Хмельницький національний університет
Факультет програмування та комп'ютерних і телекомунікаційних систем
Кафедра комп'ютерних наук та інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему *Система розпізнавання образів перцептроном із стохастичним алгоритмом навчання*

Галузь знань *12 – Інформаційні технології*
Шифр і назва галузі знань

Спеціальність *122 – Комп'ютерні науки*
Шифр і назва спеціальності

Освітня програма *Комп'ютерні науки*
Назва освітньої програми

Виконав: *студент 4 курсу, група КН-17-1* _____ *Я.О. Пітик*
Курс, група виконавця — Підпис — Ініціали, прізвище

Керівник: *к.т.н., доцент кафедри КНІТ* _____ *О.В. Мазурець*
Науковий ступінь, посада — Підпис — Ініціали, прізвище

Нормоконтроль: *к.т.н., доцент кафедри КНІТ* _____ *Р.О. Багрій*
Науковий ступінь, посада — Підпис — Ініціали, прізвище

До захисту допускаю:
Зав. кафедри КНІТ, д.т.н., професор _____ *О.В. Бармак*
Підпис — Ініціали, прізвище

08 червня 2021 р.

Хмельницький 2021

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет *програмування та комп'ютерних і телекомунікаційних систем*

Кафедра *комп'ютерних наук та інформаційних технологій*

Освітній ступінь *бакалавр*

Галузь знань *12 – Інформаційні технології*

Спеціальність *122 – Комп'ютерні науки*

ЗАТВЕРДЖУЮ

Завідувач кафедри *комп'ютерних наук та інформаційних технологій*

(підпис)

д.т.н., професор О.В. Бармак

«*08*» *лютого* *2021* року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

1. Тема кваліфікаційної роботи бакалавра: *«Система розпізнавання образів перцептроном із стохастичним алгоритмом навчання»*

2. Завдання видано студенту *Пітику Ярославу Олександровичу*
(прізвище, ім'я, по батькові)

3. Керівник роботи *доцент кафедри КНІТ Мазурець Олександр Вікторович*
(посада, прізвище, ім'я, по батькові)

4. Затверджено наказом університету від «*05*» *лютого* *2021* р. № *11*

5. Зміст пояснювальної записки (перелік задач) та вихідні дані:

Мета роботи – розробка системи розпізнавання образів, яка використовує нейронну мережу перцептрон із стохастичним алгоритмом навчання. В створеній системі розпізнавання образів навчання нейронної мережі слід реалізувати двома алгоритмами: класичним алгоритмом зворотного поширення помилки та стохастичним алгоритмом навчання нейронної мережі. Стохастичний алгоритм навчання нейромережі містить складову випадкового підбору параметрів для кожної ітерації. У якості начальних та тестових зразків системи розпізнавання образів слід використати відбитки пальців.

Виконавець:	*студент 4 курсу, група КН-17-1*		*Я.О. Пітик*
	Курс, група виконавця	Підпис	Ініціали, прізвище
Керівник:	*к.т.н., доцент кафедри КНІТ*		*О.В. Мазурець*
	Науковий ступінь, посада	Підпис	Ініціали, прізвище

Анотація

Тема кваліфікаційної роботи бакалавра: *«Система розпізнавання образів перцептроном із стохастичним алгоритмом навчання»*

Виконавець кваліфікаційної роботи бакалавра: *студент групи КН-17-1 Пітик Ярослав Олександрович*

Керівник кваліфікаційної роботи бакалавра: *к.т.н., доцент кафедри КНІТ Мазурець Олександр Вікторович*

Кваліфікаційна робота бакалавра містить:

Пояснювальна записка				Кількість додатків
Сторінок	Рисунків	Таблиць	Джерел інформації	
55	23	4	33	3

Метою кваліфікаційної роботи бакалавра є розробка системи розпізнавання образів, яка використовує нейронну мережу перцептрон із стохастичним алгоритмом навчання. Для програмної реалізації системи розпізнавання образів на платформі .NET було використовувано середовище програмування Visual Studio і мову програмування C#. У якості начальних і тестових зразків системи розпізнавання образів використано відбитки пальців рук людей, відповідно прикладна інформаційна система позиціонується як система розпізнавання осіб за відбитками пальців їх рук.

В створеній системі розпізнавання образів навчання нейронної мережі реалізовано двома алгоритмами: класичним алгоритмом зворотного поширення помилки та стохастичним алгоритмом навчання нейронної мережі. Стохастичний алгоритм навчання нейромережі містить складову випадкового підбору параметрів для кожної ітерації.

Ключові слова: нейромережа, розпізнавання образів, алгоритм навчання, розпізнавання, відбитки пальців, стохастичний алгоритм навчання.

Виконавець: *студент 4 курсу, група КН-17-1* (Курс, група виконавця) _____ (Підпис) *Я.О. Пітик* (Ініціали, прізвище)

Зміст

Перелік скорочень

Скорочення, термін, позначення	Пояснення
ІС	Інформаційна система
ІТ	Інформаційні технології
КРБ	Кваліфікаційна робота бакалавра
КН	Комп’ютерні науки
КНІТ	Комп’ютерні науки та інформаційні технології
НМ	Нейронна мережа
ПП	Програмний продукт
ХНУ	Хмельницький національний університет
CLR	Common Language Runtime
FCL	Framework Class Library
LSA	Latent Semantic Analysis
MS	Microsoft
MFC	Microsoft Foundation Class

Вступ

В епоху стрімкого розвитку новітніх технологій процесам автоматизації відведено визначне місце, так як вони допомагають не лише у питаннях скорочення часу виконання рутинних операцій, але й сприяють зменшенню помилок, спричинених людським фактором. Процеси автоматизації займають досить широке коло застосувань. І сьогодні вони посідають чільне місце майже у всіх сферах життєдіяльності людини – на технологічних об’єктах, у медицині, соціальній сфері тощо.

Для підтримування левової частки процесів автоматизації необхідна якісна класифікація (або розпізнавання образів), виконувана програмно-математичними засобами – класифікаторами.

Проблема розпізнавання образів на сучасному етапі досить популярна. В загальному розпізнавання образів можна поділити на дві частини: навчання системи розпізнавання образів та розпізнавання образів [1].

Навчання системи відбувається за рахунок показу об’єктів навчання з відміткою про їх приналежність тому чи іншому образу. Результатом такого навчання має бути система розпізнавання, яка здатна однаково реагувати на об’єкти того ж самого образу та по-різному реагувати на об’єкти різних образів.

Процес навчання за розпізнавання образів завершується лише через покази скінченного числа образів, без будь-яких підказок.

Самі об’єкти можуть бути як картинки, або інші зображення (букви), так і різні звуки та явища навколишнього світу.

Другою частиною розпізнавання образів є розпізнавання нових образів. Саме ця частина викликає найбільше питань, оскільки, людина сама придумує образи і далі пробує навчити систему.

В сучасному світі технологій розпізнавання образів набирає популярності. Задачі розпізнавання вирішуються досить часто, наприклад, сигнали світлофора, при переході або переїзді вулиці. Розпізнавання кольору

лампи світлофора, а правила дорожнього руху дозволяють прийняти правильне рішення про перехід вулиці в певний момент.

Для розпізнавання образів серед образів важливими на рівні держави є відбитки пальців. Важливість розпізнавання відбитків пальців полягає в ідентифікації людини в тих чи інших ситуаціях. Наприклад, ідентифікацію по відбитках пальців використовують для розпізнання злочинців, в аеропортах або ж для ідентифікації людини що прийшла на роботу [2]. А власне розпізнавання образів базується на скануванні унікальних виступів на кінчиках пальців людини. Відбитки перевіряються з тими які вже є в системі, й якщо таких немає, то в розпізнавання образів додаються нові образи.

Розділ 1
Характеристика предметної області та постановка задачі

1.1 Аналіз предметної області

Одне з найбільш поширених завдань аналізу даних – розпізнавання образів. Потреба в класифікації або розпізнаванні образів виникає в самих різноманітних технічних, економічних та організаційних системах. До таких завдань зводиться діагностування захворювань у медицині, прийняття рішення про видачу кредиту клієнту банку, розпізнавання образів і звуку і багато інших завдань. На сьогоднішній день створено велику кількість різних алгоритмів класифікації [3], розроблені формалізовані теорії конструювання високоефективних композицій з цих алгоритмів [4]. Основними алгоритмами є наступні [5]: класифікація за допомогою дерев рішень; байєсівська класифікація; класифікація за допомогою штучних нейронних мереж; класифікація методом опорних векторів; статистичні методи, зокрема, лінійна регресія; класифікація CBR-методом; класифікація за допомогою методу найближчого сусіда; класифікація за допомогою генетичних алгоритмів.

З огляду суто історичних причин (термін «pattern recognition», перекладений як «розпізнавання образів», був запозичений з англомовних робіт) цей клас завдань виявився пов'язаний з поняттям «образа» («pattern») [6]. Свого часу не звернули уваги на його багатозначність – термін «pattern», крім значення «образ», має ще і значення «Модель», «стиль», «режим», «закономірність», «образ дій». В сучасному розпізнаванні образів і особливо штучному інтелекті його вживають в найширшому сенсі, маючи на увазі деякий структурно наближений опис (ескіз) досліджуваного об'єкта або явища, причому часткова визначеність опису є важливою властивістю образу. Образ допускає рекурсивне визначення: символ є образом, список символів є образом, образами є тільки ті вираження, які побудовані відповідно до двох зазначених умов. Обліковий запис

дозволяє використовувати одне і те ж уявлення для опису способу довільного типу незалежно від його «змісту». Додатковою властивістю запису цього типу служить можливість користуватися одними і тими ж алгоритмами для роботи з образами.

Природно також допускати, що образ складається з двох груп символів, що представляють відповідно змінні і постійні характеристики об'єкта опису [6].

Основне призначенців описів (образів) при розпізнаванні образів – це їх використання в процесі встановлення відповідності об'єктів, тобто при доказі їх ідентичності, аналогічності, подібності і т. п., здійснюваному шляхом порівняння (зіставлення). Два образи вважаються подібними, якщо вдається встановити їх відповідність.

Можна, зокрема, вважати, що має місце відповідність двох образів, якщо можна досягти їх ідентичності, підставляючи замість змінних будь-які вирази.

Зіставлення образів являє собою основну задачу розпізнавання і грає істотну роль в інформатиці в цілому.

Це завдання розпізнавання образів виникає, зокрема, в різних розділах штучного інтелекту, наприклад в розумінні природного мови, символьної обробки виразів алгебри, експертних системах, перетворенні і синтезі програм ЕОМ. Процедура зіставлення виявилася настільки суттєвою для штучного інтелекту й розпізнавання образів, що в багато мов програмування, які використовуються в штучному інтелекті, вона входить в якості примітиву. Відзначимо, що в різних завданнях поняттю способу надається різний зміст. Так, скажімо, в розпізнаванні (у класичних моделях) образ зазвичай описується вектором ознак, кожен елемент якого являє числове значення одною з ознак, що характеризують відповідний об'єкт. У структурній моделі розпізнавання як образу виступає деякий вислів, що породжується тією граматикою, яка характеризує клас, якому даний образ належить. У завданнях обробки тексту роль образу виконує деяка ланцюжок – в результаті процедура розпізнавання

образів і встановлення відповідностей зводиться до пошуку входжень цього ланцюжка (образа) в тексті.

Термін «розпізнавання» в рівній мірі відноситься як до процесів сприйняття і пізнання, властивим людині і живим організмам в цілому, так і до спроб реалізувати і використовувати «механічні» аналоги (по функції і результату) цих процесів, дослідження і синтез яких складають предмет розпізнавання образів як розділу інформатики.

Отже, метою створення автоматизованих систем розпізнавання є автоматизація групи процесів сприйняття і пізнання, пов'язаних з пошуком, виділенням, ідентифікацією, класифікацією та описом образів на основі аналізу реальних даних, отриманих тим чи іншим способом при розпізнаванні образів. Зазвичай пошук і виділення образів здійснюються на початковому етапі аналізу в процесі обробки вихідних даних і виконуються для того, щоб отримати деякі проміжні результати (перетворити вихідні дані в деяку іншу форму), що «краще» представляють образи з точки зору вирішення відповідного завдання. Наступний етап – розробка «класифікатора» – зазвичай включає аналіз вибіркових (перетворених) даних, синтез моделі, що враховує мінливість образів, що належать до певного класу, вибору із заданого набору характеристик деякої їх підмножини, що адекватно і цілком характеризує окремі класи об'єктів, визначення методів виділення зазначених підмножини і розробку власне алгоритму розпізнавання (Класифікації) [6].

Сам алгоритм розпізнавання образів для коректої ідентифікації образів повинен пройти процес навчання. Розглядають два класи навчальних методів: детерміністський та стохастичний [7].

Детерміністські методи покроково здійснюють процедуру корекції ваг, яка базується на використанні їх поточних значень, значень входів, фактичних виходів і бажаних виходів.

Стохастичні методи навчання при розпізнаванні образів здійснюють псевдовипадкові зміни значень параметрів ШНМ, які спонукають до покращення значень на виході.

Під час навчання системи розпізнавання образів на деякій кількості об’єктів із вказанням даних джерела (наприклад від батьків, старших і т.п.) з якого отримується інформація, встановлюється образ до якого відноситься об’єкт. Такий процес називають "навчання з учителем".

Існує ще навчання у процесі якого система розпізнавання образів навчається спонтанно виконувати поставлене завдання без участі з боку "вчителя", такий процес називається "навчання без учителя" [8].

Найбільш важливими етапами в розпізнаванні образів є:

1. Сприйняття образу.

2. Попередня обробка.

3. Виділення характеристик (індексація образу).

4. Класифікація (прийняття рішення).

Для розпізнавання сформувались основні методи, зокрема: порівняння із зразком, нейронні мережі, статистичні методи і структурні та синтаксичні методи. Однак найвищу ефективність на сьогоднішній день все ж таки мають нейромережеві методи розпізнавання образів. Нейронні мережі використовуються для вирішення складних завдань розпізнавання образів, які вимагають аналітичних обчислень подібних тим, які робить людський мозок.

Станом на початок 2021 Вікіпедія налічувала 26 типів нейромереж [9]. З них 12 називалися за іменами їх винахідників, у решти були такі назви як хаотична, сіамська, осциляторний, адаптивного резонансу і т.п. Для того щоб якось систематизувати вже наявні і майбутні нейромережі, робляться спроби їх класифікації [9]:

1. За типом вхідних даних: аналогові (на вході дійсні числа), виконавчі (на вході двійкові числа) і образні (на вході знаки, ієрогліфи, символи) нейронні мережі.

2. За характером навчання: навчання нейромережі з учителем (вихідний простір рішень нейронної мережі відомо), навчання без вчителя (вихідний простір рішень формується тільки на основі вхідних впливів, а також такі мережі називають самоорганізаційними); навчання нейромережі з підкріпленням (використовується система призначення штрафів і заохочень, що отримуються в результаті взаємодії ІНС з середовищем).

3. За характером налаштування синапсів нейромережі: мережі з фіксованими зв'язками (вагові коефіцієнти нейронної мережі вибираються відразу, виходячи з умов задачі), мережі з динамічними зв'язками (у цих мереж в процесі навчання відбувається настройка синаптичних зв'язків).

4. За часом передачі сигналу нейромережі: синхронні мережі (час передачі для кожного синаптичного зв'язку одиниця або нуль, або фіксований постійної), асинхронні мережі (час передачі для кожного зв'язку між елементами свій, але теж постійний).

5. За характером зв'язків нейромережі: мережі прямого поширення (всі зв'язки направлені строго від вхідних нейронів до вихідних), рекурентні мережі (сигнал з вихідних нейронів або нейронів прихованого шару частково передається назад на входи нейронів вхідного шару), рекуррентна мережа Хопфілда (фільтрує вхідні дані, повертаючись до стійкого стану і, таким чином, дозволяє вирішувати завдання стиснення даних і побудови асоціативної пам'яті), двонаправлені мережі (між шарами існують зв'язки як в напрямку від вхідного шару до вихідного, так і в зворотному).

Крім того, використовуються радіально-базисні нейромережі (або RBF-мережі), карти що самоорганізуються (зокрема, самоорганізована карта Кохонена) і мережі інших класів, котрі ще не цілком сформованих.

Метод розпізнавання образів за допомогою якого людину ідентифікують за відбитками пальців називається дактилоскопія. Він заснований на унікальності рисунка шкіри людини.

Ілюстрації відпечатків пальців в електронно-цифровому форматі формуються в пам'яті комп'ютера в результаті сканування поліцейських дактилокарт за допомогою планшетного сканера пальців з живим сканером, фотографування цифровим фотоапаратом слідів пальців з фотоплівки або іншого твердого носія [10]. Такі дактилоскопічні зображення за розпізнавання образів зазвичай складаються з темних кольорів, що відображається зображенням папілярних ліній. На рисунку 1.1 (ліворуч) лінії дактилоскопічного зображення контрастно відділені один від одного просвітами. Майже паралельні між собою, вони утворюють узори (рисунок 1.1 (праворуч)) – загальні ознаки, рисунки у вигляді петель, дельт і завитків.

Рисунок 1.1 – Вихідне зображення та зображення у вигляді петель і дельт

Лінії закінчуються або починаються, з'єднуються або розщеплюються, утворюючи особові ознаки: закінчення і початку, з'єднання і розгалуження ліній. Такий поділ ознак за розпізнавання образів умовний. Наприклад, можна вказати закінчення і розгалуження, фрагмент, острівець, точку, примикання, місток, гачок, дельту, перетин, потроєння, переривання лінії і інше [10]. Фактично це складові ознаки, які можуть бути утворені комбінацією закінчень і розгалужень. Щоб зменшити помилки розпізнавання образів, зображення попередньо обробляють і отримують його стилізоване уявлення – скелет, а по скелету

детектують два типи часткових ознак: закінчення і розгалуження [10]. Обробка допомагає відновити зображення, звільнивши його від шумів. Далі можна переходити до етапу навчання та розпізнавання. Для вирішення індивідуального завдання було обрано нейронну мережу типу перцептрон із стохастичним методом навчання [9]. Даний метод розпізнавання запропонував Ф. Розенблатт.

Оскільки задачі розпізнавання, як одним із найефективніших засобів вирішуються за допомогою штучного інтелекту, то їх використання буде досить доречним для вирішення поставленої задачі.

1.2 Аналіз існуючого програмного забезпечення предметної області

1.2.1 Системи розпізнавання відбитків пальців

Ідентифікація за відбитками пальців складне, але водночас важливе завданням. Серед багатьох методів та підходів можемо виділити декілька, які найбільш часто застосовуються [11]:

– кореляційне порівняння. Даний підхід полягає в попіксельному порівнянні двох зображень, для різних зрушень і кутів повороту, на основі одержаних результатів виносять рішення про збіг. (В сучасних умовах не застосовується, через високу трудомісткості);

– порівняння по особливих точках. Особливі точки – це кінцеві точки і точки розгалуження. Ці точки виділяються на обох зображення, а далі методом їх кореляційного порівняння, виноситься вердикт про відповідність відбитків. З причини своєї відносно простої реалізації і швидкості роботи, дані алгоритми більш поширені;

– порівняння по візерунку залежно від необхідної точності, зображення відбитка розбивається на області. Далі візерунок в кожній з областей описується синусоїдальної хвилею, з параметрами: початковий зсув фази, довжина хвилі,

напрямок поширення. Даний клас алгоритмів, не вимагає високої роздільної здатності при скануванні;

– зіставлення по шаблону;

– порівняння на основі графів.

Розглянемо приклад додатку, що реалізовує задачу розпізнавання відбитків пальців [5]. Додаток за допомогою різних алгоритмів знаходить основні особливості людини, такі як: відбиток, орієнтоване зображення та образ (рисунок 1.2).

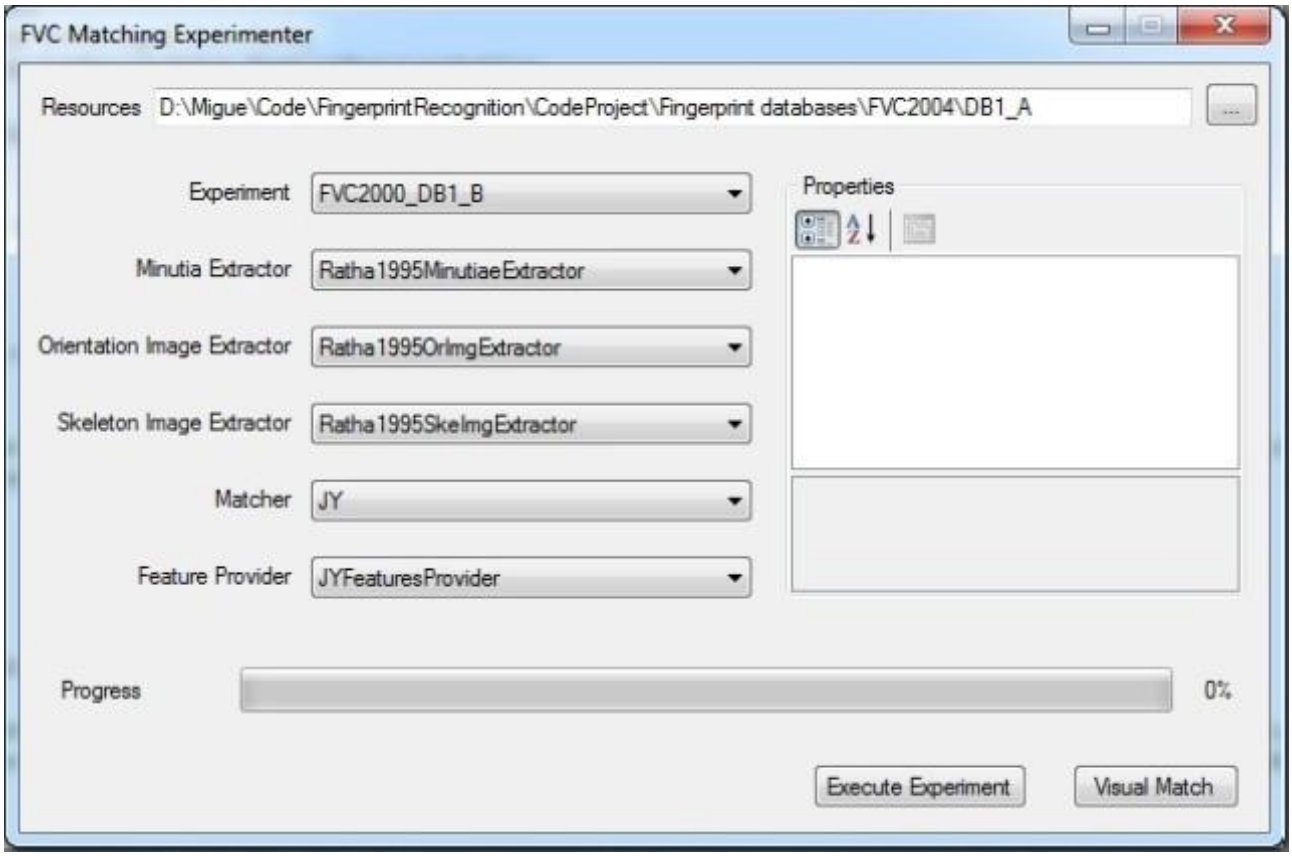


Рисунок 1.2 – Вибір алгоритму [5]

Після вибору алгоритму та інших даних, можна порівняти два відбитка і перевірити їхнє співпадіння (рисунок 1.3). Також, в програмі можна вивести на екран відбиток, тим самим візуалізувати його (рисунок 1.4).

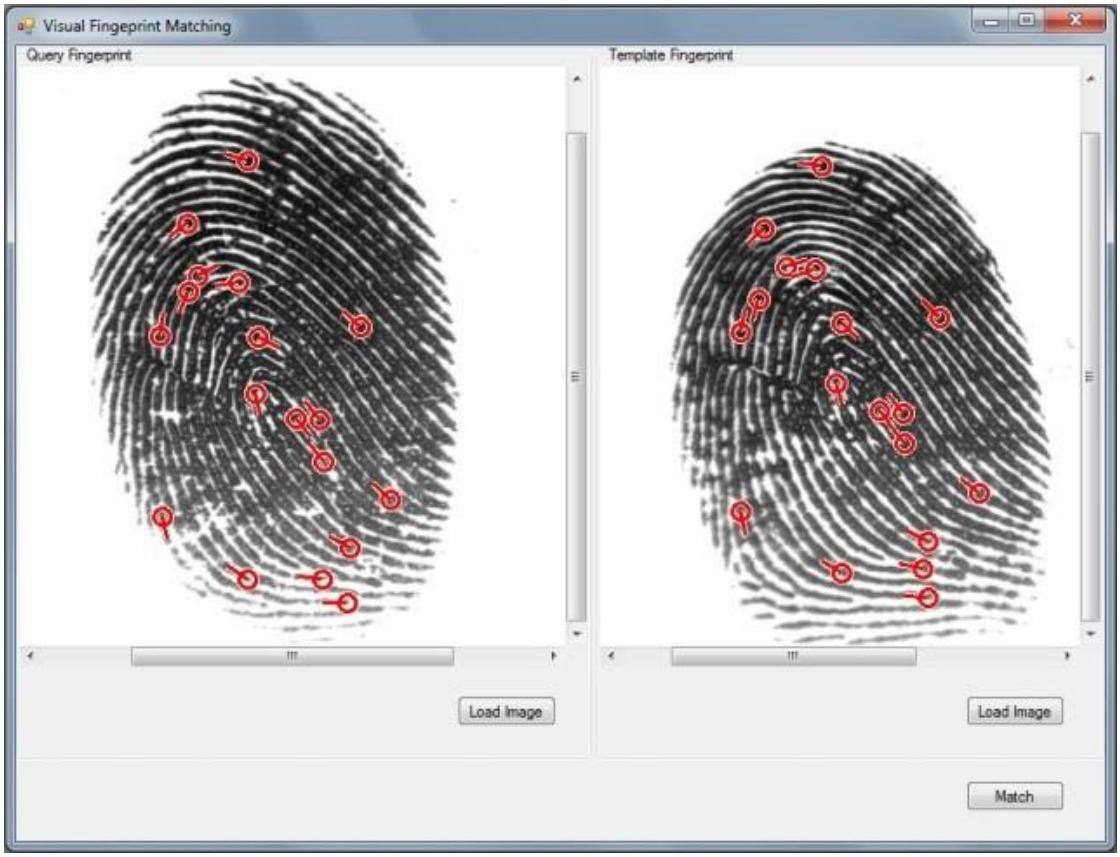


Рисунок 1.3 – Порівняння відбитків [5]

Рисунок 1.4 – Вивід відбитка на екран [5]

Провівши дослідження, виявлено що наведені приклади демонструють ефективні способи спрощення та пришвидшення виконання завдання

розпізнавання відбитків пальців, але для ефективного вирішення перспективно використовувати нейронні мережі.

1.2.2 Використання нейронних мереж у задачах розпізнавання образів

Штучні нейронні мережі з прямим зв'язком є універсальним засобом апроксимації функцій, і це дозволяє їх використовувати у вирішенні задач класифікації [12]. Як правило, нейромережі виявляються найбільш ефективним способом класифікації, адже генерують фактично велике число регресійних моделей (які використовуються у вирішенні завдань класифікації статистичними методами).

Штучною нейронною мережею є математична модель, програмна, або апаратна реалізація, побудована за принципом функціонування й водночас організації біологічних нейронних мереж [13]. Існує множина топологій нейронних мереж, як найпростіших (персептрон Розенблатта), так і складних рекурентних мереж. Серед різних структур нейронних мереж багатошарова структура однією є з найбільш відомих, в якій кожен нейрон довільного шару пов'язаний з усіма аксонами нейронів попереднього шару або, у разі першого шару, з усіма входами ІНС. Нейронні мережі власне не програмуються у звичному розумінні цього слова, втім вони навчаються. Можливість навчання – одна із головних переваг нейронних мереж над традиційними алгоритмами. Технічно навчання полягає у знаходженні коефіцієнтів зв'язків між нейронами.

При виборі архітектури мережі звичайно випробовується ряд конфігурацій з різною кількістю елементів. При цьому основним показником є об’єм навчальної множини і узагальнююча здатність цієї мережі. Найбільш оптимальним алгоритмом навчання вважається Back Propagation (зворотного поширення) із підтверджуючою множиною [14].

У задачах розпізнавання образів використовується багато видів нейронних мереж.

Хемінга / Хопфілда. Мережа Хеммінга має ряд переваг над мережею Хопфілда. Вона реалізує оптимальний класифікатор мінімуму похибки, якщо похибки вхідних бітів є випадковими, незалежними. Для функціонування мережі Хеммінга потрібно менше нейронів, оскільки середній прошарок вимагає один нейрон на клас, замість нейрону на кожен вхідний вузол. Та врешті мережа Хеммінга зазвичай не страждає від неправильних класифікацій, які можуть бути в мережі Хопфілда. В цілому, мережа Хеммінга є і швидшою, і більш точною аніж мережа Хопфілда.

Проте, недоліком є те, що ємність скерованої асоціативної пам'яті жорстко обмежена. Якщо n – кількість нейронів у вхідному прошарку, то число векторів, що можуть бути запам'ятовані в мережі не перевищує n=1024, тобто мережа здатна запам'ятати не більш 25 образів, і кожний з них повинен бути відновлюваним. Двоскерована асоціативна пам'ять має деяку непередбачуваність у процесі функціонування, тому можливі помилкові відповіді [15].

Когнітрон – штучна нейронна мережа на основі самоорганізації. Своєю архітектурою когнітрон досить схожий на будову зорової кори, має ієрархічну багатошарову організацію, в якій нейрони між шарами зв'язані тільки локально. Навчається когнітрон конкурентним навчанням без вчителя. У живих істот кожен шар мозку реалізує різні рівні узагальнення; так, вхідний шар чутливий до простих образів, таких, як лінії, і їх орієнтація в певних областях візуальної області. А реакція інших шарів є більш складною, абстрактною і незалежною від позиції образу. Аналогічні функції реалізовані в когнітроні, зокрема шляхом моделювання організації зорової кори [16]. Однак недоліком когнітрону є його нездатність розпізнавати зміщені і повернуті образи, а також досить висока складність [17].

Неокогнітрон є подальшим розвитком вихідної ідеї когнітрону. Він більш точно відображає будову біологічної зорової системи, дозволяє розпізнавати образи незалежно від їх обертань, перетворень, перекручувань, змін масштабу тощо. Він може як навчатись самостійно, так і навчатись із учителем.

Неокогнітрон отримує на вхід двовимірні образи, які аналогічні зображенням на сітківці ока, обробляє їх в наступних шарах аналогічно тому, як було це виявлено в біологічній зоровій корі. Втім, в неокогнітроні немає нічого, що обмежує його використання тільки для обробки графічних даних, неокогнітрон досить універсальний й може знайти широке застосування як узагальнена система розпізнавання образів [18].

Проте неокогнітрон також має ряд визначених недоліків: низька швидкість навчання, жосткість параметрів вхідних образів.

Іншою із числа популярних нейронних мереж для вирішення завдання розпізнавання образів є перцептрон. Це одна з найбільш часто використовуваних архітектур нейронних мереж (багатошаровий перцептрон). Багатошаровий перцептрон складається з декількох шарів, імпульс по яким поширюється послідовно від шару до шару. Шар, на який надходить вхідна інформація, називається вхідним шаром.

Вхідний шар називається шаром умовно, оскільки це формальне позначення, й завдання вхідного шару розподіляти дані на входи нейронів схованого шару. Й при цьому вхідний шар ніяким чином не впливає на дані. Шар, з якого знімається результат – вихідний шар. А всі проміжні шари називаються схованими шарами. Кількість нейронів у вхідному шарі визначається розмірністю даних, які багатошаровий перцептрон приймає на вхід (дорівнює числу вхідних змінних). Число нейронів в вихідному шарі, в свою чергу, задається наявною розмірністю даних, які багатошаровий перцептрон видає на виході (дорівнює числу вихідних змінних).

Переваги багатошарового перцептрону очевидні – незважаючи на свою простоту, у ряді задач багатошаровий перцептрон показує вищу ефективність ніж аналогові нейронні мережі [19]. Відтак, для вирішення завдання КРБ будемо використовувати перцептрон.

1.2.3 Алгоритм навчання нейронної мережі перцептрон

Перед використанням нейронної мережі багатошаровий перцептрон, вона повинна *навчитися*. Персептрон – нейронна мережа, що проходить навчання з учителем. Навчання багатошарового перцептрону відбувається алгоритмом зворотного поширення помилки (*Back-propagation algorithm*) [20].

Суть методу зворотного поширення помилки полягає у тому, що помилка поширюється від вихідного до вхідного шарів багатошарового перцептрону, корегуючи при цьому ваги синапсів, в сторону зменшення помилки.

Навчається мережа багатошаровий перцептрон на так званих «ідеальних» образах. При чому навчання образам багатошарового перцептрону відбувається не по черзі, а всім одночасно. (рисунок 1.5). Тобто корекція ваг персептрону відбувається власне після подання кожного «ідеального» образу. І одна ітерація – це підлаштування ваг для усіх вхідних образів.

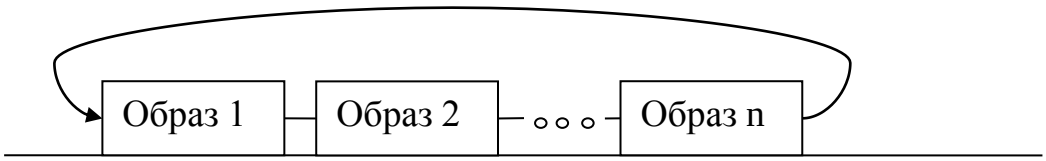


Рисунок 1.5 – Процес навчання образів багатошарового перцептрону

Процес навчання багатошарового перцептрону продовжується до тих пір, поки помилка не буде у межах допустимої (межі визначає сам користувач, в залежності від розв’язуваної задачі). Проте, не рекомендовано допустиму помилку брати більше 15 % (0,15) [20].

Обрахунок помилки та корегування ваг між вихідним та схованим шарами. Зв'язок нейронів багатошарового перцептрону між схованим та вихідним шарами зображено на рисунку 1.6.

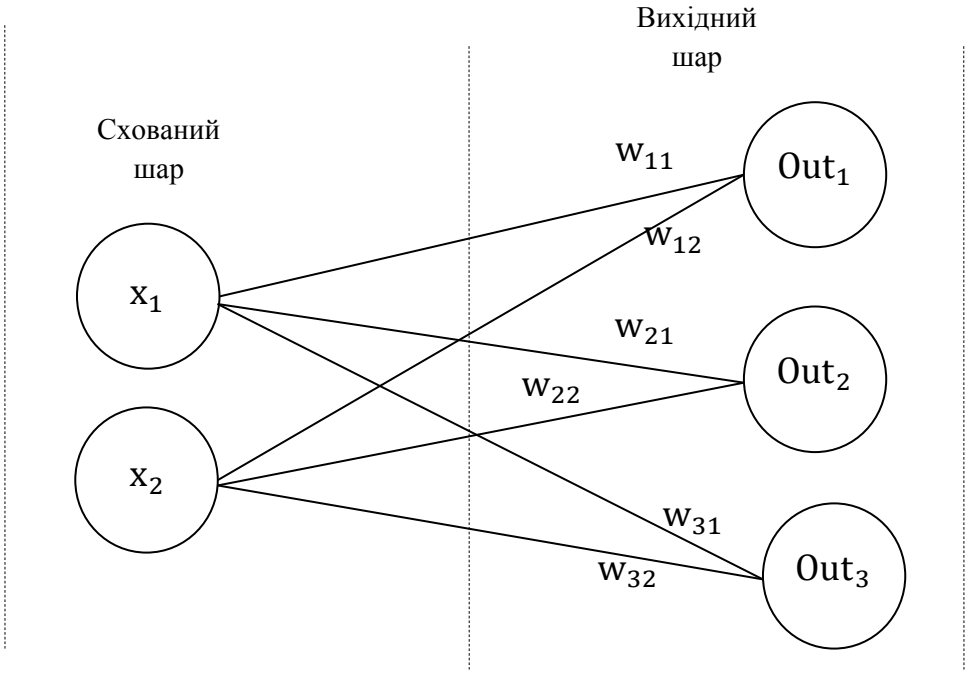


Рисунок 1.6 – Зв'язок нейронів багатошарового перцептрону між схованим та вихідним шарами

Помилка обраховується за формулою:

$$\Delta_i^{(0)} = (Out_{o_i} - Out_i) \cdot Out_i \cdot (1 - Out_i) \quad (1.1)$$

де $\Delta_i^{(0)}$ – помилка і-го нейрону вихідного шару багатошарового перцептрону, Out_{o_i} – очікуваний вихід і-го нейрону вихідного шару.

Корекція ваг:

$$w_{ij}^1 = w_{ji} + \Delta_i^{(0)} \cdot S \cdot x_j \quad (1.2)$$

де S – швидкість навчання нейронної мережі багатошаровий перцептрон (зазвичай – 0.1, але в залежності від задачі, можна підібрати більш оптимальну швидкість); w_{ij}^1 – скореговані ваги між схованим та вихідним шарами; x_j – j-й нейрон схованого шару.

Обрахунок помилки та корегування ваг між вхідним та схованим шарами. Зв'язок нейронів багатошарового перцептрону між вхідним та схованим шарами зображено на рисунку 1.7.

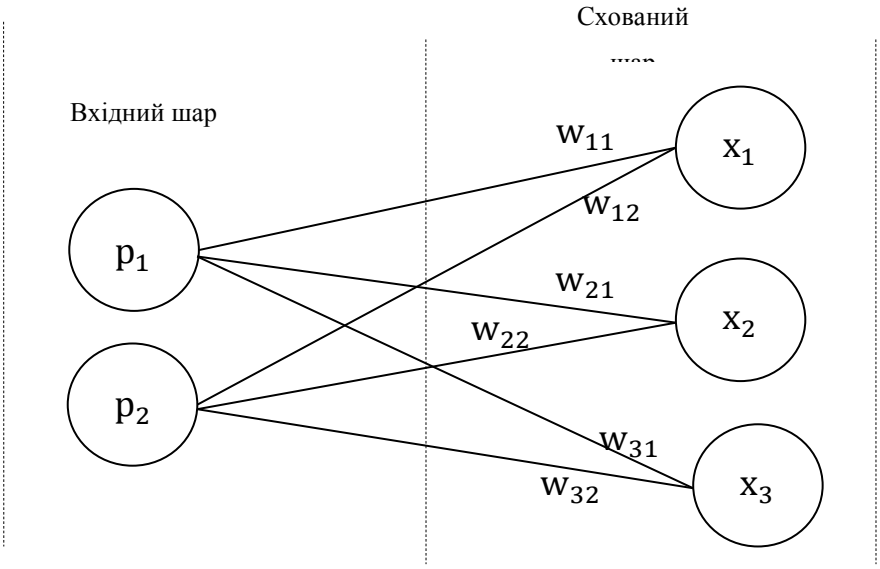


Рисунок 1.7 – Зв'язок нейронів між вхідним та схованим шарами

Помилка між вхідним та схованим шарами:

$$\Delta_{sum} = \sum_{i=1}^{3} w_{ij}^{1} \cdot \Delta_{j}^{(0)} \tag{1.3}$$

де Δ_{sum} – розраховується для знаходження $\Delta_{i}^{(1)}$; $\Delta_{j}^{(0)}$ – помилка j-го нейрону вихідного шару багатошарового перцептрону; w_{ij}^{1} – відкореговані ваги синапсів між схованим та вихідним шарами;

$$\Delta_{i}^{(1)} = x_i \cdot (1 - x_i) \cdot \Delta_{sum} \tag{1.4}$$

де $\Delta_{i}^{(1)}$ – помилка і-го нейрону схованого шару багатошарового перцептрону; x_i – вхід і-го нейрону схованого шару.

Корекція ваг між схованим та вхідним шарами:

$$w_{ij}^{2} = w_{ij} + \Delta_{i}^{(1)} \cdot S \cdot p_j \tag{1.5}$$

Помилка (*total error*), яка говорить про те, що навчання багатошарового перцептрону можна вважати завершеним обраховується по-різному (в залежності від розв’язуваної задачі).

Можна дивитись, щоб значення у рамках кожного нейрону багатошарового перцептрону не виходили за межі допустимого (ε) [20]:

$$err_i = Out_{o_i} - Out_i, \text{ де } err_i \leq \varepsilon \tag{1.6}$$

де Out_{o_i} – очікуваний вихід і-го нейрону багатошарового перцептрону вихідного шару.

Інший варіант – щоб сумарна середньоарифметична помилка не виходила за допустимі межі (ε)

$$Err_{sum} = (\sum_{i=1}^{n} err_i)/n\ , \tag{1.7}$$

$$Err_{sum} \leq \varepsilon \tag{1.8}$$

Вищерозглянутий алгоритм навчання багатошарового перцептрону та роботи мережі широко застосовується у розв'язанні задач розпізнавання образів та кластеризації [20]. У результаті роботи навченого багатошарового перцептрону, по вхідному тестовому зразку буде визначено, до якого з класів і з якою імовірністю належить тестовий зразок.

Незважаючи на велику кількість програмного забезпечення, робота у даному напрямку є доцільною. А саме у напрямі вибору більш ефективного способу навчання багатошарового перцептрону стохастичним методом для розв'язання задачі дактилоскопічного розпізнавання образів.

1.3 Аналіз сучасних засобів створення програмного забезпечення

Для створення системи розпізнавання образів, що використовує нейронну мережу перцептрон, було розглянуто варіанти веб-застосування, віконного десктоп-застосування і застосунку для мобільних пристроїв.

Веб-застосування використовується загалом для систем, в яких потрібне постійне інтернет-з'єднання. Для взаємодії з іншими користувачами, для доступу до бази даних. Основною перевагою даного типу застосування є те, що його не потрібно встановлювати, для його роботи необхідно лише мати веб-браузер [22]. Основним недоліком веб-застосування є постійна необхідність в інтернет-з'єднанні та вищої вірогідності хакерської атаки.

Застосування для мобільних пристроїв. загалом, дуже схоже на віконне (десктоп застосування), проте для нашої задачі воно недоречне.

Отож, було прийнято рішення зупинитись на десктоп-застосунку, оскільки даний тип застосунку дає більше можливостей користувачу та більшу гарантію безпеки.

Стосовно платформи, то вона обиралося з трьох найбільш популярних для створення таких застосувань: Java [22,23], PHP [24], .NET [25].

Програмна платформа Java – ряд програмних продуктів і специфікацій компанії Sun Microsystems, раніше незалежної компанії, а зараз дочірньої компанії корпорації Oracle, які спільно надають систему для розробки прикладного системного програмного забезпечення та вбудовування її в будь-яке крос-платформенне програмне забезпечення. Для програмного забезпечення Java використовується в самих різних комп'ютерних платформах від вбудованих пристроїв і мобільних телефонів в нижньому ціновому сегменті, до корпоративних серверів і суперкомп'ютерів у вищому ціновому сегменті [22]. Однак, має ряд недоліків: платне комерційне використання, низька продуктивність (наприклад, додаток очищення пам'яті – корисна функція, яка, Втім, призводить до значних проблем з продуктивністю, якщо вимагає більше 20 відсотків часу процесора), відсутність нативного дизайну, багатослівний і складний код [23].

PHP (англ. PHP: Hypertext Preprocessor – PHP: гіпертекстовій препроцесор) – мова скриптового програмування програмного забезпечення, що створена для генерації HTML-сторінок на стороні вебсервера. PHP є однією з найрозповсюдженыших мов створення програмного забезпечення, що використовують у сфері веброзробок (разом із Java, .NET, JavaScript, Python, Ruby).

PHP інтерпретується вебсервером у HTML-код. Цей код передається на сторону клієнта. PHP підтримується переважною більшістю хостинг-провайдерів [24]. На відміну від мови JavaScript, користувач програмного забезпечення не бачить PHP-коду, тому что браузер отримує готовий html-код. Це є перевага з точки зору безпеки програмного забезпечення, проте погіршує інтерактівність

сторінок. Проте оскільки зупинили свій вибір на розробці десктопної версії програмного забезпечення, нам PHP не підходить у якості вибраної платформи.

Отже, наш вибір платформи для програмного забезпечення зупиниться на *.NET*. Платформа .NET Framework складається з загальномовного середовища виконання (середовища CLR) і бібліотеки класів .NET Framework. Основою платформи .NET Framework є середовище CLR. Середовище виконання програмного забезпечення можна вважати агентом, який керує кодом під час виконання і надає основні служби, такі як керування пам'яттю, потоками програмного забезпечення і віддалену взаємодію. При цьому середовищем накладаються умови суворої типізації та інші види перевірки точності коду програмного забезпечення, що забезпечують безпеку і надійність. Код, який звертається до середовища виконання, називають керованим кодом, а код програмного забезпечення, який не обертається до середовища виконання, називають некерованим кодом. Фактично основним завданням середовища виконання є керування програмним кодом. Бібліотека класів є комплексною об'єктно-орієнтованої колекцією повторно використовуваних типів, які застосовуються для розробки програмного забезпечення – починаючи зі звичайних додатків, що запускаються з командного рядка, і до програмного забезпечення з графічним інтерфейсом (GUI) і закінчуючи додатками, що використовують останні технологічні можливості ASP.NET, такі як вебформи і вебслужби XML.

Платформа .NET Framework може розміщуватися некерованими компонентами, які завантажують середовище CLR у власні процеси і запускають виконання керованого коду програмного забезпечення, створюючи таким чином програмне середовище, що дозволяє використовувати можливості як керованого, так і некерованого виконання. Платформа .NET Framework не тільки надає ряд базових середовищ виконання, але також підтримує розробку базових середовищ виконання незалежними виробниками [25].

Наприклад, ASP.NET розміщує середовище виконання і забезпечує масштабовану середовище для керованого коду програмного забезпечення на стороні сервера. ASP.NET працює безпосередньо з середовищем виконання, щоб забезпечити виконання програмного забезпечення ASP.NET і веб-служб XML.

Оглядач Internet Explorer може служити прикладом некерованого програмного забезпечення, який розміщує середовище виконання (у вигляді розширень типів MIME). Розміщення середовища виконання в браузері Internet Explorer дозволяє впроваджувати керовані компоненти або елементи управління Windows Forms в HTML-документи. Таке розміщення середовища дозволяє виконувати керований мобільний код програмного забезпечення і користуватися його істотні переваги, зокрема виконанням в умовах неповного довіри і ізольованим зберіганням файлів.

Використовуючи дану платформу враховуючи нашу задачу можна створити програмне забезпечення на одній із мов: Visual Basic, F#, C# [26]. Отже, будемо створювати десктопний застосунок на платформі .NET.

1.4 Постановка задачі та вимоги до розробки інформаційної системи

Метою кваліфікаційної роботи бакалавра є розробка системи розпізнавання образів на платформі .NET, що використовує нейронну мережу перцептрон. Система має виконувати наступні функції:

– підготовка до навчання, що включає в себе завантаження зображень для навчання, їх бінаризацію, формування рецепторної матриці входів нейромережі, визначення приналежності зображень образам та формування стартової множини ваг синапсів;

– навчання нейронної мережі, в результаті чого одержуються проміжні дані, а саме множина ваг синапсів, що потрібні для розпізнавання;

– підготовка до розпізнавання, що полягає у завантаженні зображення для розпізнавання, його бінаризації та формуванні рецепторної матриці входів нейромережі;

– розпізнавання зображення з використанням одержаної в результаті навчання множини ваг синапсів;

– формування результату розпізнавання у вигляді цифрової оцінки приналежності зразка образам.

У якості начальних та тестових зразків системи розпізнавання образів слід використати відбитки пальців людей, таким чином прикладна інформаційна система позиціонується як така, що призначена для розпізнавання осіб за їх відбитками пальців.

Навчання нейронної мережі перцептрон, результатом якого є сформована множина ваг синапсів для розпізнавання, слід реалізувати двома наступними алгоритмами:

– класичний алгоритм навчання нейронної мережі перцептрон зворотного поширення помилки;

– стохастичний алгоритм навчання нейронної мережі перцептрон, що містить складову випадкового підбору параметрів для кожної ітерації.

Реалізація обох алгоритмів навчання перцептрона у системі розпізнавання образів дозволить у подальшому виконати порівняння їх ефективності.

Розділ 2
Проєктування інформаційної системи

2.1 Моделі, методи, інформаційна технологія системи

Розроблювана система розпізнавання образів використовує нейронну мережу перцептрон, для реалізації навчання якої передбачається реалізація двох алгоритмів розпізнавання:

– детерміністського, класичного – алгоритму зворотного поширення помилки;

– стохастичного алгоритму навчання – що має містити складову випадкового підбору параметрів.

Класичний алгоритм зворотного поширення помилки використовується для реалізації навчання нейронної мережі перцептрон та потребує вхідні дані у вигляді множини зображення для навчання та вказанням приналежності зображень образам. Спершу відбувається формування стартової множини ваг багатошарового перцептрону, після чого виконується пряме поширення сигналу. Згідно отриманих результатів на виході мережі, проводиться обрахунок помилки кожного нейрона багатошарового перцептрону, за якими проводиться обрахунок сукупної помилки нейронів, зазвичай у вигляді відхилення одержаного результату успішно розпізнаних образів від ідеального (рисунок 2.1).

При навчанні багатошарового перцептрону спершу перевіряється умова, чи навчання завершено; воно вважається завершеним, якщо відхилення одержаного результату успішно розпізнаних образів від ідеального виявляється менше певного порогового значення. У випадку, якщо навчання багатошарового перцептрону завершено, то формуються вихідні дані, а саме множина ваг синапсів. Якщо ж навчання багатошарового перцептрону не завершено, то виконується коригування всіх ваг синапсів у напрямку зменшення помилки згідно алгоритму зворотного поширення помилки.

Рисунок 2.1 – Класичний алгоритм зворотного поширення помилки для реалізації навчання нейронної мережі перцептрон

Після коригування всіх ваг синапсів багатошарового перцептрону у напрямку зменшення помилки згідно алгоритму зворотного поширення помилки, повторюється пряме поширення сигналу й обрахунок помилки кожного нейрона, за якими проводиться обрахунок сукупної помилки нейронів,

зазвичай у вигляді відхилення одержаного результату успішно розпізнаних образів від ідеального. Після чого знову перевіряється умова, чи навчання багатошарового перцептрону завершено.

На відміну від класичного алгоритму зворотного поширення помилки, при реалізації навчання нейронної мережі перцептрон за стохастичним алгоритмом на кожній ітерації змінюються не всі ваги синапсів. За стохастичного алгоритму навчання багатошарового перцептрону, який має містити складову випадкового підбору параметрів потрібних вхідних даних у вигляді ряду зображень для навчання та вказання приналежності зображень образам. Спершу відбувається формування стартової множини ваг, після чого виконується пряме поширення сигналу. Далі, згідно отриманих результатів на виході мережі, проводиться обрахунок помилки кожного нейрона, за якими проводиться обрахунок сукупної помилки нейронів, як і в попередньому випадку у вигляді відхилення одержаного результату успішно розпізнаних образів від ідеального (рисунок 2.2).

При навчанні спершу перевіряється умова, чи навчання завершено. Навчання багатошарового перцептрону вважається завершеним, якщо відхилення одержаного результату успішно розпізнаних образів від ідеального виявляється менше певного порогового значення. У випадку, якщо навчання завершено, то формуються вихідні дані – множина ваг синапсів. Якщо ж навчання не завершено, то виконується випадковий вибір одної чи кількох ваг для модифікації згідно стохастичного алгоритму навчання, що містить складову випадкового підбору параметрів. Далі виконується випадкова зміна значення обраних одної чи кількох ваг синапсів. Після коригування цих ваг синапсів, згідно стохастичного алгоритму навчання, повторюється пряме поширення сигналу й обрахунок помилки кожного нейрона багатошарового перцептрону, за якими проводиться обрахунок сукупної помилки нейронів у вигляді відхилення

одержаного результату успішно розпізнаних образів від ідеального. Після чого знов перевіряється умова, чи навчання завершено.

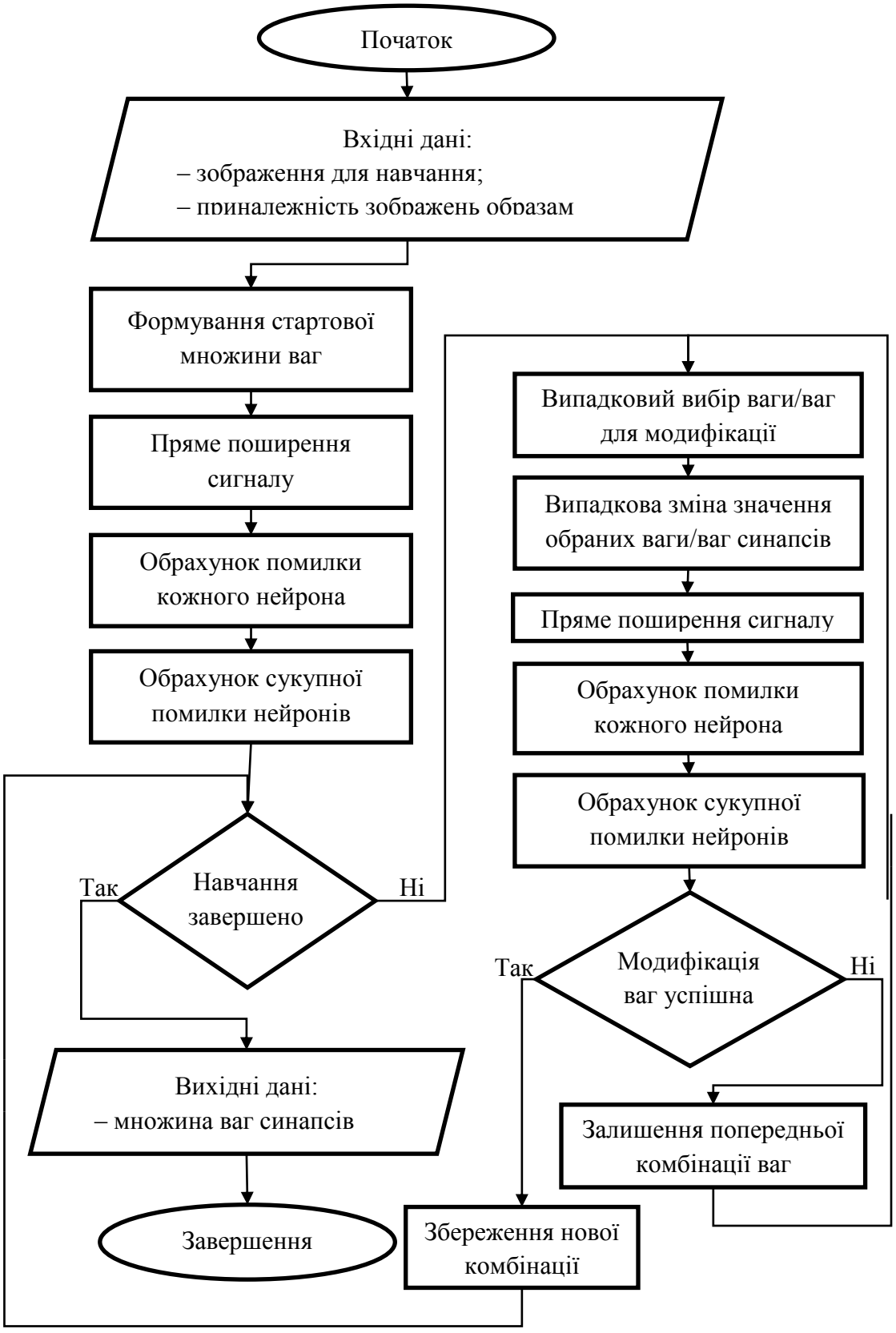


Рисунок 2.2 – Стохастичний алгоритм навчання багатошарового перцептрона

Реалізація обох алгоритмів навчання багатошарового перцептрона у системі розпізнавання образів дозволить виконати порівняння їх ефективності. Безпосередньо інформаційна технологія розпізнавання образів з використанням стохастичного алгоритму навчання розглянута в наступному підрозділі.

2.2 Інформаційна структура системи

Інформаційна технологія розпізнавання образів з використанням стохастичного алгоритму навчання передбачає вхідні дані у вигляді ряду зображень для навчання, вказівок щодо приналежності зображень образам, а також необхідний тестовий зразок для розпізнавання (рисунок 2.3).

На Кроці 1 виконується підготовка до навчання багатошарового перцептрону, що включає в себе завантаження зображень для навчання, визначення приналежності зображень образам та формування стартової множини ваг синапсів.

На Кроці 2 виконується навчання нейронної мережі з використанням стохастичного алгоритму навчання. В результаті одержуються проміжні дані, а саме множина ваг синапсів багатошарового перцептрону, що потрібні для розпізнавання.

Крок 3 складає підготовка до розпізнавання, що полягає у завантаженні зображення для розпізнавання. Завантажене зображення (тестовий зразок) на Кроці 4 розпізнається з використанням множини ваг синапсів.

Після чого на Кроці 5 виконується формування результату розпізнавання у вигляді цифрової оцінки приналежності зразка образам. Це визначає вихідні дані, які складають найбільш імовірний образ до зразка та цифрові оцінки приналежності зразка кожному з образів.

Рисунок 2.3 – Інформаційна технологія розпізнавання образів з використанням стохастичного алгоритму навчання

2.3 Вибір засобів розробки інформаційної системи

2.3.1 Вибір мови програмування

Враховуючи обрану платформу, обираємо мову програмування серед Visual Basic, F#, C# [26]. Розглянемо їх більш детально.

Visual Basic .NET (VB.NET) – об'єктно-орієнтована мова програмування, яку слід розглядати як розвиток Microsoft Visual Basic на платформі .NET. У VB.NET були зроблені суттєві зміни, й вони привели до несумісності з версіями Visual Basic, що робить автоматичне перенесення програмного забезпечення з VB 6.0 в VB.NET принаймні дуже проблематичним. Відкриття проєктів програмного забезпечення старих версій (* .vbp) можливо тільки після попереднього перетворення їх в формат VB.NET спеціальним майстром (Migration Wizard). Крім того, і це найістотніше, змінилася логіка і концепція розробки програмного забезпечення [27].

Але враховуючи, що мова Visual Basic більше не буде отримувати нові функції, його свіжі версії перестануть виходити. Його підтримка буде збережена в .NET 5.0, подальша підтримка не гарантується. На думку експертів з програмного забезпечення, Visual Basic програв конкуренцію мови C # [28].

F# – це функціонально -орієнтована, універсальна, строго-типізована, мова програмування, яка включає в себе функціональні, імперативні і об'єктно-орієнтовані методи програмування програмного забезпечення. F# найчастіше використовується в якості кроссплатформенної мови Common Language Infrastructure (CLI) в .NET, але також може генерувати код JavaScript і графічного процесора (GPU). Мова F# розроблена F# Software Foundation, Microsoft і відкритими учасниками. З відкритим вихідним кодом, крос-платформний компілятор для F# можна отримати в F# Software Foundation. F# – це повністю підтримувана мова створення програмного забезпечення в Visual Studio і JetBrains Rider. Модулі, що підтримують F#, існують для багатьох широко використовуваних редакторів, в першу чергу для розширення Ionide для Visual Studio Code і інтеграції для інших редакторів, таких як Vim і Emacs. F# є членом сімейства мов створення програмного забезпечення ML і виник як реалізація .NET Framework ядра мови програмування OCaml [29].

В цілому мови створення програмного забезпечення F# та C# універсальні. Наприклад, незважаючи на те що працювати з призначеним для

користувача інтерфейсом краще за допомогою об'єктів, для цього завдання цілком підійде і F#, тому що в ньому є підтримка ООП.

Однак все ж є завдання, коли варто віддати перевагу тій чи іншій мові:

F# краще використовувати для написання бізнес-логіки програмного забезпечення, яка вимагає серйозних обчислень (в тому числі і для роботи з Big Data).

C# більше підходить для розробки інтерфейсів програмного забезпечення, роботи з графікою або ж написання логіки, в якій потрібно працювати з об'єктами. Наприклад, функціональне програмування навряд чи підійде для створення гри, тому що в ній багато сутностей (персонажі, предмети, декорації). Саме тому серед розробників програмного забезпечення ігор більш популярний C# [30].

C# – сучасна об'єктно-орієнтована безпечна по типам мова програмування. C# дозволяє розробникам створювати безліч типів безпечного і надійного програмного забезпечення, які потребують екосистему .NET. C# належить до відомого сімейства мов C, і вона здасться добре знайомою кожному, хто працював з C, C ++, Java або JavaScript [31].

C# – це об'єктно- і компонентно-орієнтована мова програмування програмного забезпечення. C# надає мовні конструкції для безпосередньої підтримки такої концепції роботи. Завдяки цьому, C# підходить для створення та застосування програмного забезпечення. З моменту створення мова C # збагатилась функціями для підтримки нових робочих навантажень і сучасними рекомендаціями по розробці програм і програмного забезпечення.

Ось лише кілька функцій мови C#, які дозволяють створювати надійні та стійкі додатки. *Прибирання сміття* – автоматично звільняє пам'ять, зайняту недоступними невикористовуваними об'єктами. Типи, що допускають значення null, забезпечують захист від змінних, які не посилаються на виділені об'єкти. *Обробка винятків* надає структурований і розширюваний підхід до виявлення помилок та відновлення після них. *Лямбда-вирази* підтримують прийоми функціонального програмування. *Синтаксис LINQ* створює загальний шаблон

програмного забезпечення для роботи з даними з будь-якого джерела [31]. *Підтримка мов для асинхронних операцій* надає синтаксис для створення розподілених систем. У C # діє *єдина система типів.* Всі типи C#, включаючи типи-примітиви, такі як int і double, успадковують від одного кореневого типу object. Всі типи використовують загальний набір операцій, а значення будь-якого типу можна зберігати, передавати і обробляти схожим чином. Більш того, C# підтримує як визначаються користувачами посилальні типи, так і типи значень. C# дозволяє динамічно виділяти об'єкти та зберігати спрощені структури в стеці. C# підтримує універсальні методи і типи, що забезпечують підвищену безпеку типів і продуктивність програмного забезпечення. C# надає ітератори, які дозволяють розробникам класів колекцій визначати призначені для користувача варіанти поведінки для клієнтського коду.

У C# особлива увага приділяється керуванню версіями для забезпечення сумісності програмного забезпечення і бібліотек при їх зміні. Питання керування версіями істотно вплинули на такі аспекти розробки C#, як роздільні модифікатори virtual і override, правила вирішення перевантаження методів і підтримка явного оголошення членів інтерфейсу.

Враховуючи все вищеперераховане, для вирішення поставленої задачі було прийнято рішення використовувати мову програмування C#.

2.3.2 Вибір середовища розробки

Для роботи з програмним кодом мовою C# потрібно обрати сережовище розробки програмного забезпечення. Проаналізуємо найпопулярніші: Project Rider, Eclipse та Visual Studio.

Project Rider – середовище розробки програмного забезпечення від JetBrains для роботи з платформою .NET [32]. Випущена в минулому році, але вже придбала багато шанувальників.

Плюси:

– ReSharper. Це плагін, спочатку розроблений для підвищення продуктивності Visual Studio. Тепер на його основі випущена IDE.

– Підтримка повного циклу розробки програмного забезпечення. Фірмова риса продуктів JetBrains, втілена і в Project Rider. З ним можна організувати весь цикл створення програмного забезпечення: від ідеї до підтримки.

– Multiple runtime. Підтримка декількох запущених програм.

– Функціональність. Project Rider дозволяє підключити MSBuild і XBuild, працювати з CLI-проєктами і проводити тестування розроблене програмного забезпечення та налагодження застосунків .NET and Mono. Безліч опцій для швидкого створення коду покращує продуктивність.

– Кросплатформеність. Project Rider працює з Windows, Linux і MacOS.

– Контроль версій. Вбудований інструмент розробки програмного забезпечення дозволяє безпосередньо організувати роботу з Git, Mercurial і TFS.

Проте, дане середовище має ряд недоліків [32]:

– Молодість. Частина функціональності ще в розробці, не всі стартові баги виправлені.

– Вартість. Найдешевша версія Project Rider обійдеться в 139 доларів за перший рік використання. Але є тріал-версія і спеціальні пропозиції для студентів і різних непрофільних організацій.

Eclipse – одне з найпопулярніших багатомовних середовищ розробки програмного забезпечення. Орієнтоване переважно на розробку Java-застосунків, але корисна і для кодів на C#.

Переваги [32]:

– Безліч плагінів. У Eclipse чи не найбільше число надбудов - «на всі випадки життя».

– Активне співтовариство. Допомагає швидше освоїти середовище розробки програмного забезпечення, випускає нові плагіни.

– Відмінні компілятор і відладчик. Перший працює на порядок швидше, ніж у конкурентів, другий – показує потоки, перетинання, дозволяє гнучко управляти ходом налагодження програмного забезпечення.

– Кастомізація. Завдяки плагинам і налаштувань можна повністю персоналізувати Eclipse.

– Безкоштовність. Це open-source проект, абсолютно безкоштовний.

– Висока функціональність. Завдяки офіційним розробникам і членам спільноти за допомогою Eclipse можна провести будь-який C #-продукт по повному циклу розробки програмного забезпечення.

Проте, істотними мінусами є складність та відсутність гарантій надійності. Так як плагіни створюються спільнотою, за їх якість відповідає тільки розробник. Крім того, самі творці Eclipse з кожною новою версією плодять баги, не встигаючи часом виправляти старі.

Visual Studio визнана кращою IDE для розробки програмного забезпечення мовою C#[33]. З Visual Studio багато хто починає знайомитися з мовою і не розлучаються з нею протягом всієї кар'єри програміста.

У число переваг Visual Studio входить наступне [33]:

– Середовище розробки програмного забезпечення містить безліч інструментів, які дуже добре працюють на C#.

– Наявність безкоштовної версії – Community Edition.

– Community містить всі що потрібно для стороннього виробника.

– Найефективніше програмне забезпечення для розробки програмного забезпечення на будь-якій платформі, включаючи .Net і C#.

– Можливість зберігання даних у хмарі.

Отже, зважаючи на вищеописані переваги та недоліки мов програмування та середовищ розробки, для розробки програмного забезпечення використовуватимемо середовище Visual Studio і мову програмування C#.

Розділ 3

Програмна реалізація інформаційної системи

3.1 Структура та функціональне призначення складових системи

Створення та розробка будь-якого програмного забезпечення розпочинається з проектування та реалізації його структури. Для програмного забезпечення, що розробляється на мові програмування з об’єктно-орієнтованим підходом такою структурою є діаграма класів. Тому відповідно до поставленого завдання була розроблена діаграма класів майбутньої системи розпізнавання образів (рисунок 3.1).

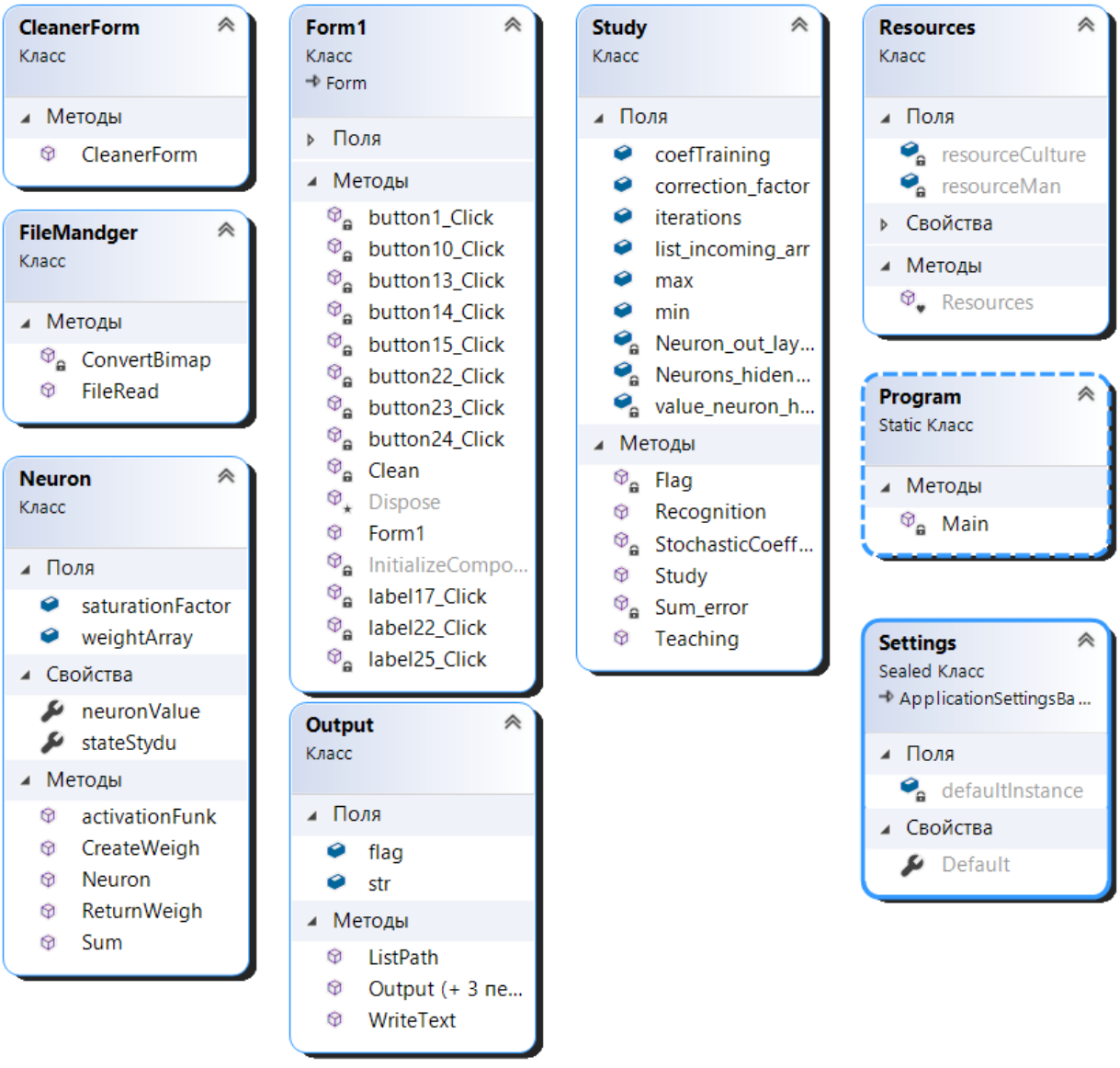


Рисунок 3.1 – Діаграма класів

У розробленій структурі важливе місце займають класи Study та Neuron оскільки саме в них проводиться реалізація основних механік нейронної мережі. Клас Neuron та його елементи відповідають за програмне представлення структури штучного нейрона. Сумування виконується за допомогою методу Sum, результат який повертає нейрон зберігається у властивості neuronValue. Функцію активації реалізовує метод activationFunk. Коефіцієнти ваг які входять до нейрона записують в масив weightArray. Клас Study реалізує алгоритм навчання нейронної мережі. Протягом роботи методу Teaching відбувається корегування коефіцієнтів ваг на всіх шарах мережі. Процес корегування закінчується коли метод Flag повертає істину, це значить ваги досягли потрібних значень та нейрона мережа може вдало розпізнати всі тестові відбитки.

Клас FileMandger за допомогою методів реалізованих в ньому проводить конвертування вхідного зображення відбитку пальця в цифрове представлення, а саме 0 та 1.

Клас Output реалізує взаємодію програми з користувачем, а саме проводить завантаження зображень по вказаному шляхові та відображує вибране зображення у потрібному елементі на формі. Клас CleanerForm проводить очищення елементів форми від введених даних.

Спроектована та розроблена структура надає можливість реалізувати функції системи розпізнавання образів, що використовує нейронну мережу перцептрон

3.2 Особливості реалізації складових системи

Програмні модулі виконують важливу функцію у розробленні програмного забезпечення, вони дозволяють розділяти програму на функціонально завершенні фрагменти. В створеному додатку реалізовано модулі для навчання та розпізнавання зображень відбитків пальців.

В модулі «Навчання» проводиться завантаження зображень відбитків пальців. Щоб вибранні зображення могли податися на входи нейронної мережі їх

потрібно перетворити в числовий масив, для цього використовується наведений нижче програмний код:

```
static double[] ConvertBimap(Bitmap srcImage)
{
   double[] mas = new double[srcImage.Height * srcImage.Width];
   int g = 0;
   for (var y = 0; y < srcImage.Height; y++)
   {
       for (var x = 0; x < srcImage.Width; x++)
       {
           Color srcPixel = srcImage.GetPixel(x, y);
           mas[g] = srcPixel.GetBrightness();
           if (mas[g] == 0)
               mas[g] = 1;
           else
               mas[g] = 0;
           g++;
   }
   return mas;
}
```

Також в модулі «Навчання» відбувається формування списку навчальних образів довільної величини. На основі сформованої переліку образів відбувається навчання. Під час формування списку навчальних образів потрібно враховувати той факт, що чим більший його розмір тим краще нейронна мережа зможе розпізнавати зображення з дефектами, але великий розмір вибірки призведе до збільшення часу навчання (рисунок 3.2).

Корегувати швидкість навчання нейронної мережі можна за допомогою кроку прогресії, коефіцієнту насичення та коефіцієнту навчання, а також змінюючи межі генерування стохастичного коефіцієнту. Під час зміни цих параметрів змінюється також час навчання та його якість. При введенні великих значення коефіцієнту навчання нейронна мережа може не навчитися розпізнати схожі відбитки, що призведе до за циклювання її роботи. В деяких випадках вирішити проблему зациклювання допомагає стохастичний коефіцієнт навчання, але якщо межі генерування стохастичного коефіцієнту будуть дуже великими то це призведе до збільшення часу навчання нейронної мережі.

Модуль «Розпізнавання» виконує ідентифікацію тестового зображення. Процес розпізнавання образу стає доступним одразу після закінчення навчання

нейронної мережі. Результатом розпізнавання є вивід імовірності збігу одного з навчальних елементів та тестового зображення. Першими виводяться елементи з найбільшою імовірністю збігу (рисунок 3.3).

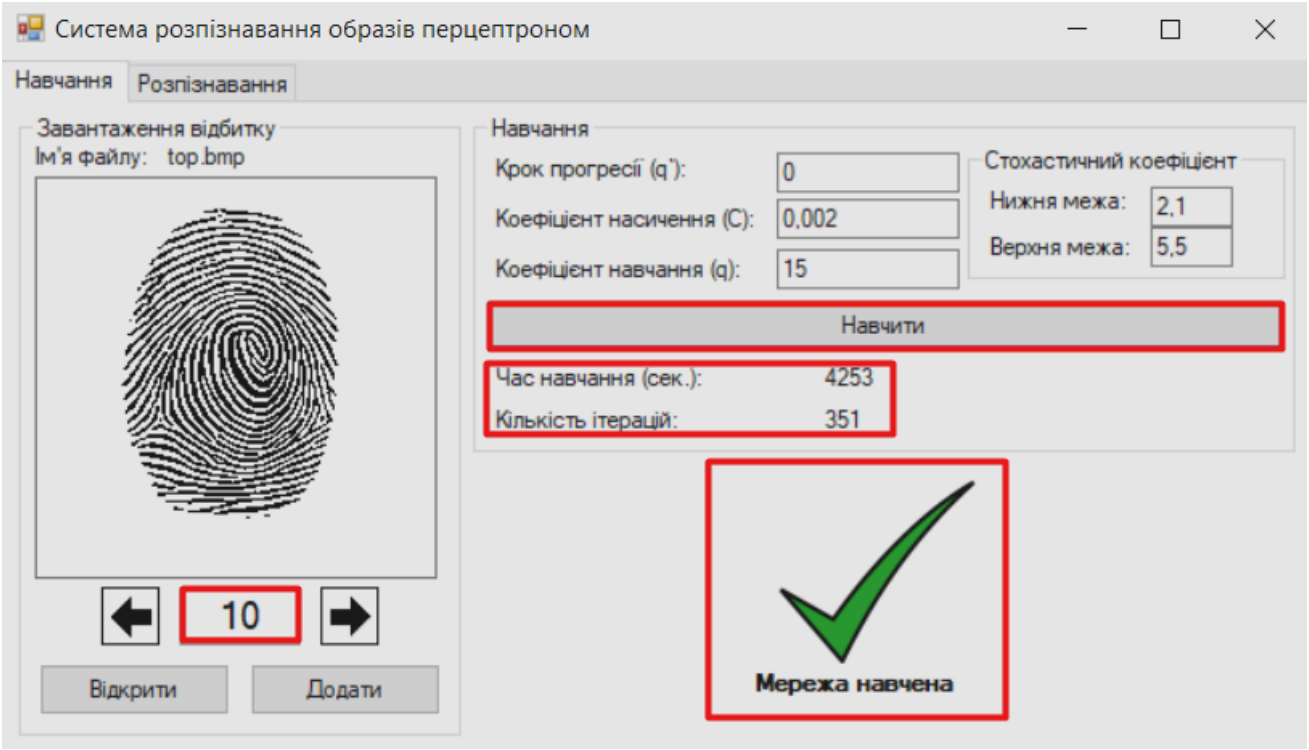


Рисунок 3.2 – Модуль «Навчання»

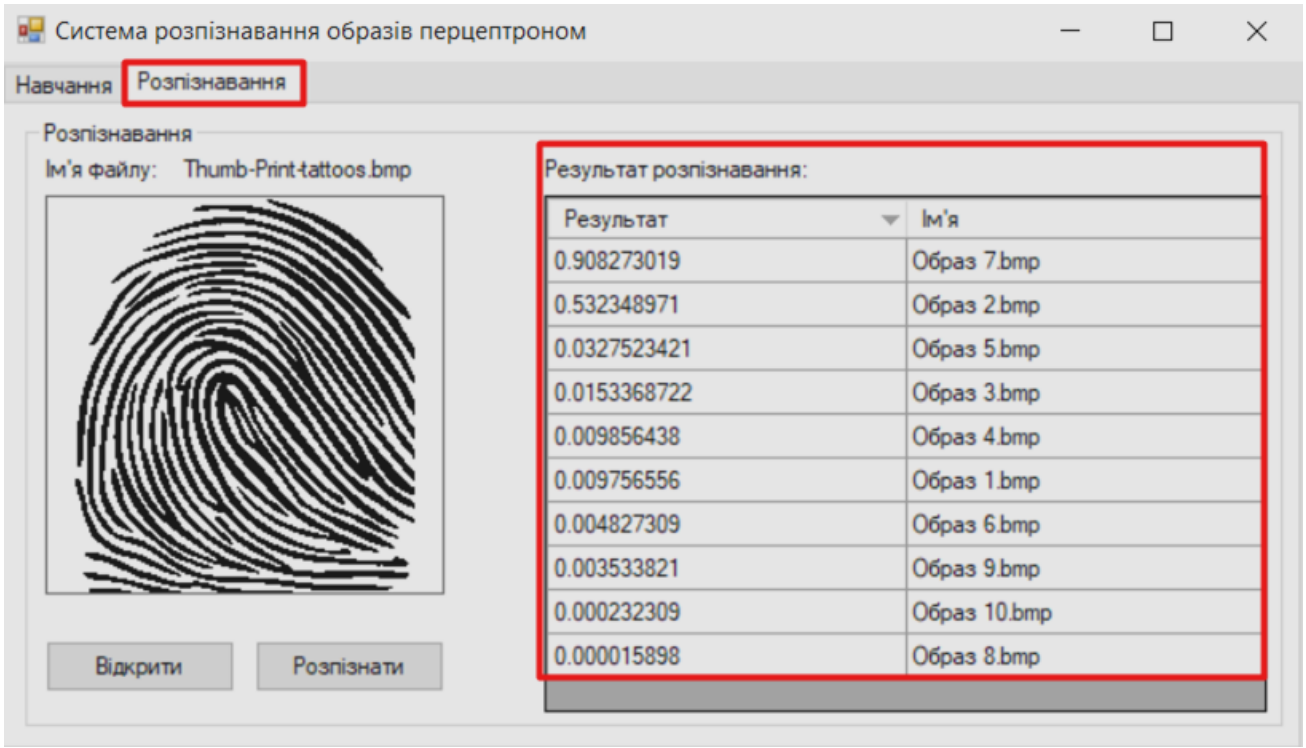


Рисунок 3.3 – Модуль «Розпізнавання»

Розпізнавання зображення відбитку пальця виконується наступним програмним кодом:

```
List<string> vrem = new List<string>();
vrem.Add(str);
List<double[]> list_result_arr = FileMandger.FileRead(vrem);
dataGridView.Rows.Clear();
for (int i = 0; i < HidenLayer.Count(); i++)
{
    HidenLayer[i].ReturnWeigh(ResultArray[0]);
    value_neuron_hiden_layer[i] = HidenLayer [i].neuronValue;
}
for (int i = 0; i < OutLayer.Count(); i++)
{
    OutLayer [i].ReturnWeigh(value_neuron_hiden_layer);
    dataGridView.Rows.Add(OutLayer[i].neuronValue,
img_path[i].Substring(img_path[i].LastIndexOf(@"\") + 1));
}
```

Розробленні модулі виконують логіне групування елементів користувацького інтерфейсу, що дозволяє розділити процес формування навчальних списків та навчання нейронної мережі від процесу розпізнавання образів.

3.3 Тестування інформаційної системи

У відповідності з поставленими завданнями розроблена програма має проводити навчання нейронної мережі та розпізнавання з її допомогою тестових зразків. Для тестування можливостей програми було розроблено набір з чотирьох тест-кейсів, які перевіряють процеси завантаження зображень, розширення списку навчальних образів, навчання нейронної мережі та розпізнавання тестових зображень.

У першому тест-кейсі виконується перевірка завантаження зображень відбитків пальців на форму. Після проходження цього тест-кейсу на формі має відобразитися вибране в спливаючому вікні зображення (таблиця 3.1).

Таблиця 3.1 – Тест-кейс TS001

Тест - кейс Id: TS001	**Пріоритет:** 1	**Розроблено:** 20.04.2021,Пітик Я.О.
Назва: Тестування можливості відображення вибраних зображень на формі **Вхідні дані:** зображення в форматі .bmp		
Кроки		**Очікуваний результат**
1. Запустити програму 2. Відкрити вкладку «Навчання» 3. Натиснути на кнопку «Відкрити» 4. Вибрати у діалоговому вікні зображення 5. Натиснути кнопку «ОК» 6. Переглянути вибране зображення на формі		Відображення вибраного зображення на формі

У результаті проведення тест – кейсу було протестована можливість програми виконувати завантаження зображень в форматі .bmp та виконувати їх вивід на форму. Отриманні результати проходження даного тест–кейсу зображенні на рисунку 3.4.

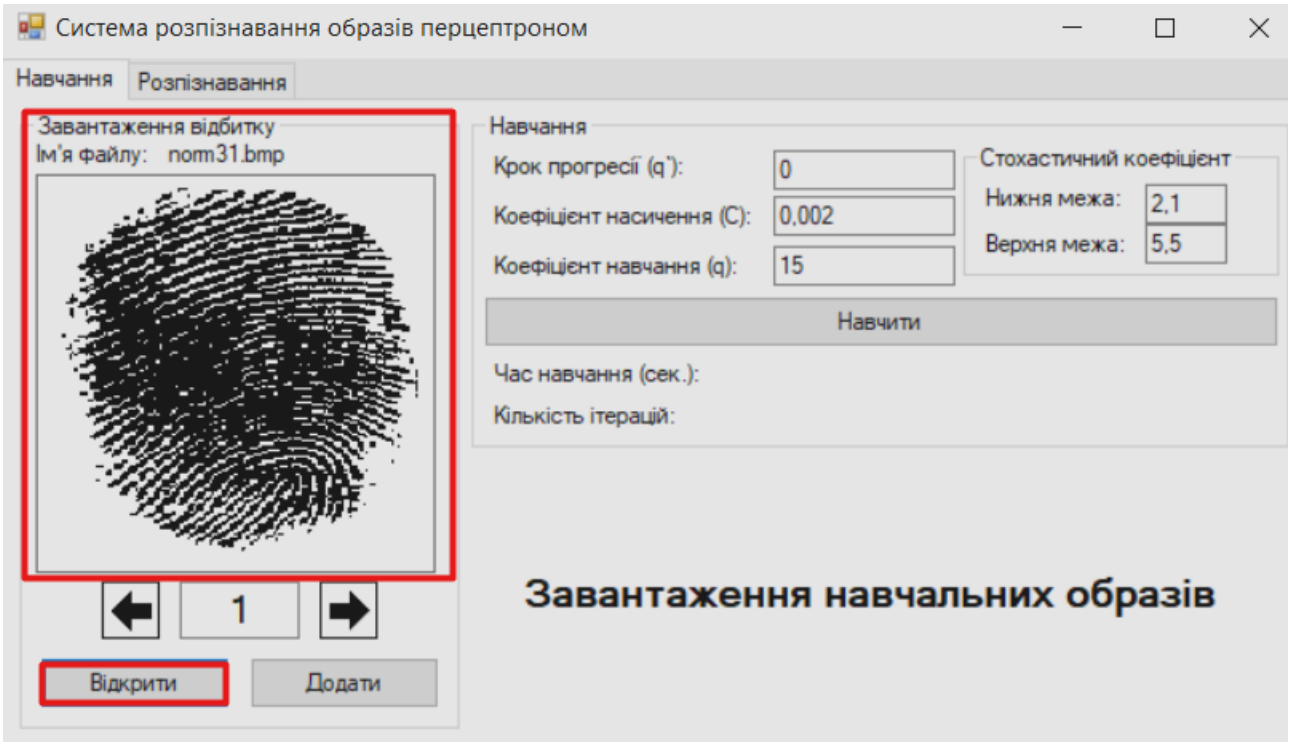


Рисунок 3.4 – Перевірка можливості завантаження зображень на форму

Тест-кейс номер два призначений для тестування можливості розширення списку навчальних образів. У результаті виконання даного тестування мають бути збільшенні розміри списку навчальних елементів (таблиця 3.1).

Таблиця 3.2 – Тест-кейс TS002

Тест - кейс Id: TS002	**Пріоритет:** 1	**Розроблено:** 20.04.2021,Пітик Я.О.
Назва: Тестування можливості розширення списку навчальних образів **Вхідні дані:** п’ять зображень відбитків пальців		
Кроки		**Очікуваний результат**
1. Запустити програму 2. Відкрити вкладку «Навчання» 3. Натиснути на кнопку «Відкрити» та вибрати зображення. 4. Натиснути кнопку «Додати» 5. Повторити операції 3 та 4 п’ять разів		Формування списку з п’яти вхідних образів

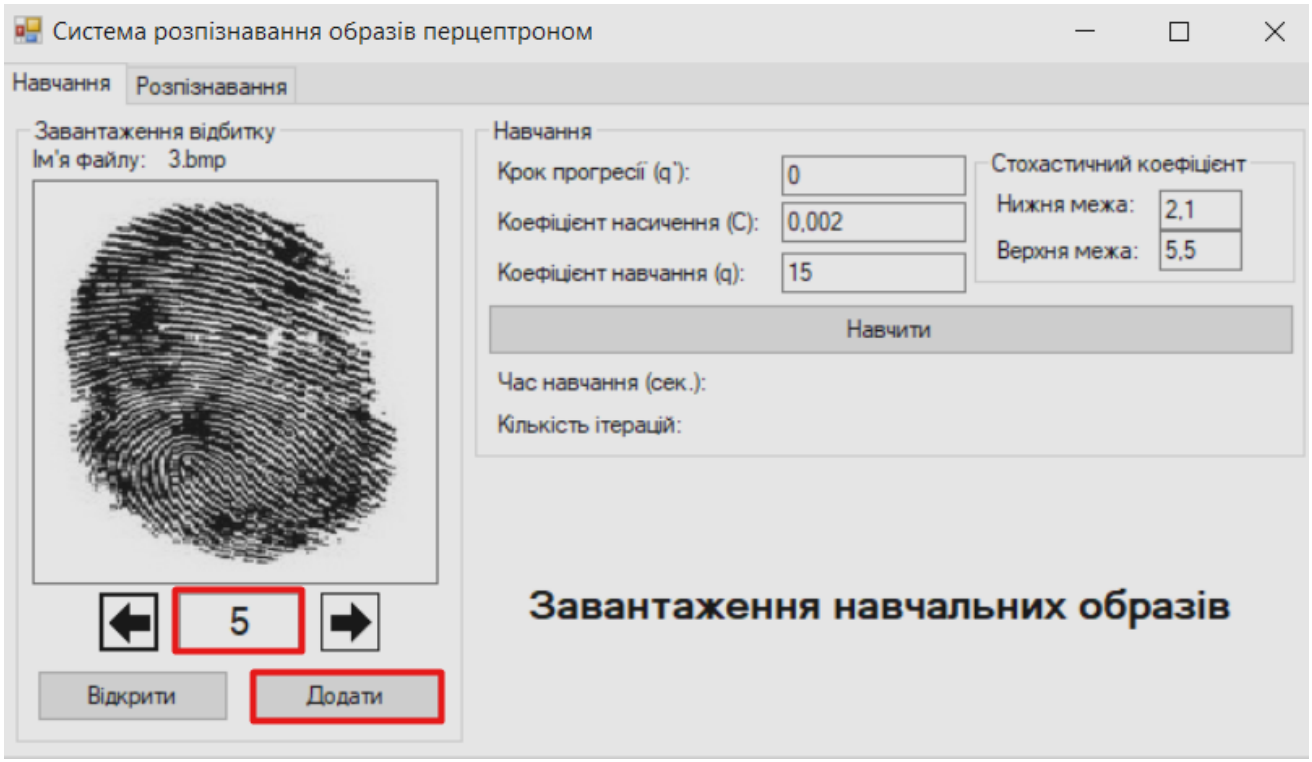


Рисунок 3.5 – Перевірка можливості розширення списку вхідних параметрів

У результаті проведення тест – кейсу номер два була перевірена можливість розширення списку вхідних параметрів, що використовуватимуться

нейронною мережею при навчанні. Результат отриманий після проведення тестування зображено на рисунку 3.5.

У третьому тест-кейсі перевіряється можливість проведення навчання нейронної мережі на п’яти вхідних образах. Після проведення тест-кейсу має бути отримана нейронна мережа готова до розпізнавання зображень відбитків пальців (таблиця 3.3).

Таблиця 3.3 – Тест -кейс TS003

Тест - кейс Id: TS003	**Пріоритет:** 1	**Розроблено:** 20.04.2021,Пітик Я.О.
Назва: Тестування можливості навчання нейронної мережі **Вхідні дані:** п’ять зображень відбитків пальців		
Кроки		**Очікуваний результат**
1. Запустити програму 2. Відкрити вкладку «Навчання» 3. Додати до навчального списку п’ять образів. 4. Натиснути кнопку «Навчання»		Отримання нейронної мережі готової до розпізнавання зображень відбитків пальців

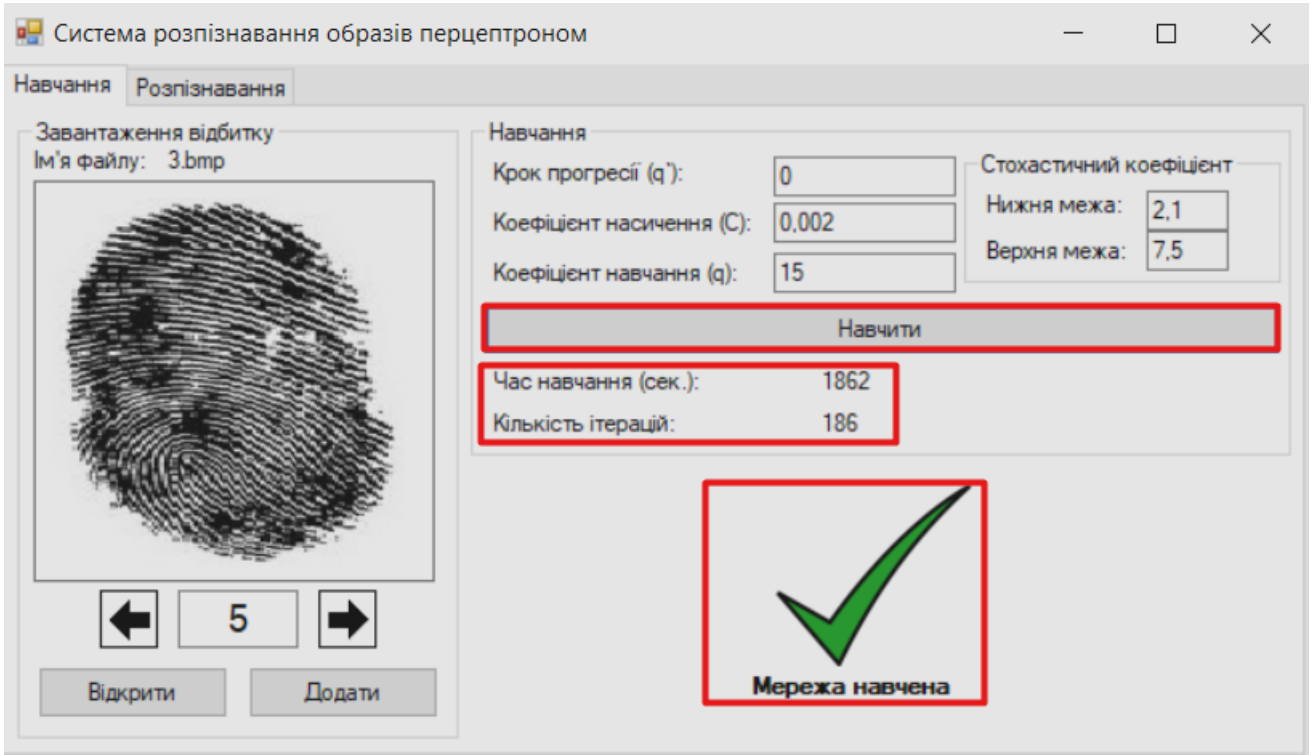


Рисунок 3.6 – Результат навчання нейронної мережі на п’яти вхідних образах

У результаті проведення тест – кейсу номер три була протестована можливість проводити навчання нейронної мережі на зазначеній кількості вхідних образів. Результат отриманий в результаті проходження тест кейсу зображений на рисунку 3.6

Четвертий тест-кейс перевіряє можливості розпізнавання тестового образу за допомогою навченої нейронної мережі. Успішним проходженням тест-кейсу вважатиметься розпізнавання тестового зображення відбитку пальця (таблиця 3.4).

Таблиця 3.4 – Тест -кейс TS004

Тест - кейс Id: TS004	**Пріоритет:** 1	**Розроблено:** 20.04.2021,Пітик Я.О.
Назва: Перевірка можливості розпізнавання тестового образу навченою нейронною мережею **Вхідні дані:** навчена нейронна мережа; тестове зображення відбитку пальця		
Кроки		**Очікуваний результат**
1. Запустити програму. 2. Відкрити вкладку «Навчання». 3. Сформувати списки навчальних вибірок. 4. Провести навчання нейронної мережі. 5. Перейти на вкладку «Розпізнавання» 6. Завантажити тестовий образ 7. Натиснути кнопку «Розпізнати»		Розпізнання нейронною мережею тестового образу

Після проходження четвертого тест-кейсу було успішно проведено розпізнавання тестового зображення відбутку пальця. Результати розпізнавання тестового образу зображено на рисунку 3.7

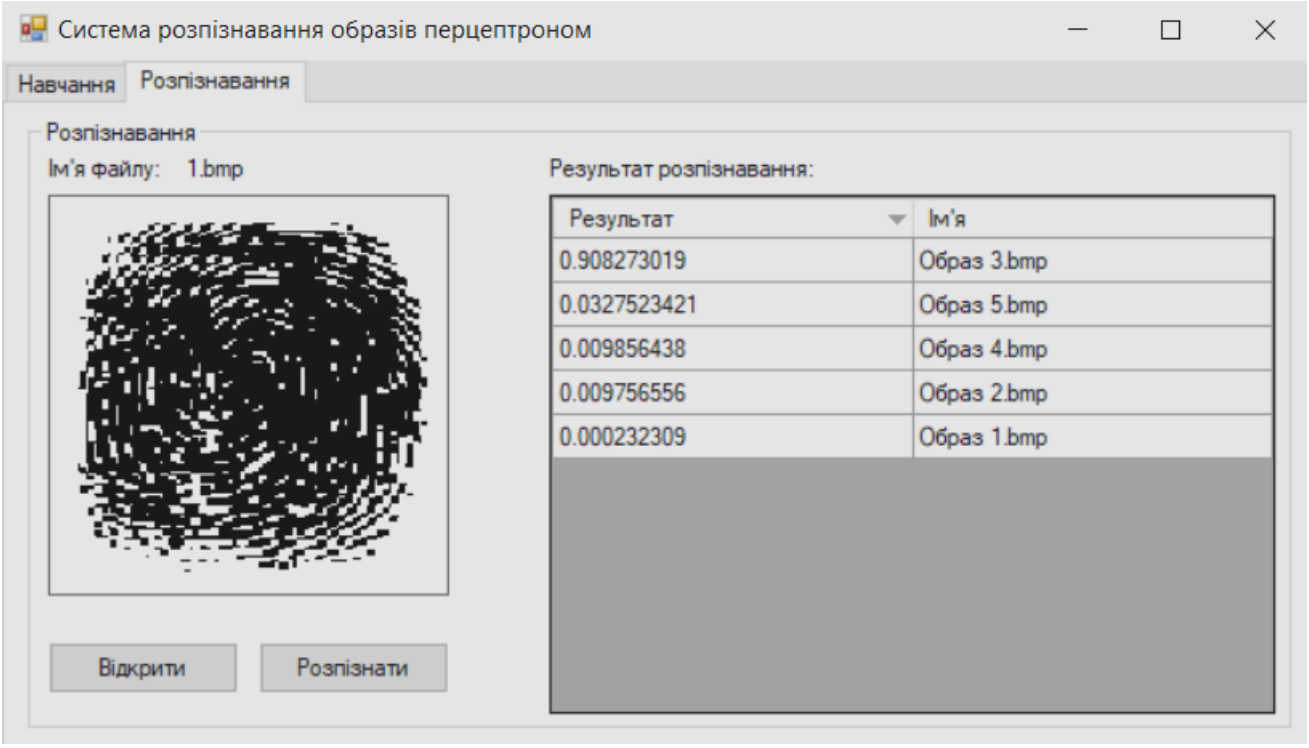


Рисунок 3.7 – Результат розпізнавання тестового зображення нейронною мережею

Тестування інформаційної системи за допомогою розроблених текст-кейсі дозволило перевірити основні моменти роботи програми. У результаті проведеного тестування недоліків роботи системи виявлено не було.

3.4 Інструкція користувача

Після запуску програми перед користувачем відкриється вкладка «Навчання». На ній проводиться завантаження відбитків та їх додавання до навчального списку. Щоб виконати додавання відбитку потрібно натиснути кнопку «Відкрити» у результаті відкриється діалогове вікно у якому користувач вибирає зображення відбитків, після чого на формі відобразиться вибране зображення та назва його файлу (рисунок 3.8).

Після завантаження зображення на форму потрібно виконати його додавання до навчального списку, для цього користувач має натиснути на кнопку «Додати». У результаті чого відбудеться розширення навчального списку

про що повідомить зміна лічильника образів та завантажиться нове полотно для додавання наступного відбутку (рисунок 3.9).

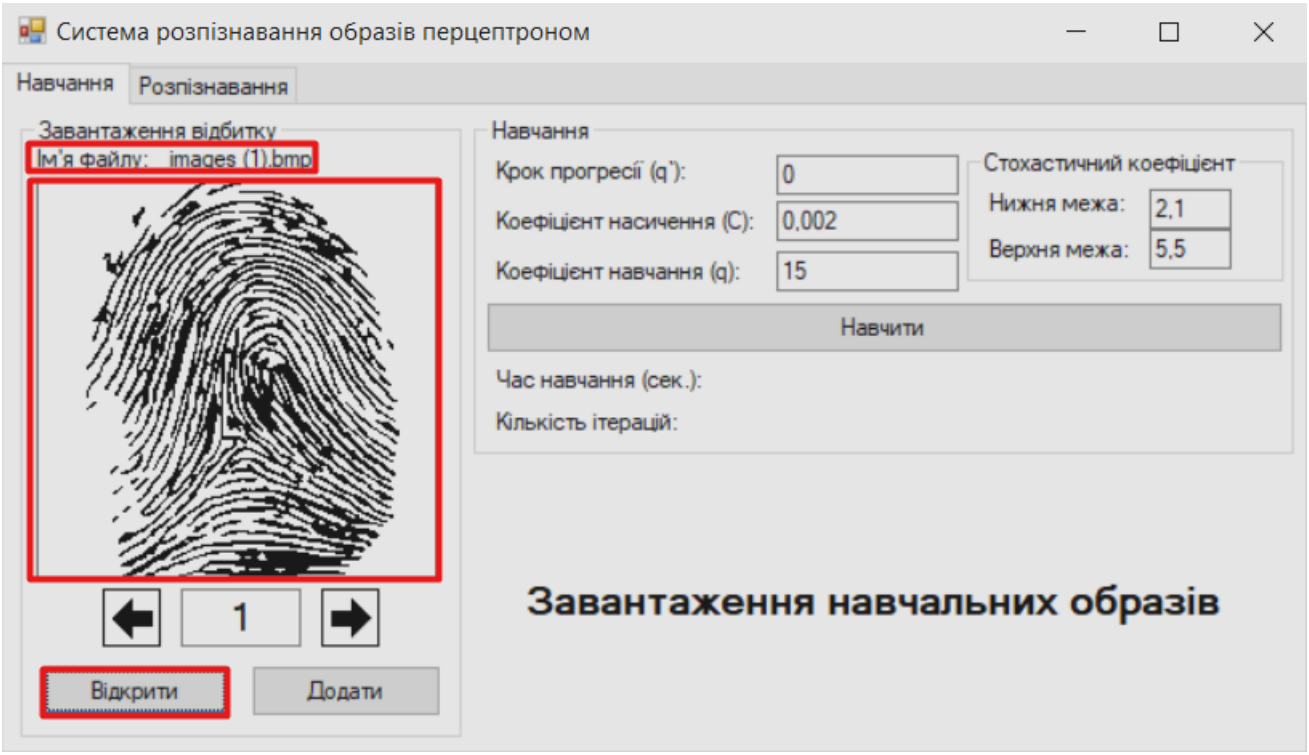


Рисунок 3.8 – Завантаження відбитку на форму

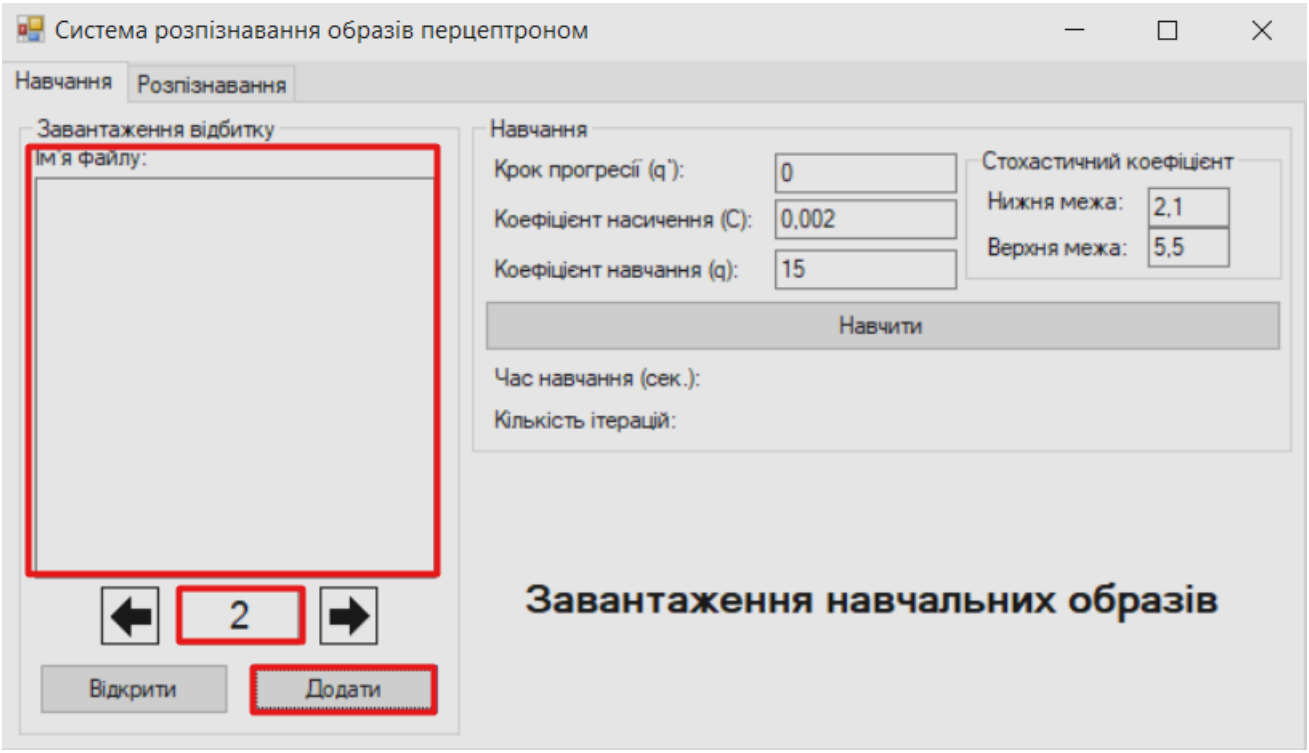


Рисунок 3.9 – Оновлення елементів форми для додавання наступного відбитку

Якщо користувач захоче переглянути список навчальних образів він має скористатися кнопками з стрілками вправо та вліво, номер елементу в списку та його ім’я дозволяють ідентифікувати вибраний відбиток (рисунок 3.10).

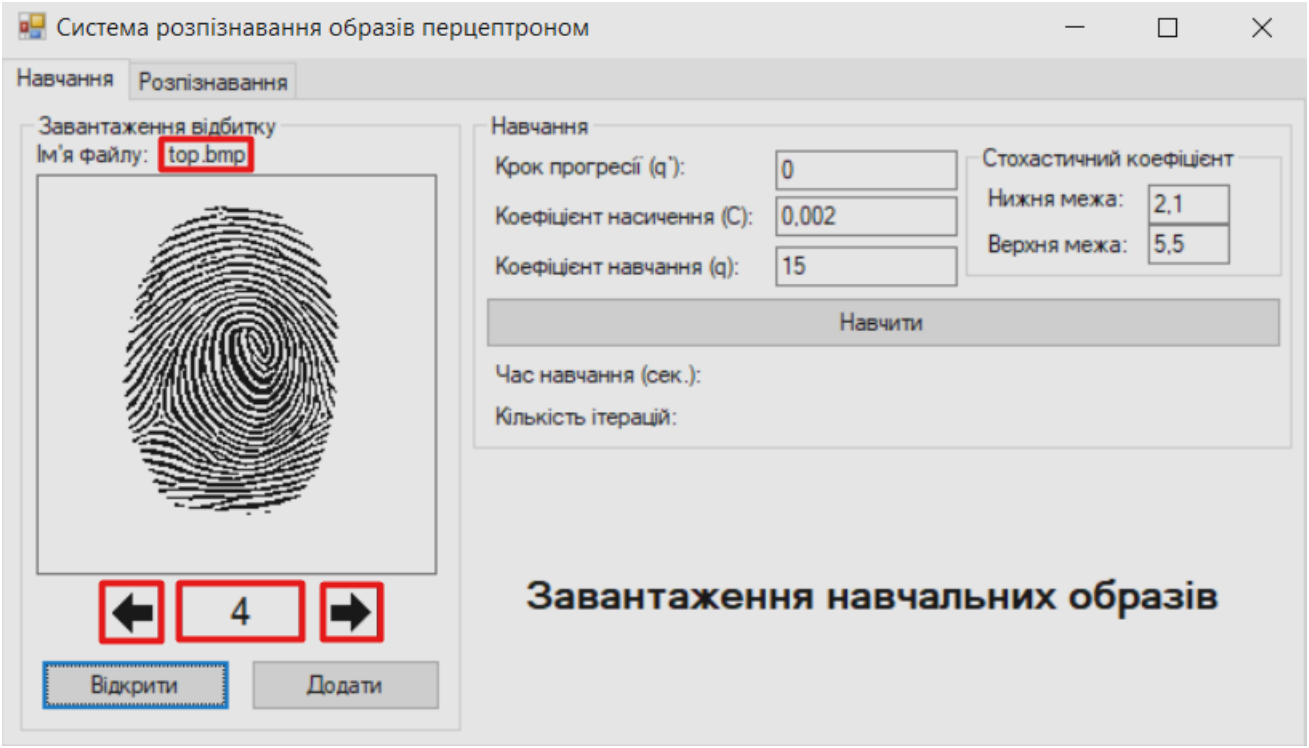


Рисунок 3.10 – Перегляд списку навчальних образів

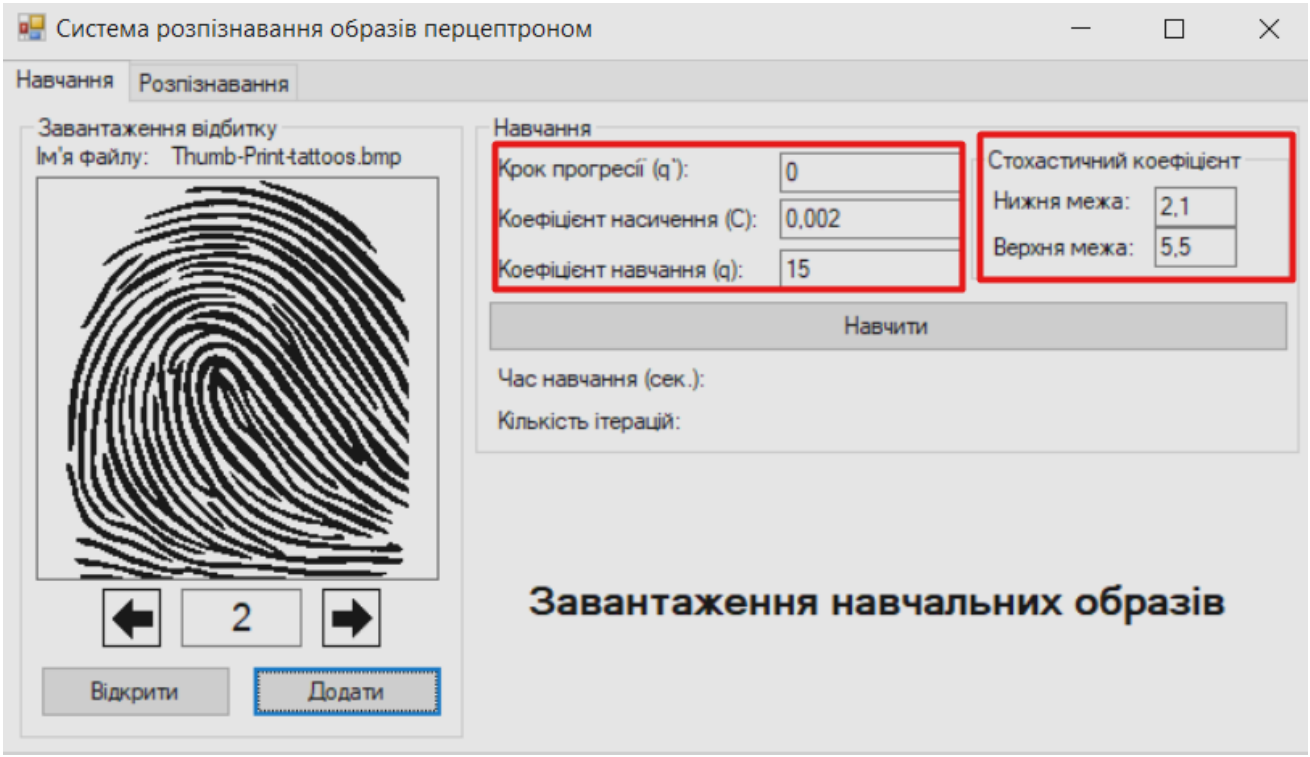


Рисунок 3.11 – Заповнення параметрів навчання нейронної мережі

Коли формування списку навчальних образів закінчено, можна перейти до введення параметрів навчання нейронної мережі. У користувача є можливість введення кроку прогресії, коефіцієнту насичення та коефіцієнту навчання, а також зміни меж генерування стохастичного коефіцієнту (рисунок 3.11).

Після заповнення всіх полів потрібно виконати навчання нейронної мережі, для цього користувач має натиснути кнопку «Навчити». У результаті чого відбудеться запуск алгоритму навчання нейронної мережі (рисунок 3.12).

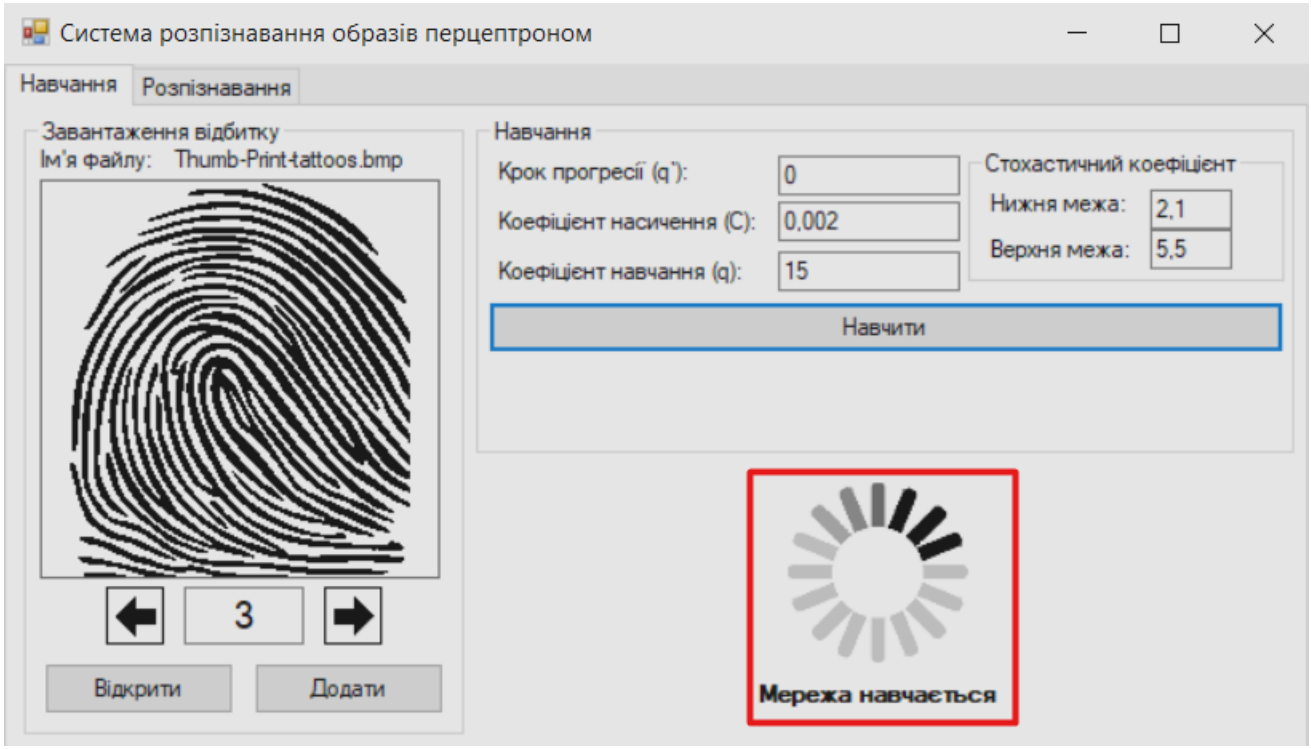


Рисунок 3.12 – Відображення стану навчання нейронної мережі

Після успішного завершення навчання можна провести розпізнавання відбитку, для цього потрібно перейти на вкладку «Розпізнавання». Далі користувач має вибрати зображення для розпізнавання, та натиснути на кнопку «Розпізнати». У результаті проведеного розпізнавання відбитку буде сформована відповідна таблиця з його результатами, де на перших позиціях відмічені образи найбільше схожі с тестовим відбитком (рисунок 3.13).

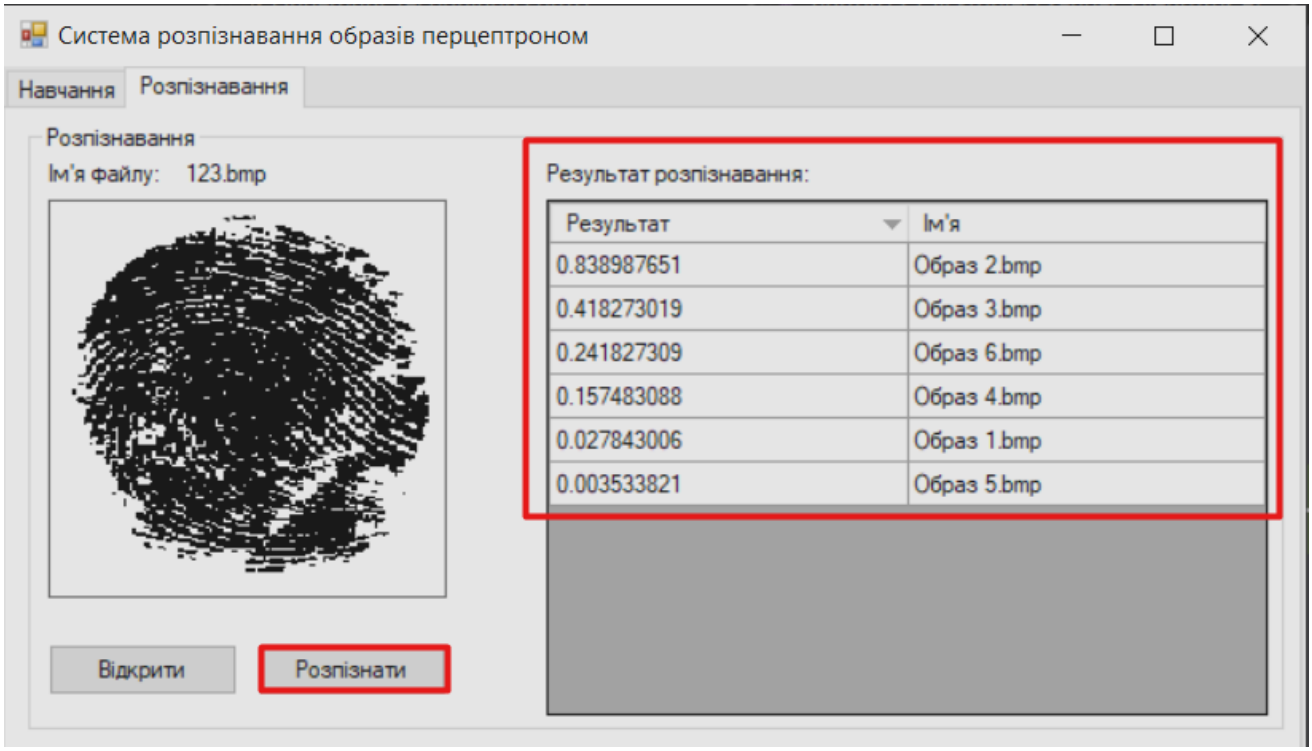


Рисунок 3.13 – Результат розпізнавання відбитка

Беручи до уваги особливості системи, є вдалим рішенням проведення групування елементів, призначених для навчання та розпізнавання у відповідні вкладки, що дозволяє користувачеві простіше працювати з програмою.

3.5 Вимоги до розгортання інформаційної системи

Мінімальні вимоги до програмних засобів:

– Операційна система: Windows 10.

– Наявність встановленого .NET Framework версії 4.6.1;

Вимоги до користувача:

– наявність списку зображень для навчання нейронної мережі;

– наявність тестового зображення.

Мінімальні вимоги до апаратних засобів:

– Процесор:Intel Core 3 або AMD схожої серії;

– Запам’ятовуючий пристрій 128GB.

– ОЗУ: 1024 MB;

Висновки

В результаті виконання кваліфікаційної роботи бакалавра розроблено систему розпізнавання образів на платформі .NET, яка використовує нейронну мережу перцептрон. Для програмної реалізації системи розпізнавання образів було використовувано середовище програмування Visual Studio і мову програмування C#.

У якості навчальних і тестових зразків системи розпізнавання образів використано відбитки пальців рук людей, відповідно прикладна інформаційна система позиціонується як систему розпізнавання осіб за відбитками пальців їх рук. Це визначає дактилоскопію як область прикладного застосування розробленої системи розпізнавання образів.

Система розпізнавання образів використовує нейронну мережу перцептрон та виконує наступні функції:

– підготовка до навчання, що полягає в завантаженні зображень для навчання, їх бінаризації, формуванні рецепторної матриці входів нейромережі, визначенні приналежності зображень образам та формуванні стартової множини ваг синапсів;

– навчання нейронної мережі для одержання проміжних даних множини ваг синапсів для розпізнавання;

– підготовка до розпізнавання, що полягає в завантаженні зображення для розпізнавання, його бінаризації й формуванні матриці входів нейронної мережі;

– розпізнавання зображення із використанням множини ваг синапсів, одержаної в результаті навчання;

– формування результату розпізнавання у вигляді цифрової оцінки приналежності зразка навченим образам.

В створеній системі розпізнавання образів навчання нейронної мережі перцептрон, результатом чого є сформована множина ваг синапсів для розпізнавання, реалізовано двома наступними алгоритмами:

– класичний алгоритм зворотного поширення помилки;

– стохастичний алгоритм навчання нейронної мережі.

Використаний стохастичний алгоритм навчання нейронної мережі містить складову випадкового підбору параметрів для кожної ітерації. Реалізація обох алгоритмів навчання перцептрона у системі розпізнавання образів дозволить у подальшому виконати порівняння їх ефективності.

Перелік посилань

1. Розробка рішень на основі Machine Learning. URL: https://evergreens.com.ua/ru/development-services/machine-learning.html

2. Біометрія від «А» до «Я» повне керівництво біометричної ідентифікації і аутентифікації.URL: https://securityrussia.com/blog/biometriya.html

3. Айвазян, С.А. Прикладна статистика. Основи економетрики / С.А. Айвазян и др. - М.: Юнити, Т1 – 2001. С. 458

4. Метод формування нечіткого класифікатора самостійного налаштування коевулюційними алгоритмами. URL: http://www.isa.ru/aidt/images/documents/2010-03/98_106.pdf

5. Порівняльний аналіз двох підходів для вирішення задач класифікації. URL: http://nbuv.gov.ua/UJRN/recs_2014_6_23

6. Загальна теорія розпізнавання. URL: http://ecat.diit.edu.ua/ft/Recognition1.pdf

7. Методи навчання штучної нейронної мережі. URL: http://ena.lp.edu.ua:8080/bitstream/ntb/36551/1/26_160-170.pdf

8. Machine Learning – машинне навчання . URL: https://www.it.ua/ru/knowledge-base/technology-innovation/machine-learning

9. Штучні нейронні мережі (ШНМ). URL: https://www.it.ua/ru/knowledge-base/technology-innovation/iskusstvennye-nejronnye-seti-ins

10. Розпізнавання дактилоскопічних зображень. URL: https:// cyberleninka.ru/article/n/raspoznavanie-daktiloskopicheskih-izobrazheniy/viewer

11. Методи розпізнавання відбитків пальців за допомогою Python. URL: https://habr.com/ru/post/116603/

12. Класифікація даних за допомогою нейронних мереж. URL: https://loginom.ru/blog/neural-classification

13. Порівняльний аналіз двох підходів для проведення розпізнавання образів. URL: http://nbuv.gov.ua/UJRN/recs_2014_6_23

14. Дослідження алгоритмів навчання нейронних мереж в задачах прогнозування. URL: http://nbuv.gov.ua/UJRN/Npchduct_2009_117_104_18

15. Дослідження компютерних систем штучного інтелекту. URL: https://www.uzhnu.edu.ua/en/infocentre/get/10973

16. Когнітрон. URL: https://intellect.icu/kognitron-6015

17. Когнітрон та неокогнітрон Фукушими. URL: https://studbooks.net/2238300/informatika/kognitron_neokognitron_fukushimy

18. Fukushima K. Neocognitron: a self-organising neural network for mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics 36, 1980, pp. 193-202.

19. Перцептрон. URL: https://znaimo.com.ua

20. Метод зворотного поширення помилки. URL: https://uk.wikipedia.org/wiki/Метод зворотного поширення помилки

21. Загальна теорія розпізнавання образів. URL: http://ecat.diit.edu.ua/ft/Recognition1.pdf

22. Java (програмна платформа). URL: https:// ru.wikipedia.org/wiki/Java_(програмна платформа)

23. Плюси та мінуси програмування на Java. URL: https:// medium.com/nuances-of-programming/плюсы-и-минусы-программирования-на-java-2861f4c2a0d5

24. PHP. URL: https://uk.wikipedia.org/wiki/PHP

25. Загалі відомості про платформу .NET. URL: https://docs.microsoft.com/ru-ru/dotnet/framework/get-started/overview

26. .NET Programming Languages. URL: https://dotnet.microsoft.com/languages.

27. VB.NET. URL: http://progopedia.ru/implementation/vbnet/

28. Закінчення легенди. Microsoft хоронить Visual Basic. URL: https://www.cnews.ru/news/top/2020-03-13_zakat_epohi_visual_basic_microsoft

29. F Sharp (мова програмування) – F Sharp (programming language URL: https://ru.xcv.wiki/wiki/F_Sharp_(programming_language)

30. Що вибрати початківцю: C# чи F# URL: https://skillbox.ru/media/code/chto_vybrat_novichku_c_ili_f/

31. Короткий огляд мови програмування C#. URL: https://docs.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/

32. Найкращі IDE для розробки на C#. URL: https://gb.ru/posts/c_sharp_ides

33. Переваги та недоліки Visual Studio, Project Rider, Eclipse. URL: https://spb.deveducation.com/blog/luchshie-ide-dlya-c-razrabotchika/

ДОДАТКИ

Додаток А

Зразки відбитків пальців рук людей для використання у системі розпізнавання образів

Навчальні відбитки:




Тестові образи:


Джерела:

1. Відбитки пальців. URL: https://cutt.ly/NnUYbDo

2. Відбитки пальців колекція зображень. URL: https://uk.crazypng.com/885.html

3. Відбитки пальців та вектори. URL:https://cutt.ly/znUYUQt

4. Як змінити відбитки пальців і чи це можливо? URL: http://bigbro.com.ua/yak-zminiti-vidbitki-paltsiv-i-chi-tse-mozhlivo/

Додаток Б

Програмні коди основних модулів системи розпізнавання образів

Лістинг Output.cs:

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Fingerprint_recognition
{
    class Output
    {
        public bool flag;
        public string str;


        public Output(string path, PictureBox picture_zoom)
        {
            OpenFileDialog openFileDialog1 = new OpenFileDialog() { Filter = "Файл
зображення (*.bmp)|*.bmp" };

            openFileDialog1.InitialDirectory = path;

            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                path = openFileDialog1.FileName;
                str = path;
                picture_zoom.Image = new Bitmap(path);
                path = path.Substring(path.LastIndexOf(@"\") + 1);
                flag = true;

            }
        }

        public Output(string path, PictureBox picture_zoom, PictureBox normal_size,
Label label)
        {
            OpenFileDialog openFileDialog1 = new OpenFileDialog() { Filter = "Файл
зображення (*.bmp)|*.bmp" };

            openFileDialog1.InitialDirectory = path;

            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                path = openFileDialog1.FileName;
                str = path;
                //   picture_zoom.Image = new Bitmap(path);
```

```
            List<string> vrem = new List<string>();
            vrem.Add(path);
            List<double[]> vream_double = FileMandger.FileRead(vrem);
            int[,] vream_int = new int[20, 20];

            for (int i = 0, z = 0; i < 20; i++)
                for (int j = 0; j < 20; j++)
                {
                    vream_int[i, j] = (Int32)vream_double[0][z];
                    z++;
                }

            path = path.Substring(path.LastIndexOf(@"\") + 1);
            label.Text = "Ім'я файлу: " + path.Substring(0, path.Length - 4);
            normal_size.Image = new Bitmap(str);
            flag = true;

        }

    }


    public Output(string path, PictureBox picture_zoom,
        TextBox name, TextBox full_path, Label label) : this(path, picture_zoom)
    {

        if (flag == true)
        {
            full_path.Text = str;
            name.Text = str.Substring(str.LastIndexOf(@"\") + 1);
        }


    }

    public Output(string path, PictureBox picture_zoom, PictureBox normal_size,
         TextBox name, TextBox full_path, Label label) : this(path, picture_zoom,
normal_size, label)
    {

        if (flag == true)
        {
            full_path.Text = str;
            name.Text = str.Substring(str.LastIndexOf(@"\") + 1);
        }


    }



    public static void ListPath(PictureBox picture_zoom, TextBox counter, int ident,
List<string> path_letters, string full_path)
    {
```

```
            if (full_path == "")
            {
                MessageBox.Show("Необхідно завантажити образ");
                return;
            }
            if (path_letters.Count < Convert.ToInt32(counter.Text))
            {
                path_letters.Add(full_path);
                CleanerForm f_clear;
                counter.Text = Convert.ToString((Convert.ToInt32(counter.Text) + 1));
                switch (ident)
                {
                    case 0:
                        f_clear = new CleanerForm();
                        f_clear.CleanImage(picture_zoom);
                        break;
                    case 1:
                        f_clear = new CleanerForm();
                        f_clear.CleanImage(picture_zoom);
                        break;

                }

            }
            else
                path_letters[Convert.ToInt32(counter.Text) - 1] = full_path;

        }

        public static void WriteText(string str, PictureBox picture_zoom,  int index,
string full_path)
        {

            if (str != "")
            {


                if (index == 1)
                {
                    List<string> vrem = new List<string>();
                    vrem.Add(str);
                    List<double[]> vream_double = FileMandger.FileRead(vrem);
                    int[,] vream_int = new int[20, 20];

                    for (int i = 0, z = 0; i < 20; i++)
                        for (int j = 0; j < 20; j++)
                        {
                            vream_int[i, j] = (Int32)vream_double[0][z];
                            z++;
                        }

                }
                else
```

```
                    picture_zoom.Image = new Bitmap(str);


                full_path = str;
                str = str.Substring(str.LastIndexOf(@"\") + 1);
            }
            else
            {
                picture_zoom.Image = null;
                full_path = "";
            }
        }
    }
}
```

Лістинг Study.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Fingerprint_recognition
{
    class Study
    {
        public static double coefTraining = 10;
        public static double correction_factor = 0;
        public static double min = 1;
        public static double max = 5;
        List<Neuron> Neuron_out_layer;
        List<Neuron> Neurons_hiden_layer;
        public List<double[]> list_incoming_arr;
        double[] value_neuron_hiden_layer;
        public int iterations = 0;



        public Study(List<string> path_letters)
        {
            Neurons_hiden_layer = new List<Neuron>();

            list_incoming_arr = FileMandger.FileRead(path_letters);


            if (list_incoming_arr[0].Count() < 1000)
                value_neuron_hiden_layer = new double[(int)Math.Ceiling((double)
                    ((list_incoming_arr[0].Count() + list_incoming_arr.Count()) / 2))];
            else
                value_neuron_hiden_layer = new double[(int)Math.Ceiling((double)
                  ((400 + list_incoming_arr.Count()) / 2))];
```

```
    for (int i = 0; i < value_neuron_hiden_layer.Count(); i++)
    {
        Neurons_hiden_layer.Add(new Neuron(list_incoming_arr[0]));
        value_neuron_hiden_layer[i] = Neurons_hiden_layer[i].neuronValue;
    }

    Neuron_out_layer = new List<Neuron>();

    for (int i = 0; i < list_incoming_arr.Count(); i++)
    {
        Neuron_out_layer.Add(new Neuron(value_neuron_hiden_layer));
    }

}


bool Flag(List<double[]> list_incoming_arr)
{
    int counter = 0;

    for (int k = 0; k < Neuron_out_layer.Count(); k++)
    {
        for (int i = 0; i < Neurons_hiden_layer.Count(); i++)
        {
            Neurons_hiden_layer[i].ReturnWeigh(list_incoming_arr[k]);
            value_neuron_hiden_layer[i] = Neurons_hiden_layer[i].neuronValue;
        }

        for (int i = 0; i < Neuron_out_layer.Count(); i++)
        {
            Neuron_out_layer[i].ReturnWeigh(value_neuron_hiden_layer);
        }

        if (Neuron_out_layer[k].neuronValue >= 0.9)
            counter++;

    }

    if (counter == Neuron_out_layer.Count())
        return false;
    else
        return true;
}

double Sum_error(int counter, double[] error_out)
{
    double count = 0;
    for (int i = 0; i < Neuron_out_layer.Count(); i++)
    {
        count += Neuron_out_layer[i].weightArray[counter] * error_out[i];
    }

    return count;
}
```

```
        public void Teaching()
        {


            double[] error_out = new double[Neuron_out_layer.Count()];


            double[] error_hiden = new double[Neurons_hiden_layer.Count()];
            int k = 0;

            while (Flag(list_incoming_arr))
            {
                if (iterations % Neuron_out_layer.Count == 0)
                {
                    coefTraining += correction_factor;

                }
                iterations++;


                if (k != 0)
                {
                    Neuron_out_layer[k].stateStydu = 1;
                    Neuron_out_layer[k - 1].stateStydu = 0;
                }
                else if (Neuron_out_layer.Count() == 1)
                    Neuron_out_layer[k].stateStydu = 1;
                else
                {
                    Neuron_out_layer[k].stateStydu = 1;
                    Neuron_out_layer[Neuron_out_layer.Count() - 1].stateStydu = 0;
                }


                for (int i = 0; i < Neurons_hiden_layer.Count(); i++)
                {
                    Neurons_hiden_layer[i].ReturnWeigh(list_incoming_arr[k]);
                    value_neuron_hiden_layer[i] = Neurons_hiden_layer[i].neuronValue;
                }




                for (int i = 0; i < Neuron_out_layer.Count(); i++)
                {
                    Neuron_out_layer[i].ReturnWeigh(value_neuron_hiden_layer);
                }


                for (int i = 0; i < Neuron_out_layer.Count(); i++)
                    error_out[i] = (Neuron_out_layer[i].stateStydu -
Neuron_out_layer[i].neuronValue)
```

```
                        * (Neuron_out_layer[i].neuronValue * (1 -
Neuron_out_layer[i].neuronValue));

                Parallel.For(0, error_hiden.Count(), i =>
                {
                    error_hiden[i] = Sum_error(i, error_out) *
                             Neurons_hiden_layer[i].neuronValue * (1 -
Neurons_hiden_layer[i].neuronValue);
                });


                for (int i = 0; i < Neuron_out_layer.Count(); i++)
                {
                    for (int j = 0; j < Neuron_out_layer[0].weightArray.Count(); j++)
                    {
                        Neuron_out_layer[i].weightArray[j] =
Neuron_out_layer[i].weightArray[j] +
                           StochasticCoefficient() * coefTraining * error_out[i] *
Neurons_hiden_layer[j].neuronValue;

                    }
                }

                for (int i = 0; i < Neurons_hiden_layer.Count(); i++)
                {
                    for (int j = 0; j < Neurons_hiden_layer[0].weightArray.Count(); j++)
                    {
                        Neurons_hiden_layer[i].weightArray[j] =
Neurons_hiden_layer[i].weightArray[j] +
                           StochasticCoefficient()* coefTraining * error_hiden[i] *
list_incoming_arr[k][j];
                    }
                }

                if (k < Neuron_out_layer.Count() - 1)
                {
                    k++;
                }
                else
                    k = 0;

            }

        }

        public void Recognition(string str, DataGridView dataGridView, List<string>
path_letters)
        {
            List<string> vrem = new List<string>();
            vrem.Add(str);
            List<double[]> list_result_arr = FileMandger.FileRead(vrem);
            dataGridView.Rows.Clear();
            for (int i = 0; i < Neurons_hiden_layer.Count(); i++)
            {
```

```
                Neurons_hiden_layer[i].ReturnWeigh(list_result_arr[0]);
                value_neuron_hiden_layer[i] = Neurons_hiden_layer[i].neuronValue;
            }

            for (int i = 0; i < Neuron_out_layer.Count(); i++)
            {
                Neuron_out_layer[i].ReturnWeigh(value_neuron_hiden_layer);
                dataGridView.Rows.Add(Neuron_out_layer[i].neuronValue,
path_letters[i].Substring(path_letters[i].LastIndexOf(@"\") + 1));
            }
        }

        double StochasticCoefficient()
        {
            Random random = new Random();
            return random.NextDouble() * (max - min) + min;
        }

    }
}
```

Лістинг FileMandger.cs:

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Fingerprint_recognition
{
    class FileMandger
    {
        static double[] ConvertBimap(Bitmap srcImage)
        {

            if (srcImage == null)
                throw new ArgumentNullException("srcImage");

            double[] mas = new double[srcImage.Height * srcImage.Width];

            int g = 0;

            for (var y = 0; y < srcImage.Height; y++)
            {
                for (var x = 0; x < srcImage.Width; x++)
                {
                    Color srcPixel = srcImage.GetPixel(x, y);
                    mas[g] = srcPixel.GetBrightness();
                    if (mas[g] == 0)
                        mas[g] = 1;
                    else
                        mas[g] = 0;
```

```
                g++;

            }
        }
        return mas;
    }

    public static List<double[]> FileRead(List<string> path_letters)
    {
        List<double[]> list_intput_letters = new List<double[]>();

        for (int i = 0; i < path_letters.Count(); i++)
        {
            Bitmap bitmap = new Bitmap(path_letters[i]);
            list_intput_letters.Add(ConvertBimap(bitmap));
        }

        return list_intput_letters;
    }
  }
}
```

Лістинг Neuron.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Fingerprint_recognition
{
    class Neuron
    {
        public double[] weightArray;
        public int stateStydu { get; set; }
        public double neuronValue { get; set; }
        public static double saturationFactor = 0.002f;
        //    double saturationFactor;
        public Neuron(double[] arr_x)
        {
            weightArray = new double[arr_x.Count()];
            Array.Copy(CreateWeigh(weightArray.Count()), weightArray,
weightArray.Count());

            neuronValue = activationFunk(Sum(arr_x, weightArray));

        }

        public double activationFunk(double sum)
        {
            return (double)(1 / (1.0 + Math.Pow(Math.E, -sum * saturationFactor)));
        }
```

```
        public double Sum(double[] array_x, double[] array_w)
        {
            double sum = 0;

            for (int i = 0; i < array_x.Count(); i++)
                sum += array_x[i] * array_w[i];

            return sum;
        }

        public double[] CreateWeigh(int Length)
        {
            double[] arr = new double[Length];
            Random ran = new Random();
            for (int i = 0; i < arr.Count(); i++)
                arr[i] = (double)ran.NextDouble();

            return arr;
        }

        public void ReturnWeigh(double[] arr_x)
        {
            neuronValue = activationFunk(Sum(arr_x, weightArray));
        }

    }
}
```

Лістинг Form1.cs:

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

namespace Fingerprint_recognition
{
    public partial class Form1 : Form
    {
        List<string> path_letters = new List<string>();
        Study study;

        public Form1()
        {
            InitializeComponent();
        }

        private void button23_Click(object sender, EventArgs e)
```

```
        {
            if (Convert.ToInt32(Counter.Text) > 1)
            {
                Counter.Text = Convert.ToString((Convert.ToInt32(Counter.Text) - 1));
                Clean(path_letters[Convert.ToInt32(Counter.Text) - 1]);
            }
        }

        private void button22_Click(object sender, EventArgs e)
        {
            if (path_letters.Count() == Convert.ToInt32(Counter.Text))
            {
                Counter.Text = Convert.ToString((Convert.ToInt32(Counter.Text) + 1));
                Clean("");
            }
            else if (path_letters.Count() > Convert.ToInt32(Counter.Text))
            {
                Clean(path_letters[Convert.ToInt32(Counter.Text)]);
                Counter.Text = Convert.ToString((Convert.ToInt32(Counter.Text) + 1));
            }
            else
            {
                MessageBox.Show("Для переходу до наступного образу необхідно запам'ятати
поточний зразок");
                return;
            }
        }

        private void button24_Click(object sender, EventArgs e)
        {
            OpenFileDialog openFileDialog1 = new OpenFileDialog() { Filter = "Файл
зображення (*.bmp)|*.bmp" };

            string directory = @"::{20D04FE0-3AEA-1069-A2D8-08002B30309D}";

            openFileDialog1.InitialDirectory = directory;


            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                directory = openFileDialog1.FileName;
                FullPath.Text = directory;
                pictureBox11.Image = new Bitmap(directory);
                directory = directory.Substring(directory.LastIndexOf(@"\") + 1);
                label4.Text = directory.Substring(0, directory.Length);
            }



        }

        private void button15_Click(object sender, EventArgs e)
        {
```

```
        }


        private async void button10_Click(object sender, EventArgs e)
        {
            label22.Visible = false;
            label17.Visible = false;
            label2.Visible = false;
            if (path_letters.Count == 0)
            {
                MessageBox.Show("Необхідно додати образ для навчання");
                return;
            }

            bool result = double.TryParse(textBox9.Text, out Study.coefTraining);
            bool result1 = double.TryParse(textBox8.Text, out Study.correction_factor);
            bool result2 = double.TryParse(textBox7.Text, out Neuron.saturationFactor);
            bool result3 = double.TryParse(textBox1.Text, out Study.min);
            bool result4 = double.TryParse(textBox2.Text, out Study.max);
            if (result == false || result1 == false || result2 == false)
            {
                MessageBox.Show("Некоректні значення коефіцієнтів");
                return;
            }
            pictureBox1.Visible = true;
            label1.Visible = true;

            Stopwatch stwatch = new Stopwatch();

            stwatch.Start();

            study = new Study(path_letters);
            await Task.Run(() => study.Teaching());
            stwatch.Stop();


            label18.Visible = true;
            label17.Visible = true;
            label22.Visible = true;
            label21.Visible = true;

            textBox9.Text = Convert.ToString(Study.coefTraining);
            label22.Text = Convert.ToString(stwatch.ElapsedMilliseconds / 1000.0);
            label17.Text = Convert.ToString(study.iterations /
study.list_incoming_arr.Count());
            pictureBox2.Visible = true;
            pictureBox1.Visible = false;
            label1.Visible = true;
            label1.Text = Convert.ToString("Мережа навчена");

        }

        private void button14_Click(object sender, EventArgs e)
        {
```

```
            OpenFileDialog openFileDialog1 = new OpenFileDialog() { Filter = "Файл
зображення (*.bmp)|*.bmp" };

            string directory = @"::{20D04FE0-3AEA-1069-A2D8-08002B30309D}";

            openFileDialog1.InitialDirectory = directory;


            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                directory = openFileDialog1.FileName;
                Result.Text = directory;
                pictureBox8.Image = new Bitmap(directory);
                directory = directory.Substring(directory.LastIndexOf(@"\") + 1);
                label8.Text = directory.Substring(0, directory.Length);
            }
        }

        private void button13_Click(object sender, EventArgs e)
        {
            dataGridView2.Rows.Clear();




            dataGridView2.DefaultCellStyle.SelectionBackColor =
dataGridView2.DefaultCellStyle.BackColor;
            dataGridView2.DefaultCellStyle.SelectionForeColor =
dataGridView2.DefaultCellStyle.ForeColor;

            if (pictureBox8.Image != null && study != null)
            {
                study.Recognition(Result.Text, dataGridView2, path_letters);
            }
            else
            {
                if (study == null)
                    MessageBox.Show("Необхідно навчити нейронну мережу");
                else
                    MessageBox.Show("Необхідно відкрити файл для розпізнавання");
            }
        }

        private void label22_Click(object sender, EventArgs e)
        {

        }

        private void label17_Click(object sender, EventArgs e)
        {

        }
```

```
        private void button1_Click(object sender, EventArgs e)
        {
            if (FullPath.Text == "")
            {
                MessageBox.Show("Необхідно завантажити образ");
                return;
            }

            if (path_letters.Count < Convert.ToInt32(Counter.Text))
            {
                path_letters.Add(FullPath.Text);
                Counter.Text = Convert.ToString((Convert.ToInt32(Counter.Text) + 1));
                pictureBox11.Image = null;
                FullPath.Text = "";
                label4.Text = "";
            }
            else
                path_letters[Convert.ToInt32(Counter.Text) - 1] = FullPath.Text;
        }

        void Clean(string path)
        {
            if (path != "")
            {
                pictureBox11.Image = new Bitmap(path);
                FullPath.Text = path;
                path = path.Substring(path.LastIndexOf(@"\") + 1);
                label4.Text = path;
            }
            else
            {
                label4.Text = "";
                pictureBox11.Image = null;
                FullPath.Text = "";
            }
        }

        private void label25_Click(object sender, EventArgs e)
        {

        }
    }
}
```

Додаток В

Презентаційний матеріал

Кваліфікаційна робота бакалавра

Система розпізнавання образів перцептроном із стохастичним алгоритмом навчання

Виконав: студент 4 курсу групи КН-17-1 Пітик Я.О.
Керівник: к.т.н., доцент кафедри КНІТ Мазурець О.В.

Актуальність

В сучасному світі технологій розпізнавання образів набирає популярності. Задачі розпізнавання образів вирішуються досить часто, наприклад, сигнали світлофора, при переході або переїзді вулиці. Отже, проблема розпізнавання образів на сучасному етапі досить популярна. Одним з прикладів використання розпізнавання образів є розпізнавання відбитків пальців, що дозволяє провести ідентифікацію людини

Завдання

Метою кваліфікаційної роботи бакалавра є розробка системи розпізнавання образів, яка використовує нейронну мережу перцептрон із стохастичним алгоритмом навчання.

Навчання нейронної мережі перцептрон, результатом якого є сформована множина ваг синапсів для розпізнавання, слід реалізувати двома наступними алгоритмами:

– класичний алгоритм навчання нейронної мережі перцептрон зворотного поширення помилки;

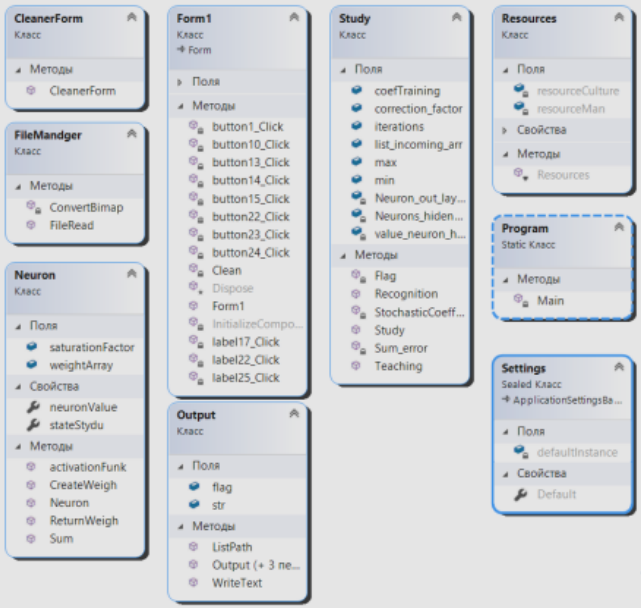
– стохастичний алгоритм навчання нейронної мережі перцептрон, що містить складову випадкового підбору параметрів для кожної ітерації.

Інформаційна технологія розпізнавання образів з використанням стохастичного алгоритму навчання

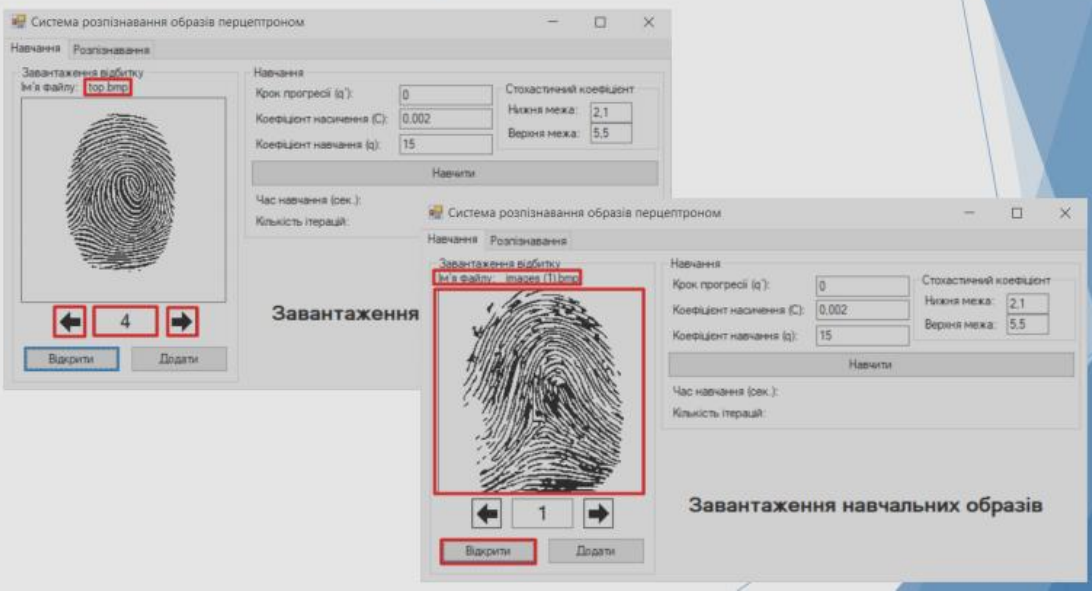
Класичний алгоритм зворотного поширення помилки

Стохастичний алгоритм навчання перцептрону

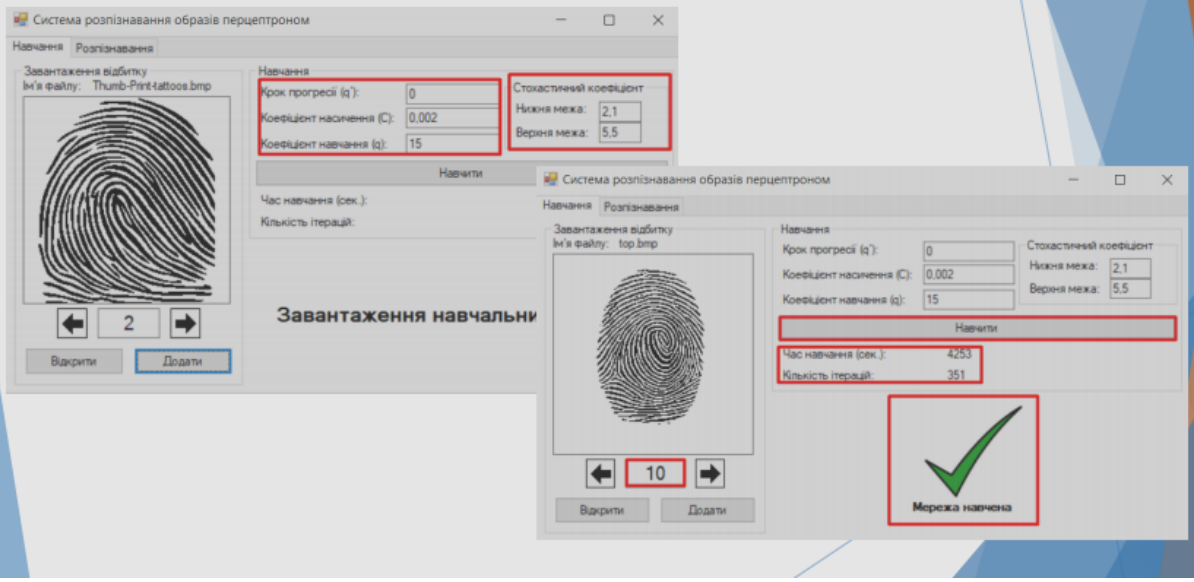
Діаграма класів



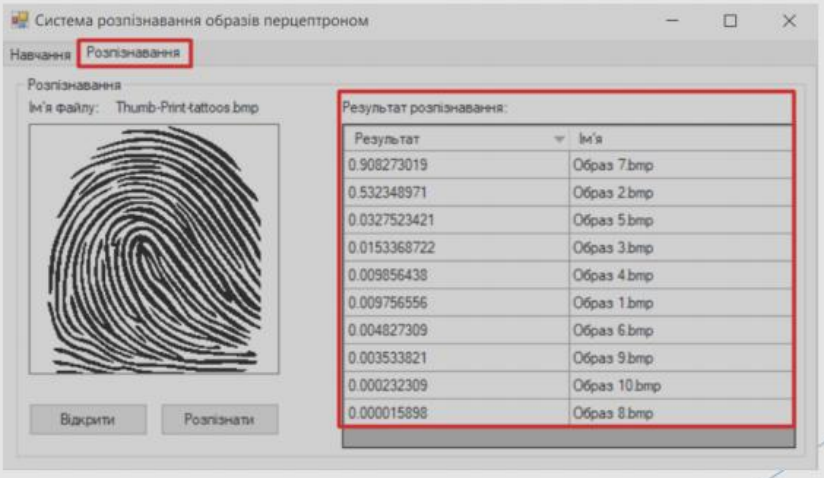
Формування списку навчальних образів



Навчання нейронної мережі



Розпізнавання тестового зображення



Висновки

В результаті виконання даної кваліфікаційної роботи бакалавра на платформі .NET розроблено систему розпізнавання образів, яка використовує нейронну мережу перцептрон. Для програмної реалізації системи розпізнавання образів було використовувано середовище програмування Visual Studio і мову програмування C#.

У якості навчальних і тестових зразків системи розпізнавання образів використано відбитки пальців рук людей, відповідно прикладна інформаційна система позиціонується як систему розпізнавання осіб за відбитками пальців їх рук. Це визначає дактилоскопію як область прикладного застосування розробленої системи розпізнавання образів.

Використаний стохастичний алгоритм навчання нейронної мережі містить складову випадкового підбору параметрів для кожної ітерації. Реалізація обох алгоритмів навчання перцептрона у системі розпізнавання образів дозволить у подальшому виконати порівняння їх ефективності.

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ГОЛОВІ ЕКЗАМЕНАЦІЙНОЇ КОМІСІЇ

Направляється студент Пітик Я. О. на захист дипломного проекту (роботи)
(прізвище, ініціали)

за спеціальністю 122 - Комп'ютерні науки

На тему: Система розпізнавання образів перцептроном із стохастичним алгоритмом навчання

Дипломний проект (робота), рецензія і довідка про перевірку на плагіат додаються.

Декан факультету _____ САВЕНКО О. С.,
(підпис) (прізвище та ініціали)

ДОВІДКА УСПІШНОСТІ

Пітик Я. О. за період навчання на факультеті програмування та компютерних і телекомунікаційних систем з 2017 по 2021 роки. повністю виконав навчальний план спеціальності з такими розподілом оцінок за:
національною шкалою: відмінно 81,25 %, добре 18,75 %, задовільно 0,00 %.
шкалою ЄКТС: A 83,64 %, B 9,09 %, C 5,45 %, D 0,00 %, E 1,82 %.

Методист факультету _____ _____
(підпис) (прізвище та ініціали)

ВИСНОВОК КЕРІВНИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ) ТА ОБГРУНТУВАННЯ ОЦІНКИ

Студент Пітик Я. О. при виконанні кваліфікаційної роботи вчасно й на високому фаховому рівні виконав всі необхідні етапи. Робота має вагому інноваційну та дослідницьку складову. Є перспективи подальших імплементувань та практичного використання як за прямим призначенням системи, так і в інших галузях.

Оцінка дипломного проекту (роботи) відмінно

Керівник дипломного прокту (роботи) _____ Шупенко О. В.
(підпис) (прізвище та ініціали)

"08" 06 2021 р.

ВИСНОВОК КАФЕДРИ ПРО ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ)

Дипломний проект (роботу) розглянуто. Студент Пітик Я. О. допускається до захисту цього проекту

Завідувач кафедри КНІТ
(назва)

_____ "08" 06 2021 р.
(підпис, прізвище, ініціали)

Thu Jun 10 17:32:38 EEST 2021, Петровський Сергій Степанович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальное совпадение с одним документом 7.0%

Словари проверки: en_US, ru_RU, ua_UA. **Ошибок в документах: 12%**

	Документ		Суммарное совпадение по Базе Данных	
ID: 93199 Название: Система розпізнавання образів перцептроном із стохастичним алгоритмом навчання Добавлено в БД: 2021-06-10 Авторы: Я.О. Пітик Руководители: О.В. Мазурець Консультанты: Опоненты:	Символы	Лексемы	Символы	Лексемы
	61826	516	8274 (13%)	95 (18%)

Источник плагиата

ID	Описание	Наличие плагиата в документе	
		Символы	Лексемы

UNICHECK

Ім'я користувача:
Кафедра КН

ID перевірки:
1008262473

Дата перевірки:
10.06.2021 18:48:25 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
10.06.2021 18:51:11 EEST

ID користувача:
100005671

Назва документа: **КРБ Пітик 20210610 3 фінал Lite**

Кількість сторінок: **28** Кількість слів: **8531** Кількість символів: **67363** Розмір файлу: **3.62 MB** ID файлу: **1008333395**

12.9%
Схожість

Найбільша схожість: **2.5%** з джерелом з Бібліотеки (ID файлу: **1008318499**)

10.1% Джерела з Інтернету	161	Сторінка 30
4.42% Джерела з Бібліотеки	58	Сторінка 32

0% Цитат

Включення цитат вимкнене

Включення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	2

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: *Система розпізнавання образів перцептроном із стохастичним алгоритмом навчання*

Автор: *студент групи КН-17-1 Пітик Ярослав Олександрович*

Спеціальність: *122 – Комп'ютерні науки*

Освітня програма: *освітньо-професійна*

Науковий керівник: *доцент кафедри КНІТ Мазурець О.В.*

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	*відповідає*
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;

2) усі запозичення фрагментарні;

3) до запозичень входять фрагменти програмного коду, що на мають авторства і містять поширені конструкції;

4) серед запозичень знаходяться загальновідомі терміни, скорочення та визначення.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості, складає 12.9 % і адресується до першоджерел, що, з урахуванням наведених обґрунтувань, відповідає характеру дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ О. В. Мазурець

Гарант ОП _____ О. В. Мазурець

Завідувач кафедри КНІТ _____ О. В. Бармак

Хмельницький національний університет
Кафедра комп'ютерних наук та інформаційних технологій

РЕЦЕНЗІЯ

на кваліфікаційну роботу бакалавра

студента *гр. КН-17-1 Пітика Ярослава Олександровича*

за темою *Система розпізнавання образів перцептроном із стохастичним алгоритмом навчання*

1. Актуальність і значення теми *Важливість розпізнавання відбитків пальців полягає в ідентифікації людини в тих чи інших ситуаціях. Власне розпізнання зазвичай базується на скануванні унікальних виступів на кінчиках пальців людини. Оскільки задачі розпізнавання вирішуються за допомогою штучного інтелекту, то розробка системи розпізнавання образів, що використовує нейронну мережу перцептрон для розпізнавання відбитків пальців є досить актуальною. Стохастичний алгоритм навчання перцептрона може зменшити час на навчання та уникнути критичних випадків при навчанні.*

2. Оцінка запропонованих моделей, підходів, алгоритмів, інформаційної складової та засобів розробки *Розроблені автором схема і компоненти інформаційної технології розпізнавання образів дозволяють проводити розпізнавання відбитків пальців використовуючи при цьому нейронну мережу персептрон, яка може навчатися за детерміністським та стохастичним алгоритмом. Реалізація обох алгоритмів навчання перцептрона у системі розпізнавання образів дозволить у подальшому виконати порівняння їх ефективності.*

3. Оцінка розробленої інформаційної системи, її практична цінність та економічна доцільність *У якості навчальних і тестових зразків системи розпізнавання образів використано відбитки пальців рук людей, відповідно прикладна інформаційна система позиціонується як система розпізнавання осіб за відбитками пальців їх рук. Тому дактилоскопія – область прикладного застосування розробленої інформаційної системи.*

4. Загальний висновок та оцінка *Всі поставлені завдання виконані повністю і на високому професійному рівні. За своєю структурою, практичними цінностями, поставленій меті та вирішеними завданнями робота відповідає вимогам вищої школи і вимогам, що пред'являються до освітньо-кваліфікаційного рівня «бакалавр», а її автор Пітик Я.О. заслуговує присвоєння кваліфікації бакалавра з комп'ютерних наук.*

Робота заслуговує на оцінку «відмінно».

Рецензент к.т.н., доц. Медведчук І.К.