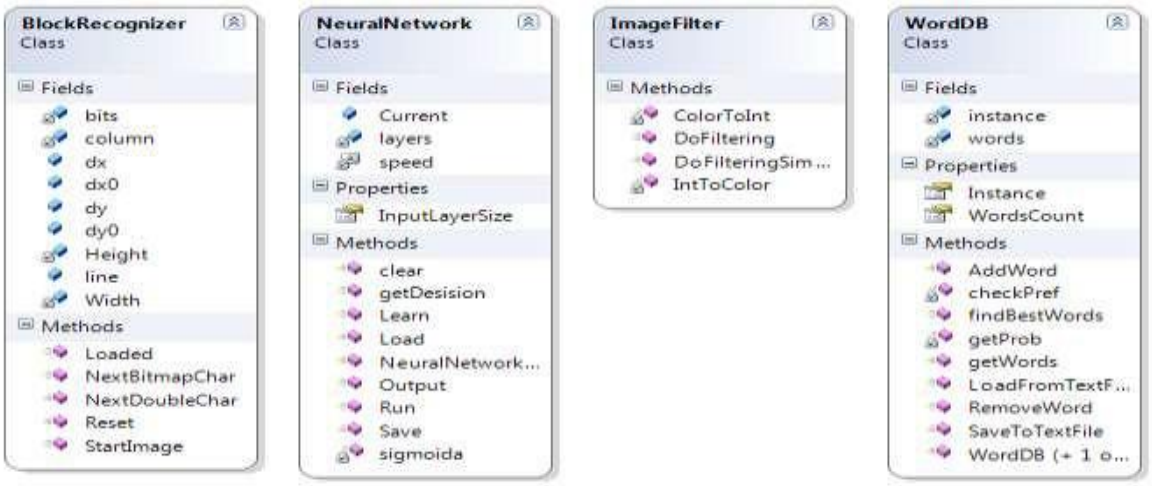


Рисунок 3.3 – Основні класи програми

Одразу після запуску програми з’являється головне вікно (рис. 3.4). Головне вікно повністю відповідає стандартному інтерфейсу Windows. Вікно складається з робочої області та віконного меню.

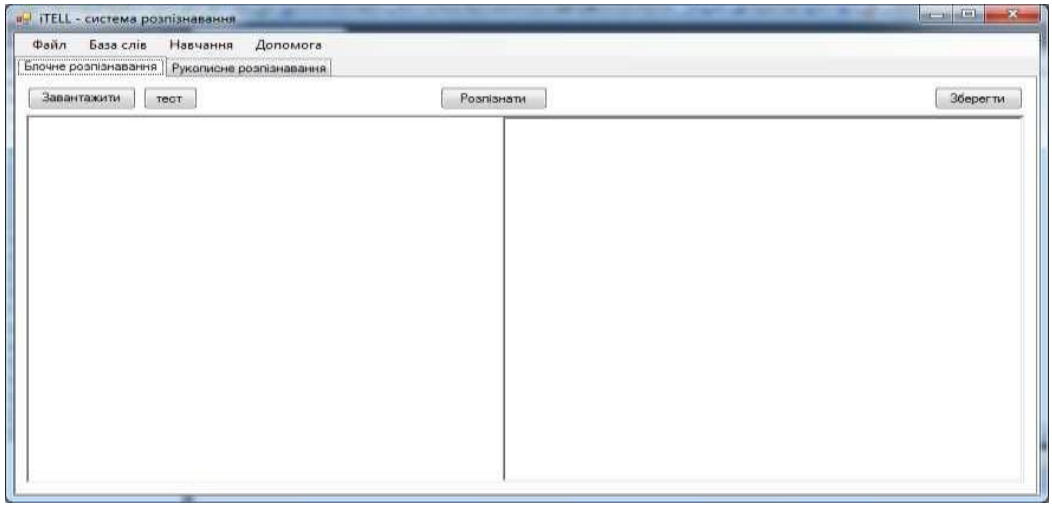


Рисунок 3.4 – Головне вікно програми

Для ідентифікації достатньо кількох кліків. Для цього необхідно завантажити зображення (за допомогою кнопки «Завантажити» або відповідного підпункту меню файлів). Далі виконайте розпізнавання («Розпізнавання») і результати відобразяться в полі виведення (права частина вікна). Приклад результату розпізнавання наведено на рисунку 3.5.

Інтелектуальна програма складається з модуля налаштування зображення, модуля тезауруса, модуля навчання та генерації нейронної мережі та інших модулів.

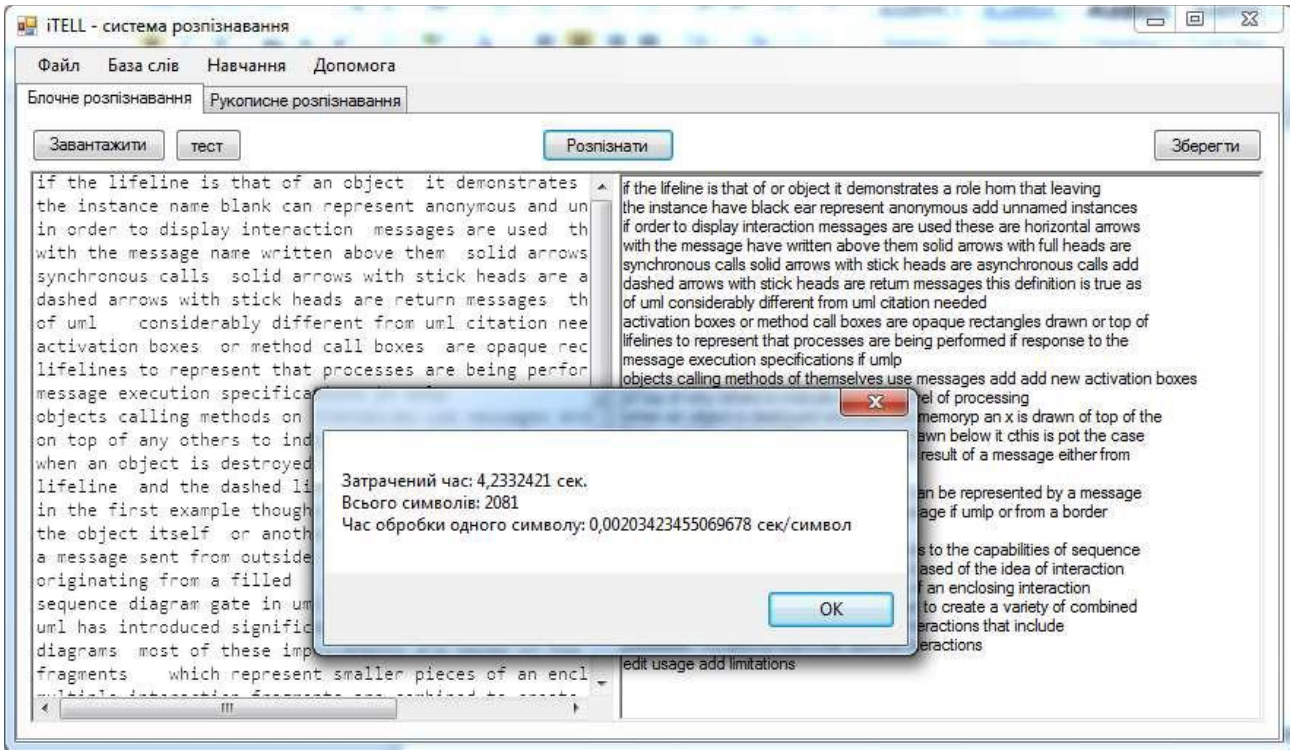


Рисунок 3.5 – Результат розпізнавання

По-перше, врахуйте налаштування користувача. Використовуйте пункт меню «Файл > Параметри», щоб викликати налаштування. Вікно налаштувань показано на малюнку 3.6.

Ці налаштування дозволяють вибрати файл нейронної мережі та файл бази слів за замовчуванням. Для редагування бази слів скористайтеся редактором «База слів > Редагувати базу». Редактор (рис. 3.7) дозволяє додавати слова, видаляти групи слів і здійснювати пошук у базі даних. Програма дозволяє зберігати бази слів у файли у звичайному форматі.

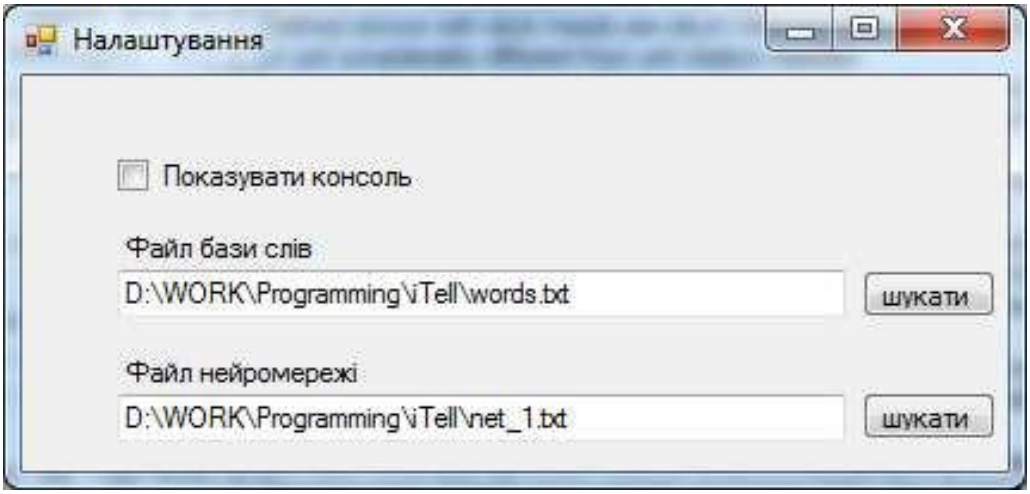


Рисунок 3.6 – Вікно налаштувань

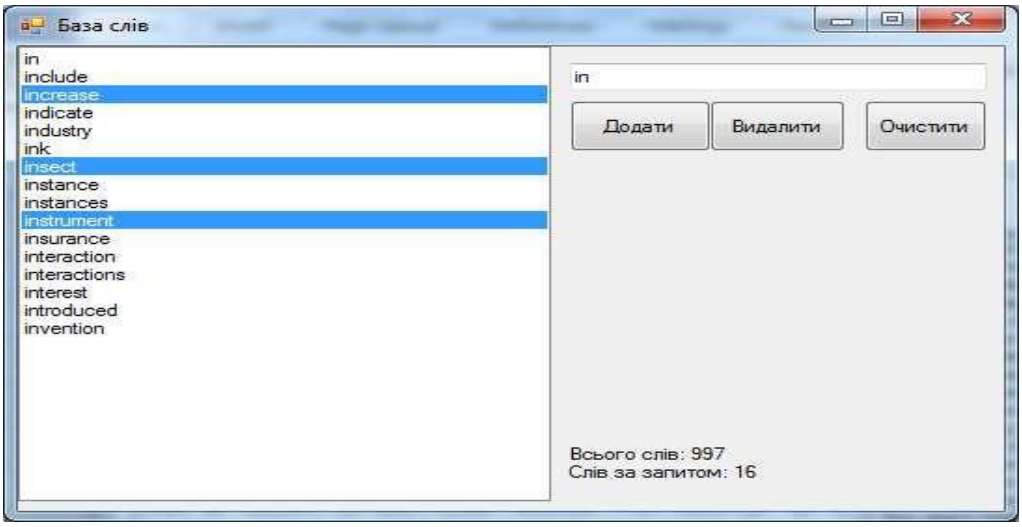


Рисунок 3.7 – Редактор бази слів

Для вивчення/завантаження/збереження нейронних мереж ми розробили редактор, доступний у розділі «Навчання». Для навчання потрібно вибрати каталог із прикладами символів (кожен символ в окремому файлі, формат зображення BMP). Далі виберіть кількість ітерацій, які потрібно виконати на наборі, і натисніть «Почати навчання». Коли мережа тренується, деякі статистичні дані будуть відображатися в полі виводу внизу вікна (рис. 3.8).

У цьому полі відображається кількість правильно визначених спроб і загальна кількість спроб. Для покращення видимості відображається стрічка,

на якій кожна спроба в хронологічному порядку замінюється символом «+» або «o», що вказує на те, що спроба була ідентифікована правильно чи неправильно відповідно.

Для точного налаштування модуля обробки блоків можна викликати додатковий редактор з головного вікна, натиснувши кнопку «Тестувати» праворуч від кнопки «Завантажити».

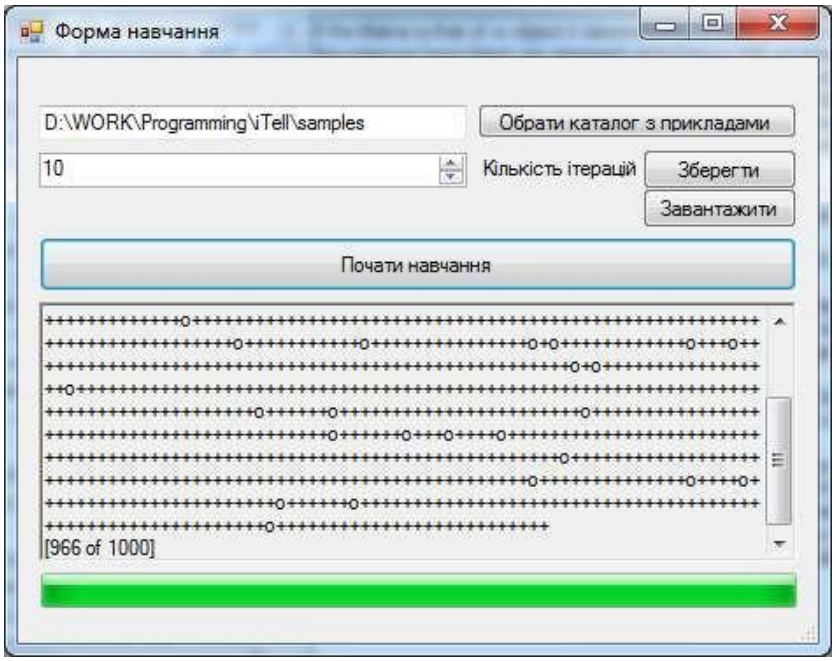


Рисунок 3.8 – Навчання нейронної мережі

Редактор зображено на рисунку 3.9. Це дозволяє вам точно налаштувати обробку блоків, наприклад додавати зміщення або встановлювати розміри символів. Можна також використовувати для підготовки символів для навчання мережі.

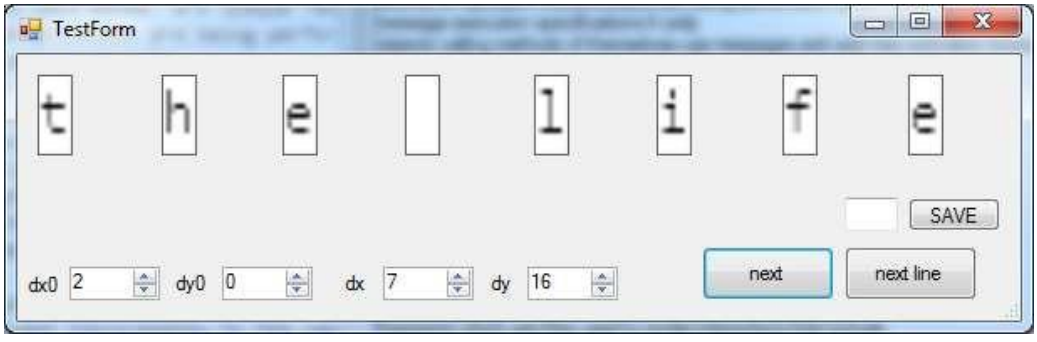


Рисунок 3.9 – Налаштування модуля блочної обробки

3.4 Тестування та аналіз результатів роботи додатку розпізнавання символів на тлі завад

Для аналізу роботи програми було використано навчання нейронної мережі з 10 000 випадкових символів. Візьміть 1000 найпоширеніших слів англійською як тезаурус і використовуйте статтю Вікіпедії як текст для ідентифікації.

Перший тест – це чисте зображення без відволікаючих факторів.

З початку розпізнавання (рис. 3.10) програма розпізнала майже всі символи з точністю 98,5%.

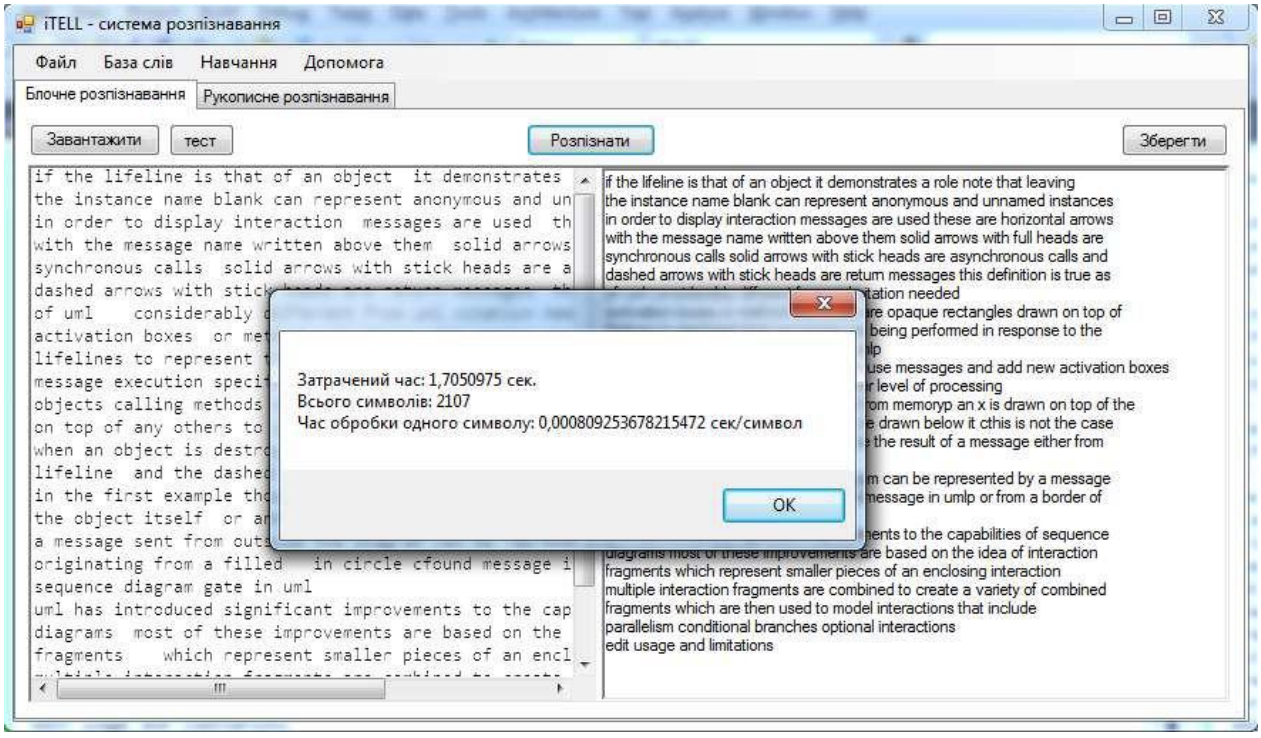


Рисунок 3.10 – Тестове розпізнавання зображення без завад

Швидкість обробки 2107 символів становить 1,7 секунди. Тобто персонаж

Середній час обробки становив 0,0008 секунди. Така швидкість пояснюється використанням простих нейронних мереж і перенесенням обчислювальної складності на блок обробки результатів.

Були отримані наступні результати у вигляді розмиття та шуму для 5% порушення: Час обробки: 1,6 секунди Точність: 92,4% (залишилося в межах встановленої цілі). Подальше збільшення перешкод призвело до таких результатів:

10% - Показник точності становить 82,3%, а швидкість залишається незмінною;

20% - Показник точності становить 70,4%, а швидкість залишається незмінною;

Пошкоджений текст (випадковий нерегулярний шум, вставка додаткових чисел, неправильні символи під час відображення) - точність 85,971% постійної швидкості.

Для визначення сильних сторін розробленої програми було порівняно надійність розпізнавання символів у контексті інтерференції між розробленою та змодельованою програмами. Як програму моделювання було обрано програму MeOCR 1.0 [14]. Він перетворює скановані документи на текстові документи, які можна редагувати, і експортує їх у Microsoft Word. Результати дослідження надійності розпізнавання двома програмами одного тексту з різними значеннями шуму наведені в таблиці. 3.1. Приклад тексту з перешкодами показано на малюнку. 3.11.

Таблиця 3.1 - Достовірність розпізнавання символів на тлі завад

	Наявність завад			
	0%	5%	10%	20%
Розроблена програма	98,5%	92.4%	82.3%	70.4%,
Аналог (програма MeOCR)	98,1%	90.5%	80.2%	67,9%,
Різниця	0,4%	1,9%	2,1%	2,5%

Огляд результатів тестування (табл. 3.1) показує, що розроблена програма має кращий показник надійності розпізнавання символів на фоні перешкод. Таким чином, поставлені цілі були досягнуті, оскільки надійність

ідентифікації розробленої програми вища, ніж у симуляції (0% шуму - на 0,4% вище, 5% шуму - на 1,9% вище, 10% шуму - на 2,1% вище, 20% шум - на 2,5% вище). Для чистих зображень якість розпізнавання наближається до 100%. На практиці частота відмов становить менше 1%, а за такими вхідними даними середня похибка комплексу не перевищує 2,2%.

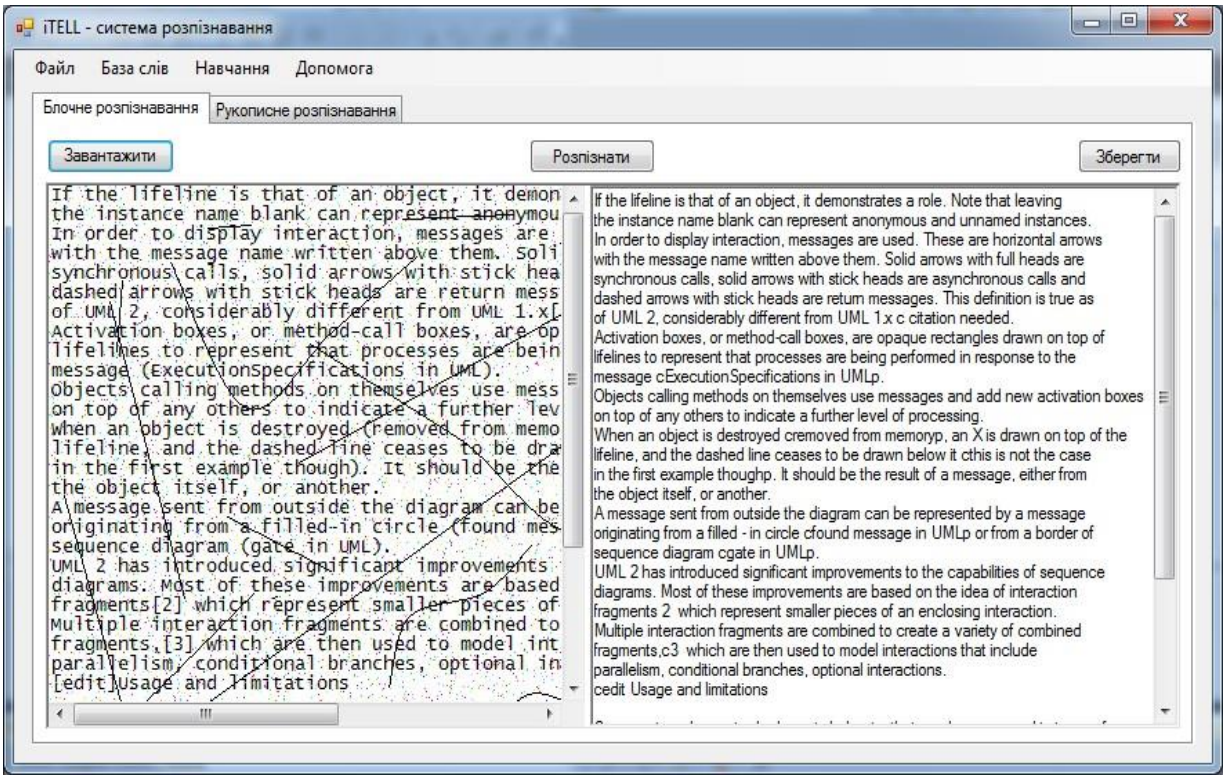


Рисунок 3.11 – Тестове розпізнавання зображення текста з завадами

Програма реалізована на базі платформи .NET. Хоча платформа .NET не є найефективнішим рішенням з точки зору продуктивності, вона допомагає перенести складність програмного забезпечення на інші платформи.

У цьому розділі розробляються алгоритми роботи програмних засобів розпізнавання символів в умовах перешкод і демонструється вибір мови програмування Microsoft Visual C# і середовища програмування Microsoft Visual Studio. Також була проведена розробка програмного засобу для розпізнавання символів у контексті перешкод. Програма складається з двох збірок: iTell.exe і iTellUtils.dll. Тести показують надійну роботу розроблених

програм. Аналіз результатів роботи доводить, що розроблена програма має кращий показник надійності розпізнавання символів, ніж змодельована програма на фоні перешкод. Наразі поставлені в роботі цілі досягнуті

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Комерційний та технологічний аудит науково-технічної розробки

Метою даного розділу є проведення технологічного аудиту, в даному випадку нового програмного продукту аналітичної веб-система побутового споживання енергії.

Особливістю програми є те, що дана технологія призначена для інтерактивного моніторингу побутового використання енергії на основі якого буде відбуватися коригування навантаження на електричні мережі для його зменшення.

Аналогом може бути Bissoft, ціна 1500 грн; Smart maic, ціна 920 грн.

Для проведення комерційного та технологічного аудиту залучають не менше 3-х незалежних експертів. Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням п’ятибальної системи оцінювання за 12-ма критеріями, у відповідності із табл. 4.1.

Таблиця 4.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздат-ність продук-ту в реаль-них умовах

2	Багато аналогів на малому ринку	Ринкові п Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку

Продовження табл. 4.1

Ринкові переваги					
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практик на здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так із комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві

Продовження табл. 4.1

11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Усі дані по кожному параметру занесено в таблиці 4.2

Таблиця 4.2 – Результати оцінювання комерційного потенціалу розробки

Критерії оцінювання	ПІБ експертів		
	Експерт 1	Експерт 2	Експерт 3
	Бали		
Технічна здійсненність концепції	3	3	4
Наявність аналогів на ринку	4	3	4
Цінова політика	4	4	3
Технічні та споживчі властивості виробу	4	3	4
Експлуатаційні витрати	3	4	3
Ринок збуту	4	3	4
Конкурентоспроможність	3	4	3
Фахівці з технічної і комерційної реалізації	4	3	4
Фінансування	4	4	3
Матеріально-технічна база	3	4	4
Термін реалізації ідеї	4	4	4
Супровідна документація	3	3	4
Сума	43	42	44
Середньоарифметична сума балів	(43+42+44) / 3 = 43		

За даними таблиці 4.2 можна зробити висновок щодо рівня комерційного потенціалу даної розробки. Для цього доцільно скористатись рекомендаціями, наведеними в таблиці 4.3.

Таблиця 4.3 - Рівні комерційного потенціалу розробки

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

Як видно з таблиці, рівень комерційного потенціалу розроблюваного нового програмного продукту є високим, що досягається за рахунок того, що програмний продукт відрізняється від існуючих тим, що дана технологія призначена для інтерактивного моніторингу побутового використання енергії на основі якого буде відбуватися коригування навантаження на електричні мережі для його зменшення.

4.2 Прогнозування витрат на виконання науково-дослідної (дослідно-конструкторської) роботи

4.2.1 Основна заробітна плата розробників, яка розраховується за формулою:

$$З_о = \frac{M}{T_р} \cdot t, \qquad (4.1)$$

де М – місячний посадовий оклад конкретного розробника (дослідника), грн.;

$T_р$ – число робочих днів в місяці, 20 днів;

t – число днів роботи розробника (дослідника).

Результати розрахунків зведемо до таблиці 4.1.

Таблиця 4.1 – Основна заробітна плата розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник проекту	44000	2200,00	35	77000,000
Програміст	40000	2000,00	35	70000,000
Всього				147000,00

Так як в даному випадку розробляється програмний продукт, то розробник виступає одночасно і основним робітником, і тестувальником розроблюваного програмного продукту.

4.2.2 Додаткова заробітна плата розробників, які приймали участь в розробці обладнання.

Додаткова заробітна плата прийнято розраховувати як 14,5 % від основної заробітної плати розробників та робітників:

$$З_д = З_о \cdot 14{,}5\ \% / 100\ \% \qquad (4.2)$$

$$З_д = (147000{,}00 \cdot 14{,}5\ \% / 100\ \%) = 21315{,}00 \text{ (грн.)}$$

4.2.3 Нарахування на заробітну плату розробників.

Згідно діючого законодавства нарахування на заробітну плату складають 22 % від суми основної та додаткової заробітної плати.

$$Н_з = (З_о + З_д) \cdot 22\ \% / 100\% \qquad (4.3)$$

$$Н_з = (147000{,}00 + 21315{,}00) \cdot 22\ \% / 100\ \% = 37029{,}30 \text{ (грн.)}$$

4.2.4. Оскільки для розроблювального пристрою не потрібно витрачати матеріали та комплектуючи, то витрати на матеріали і комплектуючі дорівнюють нулю.

4.2.5 Амортизація обладнання, яке використовувалось для проведення розробки.

Амортизація обладнання, що використовувалось для розробки в спрощеному вигляді амортизація обладнання, що використовувалась для розробки розраховується за формулою:

$$A = \frac{Ц}{Тв} \cdot \frac{t_{вик}}{12} \text{ [грн.].} \quad (4.4)$$

де Ц – балансова вартість обладнання, грн.;

Т – термін корисного використання обладнання згідно податкового законодавства, років

$t_{вик}$ – термін використання під час розробки, місяців

Розрахуємо, для прикладу, амортизаційні витрати на комп'ютер балансова вартість якого становить 20000 грн., термін його корисного використання згідно податкового законодавства – 2 роки, а термін його фактичного використання – 1,75 міс.

$$A_{обл} = \frac{20000}{2} \times \frac{1{,}75}{12} = 1458{,}33 \; грн.$$

Аналогічно визначаємо амортизаційні витрати на інше обладнання та приміщення. Розрахунки заносимо до таблиці 4.2. Для розрахунку амортизації нематеріальних ресурсів використовується формула:

$$A_{н.р.} = Ц_{н.р.} * Н_а * \frac{t_{вик.}}{12} \quad (4.5)$$

Але, так як вартість ліцензійної ОС та спеціалізованих ліцензійних нематеріальних ресурсів менше 20000 грн, то даний нематеріальний актив не амортизується, а його вартість включається у вартість розробки повністю,

$B_{нем.ак.}$ = 1390 грн.

Таблиця 4.2 – Амортизаційні відрахування матеріальних і нематеріальних ресурсів для розробників

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн.
Комп'ютер та комп'ютерна переферія	20000	2	1,75	1458,333
Офісне обладнення	25000	4	1,75	911,458
Приміщення	850000	20	1,75	6197,917
Всього				8567,71

4.2.6 Тарифи на електроенергію для непобутових споживачів (промислових підприємств) відрізняються від тарифів на електроенергію для населення. При цьому тарифи на розподіл електроенергії у різних постачальників (енергорозподільних компаній), будуть різними. Крім того, розмір тарифу залежить від класу напруги (1-й або 2-й клас). Тарифи на розподіл електроенергії для всіх енергорозподільних компаній встановлює Національна комісія з регулювання енергетики і комунальних послуг (НКРЕКП). Витрати на силову електроенергію розраховуються за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п, \quad (4.6)$$

де В – вартість 1 кВт-години електроенергії для 1 класу підприємства, В = 6,2 грн./кВт;

П – встановлена потужність обладнання, кВт. П = 0,6 кВт;

Ф – фактична кількість годин роботи обладнання, годин.

$К_п$ – коефіцієнт використання потужності, $К_п$ = 0,9.

$B_е = 0,9 \cdot 0,6 \cdot 8 \cdot 35 \cdot 6,2 = 937,44$ (грн.)

4.2.7 Інші витрати та загальновиробничі витрати.

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками. Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.7)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

$I_в = 147000,00 * 75\% / 100\% = 110250$ (грн.)

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін. Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників:

$$Н_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.8)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

$H_{нзв}$ = 147000,00 * 140 % / 100 % = 205800 (грн.)

4.2.8 Витрати на проведення науково-дослідної роботи.

Сума всіх попередніх статей витрат дає загальні витрати на проведення науково-дослідної роботи:

$B_{заг}$ = 147000,00+21315,00+37029,30+8567,71+1390+937,44+110250+
+205800 = 532289,45 грн.

4.2.9 Розрахунок загальних витрат на науково-дослідну (науково-технічну) роботу та оформлення її результатів.

Загальні витрати на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються *ЗВ*, визначається за формулою:

$$ЗВ = \frac{B_{заг}}{\eta} \text{ (грн)}, \qquad (5.9)$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи.

Так, якщо науково-технічна розробка знаходиться на стадії: науково-дослідних робіт, то η=0,1; технічного проектування, то η=0,2; розробки конструкторської документації, то η=0,3; розробки технологій, то η=0,4; розробки дослідного зразка, то η=0,5; розробки промислового зразка, то η=0,7; впровадження, то η=0,9. Оберемо η = 0,5, так як розробка, на даний момент, знаходиться на стадії дослідного зразка:

ЗВ = 532289,45 / 0,5 = 1064579 грн.

4.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

В ринкових умовах узагальнювальним позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку. Саме зростання чистого прибутку забезпечить потенційному інвестору надходження додаткових коштів, дозволить покращити фінансові результати його діяльності, підвищить конкурентоспроможність та може позитивно вплинути на ухвалення рішення щодо комерціалізації цієї розробки.

Для того, щоб розрахувати можливе зростання чистого прибутку у потенційного інвестора від можливого впровадження науково-технічної розробки необхідно:

а) вказати, з якого часу можуть бути впроваджені результати науково-технічної розробки;

б) зазначити, протягом скількох років після впровадження цієї науково-технічної розробки очікуються основні позитивні результати для потенційного інвестора (наприклад, протягом 3-х років після її впровадження);

в) кількісно оцінити величину існуючого та майбутнього попиту на цю або аналогічні чи подібні науково-технічні розробки та назвати основних суб’єктів (зацікавлених осіб) цього попиту;

г) визначити ціну реалізації на ринку науково-технічних розробок з аналогічними чи подібними функціями.

При розрахунку економічної ефективності потрібно обов’язково враховувати зміну вартості грошей у часі, оскільки від вкладення інвестицій до отримання прибутку минає чимало часу. При оцінюванні ефективності інноваційних проектів передбачається розрахунок таких важливих показників:

- абсолютного економічного ефекту (чистого дисконтованого доходу);
- внутрішньої економічної дохідності (внутрішньої норми дохідності);
- терміну окупності (дисконтованого терміну окупності).

Аналізуючи напрямки проведення науково-технічних розробок, розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором можна об'єднати, враховуючи визначені ситуації з відповідними умовами.

4.3.1 Розробка чи суттєве вдосконалення програмного засобу (програмного забезпечення, програмного продукту) для використання масовим споживачем.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

$$\Delta П_i = (\pm \Delta Ц_0 \cdot N + Ц_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \qquad (4.10)$$

де $\pm\Delta Ц_о$ – зміна вартості програмного продукту (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу;

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

$Ц_о$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки, $Ц_о = Ц_б \pm \Delta Ц_о$;

Цб – вартість програмного продукту у році до впровадження результатів розробки;

ΔN – збільшення кількості споживачів продукту, в аналізовані періоди часу, від покращення його певних характеристик;

λ – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт $\lambda = 0{,}8333$.

p – коефіцієнт, який враховує рентабельність продукту;

ϑ – ставка податку на прибуток, у 2022 році ϑ =18%.

Припустимо, що при прогнозованій ціні 500 грн. за одиницю виробу, термін збільшення прибутку складе 3 роки. Після завершення розробки і її вдосконалення, можна буде підняти її ціну на *50* грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року – на 13000 шт., протягом другого року – на 23000 шт., протягом третього року на 33000 шт. До моменту впровадження результатів наукової розробки реалізації продукту не було:

$\Delta П_1 = (0*50 + (500 + 50)*13000)* 0{,}8333* 0{,}43) * (1 - 0{,}18) = 1909916{,}590$ грн.

$\Delta П_2 = (0*50 + (500 + 50)*(13000+23000)* 0{,}8333* 0{,}43) * (1 - 0{,}18) = 5817899{,}767$ грн.

$\Delta П_3 = (0*50 + (500 + 50)*(13000+23000+33000)* 0{,}8333* 0{,}43) * (1 - 0{,}18) = 11150974{,}554$ грн.

Отже, комерційний ефект від реалізації результатів розробки за три роки складе 18878790,91 грн.

4.3.2 Розрахунок ефективності вкладених інвестицій та періоду їх окупності.

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

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (5.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої науково-дослідної (науково-технічної) роботи, грн;

Т – період часу, протягом якою виявляються результати впровадженої науково-дослідної (науково-технічної) роботи, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0,05...0,15$;

t – період часу (в роках).

Збільшення прибутку ми отримаємо починаючи з першого року:

ПП= $(1909916,590/(1+0,1)^1)+(5817899,767/(1+0,1)^2)+(11150974,554/(1+0,1)^3)$ = 1736287,81 + 4808181,626 + 8377892,227 = 14922361,66 грн.

Далі розраховують величину початкових інвестицій *PV*, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{інв} * ЗВ, \quad (4.12)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{інв}=2...5$, але може бути і більшим;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

PV = 2 * 1064579 = 2129157,79 *грн.*

Тоді абсолютний економічний ефект $E_{абс}$ або чистий приведений дохід *(NPV, Net Present Value)* для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV, \tag{4.13}$$

$E_{абс}$ = 14922361,66 – 2129157,79 = 12793203,87 грн.

Оскільки $E_{абс} > 0$ то вкладання коштів на виконання та впровадження результатів даної науково-дослідної (науково-технічної) роботи може бути доцільним.

Для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність або показник внутрішньої но-рми дохідності *(IRR, Internal Rate of Return)* вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього використаємо формулу:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{4.14}$$

$T_ж$ – життєвий цикл наукової розробки, роки.

$E_в = \sqrt[3]{(1 + 12793203,87/2129157,79} - 1 = 0,914$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f, \tag{4.15}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2022 році в Україні d = (0,09…0,14);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,5).

$$\tau_{\min} = 0,14 + 0,05 = 0,19.$$

Так як $E_в > \tau_{min}$, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$T_{ок} = \frac{1}{E_в}, \tag{4.16}$$

$T_{ок}$ = 1 / 0,914 = 1,09 р.

Оскільки $T_{ок}$ < 3-х років, а саме термін окупності рівний 1,09 роки, то фінансування даної наукової розробки є доцільним.

Висновки до розділу: економічна частина даної роботи містить розрахунок витрат на розробку нового програмного продукту, сума яких складає 1064579 гривень. Було спрогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення, розраховано період окупності витрат для інвестора та економічний ефект при використанні даної розробки. В результаті аналізу розрахунків можна зробити висновок, що розроблений програмний продукт за ціною дешевший за аналог і є висококонкурентоспроможним. Період окупності складе близько 1,09 роки.

ВИСНОВКИ

У магістерській кваліфікаційній роботі розробка цієї інформаційної технології та програми розпізнавання знаків у контексті перешкод виявилася актуальною та вигідно порівняно з розробкою аналогів. Результати аналізу предметної області, яка враховує сучасні методи розпізнавання образів, сучасні методи попередньої обробки зображень, методи представлення зображень у нейронних мережах, а також основну топологію нейронних мереж і алгоритми їх навчання, які стали можливими. згідно з алгоритмом зворотного поширення помилок, щоб обґрунтувати вибір нейронної мережі багатошарового персептрона зі здатністю до навчання. Показано основні недоліки реалізації рішень на основі інших топологій і алгоритмів.

Всі поставлені завдання виконано. Також описано математичний апарат для виконання поставленого завдання, а саме розпізнавання символів у контексті перешкод. Наведено алгоритм роботи програми розпізнавання символів на фоні перешкод. Основною перевагою розробленої програми розпізнавання є нечіткий пошук очікуваного результату. Такий додатковий процес дозволяє, по-перше, більш якісно провести ідентифікацію, а по-друге, дозволяє ідентифікувати блоки символів з перешкодами.

Програмне забезпечення розроблено на мові програмування C# та скомпільовано для платформи .NET у середовищі розробки Microsoft Visual Studio 2012 (надано відділом CS у рамках програми DreamSpark (MSDN AA)).

Цілі, поставлені в роботі, досягнуті, оскільки надійність розпізнавання розробленої програми вища, ніж у симуляції (0% шуму - на 0,4% вище, 5% шуму - на 1,9% вище, 10% шуму - на 2,1% вище , 20% шуму – 2,5% вище).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Чуба Б. О. Програмний засіб розпізнавання символів на тлі завад на основі нейронної мережі / Б. О. Чуба, О. К. Колесницький // Науково-практична конференція «Сучасні тенденції розвитку системного програмування» (25-26 листопада 2016 р., Україна, м. Київ.
2. Алексєєв А. Алгоритм розпізнавання символів на основі структурного підходу / А. Алексєєв, В. Заяць, Д. Іванов // Вісник Нац. ун-ту „Львівська політехніка‖ „Комп‘ютерна інженерія та інформаційні технології". – 2002. – № 468. – С.129 - 133.
3. Горелик А.Л. Методы распознавания / А.Л. Горелик, В.А. Скрипник. – М. : Высшая школа, 1989. – 232 с.
4. Фукунага К. Введение в статистическую теорию распознавания/ К.Фукунага. – М.: Наука, 1979. – 512 С.
5. Дуда Р. Распознавание образов и анализ сцен / Р. Дуда, П. Харт. – М. : Мир, 1976. – 512с.
6. Применение нейросетей в распознавании изображений [Електронний ресурс] // Режим доступу: http://habrahabr.ru/blogs/artificial_intelligence/74326/
7. Реалізація нейронних мереж [Електронний ресурс] // Режим доступу: http://alexandria.nu/ai/neural_net_demo
8. Математические основы нейронных сетей [Електронний ресурс] // Режим доступу: http://5ka.ru/67/26159/1.html
9. Калан Роберт Основные концепции нейронных сетей / Роберт Каллан; – М. : Вильямс, 2001. – 288 с.
10. Хайкин Саймон Нейронные сети: полный курс / Саймон Хайкин; – М.: Вильямс, 2006. – 1104 с.
11. Сотник С. Л. Конспект лекций по курсу "Основы проектирования систем искусственного интеллекта" / С. Л. Сотник. – Москва : Наука, 1998. –

284с.

12. ABBYY FineReader [Електронний ресурс] // Режим доступу: http://help.abbyy.com/FineReader/FineReader12/Russian/QuickTasks/general.htm)
13. Програма CuneiForm [Електронний ресурс] // Режим доступу: http://programdownloadfree.com/load/text/scanning_recognition/ocr_cuneiform/74-1-0-131
14. MeOCR - OCR Image to Text Converter [Електронний ресурс] // Режим доступу: http://www.meocr.com/
15. Горбань А.Н. Методы Нейроинформатики, сборник научных трудов/ А.Н. Горбань. – Красноярск: КГТУ, 1998. – 341с.
16. Заяць В.М. Архітектура подіє – орієнтованих систем на прикладі системи розпізнавання рукописного тексту / В.М. Заяць, Д.О. Іванов // Вісник Нац. ун-ту „Львівська політехніка „Комп‘ютерна інженерія та інформаційні технології". – 2004. – № 530. – С. 78 - 83.
17. Ротштейн А.П. Интеллектуальные технологии идентификации: нечеткие множества, генетические алгоритмы, нейронные сети. – Винница: «УНІВЕРСУМ-Вінниця», 1999
18. Троелсен Э. C# и платформа .NET. Библиотека программиста. - СПб.: Питер, 2007. - 796 с.
19. НЕЙРОННЫЕ СЕТИ НА C#. [Електронний ресурс] – Режим доступу: http://www.cyberguru.ru/algorithms/algorithms-theory/algorithms-neural- networks-written-on-csharp.html
20. Методичні вказівки до виконання студентами-магістрантами економічної частини магістерських кваліфікаційних робіт / Уклад. В. О. Козловський – Вінниця: ВНТУ, 2012. – 22 с.

ДОДАТКИ

Додаток А
(обов’язковий)

ЗАТВЕРДЖЕНО
Зав. кафедри КСУ ВНТУ,
д.т.н., доцент

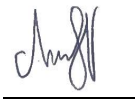

В’ячеслав КОВТУН

“04” 10 2022 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на виконання магістерської кваліфікаційної роботи
“ Нейромережевий застосунок для розпізнавання текстових зображень із завадами ”

08-03.МКР.007.03.000 ТЗ

Студент групи 2АКІТ-21м


Анатолій ПОЛІЩУК
Підпис *Ім’я*
ПРІЗВИЩЕ

Керівник д.т.н., доцент, проф. каф. КСУ


В’ячеслав КОВТУН
Підпис *Ім’я*
ПРІЗВИЩЕ

Вінниця 2022

1. Назва та галузь застосування

1.1. Нейромережевий застосунок для розпізнавання текстових зображень із завадами.

1.2. Галузь застосування – розпізнавання тексту.

2. Підстава для проведення розробки.

Тема магістерської кваліфікаційної роботи затверджена наказом по ВНТУ № 203 від 14.09.2022 р.

3. Мета та призначення розробки.

Метою магістерської кваліфікаційної роботи є розробка застосунку для розпізнавання зображень на тлі завад.

4. Джерела розробки.

Магістерська кваліфікаційна робота виконується вперше. В ході проведення розробки повинні використовуватись такі документи:

1. Калан Роберт Основні концепції нейронних мереж / Роберт Каллан; – М. : Вільямс, 2001. – 288 с
2. Нейронні мережі на C#. [Електронний ресурс] – Режим доступу: http://www.cyberguru.ru/algorithms/algorithms- theory/algorithms-neural- networks-written-on-csharp.html

5. Вимоги до розробки.

5.1. Перелік головних функцій:

- розпізнавання символів на тлі завад;
- генерація тексту;
- формування результату.

5.2. Основні технічні вимоги до розробки.

5.2.1. Вимоги до програмної платформи:

- WINDOWS 10;
- Visual Studio 11.

5.2.2. Умови експлуатації системи:

- робота на веб-додатках;
- розпізнавання зображень.

6. Стадії та етапи розробки.

6.1 Пояснювальна записка:

1. Огляд предметної області. Постановка задач дослідження «04» 10 2022 р.
2. Розробка програмного забезпечення та експериментальні дослідження. «22» 10 2022 р.

6.2 Графічні матеріали:

1. Розробка UML-діаграм системи «4» 11 2022 р.
2. Розробка архітектури мережі «10» 11 2022 р.
3. Тестування програмного забезпечення «18» 11 2022 р.

7. Порядок контролю і приймання.

7.1. Хід виконання роботи контролюється керівником роботи. Рубіжний контроль провести до «28» 11 2022 р.

7.2. Атестація МКР здійснюється на попередньому захисті. Попередній захист магістерської кваліфікаційної роботи провести до «16» 12 2022 р.

7.3. Підсумкове рішення щодо оцінки якості виконання роботи приймається на засіданні ЕК. Захист магістерської кваліфікаційної роботи провести до «23»12 2022р.

Додаток Б

Лістинг програми

Файл NeuralNetwork

using System;

using System.Collections.Generic; using System.Linq;

using System.Text; using System.IO;

namespace iTellUtils

{

public class Vertex

{

public List<Vertex> next; public List<double> wnext; public double delta; public double value; public

double threshold;

public Vertex()

{

next = new List<Vertex>(); wnext = new List<double>();

Random rand = new Random((int)DateTime.Now.Ticks); threshold = rand.NextDouble() * 0.4;

delta = 0.0;

value = 0.0;

}

}

public class Layer

{

public List<Vertex> vertex; public Layer(int size)

```
{
vertex = new List&lt;Vertex&gt;(); for (int i = 0; i &lt; size; ++i)
vertex.Add(new Vertex());
}
}
public class NeuralNetworkException: Exception
{
public NeuralNetworkException(string message)
: base(message) { }
}
public class NeuralNetwork
{

private const double speed = 0.05; private List&lt;Layer&gt; layers;
static public NeuralNetwork Current = null; public int getDesision() {
double best = -1e100;
int bestv = -1; double[] x = Output();
for(int i=0; i&lt;x.Length; ++i)
{
if (x[i] &gt; best)
{
best = x[i]; bestv = i;
}
}
```

```
return bestv;

}

public double[] Output()

{

double[] result = new double[layers[layers.Count - 1].vertex.Count];


for(int i=0; i&lt;result.Length; ++i)

{

result[i] = layers[layers.Count - 1].vertex[i].value;

}

return result;

}

public void Save(string filename)

{

StreamWriter sw = null; try

{

sw = new StreamWriter(filename);

sw.WriteLine(layers.Count);

for (int i = 0; i &lt; layers.Count; ++i)
sw.Write(layers[i].vertex.Count.ToString() + &quot; &quot;);

sw.WriteLine();

for (int i = 0; i &lt; layers.Count; ++i)

for (int j = 0; j &lt; layers[i].vertex.Count; ++j)

{
```

```
Vertex v = layers[i].vertex[j];

sw.Write(v.threshold.ToString() + &quot; &quot; + v.next.Count.ToString()
+ &quot; &quot;); for (int k = 0; k &lt; v.next.Count;

++k)

{



sw.Write(v.wnext[k]);

sw.Write(&quot; &quot;);

}

sw.WriteLine();

}

}

catch (Exception e)

{

throw new NeuralNetworkException(e.Message);

}

finally

{

if (sw != null)

{

sw.Flush();

sw.Close();

}

}
```

```
        }

        public void Load(string filename)

        {

        StreamReader sw = null; try

        {

        sw = new StreamReader(filename); string s = sw.ReadToEnd();

        string[] _ss = s.Split(&#39; &#39;, &#39;\n&#39;, &#39;\r&#39;);
List&lt;string&gt; ss = new List&lt;string&gt;(); for (int i = 0; i &lt; _ss.Length;
++i)

        if (_ss[i].Length &gt; 0) ss.Add(_ss[i]);

        int p = 0;

        int n = int.Parse(ss[p++]); int[] size = new int[n];

        for (int i = 0; i &lt; n; ++i) size[i] = int.Parse(ss[p++]);

        layers = new List&lt;Layer&gt;(); for (int i = 0; i &lt; n; ++i)

        layers.Add(new Layer(size[i]));

        for (int i = 0; i &lt; n - 1; ++i)

        for (int j = 0; j &lt; layers[i].vertex.Count; ++j)

        for (int k = 0; k &lt; layers[i + 1].vertex.Count; ++k)
layers[i].vertex[j].next.Add(layers[i + 1].vertex[k]);

        for (int i = 0; i &lt; n; ++i)



        {

        for (int j = 0; j &lt; size[i]; ++j)

        {

        Vertex v = layers[i].vertex[j]; v.threshold = double.Parse(ss[p++]); int m =
int.Parse(ss[p++]);
```

```
for (int k = 0; k &lt; m; ++k) v.wnext.Add(double.Parse(ss[p++]));

}

}

}

catch (Exception e)

{

throw new NeuralNetworkException(e.Message);

}

finally

{

if (sw != null) sw.Close();

}

}

public NeuralNetwork(string filename)

{

Load(filename);

}

public NeuralNetwork(int[] sizes)

{

Random rand = new Random((int)DateTime.Now.Ticks); layers = new List&lt;Layer&gt;();

for (int i = 0; i &lt; sizes.Length; ++i) layers.Add(new Layer(sizes[i]));

for (int i = 0; i &lt; sizes.Length - 1; ++i)

{
```

```
for (int j = 0; j &lt; layers[i].vertex.Count; ++j)

for (int k = 0; k &lt; layers[i + 1].vertex.Count; ++k)

{

layers[i].vertex[j].next.Add(layers[i + 1].vertex[k]);
layers[i].vertex[j].wnext.Add(rand.NextDouble() * 0.4

- 0.2);

}

}

}

public void clear()

{

for (int i = 0; i &lt; layers.Count; ++i)



for (int j = 0; j &lt; layers[i].vertex.Count; ++j) layers[i].vertex[j].value =
0.0;

}

private double sigmoida(double x)

{

return 1.0 / (1.0 + Math.Exp(-x));

}

public int InputLayerSize { get { return layers[0].vertex.Count; } } public
void Run(double[] input)

{

clear();

if (input.Length != layers[0].vertex.Count) throw new
Exception(&amp;quot;Input layer and input has different
```

```
number of items&amp;quot;);

// init first layer

for (int i = 0; i &amp;lt; input.Length; ++i)

{

layers[0].vertex[i].value = input[i];

for (int j = 0; j &amp;lt; layers[0].vertex[i].next.Count; ++j)
layers[0].vertex[i].next[j].value += input[i] *

layers[0].vertex[i].wnext[j];

}

// next layers

for (int t = 1; t &amp;lt; layers.Count; ++t)

for (int i = 0; i &amp;lt; layers[t].vertex.Count; ++i)

{

Vertex v = layers[t].vertex[i];

v.value = sigmoida(v.value - v.threshold); for (int j = 0; j &amp;lt; v.next.Count;
++j)

v.next[j].value += v.value * v.wnext[j];

}

}

public void Learn(double[] input, double[] need)

{

if (need.Length != layers[layers.Count - 1].vertex.Count)

throw new Exception(&amp;quot;Test is something different from
expected&amp;quot;); Run(input);

// last layer:
```

```
for (int i = 0; i &lt; layers[layers.Count - 1].vertex.Count; ++i)

{

Vertex v = layers[layers.Count - 1].vertex[i];



v.delta = v.value * (1.0 - v.value) * (need[i] - v.value);

}

// other layers:

for (int t = layers.Count - 2; t &gt;= 0; --t) foreach (Vertex v in
layers[t].vertex)

{

v.delta = 0.0;

for (int i = 0; i &lt; v.next.Count; ++i) v.delta += v.next[i].delta * v.wnext[i];

v.delta *= v.value * (1.0 - v.value);

}

// learning:

for (int t = layers.Count - 1; t &gt;= 0; --t)

for (int i = 0; i &lt; layers[t].vertex.Count; ++i)

{

Vertex v = layers[t].vertex[i];

for (int j = 0; j &lt; v.next.Count; ++j)

{

Vertex y = v.next[j];

v.wnext[j] += speed * y.delta * v.value;

}
```

```
}

}

}

}
```

Файл WordDB

```
using System;

using System.Collections.Generic; using System.Linq;

using System.Text; using System.IO;

using System.Runtime.Serialization.Formatters.Binary; using
System.Runtime.Serialization;

namespace iTellUtils

{

public class WordDBCantSaveException: Exception

{

public WordDBCantSaveException(string s):base(s) { }

}

public class WordDBCantLoadException : Exception

{

public WordDBCantLoadException(string s) : base(s) { }



}

[Serializable] public class WordDB

{

static private WordDB instance = null;
```

```
        static public WordDB Instance { get { if (instance == null) instance = new
WordDB(); return instance; } }

        private SortedSet&lt;string&gt; words = new SortedSet&lt;string&gt;();

        private double getProb(string s, FuzzyVector v) { if (s.Length !=
v.chars.Count) return 0.0; double prob =

        1.0;

        for(int i=0; i&lt;s.Length; ++i)

        prob *= v.chars[i][(int)(s[i]-&#39;a&#39;)]; return prob;

        }

        public int WordsCount { get { return words.Count; } } private bool
checkPref(string s1, string s2)

        {

        if (s1.Length&lt;s2.Length) return false;

        int i = 0; while (i &lt; s2.Length) if (s1[i] != s2[i]) return false; else ++i;
return true;

        }

        public string[] getWords(string pref)

        {

        string[] w = words.ToArray(); List&lt;string&gt; list = new
List&lt;string&gt;(); for (int i = 0; i &lt; w.Length; ++i)

        {

        if (checkPref(w[i], pref)) list.Add(w[i]);

        }

        return list.ToArray();

        }

        public void LoadFromTextFile(string filename)
```

```
{
try
{
words.Clear();
string[] w = File.ReadAllLines(filename);
for (int i = 0; i &lt; w.Length; ++i) if (w[i].Length &gt; 0)
{
AddWord(w[i]);
}

}
catch (Exception e)
{
throw new WordDBCantLoadException(e.Message);
}
}
public void SaveToTextFile(string filename)
{
try
{

}
string[] s = words.ToArray(); File.WriteAllLines(filename, s);
catch (Exception e)
```

```
{

throw new WordDBCantSaveException(e.Message);

}

}

public WordDB() { }

public WordDB(string[] words)

{

for (int i = 0; i &lt; words.Length; ++i) this.words.Add(words[i]);

}

public void AddWord(string s)

{

string ss=&quot;&quot;;

for (int i = 0; i &lt; s.Length; ++i) ss += Char.ToLower(s[i]); if
(words.Contains(ss)) return;

words.Add(ss);

}

public void RemoveWord(string s)

{

if (!words.Contains(s)) return; words.Remove(s);

}

public WordDBSearchResult findBestWords(FuzzyVector word)

{



WordDBSearchResult res = new WordDBSearchResult(10); string[] W =
getWords(&quot;&quot;);
```

```
for (int i = 0; i &lt; W.Length; ++i)
{
res.AddWord(W[i], getProb(W[i], word));
}
return res;
}
}
public class WordDBSearchResult
{
private List&lt;double&gt; prob; private List&lt;string&gt; word;
public WordDBSearchResult(int limit) { prob = new List&lt;double&gt;();
word = new List&lt;string&gt;();
}
public void AddWord(string word, double prob)
{
this.prob.Add(prob); this.word.Add(word);
}
public int Count()
{return 0;}
public string GetFirstWord()
{
double best = 0.0; string res = &quot;&quot;;
for (int i=0; i &lt; prob.Count; ++i)
{
```

```
if (prob[i] &gt; best)
{
best = prob[i]; res = word[i];
}
}
return res;
}
public string GetWordAt(int index)
{ return null; }
public double GetProbAt(int index)
{ return 0.0; }
}

public class FuzzyVector
{
public List&lt;double[]&gt; chars = new List&lt;double[]&gt;();
}
}
```

Файл BlockRecognizer

```
using System;
using System.Collections.Generic; using System.Linq;
using System.Text; using System.Drawing;
namespace iTellUtils
{
```

```
public class BlockRecognizer

{

static private int Width; static private int Height; static int[,] bits = null;

static public bool Loaded() { return (bits != null); } static public void StartImage(int [,] img, int W, int H)

{

Width = W; Height = H; bits = img; line = 0;

column = 0;

}

static public void Reset()

{

line = column = 0;

}

public static int dx0 = 2; public static int dy0 = 0; public static int dx = 7; public static int dy = 16;

static public int line; static private int column;

static public Bitmap NextBitmapChar()

{

if ((line+1)*dy + dy0 &gt; Height) return null; int y = line * dy + dy0;

int x = column * dx + dx0; if (x + dx &gt; Width) {

++line; column = 0;

return NextBitmapChar();

}

Bitmap res = new Bitmap(dx, dy); int sum = 0;

for (int i = 0; i &lt; dx; ++i) for (int j = 0; j &lt; dy; ++j) {
```

```
//res.SetPixel(i, j, Color.Red); sum += bits[x + i, y + j];

res.SetPixel(i, j, Color.FromArgb(255 - bits[x + i, y + j], 255 - bits[x + i, y + j], 255

- bits[x + i, y + j]));

}

++column; return res;

}

static public double[] NextDoubleChar()

{

if ((line + 1) * dy + dy0 &gt; Height) return null; int y = line * dy + dy0;

int x = column * dx + dx0; if (x + dx &gt; Width)

{

++line; column = 0;

return NextDoubleChar();

}

double[] res = new double[dx*dy]; int sum = 0;

for (int j = 0; j &lt; dy; ++j) for (int i = 0; i &lt; dx; ++i)

{

res[sum++]= (double)bits[x + i, y + j]/255.0;

}

++column; return res;

}

}
```

```
}
```

Файл ImageFilter

```
using System;
using System.Collections.Generic; using System.Linq;
using System.Text; using System.Drawing;
namespace iTellUtils
{
public class ImageFilter
{
static int ColorToInt(Color v)
{
return Math.Max(v.B,Math.Max(v.G,v.R));
}
static Color IntToColor(int x) {

return Color.FromArgb(x, x, x);

}
public static Bitmap DoFilteringSimple(Bitmap source) { Bitmap result =
(Bitmap)source.Clone();
for (int x = 0; x &lt; result.Width; ++x)
for (int y = 0; y &lt; result.Height; ++y)
{
result.SetPixel(x,y,IntToColor(ColorToInt(source.GetPixel(x, y))));
```

```
}

return result;

}

public static Bitmap DoFiltering(Bitmap source)

{

Bitmap result = new Bitmap(source.Width, source.Height); int[] pixels =
new int[8];

int[] dx = {-1,-1,-1,0,1,1, 1, 0};

int[] dy = {-1, 0, 1,1,1,0,-1,-1};

for (int x = 0; x &lt; result.Width; ++x)

{

for (int y = 0; y &lt; result.Height; ++y)

{

if (x == 317 &amp;&amp; y == 106) { Console.WriteLine();

}

int n = 0;

for(int i=0; i&lt;8; ++i)

{

int xx=x+dx[i]; int yy=y+dy[i];

if (xx&lt;0 || yy&lt;0 || xx&gt;=source.Width || yy&gt;=source.Height)
continue; pixels[n++] =

ColorToInt(source.GetPixel(xx,yy));

}

for(int i=0; i&lt;n-1; ++i) for(int j=0; j&lt;n-1; ++j)

if (pixels[j]&gt;pixels[j+1])
```

```
{

}

if (n&lt;8) {

pixels[j]^=pixels[j+1]; pixels[j+1]^=pixels[j]; pixels[j]^=pixels[j+1];

result.SetPixel(x,y,IntToColor(ColorToInt(source.GetPixel(x,y))));

}

else

{


int X = ColorToInt(source.GetPixel(x,y));

double med = (double)(pixels[3]+pixels[4])/2.0d; double r1 =
((double)X&lt;=med)?pixels[0]-X:X-pixels[7];

double r2 = ((double)X&lt;=med)?pixels[1]-X:X-pixels[6]; double r3 =
((double)X&lt;=med)?pixels[2]-X:X-pixels[5];

double r4 = ((double)X&lt;=med)?pixels[3]-X:X-pixels[4];

if (r1&gt;8.0 || r2&gt;20.0 || r3&gt;40.0 || r4&gt;80.0)

{

}

else

{

}

}

}

}

result.SetPixel(x,y,IntToColor( (int)(1.0*med + 0.0*X + 0.5) ));
```

```
result.SetPixel(x,y,IntToColor( (int)(0.0*med + 1.0*X + 0.5) ));
return result;
}
}
}
```

Файл Recognizer

```
using System;
using System.Collections.Generic; using System.Linq;
using System.Text; using iTellUtils;

using System.Drawing;
using System.Drawing.Imaging;
namespace iTell
{
public class Recognizer
{
static public List&lt;double[]&gt; getData(int[][] images)

{
List&lt;double[]&gt; res = new List&lt;double[]&gt;(); for (int i = 0; i &lt; images.Length; ++i)
{
double[] a = new double[NeuralNetwork.Current.InputLayerSize]; for (int r = 0; r &lt; images[i].Length; ++r)
```

```
a[r] = (double)images[i][r] / 256.0; res.Add(a);

}

return res;

}

static public FuzzyVector GetVector(List&lt;double[]&gt; data)

{

FuzzyVector vect = new FuzzyVector(); for (int i = 0; i &lt; data.Count; ++i)
{

NeuralNetwork.Current.Run(data[i]); double[] v =
NeuralNetwork.Current.Output(); double sum = 0.0;

for (int j = 0; j &lt; 26; ++j) sum += v[j]; for (int j = 0; j &lt; 26; ++j) v[j] /=
sum; vect.chars.Add(v);

}

return vect;

}

static public FuzzyVector GetVector(string word)

{

FuzzyVector vect = new FuzzyVector(); for (int i = 0; i &lt; word.Length;
++i)

{

double[] y = new double[26]; for (int j = 0; j &lt; 26; ++j)

y[j]=((int)(word[i]-&#39;a&#39;)==j)?1.0:0.0;

vect.chars.Add(y);

}

return vect;

}
```

}

}

Файл MainForm

using System;

using System.Collections.Generic; using System.ComponentModel; using System.Data;

using System.Drawing; using System.Linq; using System.Text;

using System.Windows.Forms; using iTellUtils;

using System.IO;

namespace iTell

{

public partial class MainForm : Form

{

private ConsoleForm consoleForm = new ConsoleForm(); public void showConsole(bool yes)

{

UserSettings.Instance.ShowConsole = yes; вивестиКонсольToolStripMenuItem.Checked = yes; if (yes)

consoleForm.Show(); else

consoleForm.Hide();

}

public MainForm()

{

InitializeComponent();

```
}

private void LoadWordDB(string filename)

{

Console.Message(&quot;Завантаження бази слів з файлу &quot; +
filename); bool ok = true;

try { WordDB.Instance.LoadFromTextFile(filename); }

catch (WordDBCantLoadException e) { ok = false;
Console.Error(&quot;LoadWordDB&quot;, &quot;Помилка

завантаження слів з файлу: &quot; + e.Message); }

if (ok) Console.Message(&quot;База слів завантажена&quot;);

}

private void SaveWordDB(string filename)

{

Console.Message(&quot;Збереження бази слів до файлу &quot; +
filename); bool ok = true;

try { WordDB.Instance.SaveToTextFile(filename); }

catch (WordDBCantSaveException e) { ok = false;
Console.Error(&quot;SaveWordDB&quot;, &quot;Помилка збереження

слів до файлу: &quot; + e.Message); }

if (ok) Console.Message(&quot;База слів збережена&quot;);

}

private void Form1_Load(object sender, EventArgs e)

{

consoleForm.Width = 600;

consoleForm.Height = 200; Console.setForm(consoleForm);

if (UserSettings.Instance.ShowConsole)
```

```
        {
        consoleForm.Show(); вивестиКонсольToolStripMenuItem.Checked = true;
        }
        else
        {
        consoleForm.Hide(); вивестиКонсольToolStripMenuItem.Checked = false;


        }
        // todo loading words
        if (UserSettings.Instance.CurrentDB != &amp;quot;&amp;quot;)
LoadWordDB(UserSettings.Instance.CurrentDB);
        // todo loading net
        if (UserSettings.Instance.CurrentNetwork != &amp;quot;&amp;quot;)
        NeuralNetwork.Current = new
NeuralNetwork(UserSettings.Instance.CurrentNetwork); else
        {
        int[] sizes = new int[] { 112,100,26 }; NeuralNetwork.Current = new
NeuralNetwork(sizes);
        }
        }
        private void button1_Click(object sender, EventArgs e)
        {
        }
        private void button2_Click(object sender, EventArgs e)
        {
```

```
        }

        private void button1_Click_1(object sender, EventArgs e)

        {

        if (openFileDialog1.ShowDialog() == DialogResult.OK)

        {

        pictureBox1.Load(openFileDialog1.FileName);

        Bitmap bits = ImageFilter.DoFilteringSimple(new
Bitmap(pictureBox1.Image)); pictureBox1.Image = bits;

        int [,] img = new int[bits.Width,bits.Height]; for (int i = 0; i &lt; bits.Width;
++i)

        for (int j = 0; j &lt; bits.Height; ++j)

        {

        img[i, j] = 255-(int)bits.GetPixel(i, j).B;



        }

        BlockRecognizer.StartImage(img, bits.Width, bits.Height);

        }

        }

        string goWork(FuzzyVector v, List&lt;double[]&gt; input)

        {

        string s = WordDB.Instance.findBestWords(v).GetFirstWord(); for (int i = 0;
i &lt; s.Length; ++i) {

        int a = (int)s[i] - &#39;a&#39;; double[] need = new double[26];



        for (int j = 0; j &lt; s.Length; ++j)
```

```
        {
        need[j] = (j == a) ? 1.0 : 0.0;
        }
        NeuralNetwork.Current.Learn(input[i], need);
        }
        return s;
        }
        private void button2_Click_1(object sender, EventArgs e)
        {
        DateTime beginTime = DateTime.Now; int total=0;
BlockRecognizer.Reset();
        FuzzyVector word = new FuzzyVector(); List&lt;double[]&gt; imgs = new
List&lt;double[]&gt;(); StringBuilder sb =
        new StringBuilder(); int lastline = BlockRecognizer.line;
        int last = 0; while(true) {
        last = BlockRecognizer.line;
        double[] x = BlockRecognizer.NextDoubleChar();
        ++total;
        if (x == null) break;
        int spaces = 0; for (int i = 0; i &lt; x.Length; ++i) if (x[i] &lt; 0.1) spaces++;
if (spaces &gt; x.Length * 90 / 100)
        {
        if (word.chars.Count &gt; 0)
        {
        if (last != lastline)
```

```
        {
        sb.Append(&quot;\n&quot;);
        }
        }
        }
        else
        {
        lastline=last; sb.Append(goWork(word, imgs)+&quot; &quot;);
word.chars.Clear();
        imgs.Clear();
        imgs.Add(x); NeuralNetwork.Current.Run(x);
        word.chars.Add(NeuralNetwork.Current.Output());
        }
        }
        if (word.chars.Count &gt; 0)
        {

        sb.Append(goWork(word, imgs)); word.chars.Clear(); imgs.Clear();
        }
        richTextBox1.Text = sb.ToString();
        double v=(DateTime.Now - beginTime).TotalSeconds;
MessageBox.Show(&quot;Затрачений час: &quot; +
        v.ToString() + &quot; сек.\n&quot; +
        &quot;Всього символів: &quot; + total.ToString() + &quot;\n&quot; +
```

```
&amp;quot;Час обробки одного символу: &amp;quot; + (v / (double)total).ToString() + &amp;quot;

сек/символ&amp;quot;);

}

private void вивестиКонсольToolStripMenuItem_Click(object sender, EventArgs e)

{

showConsole(!UserSettings.Instance.ShowConsole);

}

private void вихідToolStripMenuItem_Click(object sender, EventArgs e)

{

Close();

}

private void налаштуванняToolStripMenuItem_Click(object sender, EventArgs e)

{

SettingsForm settingsForm=new SettingsForm(this); settingsForm.ShowDialog();

}

private void редагуванняБазиToolStripMenuItem_Click(object sender, EventArgs e)

{

DBForm form = new DBForm(); form.ShowDialog();

}

private void імпортуватиЗФайлуToolStripMenuItem_Click(object sender, EventArgs e)

{
```

```
        if (openFileDialog1.ShowDialog() == DialogResult.OK)
LoadWordDB(openFileDialog1.FileName);

        }

        private void експортуватиВФайлToolStripMenuItem_Click(object sender,
EventArgs e)

        {

        if (saveFileDialog1.ShowDialog()==DialogResult.OK)
SaveWordDB(saveFileDialog1.FileName);

        }

        private void button4_Click(object sender, EventArgs e)

        {



        if (BlockRecognizer.Loaded()) { TestForm form = new TestForm();
form.ShowDialog();

        }

        }

        private void
завантажитиНейроннуМережуToolStripMenuItem_Click(object sender,
EventArgs e)

        {

        if (openFileDialog1.ShowDialog() == DialogResult.OK)

        NeuralNetwork.Current.Load(openFileDialog1.FileName);

        }

        private void зберегтиНейроннуМережуToolStripMenuItem_Click(object
sender, EventArgs e)

        {

        if (saveFileDialog1.ShowDialog() == DialogResult.OK)
```

```
NeuralNetwork.Current.Save(saveFileDialog1.FileName);

}

private void навчанняНаОсновіШаблонних СимволівToolStripMenuItem_Click(object sender,

EventArgs e)

{

LearningForm form = new LearningForm(); form.ShowDialog();

}

private void button3_Click(object sender, EventArgs e)

{

if (saveFileDialog1.ShowDialog() == DialogResult.OK) {

File.WriteAllText(saveFileDialog1.FileName,richTextBox1.Text);

}

}

private void завантажитиЗображенняToolStripMenuItem_Click(object sender, EventArgs e)

{

button1_Click_1(sender, e);

}

private void зберегтиРезультатиРозпізнаванняToolStripMenuItem_Click(object sender, EventArgs e)

{

button3_Click(sender, e);

}
```

```
}

}
```

Файл LearningForm

```
using System;

using System.Collections.Generic; using System.ComponentModel; using System.Data;

using System.Drawing; using System.Linq; using System.Text;

using System.Windows.Forms; using iTellUtils;

namespace iTell

{

public partial class LearningForm : Form

{

public LearningForm()

{

InitializeComponent();

}

private void button1_Click(object sender, EventArgs e)

{

if (folderBrowserDialog1.ShowDialog() == DialogResult.OK)

{

textBox1.Text = folderBrowserDialog1.SelectedPath;

}

}

Random rand = new Random((int)DateTime.Now.Ticks); private void button2_Click(object sender,
```

```
        EventArgs e)

        {

        List&lt;double[]&gt; bits = new List&lt;double[]&gt;();
List&lt;double[]&gt; need = new List&lt;double[]&gt;(); for(int i=0;

        i&lt;26; ++i)

        {

        double[] res = new double[7*16];

        Bitmap p = new
Bitmap(textBox1.Text+&quot;\\&quot;+(char)((int)&#39;A&#39;+i)+&quot;.bmp
&quot;); int pos=0;

        for(int y=0; y&lt;16; ++y) for(int x=0; x&lt;7; ++x) res[pos++] =
(double)(255- p.GetPixel(x,y).B)/255.0;

        bits.Add(res);

        }

        for(int i=0; i&lt;26; ++i)

        {

        double[] a = new double[26];

        for(int j=0; j&lt;26; ++j) a[j] = (i==j)?1.0:0.0; need.Add(a);

        }

        StringBuilder sb = new StringBuilder(); int iter =
(int)numericUpDown1.Value; int total=0;

        int wins=0; progressBar1.Minimum=0; progressBar1.Maximum=iter;

        for (int i = 0; i &lt; iter; ++i)

        {

        progressBar1.Value = i; Application.DoEvents();

        for (int a = 0; a &lt; 100; ++a)
```

```
        {
        int x = rand.Next() % 26;
        NeuralNetwork.Current.Run(bits[x]);
        if (NeuralNetwork.Current.getDesision()==x)
{sb.Append(&quot;+&quot;);++wins;} else sb.Append(&quot;o&quot;);
        ++total;
        NeuralNetwork.Current.Learn(bits[x], need[x]);
        }
        progressBar1.Value = i + 1;
        }
        richTextBox1.Text = sb.ToString() +
&quot;\n&quot;+&quot;[&quot;+wins.ToString()+&quot; of
&quot;+total.ToString()+&quot;]&quot;;
        }
        private void button4_Click(object sender, EventArgs e)
        {
        if (openFileDialog1.ShowDialog() == DialogResult.OK)
        NeuralNetwork.Current.Load(openFileDialog1.FileName);
        }
        private void button3_Click(object sender, EventArgs e)
        {
        if (saveFileDialog1.ShowDialog() == DialogResult.OK)
        NeuralNetwork.Current.Save(saveFileDialog1.FileName);
        }
        }
```

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: «Нейромережевий застосунок для розпізнавання текстових зображень із завадами»

Тип роботи: Магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ КСУ, ФІТА
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 92,6% Схожість 7,4%

Аналіз звіту подібності (відмітити потрібне)

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

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

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

Особа, відповідальна за перевірку _____ Галущак А.В.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unichek щодо роботи.

Автор роботи _____ Поліщук А.А.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Ковтун В.В.
(підпис) (прізвище, ініціали)

Додаток В
(обов’язковий)

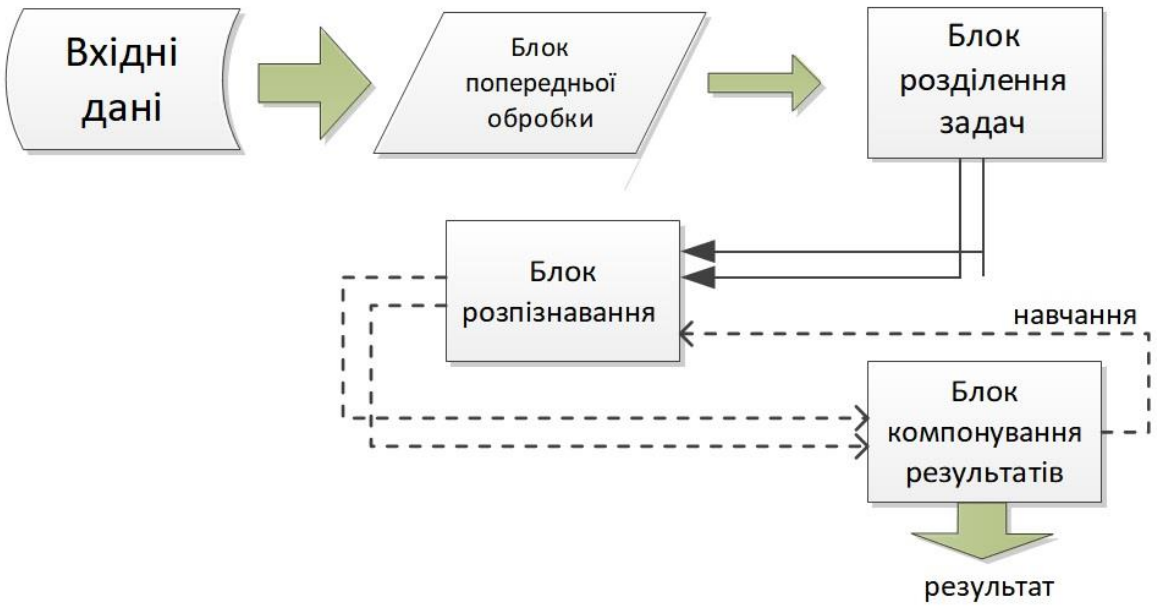
ІЛЮСТРАТИВНА ЧАСТИНА
НЕЙРОМЕРЕЖЕВИ ЗАСТОСУНОК ДЛЯ РОЗПІЗНАВАННЯ ТЕКСТОВИХ ЗОБРАЖЕНЬ ІЗ ЗАВАДАМИ

Студент групи 2АКІТ-21м

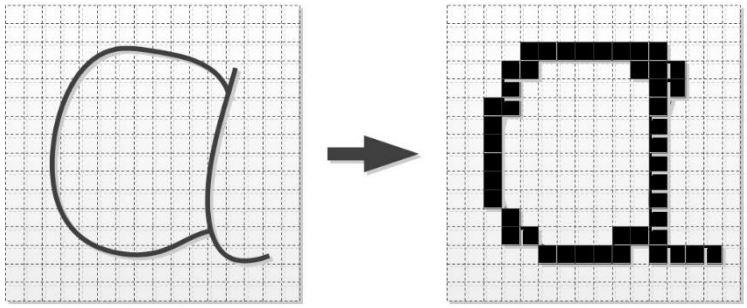
Підпис Анатолій ПОЛІЩУК
Ім’я ПРІЗВИЩЕ

Керівник д.т.н., доцент, проф. кафедри КСУ

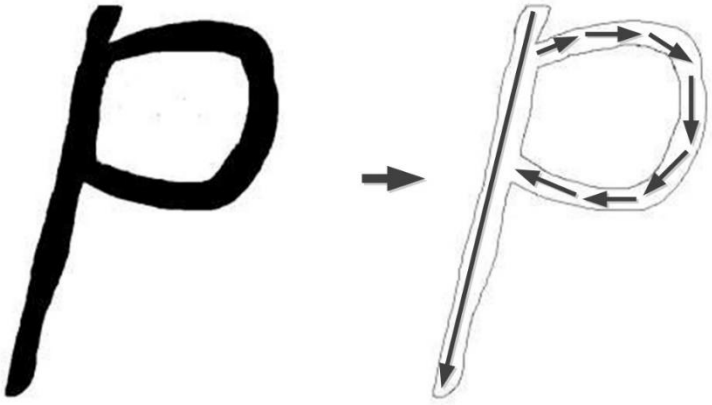
Підпис В’ячеслав КОВТУН
Ім’я ПРІЗВИЩЕ



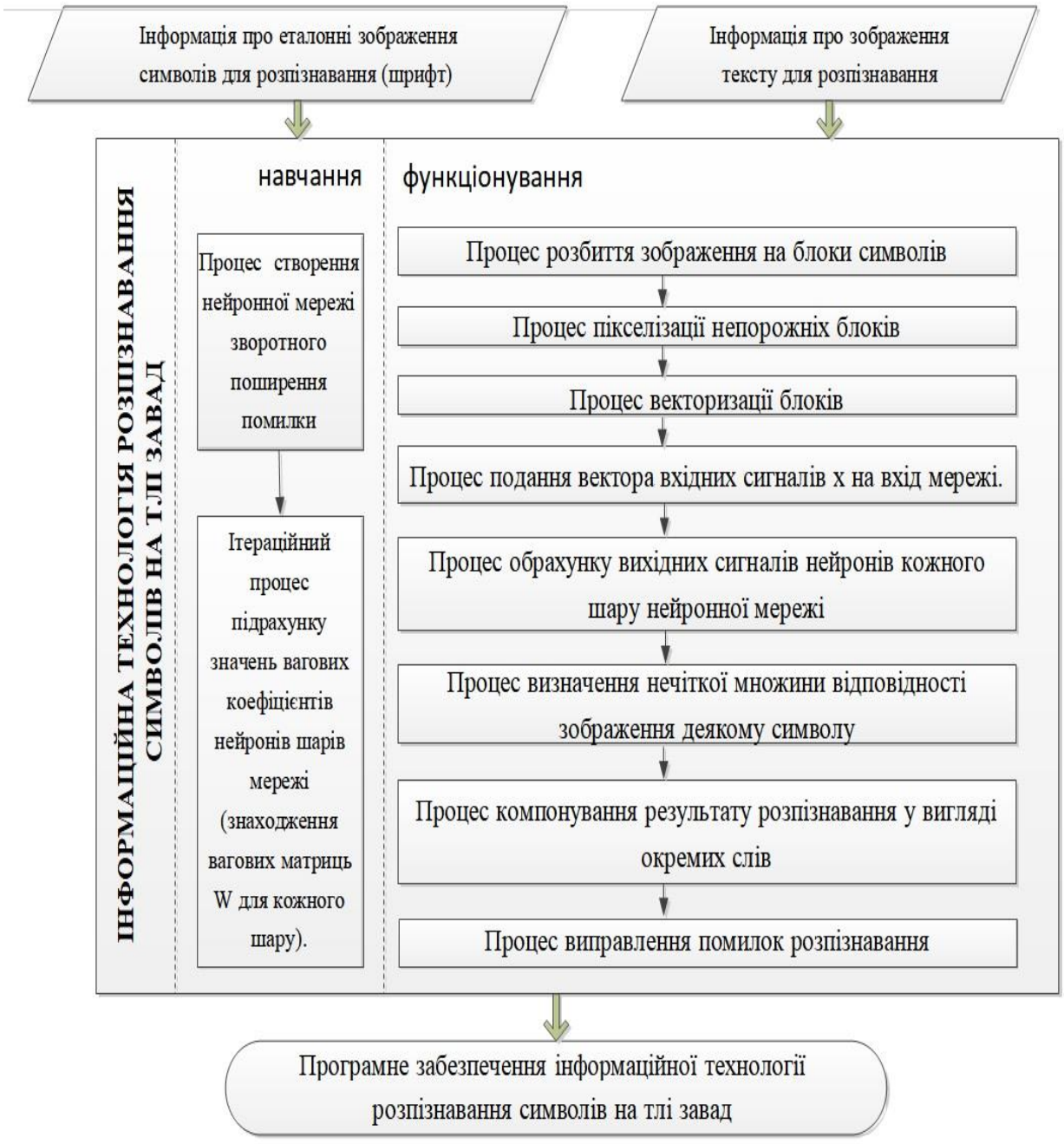
Архітектура додатку розпізнавання символів на тлі завад



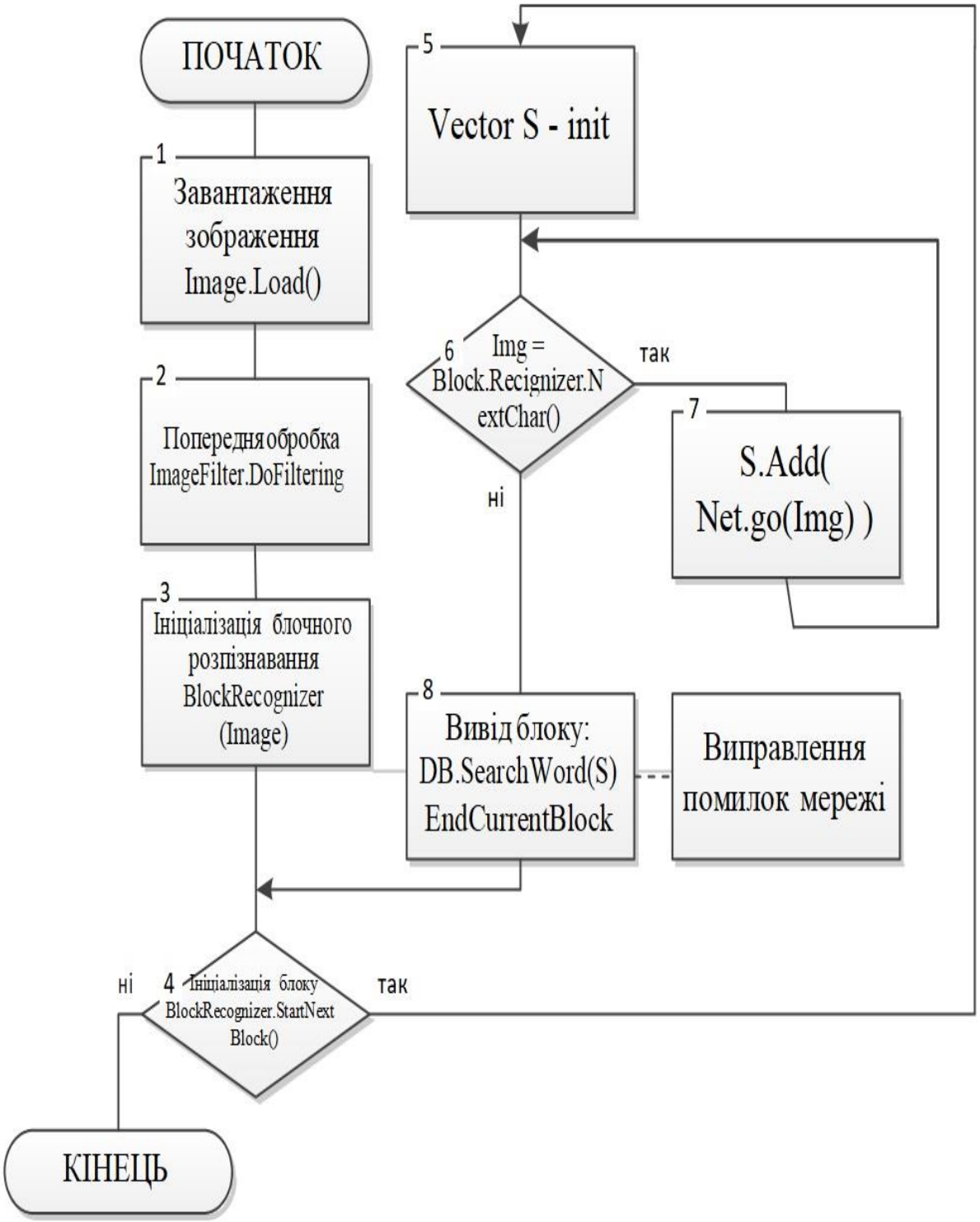
Пікселізація зображення



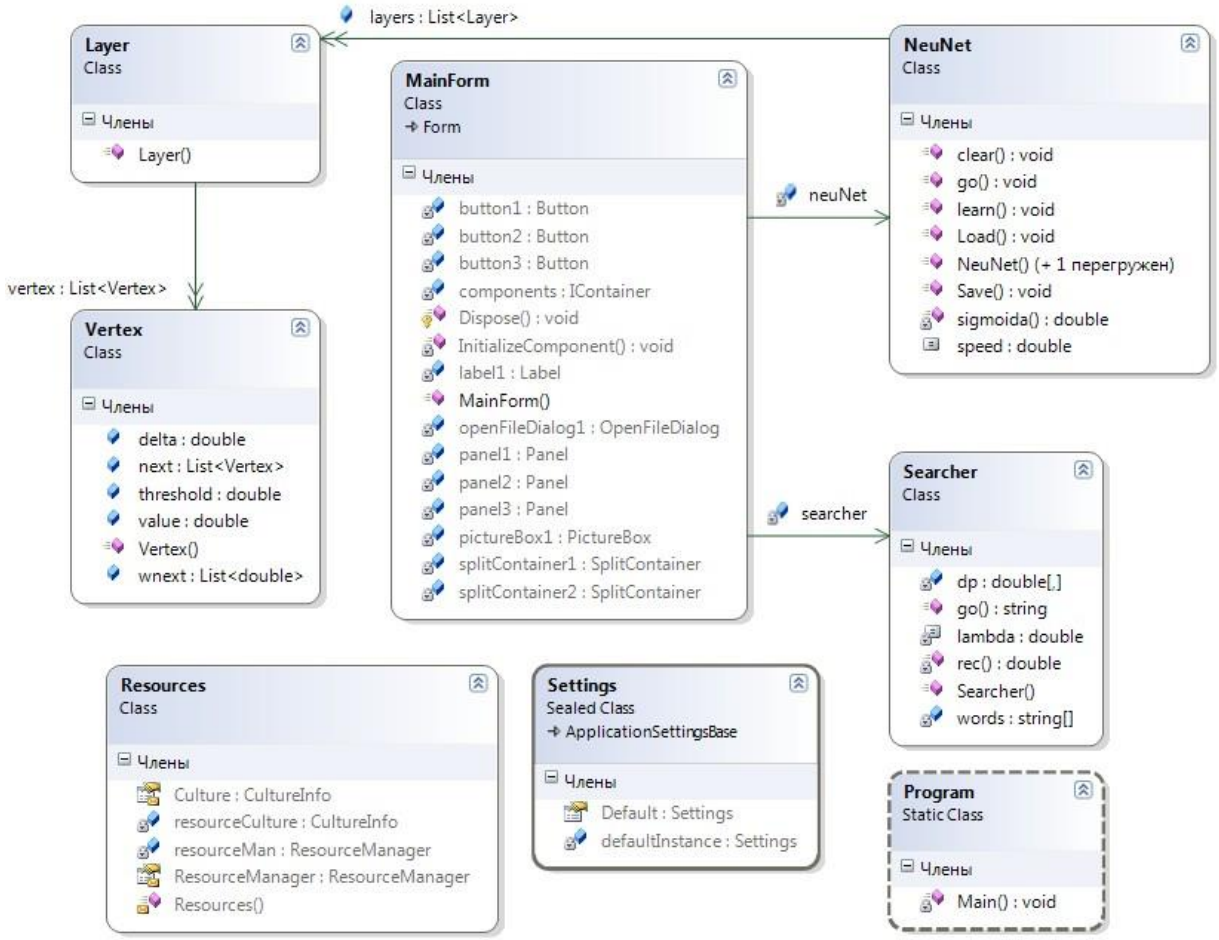
Векторизація зображення



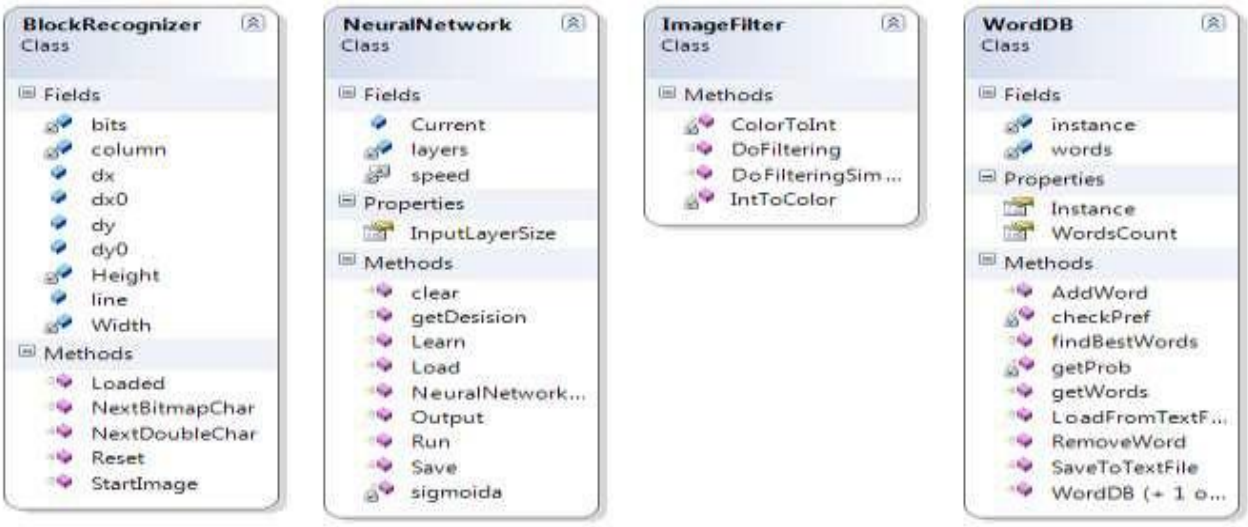
Структура додатку розпізнавання символів на тлі завад



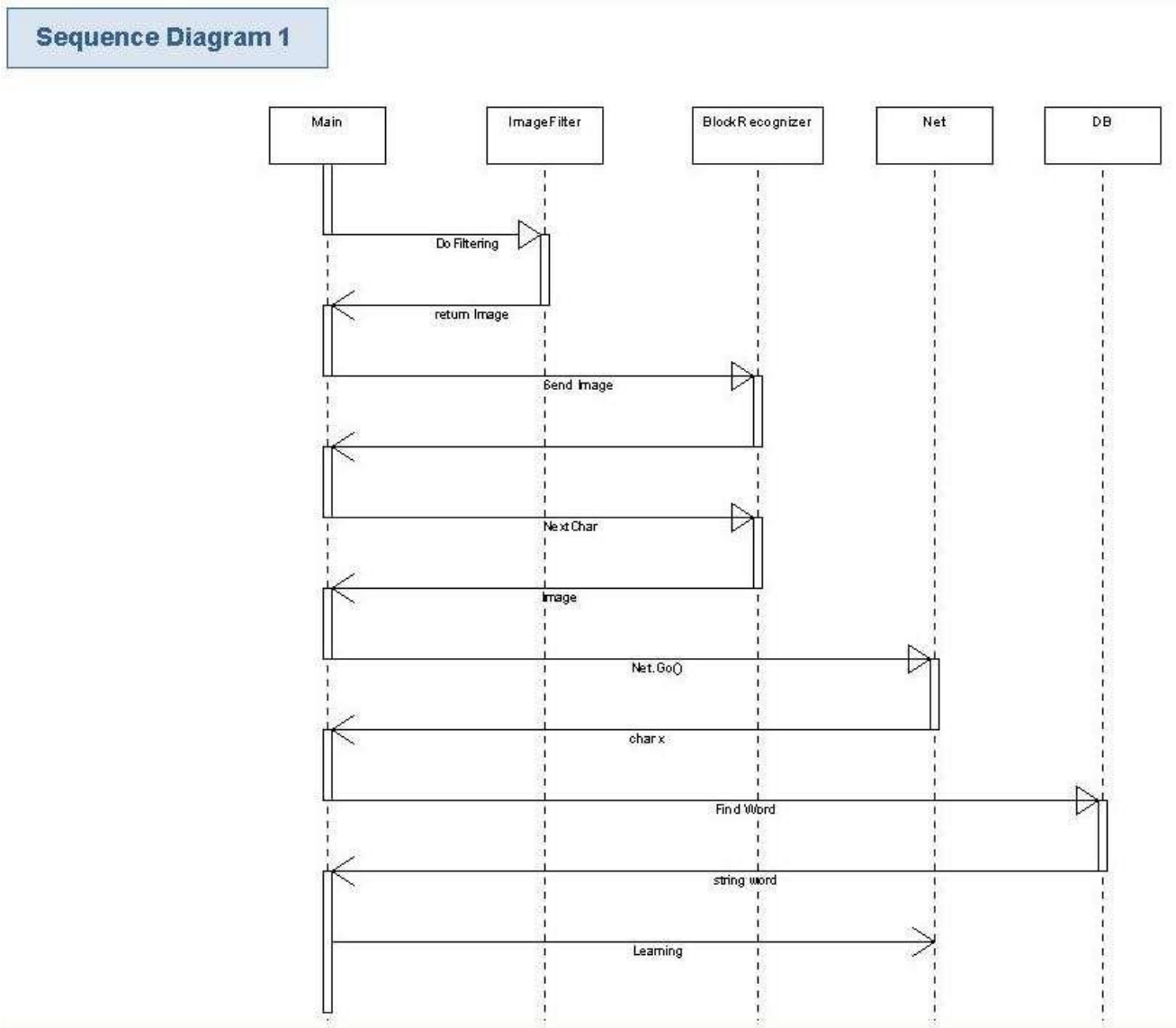
Алгоритм безпосередньо розпізнавання символів на тлі завад



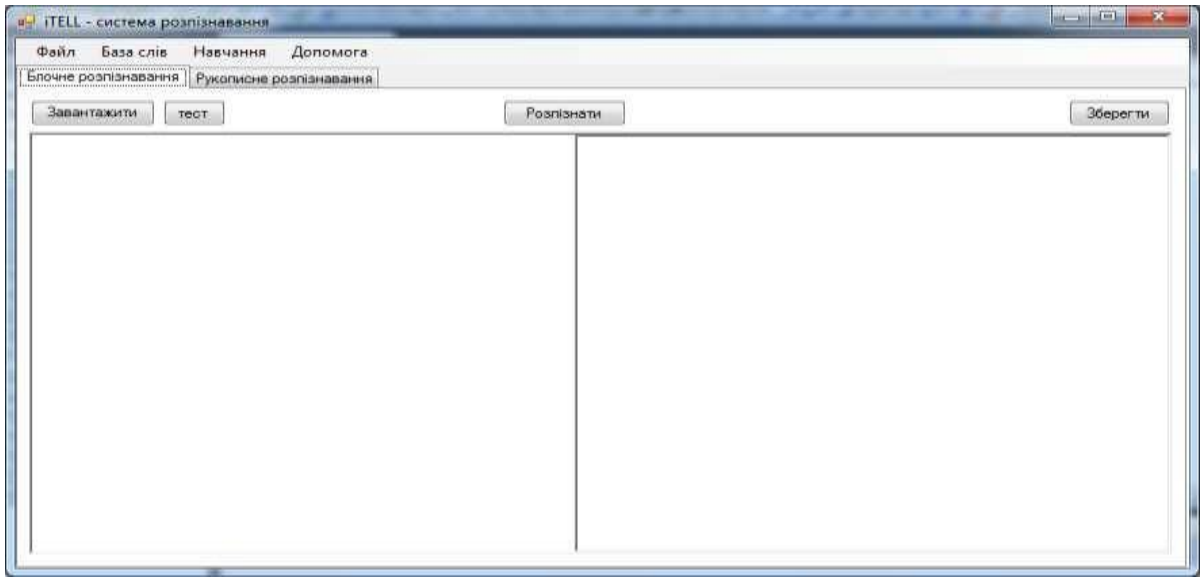
Діаграма класів



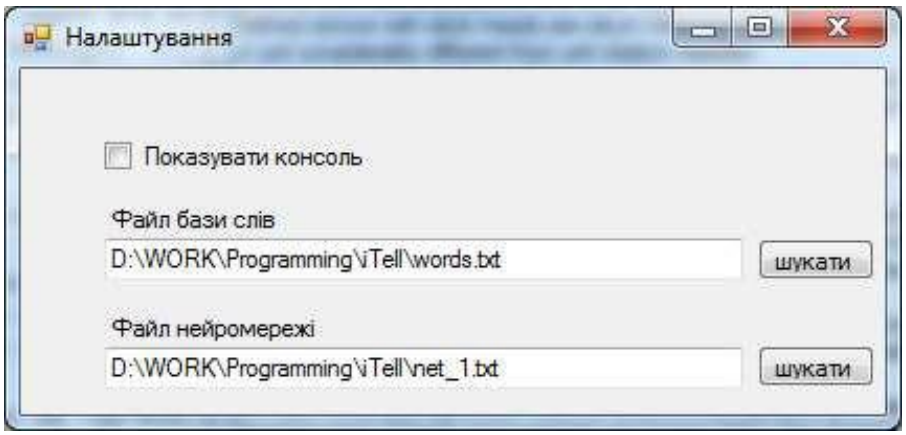
Основні класи програми



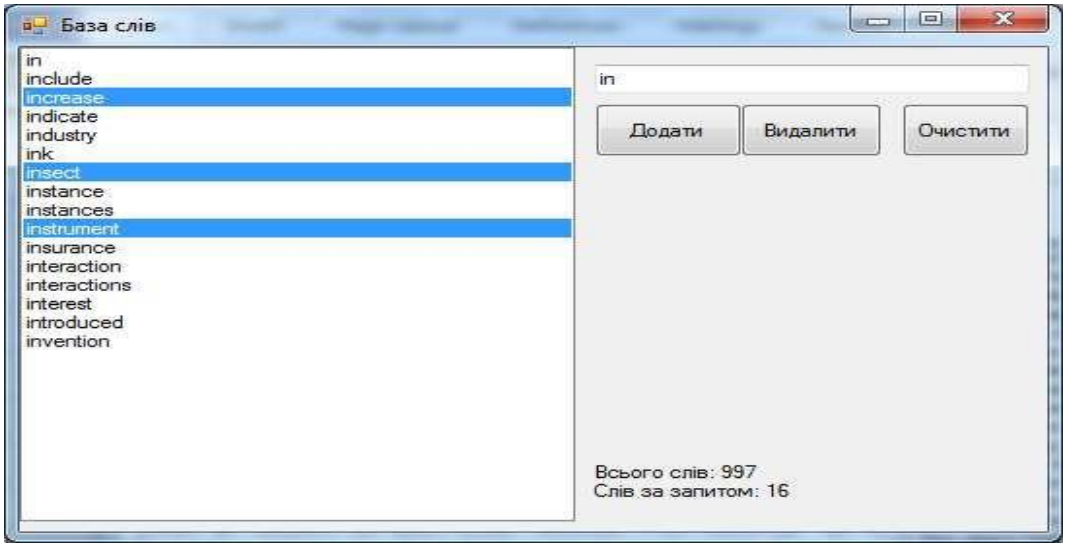
Діаграма послідовності роботи програми



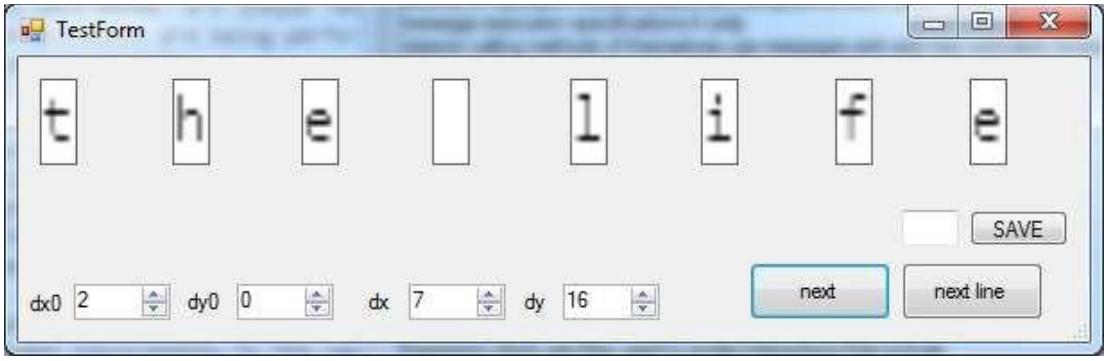
Головне вікно програми



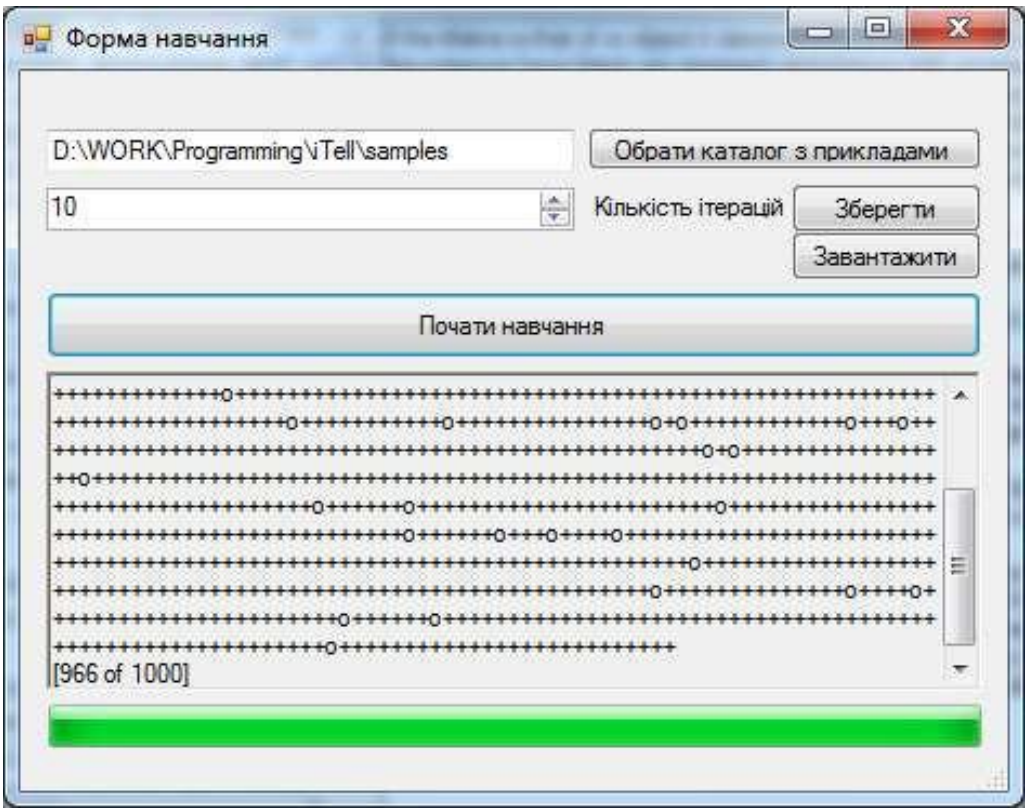
Вікно налаштувань



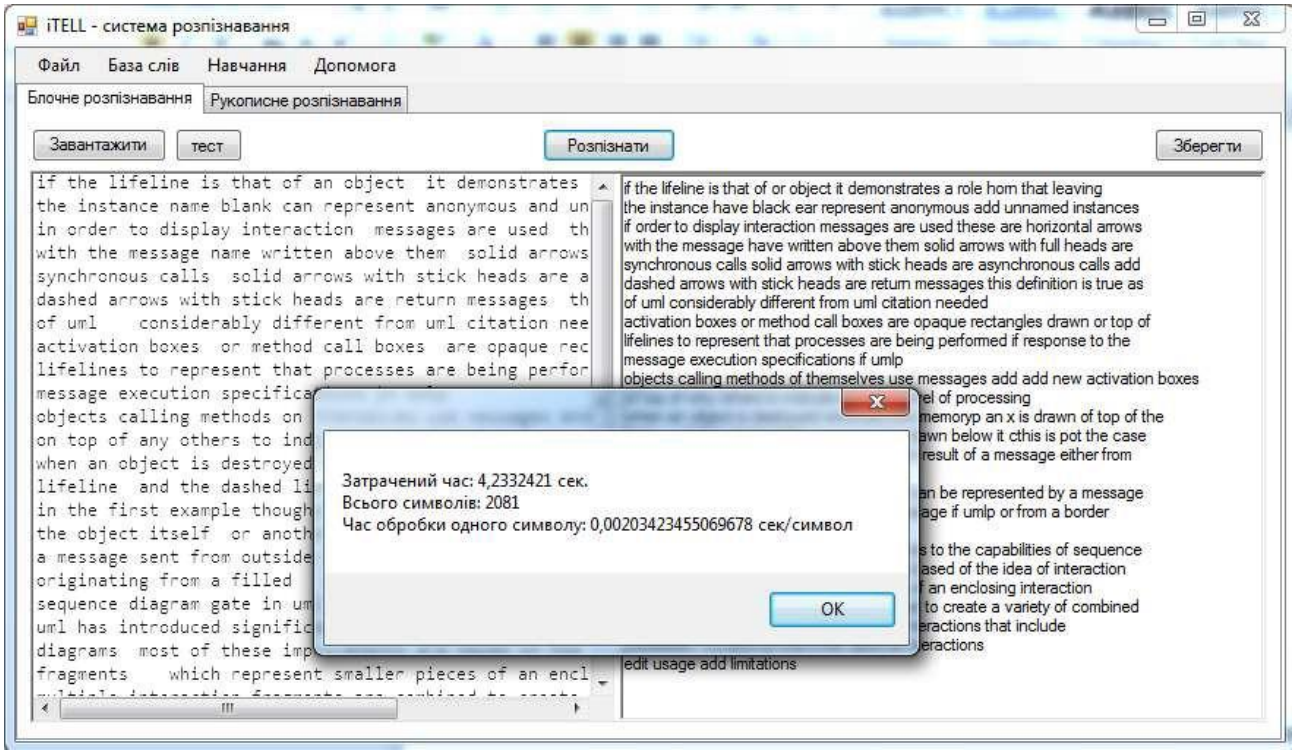
Редактор бази слів



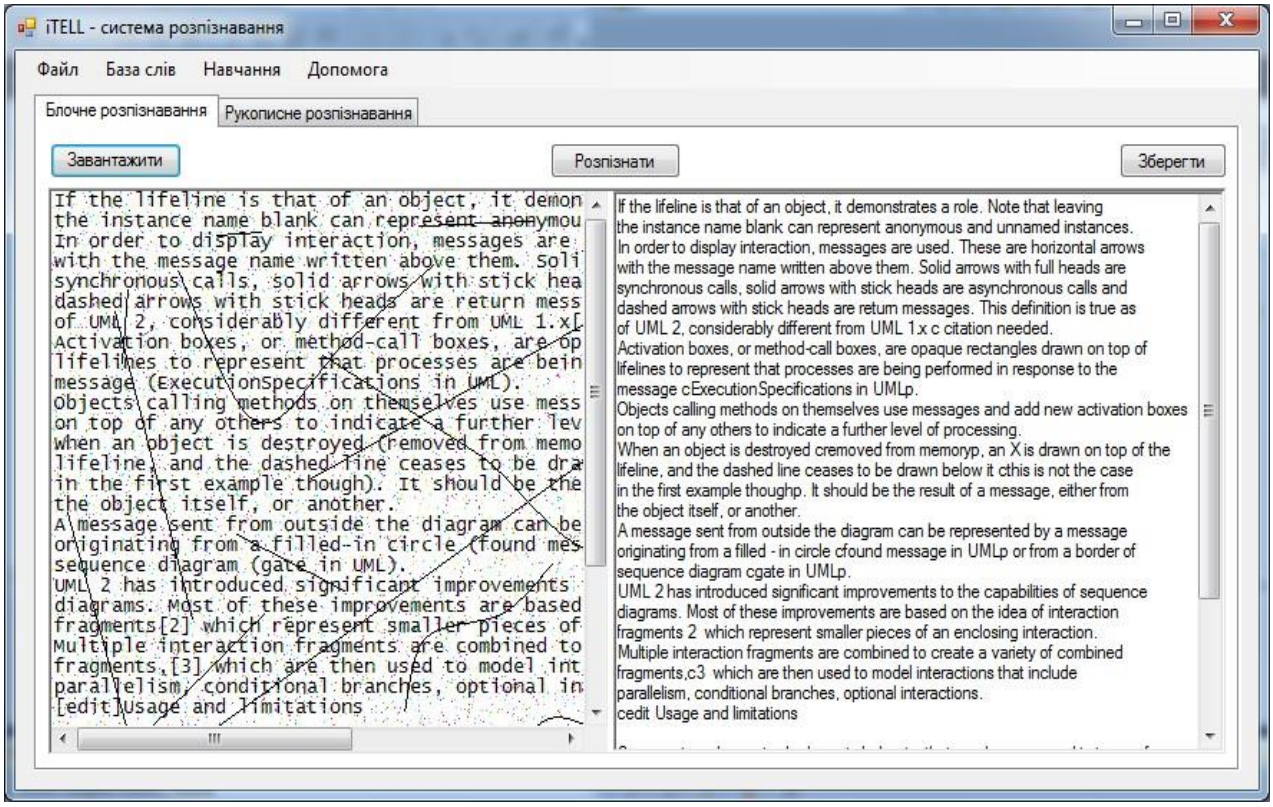
Налаштування модуля блочної обробки



Навчання нейронної мережі



Результат розпізнавання



Тестове розпізнавання зображення із завадами

	Наявність завад			
	0%	5%	10%	20%
Розроблена програма	98,5%	92.4%	82.3%	70.4%,
Аналог (програма MeOCR)	98,1%	90.5%	80.2%	67,9%,
Різниця	0,4%	1,9%	2,1%	2,5%

Достовірність розпізнавання символів на тлі завад