Міністерство освіти і науки України

Одеський національний політехнічний університет

Навчально-науковий інститут комп’ютерних систем

Кафедра системного програмного забезпечення

*Невзоров Володимир Дмитрович*,

студент групи АС-151

**КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА**

*Програма для автентифікації рукописного підпису з використанням нейромережі*

Спеціальність:

121 – Інженерія програмного забезпечення

Спеціалізація:

Інженерія програмного забезпечення

Керівник:

*Комлева Наталія Олегівна,*

*канд. техн. наук, доцент*

Одеса – 2020

**ЗМІСТ**

|  |  |
| --- | --- |
| ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ | 5 |
| ЗАВДАННЯ НА РОЗРОБКУ РОЗДІЛУ «ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ» | 7 |
| АНОТАЦІЯ | 8 |
| ВСТУП | 9 |
| РОЗДІЛ 1 КРИТИЧНИЙ АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ДЛЯ РОЗПІЗНАВАННЯ ПІДПИСУ | 12 |
| 1.1Аналіз особливостей розпізнавання рукописного підпису | 12 |
| 1.2 Використання нейромережі в задачах розпізнавання | 14 |
| 1.3 Огляд існуючих програмних аналогів | 17 |
| РОЗДІЛ 2 РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ РОЗПІЗНАВАННЯ ПІДПИСУ | 20 |
| 2.1 Збір та розпізнавання рукописних підписів | 20 |
| 2.1.1 Словесний алгоритм збору та розпізнавання | 20 |
| 2.1.2 Використання порогової бінаризації | 22 |
| 2.1.3 Алгоритм скелетизації | 23 |
| 2.1.4 Використання вектору ознак | 23 |
| 2.2 Застосування нейронної мережі для розпізнавання | 25 |
| 2.3 Принцип верифікації результатів роботи нейромережі | 26 |
| РОЗДІЛ 3 СПЕЦИФІКАЦІЯ ВИМОГ ДО ПРОГРАМИ | 30 |
| 3.1 Варіанти використання системи | 30 |
| 3.2 Вимоги до нефункціональних характеристик | 34 |
| 3.3 Системні вимоги | 36 |
| РОЗДІЛ 4 ПРОЕКТУВАННЯ ПРОГРАМИ ДЛЯ АВТЕНТИФІКАЦІЇ РУКОПИСНОГО ПІДПИСУ | 37 |
| 4.1 Проєктування архітектури системи | 37 |
| 4.2 Основні алгоритми в системі класифікації | 41 |
| 4.3 Основні алгоритми в системі навчання | 41 |
| 4.4 Проектування структури та організації класів | 44 |
| 4.4.1Організація класів для модуля класифікації | 44 |
| 4.4.2 Організація класів для модуля навчання | 46 |
| РОЗДІЛ 5 ПРОГРАМНА РЕАЛІЗАЦІЯ РОЗРОБЛЮВАНОЇ СИСТЕМИ | 49 |
| 5.1 Особливості створення програмних модулів з урахуванням мови програмування | 49 |
| 5.2 Розробка класифікаторів на основі нейронних мереж | 54 |
| 5.2.1 Опис принципу singleton класифікатора | 54 |
| 5.2.2 Класифікація за допомогою дерева рішень | 55 |
| 5.2.3 Класифікація за допомогою мажоритарної функції | 56 |
| 5.2.4 Пошук оптимальних налаштувань для кожного класифікатора | 56 |
| 5.3 Алгоритм донавчання нейронної мережі | 57 |
| 5.4 Структури даних в розроблюваній системі | 58 |
| 5.5 Особливості підключення бібліотеки реалізації нейронних мереж до програмної системи | 60 |
| РОЗДІЛ 6 ВИЗНАЧЕННЯ ВЛАСТИВОСТЕЙ ПРОГРАМИ SIGNATURE SCAN TOOL | 62 |
| 6.1 Тестування моделі класифікатора | 62 |
| 6.1.1 Тестування єдиної нейронної мережі | 63 |
| 6.1.2 Тестування глибокого дерева рішень | 63 |
| 6.1.3 Тестування широкого дерева рішень | 64 |
| 6.2 Тестування системи з використанням функціональних діаграм | 64 |
| 6.3 Приклади використання програмної системи | 67 |
| РОЗДІЛ 7 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ  СИТУАЦІЯХ | 71 |
| ВИСНОВКИ | 73 |
| СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ | 74 |
| Додаток А. МАТЕРІАЛИ ТЕЗИСІВ | 76 |
| Додаток Б. ЛІСТИНГ ПРОГРАМИ | 79 |
| Додаток В. ПИТАННЯ З ОХОРОНИ ПРАЦІ ТА БЕЗПЕКИ У  НАДЗВИЧАЙНИХ СИТУАЦІЯХ | 93 |
| В.1 Організація та управління охороною праці у фірми розробника програмного забезпечення для необслуговуємих приміщень | 93 |
| В.2 Обґрунтування заходів з покращення умов охорони праці | 94 |
| В.3 Індивідуальне завдання. Розрахунок захисного занулення | 96 |
| В.4 Надзвичайні ситуації та шляхи їх запобігання | 97 |
| В.5 Індивідуальне завдання. Безпека в надзвичайної ситуації радиоактивного забруднення | 99 |
| В.6 Заходи з охорони праці та безпеки у надзвичайних ситуаціях | 101 |

Міністерство освіти і науки України

Одеський національний політехнічний університет

Навчально-науковий інститут комп’ютерних систем

Кафедра системного програмного забезпечення

Рівень вищої освіти: другий (магістерський)

Спеціальність: 121 – Інженерія програмного забезпечення

Спеціалізація: Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Крісілов В.А.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ р.

**ЗАВДАННЯ**

**НА КВАЛІФІКАЦІЙНУ РОБОТУ**

*Невзорова Володимира Дмитровича*, група АС-151

1. Тема роботи: *Програма для автентифікації рукописного підпису з використанням нейромережі*

Керівник роботи: *Комлева Наталія Олегівна, канд. техн. наук, доцент*

затверджені наказом ректора від «29»жовтня 2020 р. № 412-в.

1. Зміст роботи: проведення критичного аналізу існуючих рішень для розпізнавання та автентифікації рукописного підпису з використанням нейромережі; розробка засобу для розпізнавання, вибір алгоритмів; специфікація вимог до програми; проектування програмного засобу; програмна реалізація та визначення властивостей засобу для розпізнавання; заходи з охорони праці та безпеки у надзвичайних ситуаціях.
2. Перелік ілюстративного матеріалу: згідно зі слайдами презентації.
3. Консультанти розділів роботи

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
| завдання видав | завдання прийняв |
| Охорона праці | доц. Москалюк А.Ю. |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. Дата видачі завдання: «\_02\_\_» \_\_\_\_\_\_09\_\_\_\_\_\_\_\_ 2020 \_\_\_\_ р.

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| № з/п | Назва етапів кваліфікаційної роботи | Строк виконання | Примітка |
| 1 | Аналіз існуючих рішень | 2.09.2020 – 16.09.2020 | вик. |
| 2 | Специфікація вимог до програми | 17.09.2020 – 30.09.2020 | вик. |
| 3 | Проєктування програмної системи | 01.10.2020 – 17.10.2020 | вик. |
| 4 | Програмна реалізація та тестування системи | 18.10.2020 – 11.11.2020 | вик. |
| 5 | Завдання з охорони праці та БНС | 14.11.2020 – 28.11.2020 | вик. |
| 6 | Оформлення пояснювальної записки та графічного матеріалу | 29.11.2020 – 10.12.2020 | вик. |

**Здобувач вищої освіти** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *В.Д. Невзоров*

**Керівник роботи** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *Н. О. Комлева*

**ЗАВДАННЯ**

на розробку розділу «Охорона праці та безпека в надзвичайних ситуаціях»

*Невзорова Володимира Дмитровича*, група АС-151

Навчально-науковий інститут комп’ютерних систем

Кафедра системного програмного забезпечення

Тема роботи: *Програма для автентифікації рукописного підпису з використанням нейромережі*

Зміст розділу:

1. Аналіз умов праці і вибір заходів і засобів захисту від небезпечних і шкідливих виробничих факторів.
2. Аналіз техногенних небезпек і вибір заходів і засобів забезпечення безпеки у надзвичайних ситуаціях
3. Індивідуальне завдання

Керівник роботи Консультант з охорони праці та БНЗ

\_\_\_\_\_\_\_\_\_\_\_\_\_ *Н. О. Комлева* \_\_\_\_\_\_\_\_\_\_\_\_\_\_ *А.Ю. Москалюк*

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 20 \_\_\_ р. «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 20 \_\_\_ р.

# АНОТАЦІЯ

Метою роботи є підвищення точності розпізнавання рукописного підпису завдяки адаптації методу розпізнавання до кінцевого користувача з використанням нейромережі. Використано методи цифрової обробки зображень, аналітичної та обчислювальної геометрії для виявлення особливостей рукописного підпису. Методи розробки базуються на технологіях Java, C++ з використанням FANN та JNI, XML. Як результат роботи розроблено програмний засіб, що дозволяє виконувати автентифікацію рукописного підпису.

Ключові слова: рукописний підпис, розпізнавання, нейромережа, цифрова обробка зображень, Android, Java, C++, FANN, JNI, XML.

**ABSTRACT**

The aim of the work is to improve the quality of handwriting recognition through adaptation of the recognition method to the end user using a neural network. Methods of digital image processing, analytical and computational geometry were used to identify the features of text caption depending on the degree of human emotional state. Development methods are based on Java, C ++ technologies using FANN and JNI, XML. As a result, a software tool has been developed that allows handwritten signature authentication.

Keywords: handwritten signature, recognition, neural network, digital image processing, Android, Java, C++, FANN, JNI, XML.

**ВСТУП**

Розпізнавання рукописного підпису – проблема, яка з’явилася кілька багато тому і досі залишається актуальною. Почерк у різних людей сильно різниться, тому складно створити програму, яка може розпізнати будь-який із рукописних підписів. На даний момент досягнута точність розпізнавання рукописних букв та окремих елементів у підпису нижча, ніж для літер друкованого тексту [1].

Багато завдань розпізнавання особливостей напису рукописного тексту можна звести до розпізнавання фігур – це особливий випадок розпізнавання зображень. Існуючі методи представлення фігур можна класифікувати наступним чином: це методи, засновані на контурі, та методи, засновані на домені, просторовій області та домені перетворення; інформаційно-збережуючі та інформаційно-незбережуючі методи [2, 3].

Для кожної людини при написанні підпису можна виявити деякі унікальні характеристики. Багато дослідженнь припускають вибір оптимального способу порівняння двох варіантів рукописного підпису для певної людини. Існує велика кількість алгоритмів для виявлення різних ознак та характеристик підпису і подальшої автентифікації. Але тому, що різні алгоритми відображають різні особливості підпису, у загальному випадку важко порівнювати між собою ці алгоритми. Все вищесказане показує *доцільність та актуальність* розробки програмного засобу, що дозволяє виконувати автентифікацію рукописного підпису з високим рівнем точності.

*Метою роботи* є підвищення точності розпізнавання та автентифікації рукописного підпису.

Для досягнення мети були поставлені і вирішені *наступні завдання*:

* дослідження і аналіз існуючих методів щодо ефективного розпізнавання рукописного тексту та підписів;
* дослідження застосування нейронних мереж в задачах розпізнавання;
* вибір методу розпізнавання рукописного підпису;
* розробка алгоритмів попередньої обробки і розпізнавання підпису;
* програмна реалізація алгоритмів;
* тестування програмної системи.

Для досягнення поставленої мети було застосовано наступні *методи досліджень*:

* методи розпізнавання з використанням нейромережі;
* методи цифрової обробки зображень;
* методи аналітичної та обчислювальної геометрії.

Означені *результати опубліковано* у збірнику доповідей XIІI міжнародної науково-практичної конференція «Інформаційні технології і автоматизація – 2020» [4].

*Практична цінність одержаних результатів*. Розроблено мобільний програмний Android-засіб для розпізнавання та автентифікації рукописного підпису. Точність розпізнавання склала 93.28% при при застосуванні широкого дерева рішень.

Даний звіт є результатом виконання дипломного проектування. Він містить вступ, сім розділів, висновки, список літератури та три додатки.

У першому розділі наведено опис предметної області та проаналізовано особливості розпізнавання рукописного підпису. Виконано критичний аналіз існуючих засобів для розпізнавання рукописного підпису.

У другому розділі наведено опис методу для розпізнавання та автентифікації рукописного підпису, що містить сукупність алгоритмів обробки зображень підпису, в тому числі порогову бінеризацію, скелетизацію та отримання вектора ознак. Проанализовано особливості застосування різних структур нейромережі для задач розпізнавання, обрано відповідну структуру нейромережі.

Третій розділ містить специфікацію вимог до програмного продукту. Детально описано функціонал програми для автентифікації рукописного підпису. опис функціоналу програмного засобу. Наведено вимоги до нефункціональних характеристик.

Четвертий розділ присвячений проєктуванню програми. Визначено архітектуру системи, описано основні алгоритми для навчання та класифікації. Виконано проектування структури та організації класів.

П’ятий розділ містить опис програмної реалізації системи з визначенням особливостей створення програмних модулів з урахуванням обраних технологій та мови програмування.

У шостому розділі наведено експериментальне визначення властивостей розроблюваного програмного засобу. Проведено тестування функціональності та визначення точності автентифікації рукописного підпису.

Сьомий розділ містить опис заходів з охорони праці та безпеки у надзвичайних ситуаціях.

### **КРИТИЧНИЙ АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ДЛЯ РОЗПІЗНАВАННЯ ПІДПИСУ**

**1.1 Аналіз особливостей розпізнавання рукописного підпису**

Рукописний підпис з давніх часів використовується для підтвердження авторства створених документів чи інших паперів, або для підтвердження згоди з їх змістом. Рукописний підпис може містити окремі рукописні літери, що зазвичай співпадають з першими буквами імені та / чи прізвища, та окремі графічні елементи, які можуть накладатися на літери та перетинати їх. Літери можуть бути спеціальним чином спотворені за бажанням автора підпису (підписанта). У загальному випадку літери можуть бути відсутні зовсім, тому доцільніше розглядати рукописний підпис як зображення, створене з ліній певного кольору на фоні певної якості.

Процес отримання підпису залежить вид класу задачі. Підпис, отриманий онлайн, легше розпізнати тому, що реалізована на основі певного методу система може використовувати додаткові ознаки підпису, такі як: швидкість створення певних елементів, силу натиску та ін. Якщо система дозволяє графічно інтерпретувати силу натиску на стілус як товщину лінії, то це дозволяє зробити візуальний підпис більш похожим на його оригінал. Найбільш зручний спосіб онлайн-створення рукописного підпису – це використання стілусу на планшеті, смартфоні чи спеціальному сенсорному екрані.

При онлайн-роботі з підписом ніякі дані щодо процесу його створення недоступні. Даними для розпізнавання та автентифікації підпису є тільки статичне зображення з його графічними елементами. Це зображення може бути також введено за допомогою стілусу на сенсорному екрані, але найчастіше зображення рукописного підпису читається з графічного файлу типу jpeg, png, bmp та ін. Цей файл може бути отриманий різними засобами, найпростішим є фотографування паперового носія з підписом [5].

Будь-які графічні зображення можна подавати чи як сукупність точок, чи як сукупність векторів. Точки, або пікселі, утворюють растрове зображення, у якому кожний піксель містить розподіл кольорів RGB. При роботі з підписами робота з відтінками кольорів є непріоритетною. Зазвичай розпізнаванню кольорового зображення передує перетворення його до монохромного вигляду у відтінках сірого.

У векторних зображеннях елементами є опорні точки та відповідні криві. При цьому у зображення фактично покладені певні математичні співвідношення та формули, обчислення яких потребує додаткового часу і обчислювальних потужностей.

Формально завдання розпізнавання та автентифікація рукописних підписів можна сформулювати так. Потрібно знайти мінімальну розбіжність між шаблоном для підпису та поточним поданням зображення, представленим у вигляді набору ознак. З урахуванням мінімально припустимої невідповідності поточного зображення шаблону підпису розпізнане зображення підпису має бути індексовано.

Розглянемо основні засоби реалізації цієї функціональності.

Для аналізу та розпізнавання рукописних підписів потрібні два процеси. Перший процес – це навчити систему особливостям кожного з підписів, які ви хочете розпізнати. Другий процес полягає у використанні інформації про набір підписів, наявних у системі, та порівнянні визначення того, чи можна визначити новий підпис як певний із заданого набору шаблонів, чи він взагалі не належить йому. Кількісний критерій успішного завершення цього етапу – це величина ймовірності, з якою розпізнається новий підпис. На обох цих етапах доцільно використовувати нейронну мережу [6].

Для навчання системи розпізнаванню рукописних підписів потрібно, по-перше, отримати навчальні зразки (шаблони) підпису від конкретних підписантів, тобто тих, для кого потім буде виконуватись автентифікація. Далі треба розглянути усі особливості підписів та підібрати набор локальних параметрів, що описують усі характерні особливості. Далі потрібно вивчити загальні умови, що впливають на якість усіх зображень зібраних підписів, та сформувати множину глобальних параметрів. Значення локальних та глобальних параметрів повинні бути пронормовані з метою виключення впливу зовнішніх факторів на різних умов отримання підписів на результат навчання системи та подальшої автентифікації підпису.

Для того, щоб розпізнавати підписи, потрібно пройти такі етапи: 1) реєстрація нового підпису; 2) обробка зображення; 3) виділення ознак зображення (структурні ознаки – для літер у підпису, характерні ознаки – для нелітерних елементів, змішаний варіант); 4) перевірка зображення; 5) ідентифікація визнаного підпису.

Щоб збільшити продуктивність, нейронні мережі, що включають багато шарів і нейронів у кожен шар із початковими вагами, повинні бути правильно розроблені.

Найпоширенішим алгоритмом для навчання багатошарових нейронних мереж прямого поширення є алгоритм зворотного поширення. Ось чому таку нейронну мережу часто називають зворотною мережею. Мережа із зворотним розповсюдженням помилок навчається за обмеженою кількістю стандартів, тобто пар моделей, що складаються з наборів вхідних значень джерела та наборів бажаних вихідних значень.

При розпізнаванні рукописних підписів вирішало вимогою є висока точність розпізнавання. Точність може опинитися суттєво нижчою у випадках, коли підписант має високу варіативність у розмірі, нахилу власного підпису, по-різному з’єднує окремі елементи підпису, скорочує підпис, тощо [7]. Ускладнює розпізнавання ситуація, коли елементи підпису мають перетини один з іншим чи взагалі накладаються один на одного. Паперові дефекти, плями, потертості, неоднорідний колір паперу, а також дефекти сканування чи фотографування також знижують точність розпізнавання зображення підпису.

* 1. **Використання нейромережі в задачах розпізнавання**

Першою штучною мережею став перцептрон, який у 1957 році запропонував Розенблатт, та для якого було розроблено електронна машина, що оперувала принципами функціонування механизмів мозку в перекладі на електронно-обчислювальний підхід.

Перцептрон містить 3 типи елементів: це S-елементи – джерела (1), що поставляють певні сигнали, які дорівнюють 0 у стані спокою чи 1 у стані збудження; асоціативні (2) А-елементи, які при перевищенні порогового значення та збудженні формирують сигнал рівний 1; реагуючі R-елементи (3), які складають один з одним зважені дані від елементів попереднього шару та при перевищенні порогу генерують «одиницю» на виході. Ця послідовність забезпечує набір взаємозв’язаних імпульсів між вхідними стимулами та відповідною реакцією як результату роботи мережі. Штучні мережі перцептрони можуть бути класифіковано за трьома типами:

* з одним прихованим обчислювальним шаром (прихований шар виконую перетворення вхідних даних);
* з [пороговою передавальною функцією](https://uk.wikipedia.org/wiki/%D0%A8%D1%82%D1%83%D1%87%D0%BD%D0%B8%D0%B9_%D0%BD%D0%B5%D0%B9%D1%80%D0%BE%D0%BD#%D0%9F%D0%BE%D1%80%D0%BE%D0%B3%D0%BE%D0%B2%D0%B0_%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D0%B2%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D1%96%D1%8F), що визначає вихідне значення нейрону в залежності від значення порогу та сумарного значення вхідних сигналів;
* з [прямим розповсюдженням сигналу](https://uk.wikipedia.org/wiki/%D0%9D%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D0%B0_%D0%BC%D0%B5%D1%80%D0%B5%D0%B6%D0%B0_%D0%BF%D1%80%D1%8F%D0%BC%D0%BE%D0%B3%D0%BE_%D0%BF%D0%BE%D1%88%D0%B8%D1%80%D0%B5%D0%BD%D0%BD%D1%8F), при якому кожний наступний шар приймає значення з попереднього шару мережі та виконує операції з множення та активації.

За термінологією Розенблатта так нейронна мережа є тришаровою, однак пізніше вона була названа одношаровою за кількістю прихованих шарів (рис. 1.1).

Персептрон виконує класифікацію на основі прямої, яка розділяє два класи (категорії) та має рівняння

де х – вектор-функція;

w – ваги її елементів;

b – зміщення, що може бути як позитивним, так і від’ємним.

Виходом персептрону є результат обчислення функції, що пропускає зважену комбінацію вхідних даних через функцію активації, яка забезпечує нелінійність, безперервну диференційованість та монотонність значень. Наведемо її математичний запис у загальному вигляді:

(1.1)

де *w*– вектор вагових коефіцієнтів для вхідних елементів,

*x*– вектор вхідних елементів,

*b*– зсув (позитивний або) від’ємний,

*φ*– певна функція для нелінійної активації елементів мережі.

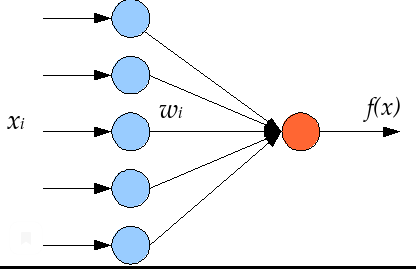


Рисунок 1.1 – Схема порогового елементу

У розроблюваному мобільному програмномузасобі використовується модель багатошарового перцептрона Румельхарта (рис. 1.2).

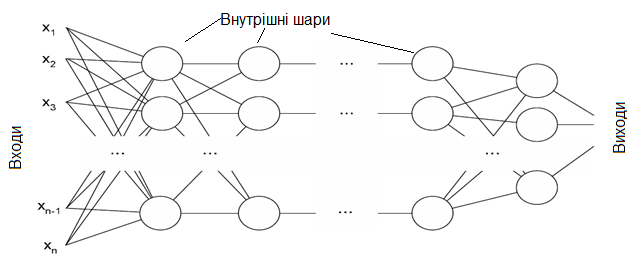


Рисунок 1.2 – Структурна схема перцептрона Румельхарта

Розглянемо чим відрізняється персептрон Румельхарта (багатошаровий) від перцептрона Розенблата (одношарового):

1. У персептроні Румельхарта найчастіше використовується сигмоїдна функція активації
2. Дуже рідко використовується один прихований шар, частіше – до трьох шарів
3. Необхідність мати на вході та виході значення, що лежать у діапазоні від -1 до +1, тому вхідні та вихідні значення потребують нормалізації
4. Можливість використання розрідженої архітектури, при якої з оригінальної повної нейромережі видаляються зв’язки, для навчання таких мереж використовують спеціальні методи, що підвищують ефективність їх використання
5. Для визначення помилки результату обчислюється певна статистична міра, що показує відхилення отриманих результатів від очікуваних
6. Навчання мережі проводиться таким чином, щоб не допустити перенавчання, при цьому отримуються оптимальні значення вагів елементів мережі.

**1.3 Огляд існуючих програмних аналогів**

Розглянемо такі ситуації, які виникають при роботі з рукописним підписом: 1) підпис переважно містить літери алфавіту; 2) підпис переважно містить елементи примітивних зображень.

Для розпізнавання літер алфавіту у відсканованім підписі можна використовувати стандартні утилітні програми. Хорошим прикладом є програма ABBYY FineReader Online, що стабільно працює на ринку вже понад десятку років. Компанія ABBYY FineReader втпускає та оновлює різні утиліти для роботи зі сканованим текстом та зображеннями. Сучасні алгоритми ABBYY FineReader Online дозволяють розпізнати понад 95% відсканованого тексту, що містить окремимі літери. Але цей продукт не можна вважати аналогом, бо наша програма повинна працювати з підписами, що не обмежуються літерами.

У другому випадку можна використовувати програми, які передбачають введення рукописного підпису з певного файлу або за допомогою миші, планшета або переміщення пальця по екрану.

Для мобільних пристроїх з операційною системою Android існує багато програмних сканерів. Кожна програма має свої особливості, але все ж такі обмежені кількістю фільтрів, які можна застосувати для обробки зображення підпису.

Базовою функцією сканер-застосування є отримання фотографічного зображення паперового документу та збереження цього зображення у файлі у пам’яті пристрою чи у хмарному середовищі. Традиційно інтерфейс таких програм не є складним та перевантаженим багатьма функціями [8].

Після отримання фото сканер-застосування пропонує набір інструментів для його редагування – повороту зображення, обрізання за контуром, видалення чи згладжування фону, налаштування кольору, регулювання контрасту і т.д.

Ясно, що чим повніше функціонал сканер-застосування, тим більші вимоги воно потребує. Це стосується як апаратних вимог, так і цінової політики при монетизації програмного продукту. До найбільш цінних можливостей, які надають повноцінні сканер-застосування, відносять високу якість зображення, використання багатосторінкових документів, можливість сканування одночасно набору зображень, розміщення результатів на сервері розробника чи на стандартному сервері і т.п.

Аналіз наведених ресурсів та сервісів наданий у табл. 1.1, де на перетині стовпця і строки стоїть «+», якщо дана програмна система вирішує проблему, і «-», якщо ні. Signature Scan Tool – власний продукт, розроблюваний в рамках дипломної роботи.

Властивості програмних продуктів: 1 – можливість розпізнавання рукописного підпису з файлу; 2 – наявність фільтрів для коригування зображень підписів; 3 – екран вільний від реклами та/або водяних знаків; 4 – безкоштовність або доступна ціна; 5 – можливість ручного вводу підпису за допомогою стілусу або іншого засобу; 6 – автентифікація підпису з використанням бази шаблонів підписів.

Таблиця 1.1 – Порівняльна характеристика програмних продуктів

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Назва продукту | Властивості продукту | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 |
| Document Scanner | + | + | - | - | - | - |
| Genius Signature Scan | + | - | + | - | - | - |
| Scan Master | + | +- | + | + | + | - |
| Handy Scanner | - | + | - | - | + | - |
| Mobile OCR Free | + | + | - | + | - | - |
| Mobile Doc Scanner Lite | - | + | + | + | + | - |
| Signature Scan Tool (власна розробка) | + | + | + | + | + | + |

Проведений аналіз доказує необхідність розробки мобільного Android-застосування для розпізнавання та автентифікації рукописного підпису з високим рівнем точності.

1. **РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ РОЗПІЗНАВАННЯ ПІДПИСУ**

**2.1 Збір та розпізнавання рукописних підписів**

**2.1.1 Словесний алгоритм збору та розпізнавання**

При розгляді рукописного підпису потрібно розглядати їх як зображення, які у загальному випадку можуть містить символи рукописних букв і додаткові графічних елементи-примітивів, які повинні також бути розглянуті. В цьому випадку потрібно виконувати обробку рукописного підпису як графічного зображення, попередньо виконуючи нормалізацію, виділення ознак з зображення таі розпізнавати наданий образ за зразками (шаблонами), які було отримано при навчанні системи.

Розглянемо кроки алгоритму збору та розпізнавання підписів.

1. При створенні бази даних шаблонів (зразків) рукописних підписів потрібно вимагати, щоб вони робились та надавались у в різних варіантах відображення. Доцільно звертати увагу на нахили підпису, відстані між елементами зображення та окремими символами, масштаб зображення підпису. Різноманітність можливих варіантів підписів дає більш повну картину при навчанні та розпізнаванні.

2. При наявності паперу у якості носія рукописного підпису потрібно змінити носій, тобто перевести підпис в електронний вигляд. При цьому кожен 24-бітний bmр-файл (можливі й інші формати), що був отриманий шляхом сканування зображення на білому фоні, містить підпис від певної людини. Якщо підписи вводяться графічно у самій програмі, вимоги щодо їх зберігання ти ж самі.

3. Для задовільнення вимогам розпізнавання, а саме – забезпечення точності – потрібно виконати цифрову обробку отриманого зображення. Зазначимо, що первинне зображення підпису може бути різнокольоровим чи однокольоровим з різними відтінками кольору завдяки застосуванню неякісного засобу для створення підпису. У такому разі:

* зображення переводять у відтінки сірого за наступною формулою:

С = (RЕD + GRЕЕN + BLUЕ) / 3; (2.1)

де С – відсотковий вклад RЕD-, GRЕЕN- та BLUЕ-компонент;

* виконуються бінерізація зображення підпису. Для цього визначається порогове значення яркості та аналізуються значення яркостей пікселів. Якщо яркість якогось пікселя перевищує яркість порогового значення, то він стає білим, а якщо нижче – то чорним;
* для уніфікації обробки підписів вони повинні мати однаковий розмір. Для цього треба змінити масштаб зображення в сторону збільшення чи зменшення. Для цього розглядаються окремі пікселі, що створюють растрове зображення. Потім елементи растрового зображення представляють у вигляді векторів. Кожний вектор має початкову точку, власну довжину та напрямок. Після цього стає можливим обчислення коефіцієнту масштабування, у загальному випадку коефіцієнти по ширині і висоті зображення можуть відрізнятись один від одного. За допомогою отриманих коефіцієнтів обчислюють нові дані векторів. Після цього виконується зворотне перетворення – векторного зображення у растрове;
* зазвичай зображення має поодинокі дефекти, що можуть бути пов’язані з низькою якістю обладнання або ж роботою процедур масштабування: це точкт, що не мають відношення до підпису, «порожки» на лініях, «білі» точки та лінії на елементах, різна товщина ліній. Для коригування цього виконується згладжування зображень елементів підпису, також забезпечується товщина ліній не менше ніж 3 пікселі. Це стане в нагоді при роботі алгоритму виділення контуру елементів зображення підпису;
* далі потрібно виділити контур зображення підпису. Це дозволяє знизити чутливість до його деформацій та мінливостей. При цьому отримують скелет зображення (остов) для зниження чутливості до незначного спотворення і деформацій. Дуже важливим є відкидання тільки тих точок, які не є елементами остова. Якщо видалити такі точки, це сильно модифікує зображення та точність подальшого розпізнавання буде низькою. Тому доцільно залишити остов та видалити межі (контур) графічного об’єкта;
* Для визначення екстремумов контурів графічного об’єкта виконується фіксація особливих точок, екстремуми відповідають кранім лівої, правої, найвищої та найнижчої точкам контуру зображення.

Метою етапу попередньої обробки є максимальне можливе поліпшення якості зображення рукописних підписів, при цьому зображення підписів провидиться до виду, найбільш підходящому для подальшої автоматичної роботи з ними. Поліпшення якості зображення підписів з метою подальшої автентифікації підписів включає в себе видалення дефектів зображення і відділення тексту від фону [9].

У розробленій системі виконується бінаризація та скелетування зображення.

**2.1.2 Використання порогової бінаризації**

Бінаризація – це процес привласнення кожному пікселю зображення рукописного підпису одного з двох можливих значень: поріг яркості не досягнуто або поріг яркості перевищено. Після такого перетворення зображення стає двоколірним [10]. У загальному випадку значення порогу може відноситись до певної області зображення, таким чином, для одного зображення може бути використано декілька порогових значень.

Якщо розглядати чорний колір переднього плану (рукописний підпис) та білий колір для заднього плану (фон), то для зображення підписів пікселі з яркістю менше порога стають білими, а пікселі з яркістю, що перевищують поріг, – чорними. У даній роботі вирішено застосовувати один поріг для усього зображення підпису.

Порогова бінаризація є найбільш простою операцією, яка допомагаю зменшити кількість інформації, що надходить від зображення:

(2.2)

Існує багато методів бінеризації. Коректно проведена бінеризація спрощує подальшу обробку зображення, помилки при проведенні бінеризації, навпаки, спотворюють зображення та знижують подальшу точність його розпізнавання.

**2.1.3 Алгоритм скелетизації**

За допомогою скелетизації можна визначити пікселі (особливі точки), які складають морфологічну основу структури рукописного підпису. Створений скелет повинен однозначно відповідати певному зображенню підпису.

Зазвичай алгоритми скелетизації є ітераційними. Виділяють шаблонну скелетизацію, волновий метод, алгоритм Зонга-Суня та Щепіна. Шаблонний алгоритм заснований нових значеннях для пікселів, які обчислюються на основі значень, що були визначені на попередньому кроці.

Розглянемо 2 підітерації. Розглядаються пікселі як елементи матриці. Зазвичай початкове зображення має розмір 3х3 пікселя.

Поточний піксель видаляється при забезпеченні умов:

1. У нього є принаймні два чорних сусіда і не більше шести сусідів взагалі.

2. Три його діагональні сусіди є білими.

Кроки повторюються багато разів. Вне залежності від кількості ітерацій область зображення рукописного підпису повинна залишатися зв’язною. Алгоритм зупиняє роботу, коли пікселі для видалення відсутні.

**2.1.4 Використання вектору ознак**

Далі вирішується завдання по отриманню вектора ознак. Для цього виконуються такі етапи перетворення зображення рукописного символу:

* для зображення визначаються контури та виділяється існуючий зовнішній контур підпису;
* на цьому зображенні шукається множина опорних точок;
* для всіх отриманих точок будуються вектора та визначаються їх координати;
* виконується побудова моделі на основі отриманих векторів;
* проводиться обробка отриманої моделі, в тому числі фільтрація та нормалізація;
* на основі знань про класи моделей отримана модель відноситься до відповідного класу.

Для визначення контуру зображення доцільно використовувати фільтр Робертса, що є нелінійним та заснований на диференціюванні. Особливою цінністю при застосуванні фільтру Робертса є висока швидкість, яка необхідна при роботі систем у реальному часі. Якщо розглянути сканування зображення зверху вниз та зліва направо, то за допомогою яркості поточного пікселя та сусідніх пікселів це перетворення визначається за формулою:

(2.3)

або

(2.4)

де *A*' – для пікселя i,j – нове значення яркості;

*А* – для пікселя i,j – поточне значення яркості;

*В* – для пікселя i+1,j – поточне значення яркості;

*С* – для пікселя i,j+1 – поточне значення яркості;

*D* – для пікселя i+1,j+1 – поточне значення яркості.

Коли всі опорні точки знайдено, будується векторна модуль, для якої визначається така множина векторів:

Model = {*V*1,*V*2,…*V*i,*V*(*i*+1)…*Vn*}. (2.5)

Усі вектори утворюють замкнутий контур, тобто кінець вектора *i* є одночасно початком вектора *i*+1.

Координати утвореного ветора аналітично визначаються таким чином:

𝑉𝑖 = (𝐿𝑒𝑛𝑖,𝐴𝑙𝑓𝑎𝑖) , (2.6)

де Len – це довжина вектора;

Alfa – це кут між вектором та горизонтом.

Для обчислення у полярних координатах застосовується формула (2.7):

(2.7)

де 𝑥𝑖, 𝑦𝑖– відносні координати вектора 𝑥𝑖 = 𝑥𝑖+1 - 𝑥𝑗, 𝑦𝑖 = 𝑦𝑖+1 - 𝑦𝑗;

𝑎𝑥𝑗, 𝑦𝑖– координати поточної опорної точки, взятої з таблиці;

𝑥𝑖+1, 𝑦𝑖+1– координати наступної опорної точки.

Таким чином, векторна модель вважається отриманою.

**2.2 Застосування нейронної мережі для розпізнавання**

Використання методу розпізнавання за допомогою нейронних мереж відрізняє більш висока ефективність і продуктивність, однак для нього потрібна або велика кількості прикладів при проведенні навчання, або створення спеціальних структур нейронної мережі, які враховують специфіку даних завдань.

Різноманітні методи з застосвуванням нейромереж є ефективними та продуктивними. Їх можливість роботи у режимі реального часу дозволяє використовувати ці методи в рішенні задач розпізнавання та автентифікації підпису. При модифікаціях, що трапляються з підписом у разі зміни умов отримання підпису, або ж фізичного стану підписанту, використовується додаткове навчання нейромережі.

Після проведення початкового навчання штучної нейромережі на векторних моделях рукописних підписів, автентифікація певного підпису відбувається з великою швидкістю, що виявляється важливим фактором для систем, які повинні працювати у режимі реального часу та володіють великими датасетами підписів. Нейромережеві підходи представляють можливість виявлення особливостей змін підписів на підставі нових даних, які введені в систему, з використанням алгоритмів аналізу мінливості структури.

У задачі автоматизованого розпізнавання підпису використання нейронної мережі є доцільним завдяки значній економити ресурсів і технічних витрат. Саме цей метод використаний в даній роботі.

**2.3 Принцип верифікації результатів роботи нейромережі**

Висока складність завдання дослідження підпису визначається наступними факторами:

* підпис короткий, найчастіше простий і малоінформативний;
* може бути скопійований із застосуванням технічних засобів;
* на перевіряючого можуть впливати збиваючи фактори;
* почерки різних людей природним чином бувають схожі;
* підпис людини варіативний.

Для тестування результатів роботи нейромережі потрібно створення бази даних з великої кількості зразків рукописних підписів. В базе даних не повинні бути дублі підписів. Кожен підпис від певного підписанту повинен бути унікальним.

На вхід системи треба подавати дані одного формату та такі, що є однакового розміру (з урахуванням нормалізації). Для забезпечення цих вимог проводиться наступна процедура обробки підпису:

1. Вибір та застосування потрібного фільтра, що усуває поміхи (перешкоди) для очищення зображення.

2. Визначення мінімального необхідного значення для порогу яскравості в залежності від завдання та переведення зображення з кольорового у чорно-біле формат з використанням вказаного порогу.

3. Розташування підпису, що аналізується, по центру, при цьому відступи до полів повинні бути мінімальними.

4. Встановлення фіксованого значення розміру отриманого зображення 40 на 80 пікселів (згідно зі стандартними рекомендаціями).

У ході експерименту потрібно визначати точність (погрішність) роботи системи при навчанні та класифікації. Для можливості конкурентного практичного використання точність розпізнавання не повинна бути меншою ніж 90%. Крім точності, потрібно визначати та оцінювати кількість епох та часу навчання у годинах. Фактично, одна епоха навчання – це один перегляд можливих прикладів навчальної вибірки, при якому проводиться корекція ваг нейромережі в залежності від правильності їх вирішення мережею. Щоб мережа навчилася, може знадобитися кілька (і навіть кілька десятків, і навіть кілька сотень) епох. Проходження епохи навчання на певному прикладі ще не означає, що мережа обов’язково скоригувала ваги мережі: для мережі, яка вже достатньо точно навчилась розпізнавати певний підпис, можна перестати надавати його зразки, а перейти до інших прикладів рукописних підписів, які поки ще вирішуються з недостатньою точністю.

Для розпізнавання рукописних підписів доцільно використовувати машинне навчання. При цьому програма повинна навчатися на великих наборах даних – датасетах, ці набори повинні бути попередньо розмічені. Дуже важливим для роботи алгоритми є релевантність вихідних даних. Наприклад, якщо для навчання програми потрібне певне число рукописних підписів однієї і тієї ж людини, то наявність натомість підписів різних людей призведе до помилкової ситуації.

При роботі з оффлайн-розпізнаванням система може аналізувати тільки вже наявні графічні дані. Важливим є те, щоб дані були отримані при тих же умовах, при яких буде надалі працювати система: тип камери, за допомогою якої отримують зображення рукописного підпису, рівень освітленості, кут нахилу камери при зйомці і т.д. Чим більшим буде подібність між умовами, при яких отримані датасет і реальні дані, що вимагають розпізнавання, тим більш достовірними будуть кінцеві результати.

При формуванні датасета важливо використовувати різні дані для навчання і для верифікації програми. Завжди слід тестувати алгоритм розпізнавання рукописного підпису на нових даних, які йому потрібно буде обробляти при реальній експлуатації, а не на тих даних, за допомогою яких проводилося навчання алгоритму.

Після формування датасета рукописних підписів, необхідно вибрати метрику (або набір метрик), для перевірки (оцінки) результатів.

За визначенням, метрика – це функція, яка:

1) в якості вхідних даних приймає результати роботи відповідного алгоритму – для ідентифікації або верифікації;

2) в якості вихідного результату повертає число, яке для заданого датасета відповідає якості роботи алгоритму.

Верифікація підпису полягає у видачі одного з двох відповідей:

1) підпис відповідає підписанту – нульова гіпотеза;

2) підпис не відповідає підписанту – альтернативна гіпотеза.

Метрики верифікації грунтуються на можливі помилки класифікації, які формалізовані для подібних завдань. З огляду на, що є 2 можливих відповіді алгоритму і 2 варіанти для об'єктивної ситуації, всього можливо 4 результати: «True Positive» – рішення вірно прийнято, «True Negative» – рішення вірно відкинуто, «False Positive» – відкинути гіпотезу Н0, в той час, як вона є вірною (помилка 1 роду), «False Negative» – прийняття альтернативної гіпотези, коли насправді вірна нульова (помилка 2 роду) [11].

Якщо розглядати систему надання доступу (або розширення прав) результат «False Negative» (пропуск цілі) передбачає надання вільного доступу підписанту, для якого така дія не була передбачена. Результат «False Positive» (помилкове спрацьовування, помилкова тривога) означає, що наша система не змогла правильно розпізнати підпис підписувача та відмовила йому в авторизації і надання доступу (або розширення прав). Ці помилки не можна вважати рівнозначними, в різних завданнях з точки зору можливих ризиків і втрат їх «вартість» різна. В задачах ідентифікації рукописного підпису результат «False Positive» означає, що потрібно повторно зажадати підпис у підписанта. Результат «False Negative» може привести до надання доступу до закритих можливостей і / або ресурсів неавторизованій особі, що може завдати шкоди у вигляді спотворення, псування, розголошення даних, втрати фінансових і матеріальних ресурсів і т.д.

Значення TP і FP використовуються для визначення точності роботи алгоритму класифікації:

Precision (точність) обчислюється як

Precision = TP / (TP + FP),

тобто показує наскільки класифікатор хороший при запобіганні помилковості FP-спрацьовувань.

Точність виражає ступінь відповідності результатів дослідження дійсному значенню величини, що досліджується.

Проблеми при використанні датасету замалого розміру.

В залежності від обраного алгоритму розпізнавання визначається мінільно припустимий об’єм датасету зображень. Невеликі датасети з декількох підписів для кожного з підписантів дадуть можливість ознайомитись з алгоритмом та його діями у складних ситуаціях: невизначеність дій, недостатність ресурсів, тощо. Більші об’єми датасетів дозволяють оцінити продуктивність алгоритму та виконати тестування навантаження. Але якщо зразків недостатньо, неможна коректно обчислити точність обраного алгоритму [12].

Якщо визначено достатній об’єм датасету зображень, то перевищення його розміру призводить до неефективних витрат пам’яти та обчислювальних ресурсів.

1. **СПЕЦИФІКАЦІЯ ВИМОГ ДО** **ПРОГРАМИ**

**3.1 Варіанти використання системи**

Для формалізації вимог до розроблюваної системі було проведено збір та аналіз інформації на основі документів «Бачення», сформованих майбутніми користувачами системи. Користувачів системи доцільно розділити на наступні категорії:

* «Користувач» (User) – користувач програмним застосуванням, актор найвищого рівня абстракції.
* «Користувач-Тренер» (UTrainer) – користувач програмним застосуванням, що працює з програмою у режимі навчання нейромережі;
* «Класифікатор» (Classifier) – класифікатор, що виконує класифікацію та автентифікацію рукописного підпису.

Розглянемо основні функціональні вимоги, що пред'являються до розроблюваної системі. Система класифікації повинна навчатися на існуючих шаблонах підписів, розпізнавати та виконувати автентифікацію підпису, донавчатися на нових даних, а також додавати нові класи станів з метою уточнення.

Формальний опис вимог представлено на рис. 3.1 у вигляді діаграми варіантів використання. Допускається розбивка підсистеми на декілька окремих компонентів, для спрощення інтеграції в систему Signature Scan Tool.

Програма навчання повинна передбачати формування класифікаторів, здатних до інтеграції в систему Signature Scan Tool, автоматичне перетворення даних у формат, придатний до навчання класифікаторів. При неможливості створення відповідних класифікаторів програма повинна повідомити про це.

Програма повинна виконувати обробку та аналіз цілісності тренувальних даних – зразків рукописних підписів, перевіряти дані на відповідність закладених критеріїв, обчислювати значення ознак відповідно до закладених методів і алгоритмів. На підставі розрахованих значень ознак зберігати моделі для подальшої інтеграції.

****

Рисунок 3.1 – Діаграма варіантів використання системи Signature Scan Tool

Так як розроблювальний програмний продукт призначений для використання звичайним користувачем, які володіють обчислювальною технікою на слабкому рівні, то він повинен бути легкий і простий у використанні, тобто володіти зручним дружнім інтерфейсом. Наочність інтерфейсу допоможе користувачеві сконцентруватися на роботі з автентифікацією рукописного підпису.

Завдання дій має здійснюватися з використанням команд меню, кнопок піктограм, основним діям повинні відповідати підказки.

*Специфікація прецедентів (варіантів використання – ВВ):*

1. *ВВ «Add data»* (Додати дані)

Передумова: Програма запущена.

1. Користувач обирає режим завантаження даних.
2. Користувач обирає файл даних

2а. Помилка відкриття файлу.

2а.1 Система виводить помилку.

1. Система завантажує файл даних.
2. Користувач обирає відповідний файл з даними.

4а. Помилка відкриття файлу.

4а.1 Система виводить помилку.

1. Користувач запускає підготовку даних.

5а. Помилка в підготовці даних.

5а.1 Система виводить помилку.

1. Система обробляє дані. Успіх.
2. *ВВ «Supervised learn»* (Навчання з вчителем)

Передумова: Дані завантажені.

1. Користувач обирає режим навчання.
2. Програма перевіряє цілісність файлів з даними.

2а. Файл пошкоджено.  
 2а.1 Перехід до ВВ 1.

1. Програма виконує підбір оптимальних налаштувань класифікаторів.
2. Програма виводить оптимальні налаштування на екран.
3. Програма створює класифікатори відповідно до оптимальних налаштувань.
4. *ВВ «Save neural networks»* (Збереження нейронних мереж)

Передумова: Класифікатори успішно навчені

1. Програма зберігає частини класифікатора.

1а. Помилка запису на носій.

1а.1 Вивід повідомлення про помилку.

1. Програма виводить повідомлення про успіх.
2. *ВВ «Save tree structure»* (Збереження структури дерева)

Передумова: Класифікатори успішно навчені.

1. Програма зберігає опис класифікатора відповідно до внутрішньої структури.

1а. Помилка запису на носій.

1а.1 Вивід повідомлення про помилку.

1. Програма виводить повідомлення про успіх.
2. *ВВ «Identify signature»* (Ідентифікація підпису)

Передумова: Користувач авторизувався.

1. Користувач обирає відповідний режим роботи.
2. Системастворює новий сеанс.
3. Виклик ВВ «Input signature».
4. Система обчислює параметри підпису.
5. Система виводить користувачу результат.
6. Система зберігає результат.
7. *ВВ «Input signature»* (Введення рукописного підпису)

Передумова: Запущено режим вводу підпису для автентифікації. Користувач обрав відповідний режим.

1. Користувач обирає інструмент для вводу підпису.
2. Система надає область для вводу.
3. Користувач вводить текст та натискає кнопку для завершення..
4. Система виконує підготовчі дії для збереження.
5. Система перевіряє цілісність даних.

5а. Помилка в даних.

5а.1 Перехід до п.1.

1. Система зберігає дані.
2. *ВВ «Filtration»* (Фільтрація)

Передумова: Запущено режим вводу підпису для автентифікації. Користувач обрав режим вводу, зображення підпису не відповідає вимогам.

1. Користувач обирає інструмент фільтрації.
2. Система коригує зображення рукописних підписів відповідно до алгоритму – порогова бінарізація, скелетизація та ін.
3. Користувач перевіряє якість зображення власного рукописного підпису.

2а. Зображення не відповідає вимогам.  
 2а.1 Перехід до п. 1.

1. Завершення фільтрації. Система повідомляє про успіх.
2. ВВ «Signs calculation» (Обчислення ознак)

Передумова: Дані щодо рукописного підпису успішно класифіковано.

1. Система обчислює відстань між центром класу та наявними даними щодо рукописного підпису.
2. Система виводить результат щодо подібності даних користувача до певного стану.
3. ВВ «*Add new sample»* (Додання нового екземпляру даних)

Передумова: Дані щодо рукописного підпису класифіковано невірно.

* + - 1. Система вносить дані до внутрішньої обробки.
      2. Система додає дані в модель.

2а. Сукупна похибка моделі збільшилась.

2а.1 Система відміняє зміни.

* + - 1. Система виводить повідомлення.

**3.2 Вимоги до нефункціональних характеристик**

Сценарії атрибутів якості

а) Надійність (Reliability)

1) Якщо користувач бажає зберегти введений підпис, то система показує повідомлення о результаті менш ніж за 2 секунди після завершення вводу підпису.

2) Якщо користувач запитує перелік файлів з власними підписами, то система показує повідомлення о результаті менш ніж за 2 секунди.

3) У випадку критичної помилки система зберігає роботоспроможність у 80% випадків.

4) Якщо система зберігає дані щодо введеного рукописного підпису, то ймовірність вдалого збереження не менше 0.9.

5) Якщо система зберігає дані щодо результатів фільтрації введеного рукописного підпису, то ймовірність вдалого збереження не менше 0.9.

б) Продуктивність (Productivity)

1) Якщо користувач бажає пройти автентифікацію, то система показує повідомлення о результаті менш ніж за 3 секунди після завершення вводу підпису.

2) Якщо користувач відправляє вже існуючий зразок підпису на перевірку, то система відповідає на запит менше ніж за 1 секунду.

3) Якщо користувач фільтрує власний зразок рукописного підпису, то система оброблює дані щодо зображення за 1,5 секунди.

5) Якщо користувач зберігає дані щодо автентифікації, то система відповідає на запит менше ніж за 1 секунду.

в) Безпека (Security)

1) Якщо система запитує інформацію щодо зображення рукописного підпису, то отримує її в цілісності не менш, ніж у 90% випадків.

2) Якщо користувач створює та та тестує підпис, то можливість «злому» його сторонньою особою менше ніж у 2% випадків.

г) Зручність користування (Usability)

1) Якщо користувач створює підпис, то для відкриття форми для цього йому потрібно не більше ніж 3 секунди.

2) Якщо користувач бажає переглянути свій підпис, то для відкриття форми для цього йому потрібно не більше ніж 4 секунди.

3) Якщо користувач бажає переглянути збережений результат класифікації, то для відкриття форми для цього йому потрібно не більше ніж 3 секунди.

д) Супроводження (Support)

1) Якщо потрібно змінити функціонал програмної системи, то зміни вступають у силу менш ніж за 2 місяці.

2) Якщо потрібно портувати систему на іншу платформу (наприклад, iOS), то поставлена задача буде виконана не більш, ніж за 6 місяців.

3) Якщо відмічено велику кількість відгуків про помилку системи, то виправлення вступлять у силу не більш, чим через 1 місяць.

є) Обмеження (Restrictions)

1) Персональні дані користувачів повинні залишатися анонімними. Результат автентифікації (особиста інформація) також повинні залишатися анонімними.

2) Набір підписів, що приймає участь у автентифікації, повинен залишатися анонімним.

3) У рамках роботи з системою користувач не має можливості суттєво змінювати свій підпис (наприклад, при зміні прізвища).

* 1. **Системні вимоги**

Основні рекомендації щодо оформлення коду для вживаних мов програмування (Google C++ Code Guidelines, Java Code Convention, etc.)

Стандарт щодо організації процесу розробки – ISO/IEC TR 19759:2005 Software Engineering – Guide to the Software Engineering Body of Knowledge (SWEBOK).

Програма має працювати з адекватним часом відгуку на більшості мобільних пристроїв та у більшості браузерів. Основна вимога – достатня швидкість супутникового зв’язку (Н+ та вище).

1. **ПРОЕКТУВАННЯ ПРОГРАМИ ДЛЯ АВТЕНТИФІКАЦІЇ РУКОПИСНОГО ПІДПИСУ**

## Проектування архітектури системи

Система розподілена на два окремих модуля, що доповнюють одне одного та інтегровані у проект Signature Scan Tool: підсистема класифікації та підсистема навчання, яка створена для генерації класифікаторів, що підключаються до модуля класифікації.

Для оцінки проектування розглянуто деякі основні сценарії використання у форматі діаграм діяльності.

На рис. 4.1 зображено діаграму взаємодії для ВВ «Add data». Варіант використання передбачає введення даних користувачем з наступною обробкою з боку системи. У разі помилки система повертається до попереднього кроку.

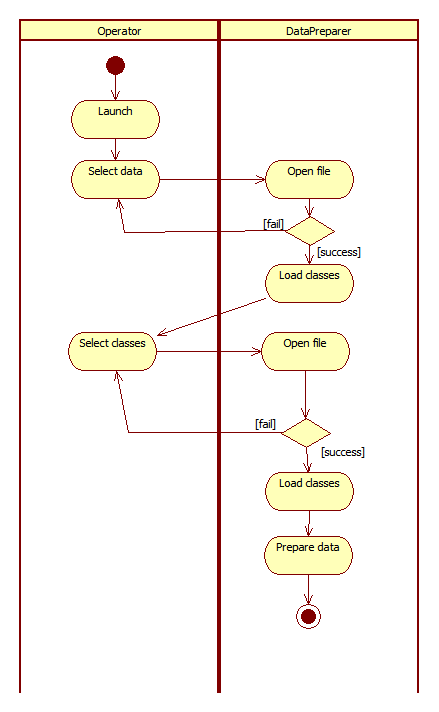


Рисунок 4.1 – Діаграма взаємодії для ВВ «Add data»

Діаграма взвємодії для ВВ «Supervised learn» зображена на рис. 4.2. Варіант використання передбачає внутрішню обробку даних з послідуючою генерацією вузлів класифікатора.

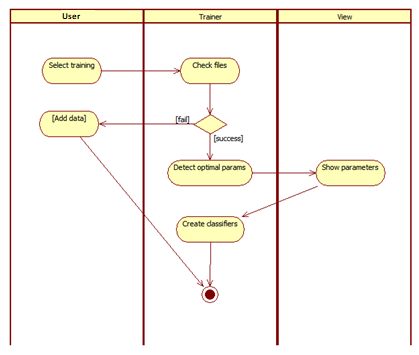


Рисунок 4.2 – Діаграма діяльності для ВВ «Supervised learn»

Оскільки ВВ «Identify Signature» відноситься до модуля класифікації даних, будуть використані різні актори (рис. 4.3).

На рис. 4.4 наведено діаграму взаємодії для ВВ «Add new sample». Як можна побачити, користувачем додається новий зразок підпису, який перевіряється на відповідність заданому формату, після чого нейромережа NN поповнюється новими даними.

На рис. 4.5 наведено діаграму взаємодії для ВВ «Input signature», яка дозволяє завантижити новий підпис для автентифікації, перевірити його на відповідність підпису для певного користувача та видати результати щодо успіху чи невдачі.

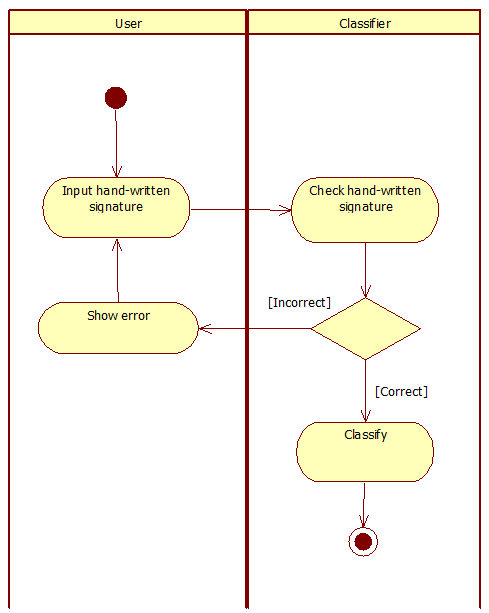


Рисунок 4.3 – Діаграма діяльності для ВВ «Identify Signature»

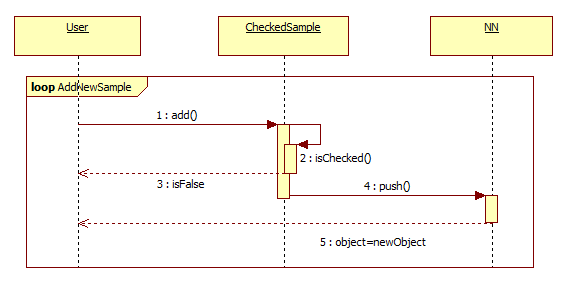


Рисунок 4.4 – Діаграма взаємодії для ВВ «Add new sample»

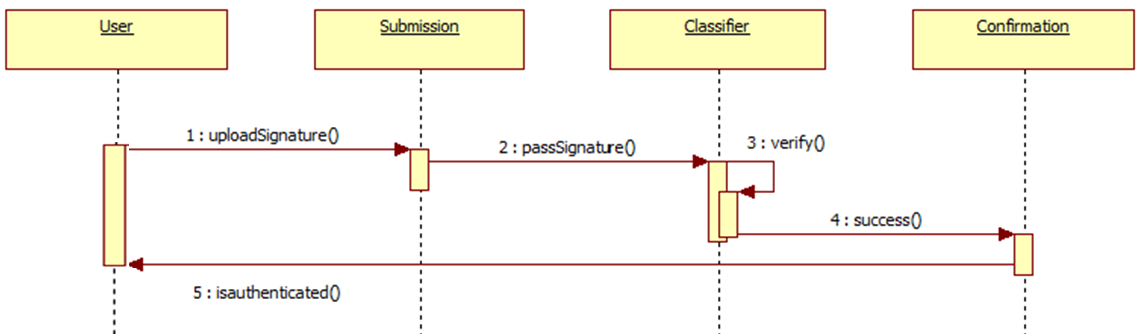


Рисунок 4.5 – Діаграма взаємодії для ВВ «Input signature»

На основі зазначених діаграм взаємодіїї створено діаграму концептуальних класів (рис. 4.6), що потім була деталізована до діаграми програмних класів.

Також, беручи до уваги те, що деякі операції з навчання класифікаторів є досит тривалими в часі, доцільно використовувати багатопоточні алгоритми, тому потрібно винести на діаграму концептуальних класів менеджер паралельних процесів.

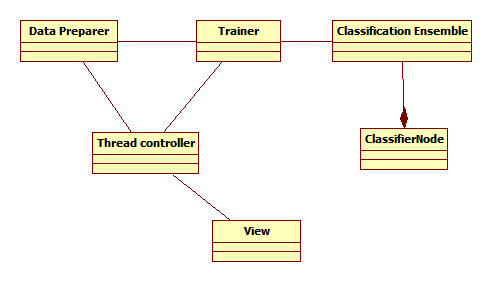


Рисунок 4.6 – Діаграма концептуальних класів програми навчання

Діаграма концептуальних класів для підсистеми класифікації зображена на рис. 4.7.

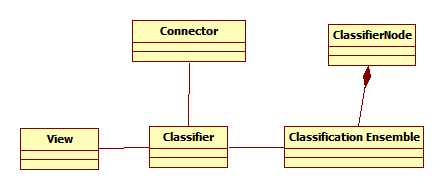


Рисунок 4.7 – Діаграма концептуальних класів підсистеми класифікації

## Основні алгоритми в системі класифікації

У цьому пункті розглянуто декілька основних підходів до класифікації багатовимірних даних, що використані у проекті. Для автентифікації рукописного підпису користувача були розроблені спеціальні класифікатори.

Для забезпечення роботи класифікатора були застосовані алгоритми статистичного аналізу і індуктивного машинного навчання.

На першому етапі розробки використовувався метод класифікації за допомогою статистичних алгоритмів, який показав результати класифікації з точністю ~ 90% в залежності від класу (персони). Однак, такий підхід не дозволив ідентифікувати дані, які не належать ні до одного з відомих класів.

Використання статистичного аналізу підтвердило припущення про відмінності між вибірками ознак для різних станів. З цього був зроблений висновок про те, що стани можна вважати непересічними класами. І, відповідно, визначило можливість переходу до використання машинного навчання для вирішення поставленого завдання методом моделювання штучної нейронної мережі.

## Основні алгоритми в системі навчання

Різновиди градієнтного алгоритму є найбільш поширеними серед ефективних методів навчання перцептронів. При цьому використовується підхід зворотного розповсюдження помилки, що дозволяє налаштувати нейромережу під вхідні дані, сигналів помилки розповсюджується від виходу до входу мережі. Для такого підходу повинні бути відомі як вхідні дані (стимули), так і вихідні (реакції), таким чином, це є методом навчання з вчителем.

Існує багато алгоритмів, що базуються на градієнтному, зокрема:

* Інкрементальний – ваги зв’язків змінюються після кожного навчального прикладу;
* Пакетний – ваги зв’язків змінюються після проходження всіх навчальних прикладів;
* RPROP (Resilient PROPagation) – алгоритм враховує лише знак похідної від функції помилки, ваги знаходяться ітеративно за допомогою метода послідовних наближень;
* QUICKPROP – використовує метод Ньютона для пошуку мінімуму поверхні функції помилки;
* SARPROP (Simulated Annealing Resilient PROPagation) – евристичне покращення методу QUICKPROP для уникнення локальних мінімумів поверхні функції.

Інкрементальні та пакетні алгоритми представляють собою достатньо прямолінійні та прості алгоритми, що не потребують великої обчислювальної складності. Розглянемо інкрементальний та пакетний алгоритми навчання. Їхній метод роботи практично ідентичний:

1. Ваги мережі ініціюються випадково, або за допомогою спеціального алгоритму.
2. Спочатку вхідний сигнал розповсюджується через нейронну мережу від входів до виходів.
3. Кожний прихований нейрон обчислює результат своєї функції активації і розсилає свій сигнал іншим нейронам.
4. Обчислюється помилка  на одному вихідному нейроні К:

, (4.1)

де – бажаний, а  – дійсний вихід нейрона К.

після цього обчислюється , шо використовується для коригування вагів:

, (4.2)

де  – похідна від функції активації нейронів.

Згідно вимог, функція активації в алгоритмі зворотного розповсюдження помилки має бути:

1. неперервною;
2. диференційованою на всьому проміжку значень;
3. монотонно спадаючою.

Цим вимогам відповідає логістична функція:

, (4.3)

Також логістична функція оптимальна тим, що її похідна виражається через саму функцію, тим самим зменшуючи складність обчислень:

. (4.4)

Для кожного попереднього шару обчислюється  на базі  (4.2) за формулою:

 (4.5)

де *К* – кількість нейронів в поточному шарі; – ваги входів;  – параметр швидкості навчання, який визначає, наскільки сильно алгоритму потрібно змінювати ваги.

## Проектування структури та організації класів

* + 1. **Організація класів для модуля класифікації**

Наведемо окремі діаграми класів для класифікатору та модулю навчання.

Для простоти інтеграції в основний продукт, класифікатор забезпечений уніфікованим інтерфейсом (*IClassifier*). Це дозволяє створювати різні реалізації системи класифікації, не змінюючи продукт в цілому. Діаграму класів зображено на рис. 4.8.

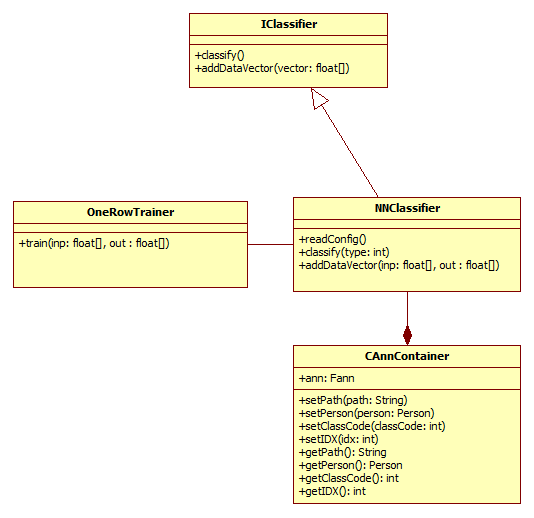


Рисунок 4.8 – Діаграма класів підсистеми класифікації

Клас *IClassifier* містить наступні методи:

classify() – виконується класифікація підпису, потім ця функція перевантажується у класі *NNClassifier*;

addDataVector – додавання вектора ознак підпису, потім ця функція теж перевантажується.

Клас *NNClassifier* містить функції:

readConfig() – отримання даних з файлу конфігурації для налаштування програми;

classify(type: int) – процедура співвіднесення підпису до певного класу;

addDataVector(int: float[], out float[]) – додавання векторів входних ознак та виходних вагів для нейромережі.

У класі *CAnnContainer* реалізований контейнер, в якому знаходиться нейронна мережа–класифікатор, а також додаткові дані для її коректної ініціалізації та роботи. Також, *CAnnContainer* є типом, що може порівнюватися. Це створено для забезпечення простої збірки глибоких дерев класифікації. Методи:

setPath(path: String) – встановлення шляху до файлу даних;

setPerson(person: Person) – задати певного підписанта;

setClassCode(classCode: int) – встановити значення коду (номер) для класу;

setIDX(idx: int) – задати ідентифікатор для нейромережі у конфігураційному файлі;

getPath(): String – отримання шляху до файлу даних;

getPerson(): Person – отримати дані щодо певного підписанта;

getClassCode(): int – визначити код для класу;

getIDX(): int – визначити ідентифікатор нейромережі у конфігураційному файлі.

Клас *OneRowTrainer* забезпечує донавчання нейронних мереж по мірі надходження даних про рукописні підписи нових осіб.

Цей клас містить метод train(inp: float[], out: float[]), який саме і виконує донавчання нейромереж. При донавчання фіксують всі параметри, крім параметрів останнього шару, і тільки потім поступово виконується підналаштування попередніх шарів, таким чином після перенавчання останнього шару донавчання вже не настільки руйнівно діє на попередні шари.

* + 1. **Організація класів для модуля навчання**

Система навчання являє собою програму, що створена для генерації нових класифікаторів з вихідних даних. Для такого завдання доцільно використовувати шаблон MVC, у якому поєднані вид та контролер для забезпечення роботи інтерфейсу, а модель займається генерацією класифікаторів. Діаграма класів зображена на рис. 4.9.

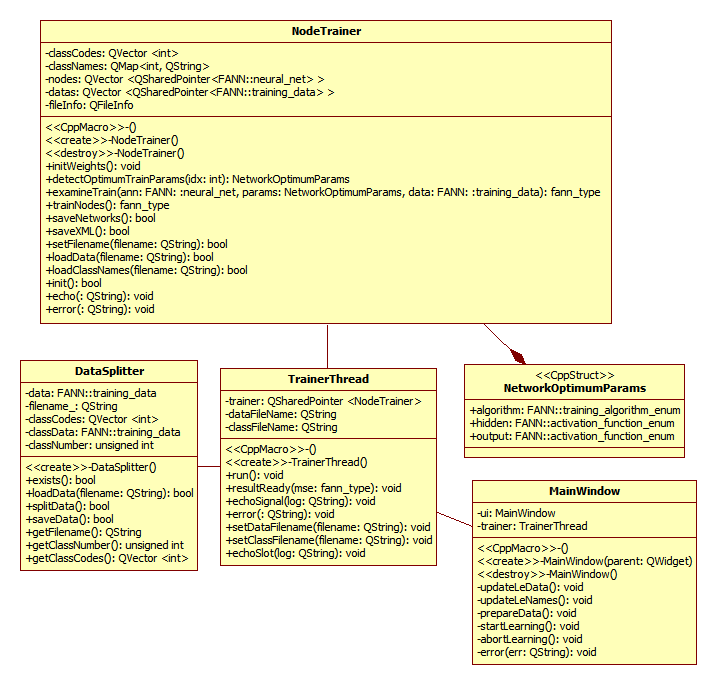


Рисунок 4.9 – Діаграма класів програми генерації класифікаторів

Така реалізація дозволяє досить просто модифікувати модель, не виконуючи змін в інтерфейсі, а також, пов’язаних з ним класами. Оскільки навчання нейронних мереж є тривалою операцією, доцільно винести модель в окремий потік, щоб не перешкоджати взаємодії користувача з інтерфейсом. Для обслуговування потоку служить клас *TrainerThread,* що також є контролером у шаблони MVC. Він містить наступні методи:

create – створення потоку;

run() – запуск на виконання;

resultReady(mse: fann\_type): void – для використання можливостей fann;

echoSignal(log: QString): void – логи сигналу для потоку;

error(err: QString): void – обробка помилок;

setDataFilename(filename: QString): void – задати ім’я для файлу даних;

setClassFilename(filename: QString): void – задати ім’я для файлу класу;

echoSlot(log: QString): void – функція, яка викликається у відповідь на певний сигнал log.

Клас *TrainerThread*, власне, розпоряджується навчанням нейронних мереж (*NodeTrainer*) та підготовкою даних (*DataSplitter*).

Клас *NodeTrainer* містить наступні методи:

initWeights() – ініціалізація вагів (це дійсні числа, що відображають коефіцієнт зв'язку між конкретними нейронами);

detectOptimumTrainParams(idx: int) – визначення параметрів мережі за певним ідентифікатором;

examineTrain(ann: FANN, net: neural\_net, params: NetworkOptimumParams, data: FANN, data: training\_data): fann\_type – навчання вказаної мережі за вказаними даними;

trainNodes(): fann\_type – ваги зв'язків мережі;

saveNetworks(): bool – збереження нейремереж;

saveXML(): bool – збереження конфігуратору;

setFilename(filename: QString): bool – встановити і’мя файлу даних;

loadData(filename: QString): bool – завантажити дані з вказаного файлу;

loadClassNames(filename: QString): bool – завантажити доступний клас;

init(): bool – початкова ініціалізація;

echo(str: QString): void – видача інформаційних повідомлень;

error(str: QString): void – видача повідомлень про помилкові ситуації.

Клас *DataSplitter* містить наступні методи:

exists(): bool – перевірка того, що кількість наявних класів дорівнює потрібній кількості;

loadData(filename: QString): bool – завантаження файла даних за указаним ім’ям;

splitData(): bool – розподіл даних по окремим класифікаційним класам;

saveData(): bool – збереження даних у відповідному файлі даних з розширенням .dat;

getFilename(): QString – отримати ім’я файлу;

getClassNumber(): unsigned int – отримати номер класу;

getClassCodes(): QVector <int> – отримати коди класів.

Клас *MainWindow*  є класом-видом, що зберігає у собі форму та інтерфейс. Він оновлюється через механізм сигналів/слотів у фреймворці Qt. Цей клас містить такі методи:

updateLeData(): void – визначення файлу з назвами класів за допомогою ui;

updateLeNames(): void – визначення файлу з назвами мереж за допомогою ui;

prepareData(): void – пошук підготовлених даних та перезапис при наявності таких даних та перевірка успішності операції;

startLearning(): void – видача інформації про початок навчання нейронних мереж та відомості про користування програмою підготовки класифікаторів системи;

abortLearning(): void – видача інформаційного повідомлення про завершення навчання нейронних мереж;

error(err: QString): void – видача повідомлень про наявність помилки.

# ПРОГРАМНА РЕАЛІЗАЦІЯ РОЗРОБЛЮВАНОЇ СИСТЕМИ

**5.1 Особливості створення програмних модулів з урахуванням мови програмування**

У даному розділі опишемо особливості використання обраної мови програмування та середовища розробки. Розглянемо особливості створення програми під Android та середовища Android Studio, що має власну систему для відлагодження програми з емуляторами мобільних пристроїв. При цьому в Java файлах потрібно описувати логіку програми – той функціонал, що повинна виконувати програма.

Інтерфейс – зовнішній вигляд програми – у вигляді макетів наводиться у XML файлах, їх може бути декілька для кожного виду програмного застосування.

Програма, написана під Android, має велику кількість файлів різного призначення та з різними розширеннями. Усі вони повинні бути скомпільованими та упакованими в .apk файл, який може бути запущеним на емуляторі чи фізичному пристрої (смартфоні).

Інтегроване середовище розробки (IDE) Android Studio містить емулятор, що імітує пристрої Android на комп’ютері, так що є можливість протестувати програму на різних пристроях та рівнях Android API, не маючи необхідності фізично підключати кожен пристрій.

Тестування програми на емуляторі є певним чином швидшим та простішим, ніж на фізичному пристрої. Наприклад, можна швидше передавати дані на емулятор, ніж на пристрій, підключений через USB.

Емулятор постачається із заздалегідь визначеними конфігураціями для різних телефонів Android, планшетів, Wear OS та Android TV. Android Studio є продуктом компанії Google. Емулятор Android застарів у червні 2019 року для 32-розрядних систем Windows. Підтримка 32-розрядного емулятора Windows триває до червня 2020 року, включаючи критичні виправлення помилок, але нові функції не додаватимуться. Тому для використання емулятора краще використовувати 64-розрядну систему Windows. При необхідності використання емулятору у 32-розрядної системі Windows, доцільно використовувати диспетчер SDK, щоб встановити останню версію емулятора для 32-розрядної Windows.

Кожен примірник емулятора Android використовує віртуальний пристрій Android (AVD) для визначення версії Android та апаратних характеристик змодельованого пристрою. Щоб ефективно протестувати свою програму, слід створити AVD, який моделює кожен пристрій, на якому призначена ваша програма. Щоб створювати AVD та керувати ними, використовуйте AVD Manager. Кожен AVD функціонує як незалежний пристрій із власним приватним сховищем для даних користувачів, SD-картою тощо. За замовчуванням емулятор зберігає дані користувача, дані SD-карти та кеш-пам’яті в каталозі, специфічному для цього AVD. Коли ви запускаєте емулятор, він завантажує дані користувача та дані SD-карти з каталогу AVD.

Щоб встановити файл APK на емульований пристрій, треба перетягнути файл APK на екран емулятора. З'явиться діалогове вікно програми встановлення APK. Коли інсталяція завершиться, можна буде переглянути програму у своєму списку програм.

Щоб додати файл до емулятора, треба перетягнути файл на екран емулятора. Файл розміщується в каталозі / sdcard / Download /. Можна буде можете переглянути файл із Android Studio за допомогою Провідника файлів пристрою або знайти його на пристрої за допомогою програми Завантаження або Файли, залежно від версії пристрою.

Знімок – це збережене зображення AVD (віртуального пристрою Android), яке зберігає весь стан пристрою на момент його збереження, включаючи налаштування ОС, стан програми та дані користувача. Можна повернутися до збереженого стану системи, завантажуючи знімок, коли завгодно, заощаджуючи час очікування перезапуску операційної системи та програм на віртуальному пристрої, а також заощаджуючи зусилля, щоб повернути свою програму до стану на якому необхідно відновити тестування. Знімки дійсні для системного образу, конфігурації AVD та функцій емулятора, за допомогою яких вони зберігаються. Будь-яке оновлення Android-емулятора, системного образу або налаштувань AVD скидає збережений стан AVD, тому наступного разу, коли ви запускаєте AVD, воно повинно виконати холодне завантаження.

Розглянемо структуру Android-проекту та визначимо які елементи проекту містять його папки:

* папка Java – містить програмний код, який реалізує бізнес-логіку. Цей код знаходиться в каталозі src\main\java в основній папці проекту;
* папка Resources – містить ресурси для сучасної візуалізації програми – статичної візуалізації та динамичної анімації;

Ця папка за умовчанням містить такі:

* папка для збереження іконок проекту *drawable*;
* папка з зовнішніми виглядами екранів (XML файли) *layout*;
* папка для списку елементів у панелі *menu*;
* папка з даними щодо розмірів, констант тощо *values*.

Файл AndroidManifest.xml повинна мати кожна Android-програма, цей файл повинен бути розташований у корені проекту. Він містить інформацію щодо версії операційної системи, інструментів збірки проекту, тощо.

Публічний клас View представляє базу для побудови окремих компонентів проекту. Він повинен виконувати обробку подій, які відбуваються у проекті, та за графічне опрацьовування цих подій. Цей клас використовують віджети для таких компонентів як поля вводу, кпонку, слайдери, тощо. Окремі компоненти можуть вкладатись у контейнери, у яких можна впорядковувати об’єкти та керувати їх розташуванням на екрані.

Віджети програм розташовуються в пакеті android.widget та являють собою мініатюрні види програм, які можна вбудовувати в інші програми (наприклад, на головний екран) і отримувати періодичні оновлення. Компонент Android-програми, який вміщує інші віджети додатків, називається хостом віджета програми.

Розмітка виду визначає зовнішній вигляд інтерфейсу Android-проекту. Для того, щоб визначити особливості інтерфейсу, можна:

1. зробити фрагмент програмного java-коду для кожного з елементів інтерфейсу, які повинні бути задіяними у проекті;
2. визначити елементи розмітки виду в XML-файлі.

Ці способи можна компонувати, з визначенням ряду елементів у XML-файлі, а іншу їх частину – у java-коді.

TextView – це елемент інтерфейсу користувача, який відображає текст користувачеві. Його підкласом є елемент EditText, що надає можливість редагування та вводу тексту.

Прості повідомлення можна виводити на екран за допомогою класу Toast. Він надає спливаюче вікно (час відображення на екрані налаштовується), у якому користувачеві наводиться певний текст. Цей клас визначається тільки у праграмному коді.

Якщо повідомлення повинно розтягуватись (масштабування користувачем), доцільно використовувати клас Snackbar (час відображення теж налаштовується у мілісекундах).

Для визначення вибору дій використовується клас android.widget.Button. Кастомізація можлива шляхом встановлення на кнопку зображень, зміни шрифта, кольору тексту, розміру тексту та ін.

Для того, щоб відмітити властивості об’єкта чи налаштувати дію проекту використовуються прапорці – елементи Checkbox, що надають можливість множинного вибору з кількох значень; кожен елемент має бінарний стан.

Якщо є потреба встановлювати текст в залежності від того який є стан – активний чи пасивний – для певного елементу, використовується клас ToggleButton.

При необхідності обрати одну дію з переліку дій (наприклад, зберегти підпис на пристрої, чи в пам’яті хмари, чи на якомусь певному сервері) використовуються радіоперемикачі класу RadioButton. Один з перемикачів обов’язково є вибраним..

Для зручного вибору та користування датой та часом використають класи DatePicker і TimePicker.

Частина роботи застосування пов’язана з графікою, тому потрібно передбачити можливість вводу підпису за допомогою полотна Canvas, що є графічною поверхнею для малювання простих рисунків. При цьому визначаються товщина та колір лінії. Для роботи Canvas треба підключити android.graphics з відповідними бібліотеками двовимірної графіки.

Для створення простого графічного зображення без змін та анімації зазвичай створюють успадкування від класу View і використовують метод onDraw().

Зображення у Android-проекті можна отримати завдяки одному з підходів:

* завдяки використанню об'єкту View;
* із за діянням графічного Canvas.

Дуже коротко опишемо ОО-мову Java, яка використовує синтаксис, що є модифікацією [C](https://uk.wikipedia.org/wiki/C_(%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) і [C++](https://uk.wikipedia.org/wiki/C%2B%2B). Мова Java сає спрощений та поліпшений процес робробки програм, використання віртуальної машини дозволяє виконувати ряд ключових дій. Java володіє меншою кількістю низькорівневих засобів для роботи з апаратним забезпеченням, що в цілому відповідає вимогам до проекту [13].

Обрана мова програмування є платформо-незалежною, таким чином, сервер буде однаково працювати на різноманітнх операційних системах.

Для побудови системи були використані програмні пакети Qt, MySQL, мови програмування Java і С++. Штучна нейронна мережа реалізована за допомогою програмної бібліотеки FANN, яка передбачає тип штучної нейронної мережі «багатошаровий перцептрон».

Використання цієї бібліотеки обумовлено її кросплатформеністю і спрямованістю на вбудовані системи. Її ядро написане мовою С за стандартом С99. Це дозволяє їй працювати приблизно в 150 разів швидше її аналогів. Використання стандартизованої мови С++ дозволяє їй працювати під різними операційними системами, як сімейства Windows, так і Linux. Вона має також біндинг під інші різні мови. Був зібраний біндинг для Java JDK8 x64. Також, для написання програми навчання нейронної мережі використовується фреймворк Qt – кросплатформовий інструментарій розробки програмного забезпечення мовою програмування C++.

## Розробка класифікаторів на основі нейронних мереж

У задачах класифікації та розпізнавання образів все частіше застосовуються штучні нейронні мережі. Штучною нейронною мережею називається математична модель, програмна, або апаратна реалізація, побудована за принципом функціонування та організації біологічних нейронних мереж. Існує множина топологій нейронних мереж, як найпростіших (перцептрон Розенблата, мережа Ворда), так і складних згорточних мереж. Серед різних структур нейронних мереж однією з найбільш відомих є багатошарова структура, в якій кожен нейрон довільного шару пов'язаний з усіма аксонами нейронів попереднього шару або, в разі першого шару, з усіма входами штучною нейронної мережі.

Невірно було б казати, що нейронні мережі програмуються на певній мові програмування, вони повинні навчатися виконувати правильні дії, у нашому випадку – класифікувати.  Технічно, навчання полягає в знаходженні коефіцієнтів зв'язків між нейронами. У ролі класифікатора для вирішення завдання, зі всієї множини штучних нейронних мереж був обраний багатошаровий перцептрон на підставі теореми про збіжність перцептрона і універсальної теореми апроксимації.

**5.2.1 Опис принципу singleton класифікатора**

Для автентифікації підписів користувачів системи, на підставі значень ознак підписів, були застосовані інкрементальний і груповий методи зворотного поширення помилки. Відповідно до їхньої роботи, після поширення вхідних сигналів через мережу обчислюється помилка і передається назад через мережу. У цей час ваги зв'язків коригуються для зменшення помилки. Штучна нейронна мережа має N виходів, з яких (N-1) відповідають наявним класам, а останній відповідає ситуації, при якій досліджуваний об'єкт не належить жодному з описаних класів. Вихідна інформація містить ймовірності віднесення об'єкта до відповідного класу. При великій кількості об'єктів, які використовуються на етапі навчання штучної нейронної мережі (>50000), ймовірність правильної класифікації наближається до 1. Але, оскільки реальна кількість користувачів програми Signature Scan Tool суттєво менша за 50000, тренування такої штучної нейронної мережі складної структури являє собою досить важку задачу.

* + 1. **Класифікація за допомогою дерева рішень**

В результаті задача автентифікації підписів була вирішена з використанням нейронних мереж, організованих в дерево класифікації для виділення окремих підписів (станів – класів даних). Кожна нейронна мережа навчається як бінарний класифікатор типу «один–проти–всіх» для кожного з станів (класів) даних. Такий підхід дозволив домогтися легкості навчання і позбутися від перенавчання мережі, пов'язаного з необхідністю виділення всього лише одного класу для кожної штучної нейронної мережі з усієї вибірки даних.

На відміну від підходу з однієї штучної нейронної мережі, підхід з розділеними нейромережами додав можливість використовувати різні функції активації і тренувальні алгоритми для кожного класифікатора. Що дозволило значно підвищити точність і швидкість навчання. Також, стало можливим визначення даних, що не належать до будь–якого з відомих класів.

Дерево класифікації будується шляхом перебору всіх можливих перестановок послідовності класифікаторів для досягнення найбільшої точності на тестовій вибірці даних – приблизно 95% (рис. 5.1).

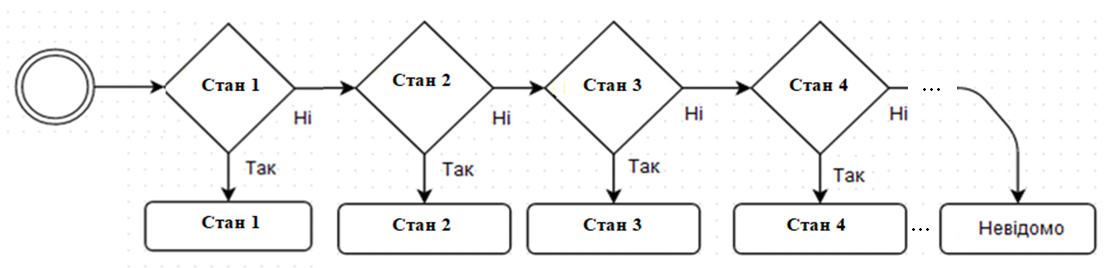


Рисунок 5.1 – DFD–діаграма дерева рішень

* + 1. **Класифікація за допомогою мажоритарної функції**

Застосування мажоритарної функції для оцінки результатів класифікації дозволяє збільшити швидкість навчання, так як не потрібно повний перебір варіантів, що вимагає О(2^N) порядку часу, де N – кількість класифікаторів. При такому підході одночасно виконуються всі бінарні класифікатори з подальшою компонуванням відповіді в єдине рішення.

Використання паралельно всіх класифікаторів дозволить тестувати ступінь подібності тестових даних середньому значенню класів і робити висновок щодо авторства підпису, що поєднує кілька осіб в разі сумнівних зразків (рис. 5.2).

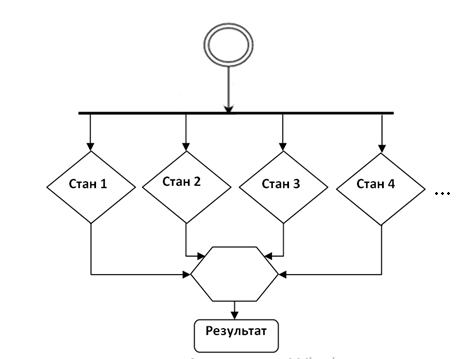


Рисунок 5.2 – DFD – діаграма паралельного включення класифікаторів

* + 1. **Пошук оптимальних налаштувань для кожного класифікатора**

Він виконується за допомогою перебору всіх можливих налаштувань з подальшим вибором найкращого з них. Алгоритм тренує спеціально підготовану нейронну мережу на наборі даних з відповідними варіантами вихідних та прихованих функцій активації, а також з усіма алгоритмами тренування, що доступні в бібліотеці.

Оскільки процес тренування на повному наборі даних з повнозв’язковою мережею є тривалою в часі операцією, виконано оптимізацію даного процесу. Оптимізація полягає в тому, що алгоритм виконується на розрідженій нейронній мережі, яка має 20% з’єднань між нейронами. Тренування виконується на протязі 100 епох (циклів навчання), замість 50000 епох для стандартного навчання. Враховуючи лінійну масштабованість перцепторнів, та необхідність лише оцінити продуктивність обраних функцій активації та алгоритму тренування, тобто динамічні характеристики мережі, дана оптимізація дозволяє припустити, що для повнозв’язкової мережі параметри залишаться тими самими.

* 1. **Алгоритм донавчання нейронної мережі**

Для забезпечення адаптування моделі та покращення її характеристик, було розроблено алгоритм додаткового навчання, що дозволяє додавати кожний окремий зразок рукописного підпису, який був визначений та віднесений до класифікаторів системи.

Алгоритм являє собою одну ітерацію алгоритму зворотного розповсюдження помилки через кожну з нейронних мереж класифікатора. Таке виконання алгоритму дозволяє регулювати кожний з класифікаторів в момент навчання.

Основною проблемою такого алгоритму може стати випадок, коли нові дані є протирічними до тих, що вже є в нейронній мережі. В такому разі сукупна похибка мережі скачкоподібно збільшується на деяку непередбачувану величину. Для запобігання такій ситуації перед донавчанням створюється копія вже існуючої нейронної мережі, на якій проводиться експеримент з додаванням нових даних. У випадку, коли сукупна помилка нової нейронної мережі стає більшою за сукупну помилку «чистої» нейронної мережі, результат видаляється. В іншому разі стара нейронна мережа замінюється на нову.

Реалізація алгоритму використовує функцію, що знаходиться в ядрі FANN, тому для неї було створено окремий Java–клас, який викликає цю функцію з визначеними параметрами. Також на низькому рівні виконується копіювання нейронних мереж, оскільки біндинг в Java оперує покажчиками на нейронні мережі, та не допускає копіювання їхнього вмісту.

* 1. **Структури даних в системі Signature Scan Tool**

Для використання навчених класифікаторів у вигляді дерева рішень було прийнято рішення використовувати xml–файл в сукупності зі стандартними розширеннями бібліотеки FANN – .net. Такий підхід дозволяє реалізувати автоматизоване навчання класифікаторів з подальшою упаковкою в єдиний файл. Такий класифікатор зможе працювати зі стандартним інтерфейсом модуля класифікації системи Signature Scan Tool.

*Збереження дерева класифікації в XML–файлі.*

При конфігуруванні класифікатора необхідно побудувати дерево рішень на підставі кількох підклассифікаторів. Для полегшення такої побудови був розроблений метод налаштувань за допомогою xml–файлу. Для користування програмою потрібно визначити назву файлу – "classifier.xml". Файл містить посилання на нейромережеві вузли кожного з вузлів в дереві рішень.

Нижче наведено приклад конфігураційного файлу:

<?xml version=**"1.0"** encoding=**"utf–8"**?>

<tree>

<classifier idx = **"1"**>

<person>**State1**</person>

<code>**1**</code>

<path>**cp–a.net**</path>

</classifier>

classifier idx = **"2"**>

<person>**State2**</person>

<code>**2**</code>

<path>**cp–a.net**</path>

</classifier>

<classifier idx = **"3"**>

<person>**State3**</person>

<code>**3**</code>

<path>**cb–a.net**</path>

</classifier>

<classifier idx = **"4"**>

<person>**State4**</person>

<code>**4**</code>

<path>**cn–a.net**</path>

</classifier>

</tree>

<? Xml version = "1.0" encoding = "utf–8"?> – версія і кодування файлу

<tree></ tree> – кореневий елемент, що описує дерево класифікації

<classifier></ classifier> – елемент, що описує один з вузлів дерева класифікації. Параметр idx вказує на його порядок в ієрархії

<path></ path> – Шлях до файлу з нейронною мережею (рядок), що відповідає за вузол класифікації (щодо файлу classifier.xml)

<code></ code> – код особи (десяткове число)

<person></ person> – ім’я особи (рядок)

Наявність всіх тегів обов'язкова.

*Внутрішнє представлення дерева класифікації.*

Всередині системи класифікації дерево, закодоване в XML (eXtensible Markup Language) розгортається в дерево рішень за допомогою парсера XML. Xml парсер – це програма, яка витягує з вихідного файлу XML формату дані і зберігає або використовує для подальших дій.

Класифікатори зберігаються у контейнерах, що окрім нейронної мережі зберігає також назву класу (особи), його код, а також забезпечує коректну обробку відкриття та модифікації нейронних мереж. Всередині ансамблевого класифікатора знаходиться список контейнерів, а також методи для його завантаження та обробки.

*Забезпечення безпеки багатопоточності в програмі навчання.*

Забезпечується завдяки використанню вбудованих в фреймворк Qt розумних покажчиків *QSharedPointer*. Для збереження в пам’яті тренувальних даних та натренованих елементів класифікатора використовуються динамічні масиви розумних покажчиків, що дозволяє не перейматися про забезпечення відсутності конкуренції за ресурси між потоками, а також за можливі витіки пам’яті при нетиповому завершенні дочірніх процесів.

*Інші структури даних.*

Також для різних задач використані такі структури даних як *QMap, QVector, QFile.* З бібліотеки FANN використані структури даних *training\_data, neural\_net.* Ці структури даних знаходяться під обгорткою на мові С++, де вони знаходяться всередині відповідних класів.

*Файл даних* являє собою файл, в якому у першому рядку міститься: кількість екземплярів даних у файлі, кількість вхідних даних, кількість виходів класифікатора (має бути завжди 1). Далі йдуть рядки, в яких міститься інформація про дані користувача та код стану (класу). У випадку невірного формату файлу, або якщо файл пошкоджено, система видає відповідну помилку.

*Файл назв* являє собою файл, в якому окремими строками ставиться відповідність між кодом особи та її ім’ям. Так, кожна строчка файлу містить натуральне число, що репрезентує код класу, а також текст, що репрезентує назву стану (класу). У разі пошкодження файлу, або його невірного формату система видає відповідну помилку.

* 1. **Особливості підключення бібліотеки реалізації нейронних мереж до системи Signature Scan Tool**

Особливим питанням є ув’язання програми на мові java з *native*–методом, при цьому виникають питання використовуваної операційної системи та орігінільній мові розробки *native*–методу. JNI (Java Native Interface) – це власний інтерфейс Java. Він визначає спосіб взаємодії байт-коду, який Android компілює з керованого коду (написаного мовами програмування Java або Kotlin), із власним кодом (написаним на C / C ++). JNI нейтральний до постачальника, має підтримку завантаження коду з динамічних спільних бібліотек, і хоча громіздкий часом є досить ефективним.

JNI є інтерфейсом скомпонованим у вигляді набору динамічних бібліотек, такий підхід дозволяє не використовувати статичне зв'язування. Використання JNI дозволяє викликати з програми на Java функції С / С++. За допомогою JNI, бібліотека FANN підключаються до Java–коду системи Signature Scan Tool.

Для організації роботи бібліотеки було перероблено JNI біндинг. Під режимом збірки тут розуміється формат пакування експортних функцій в бібліотеці.

Для перезбірки бібліотеки було використано систему збірки *cmake*, за допомогою якої було підготовано набір вихідних файлів для компілятора *mingw.* Зібрана за допомогою *mingw* бібліотека у форматі *dll* була підключена до біндингу.

# ВИЗНАЧЕННЯ ВЛАСТИВОСТЕЙ ПРОГРАМИ

# SIGNATURE SCAN TOOL

* 1. **Тестування моделі класифікатора**

Для тестування обраної технології та її якості навчання на основі набору тренувальних даних було сформовано набір тестових даних за наступним принципом:

(6.1)

де – тестовий вектор,

– вектор з тренувального набору.

Число 25 обрано як перша квартиль величини, що дозволило створити статистично значимі тестові дані.

Вибірка фрагменту з бази даних рукописних підписів наведена на рис. 6.1. Для коректного тестування мережі використані дані з бази даних (відкритої) з кількістю підписантів = 500, кількість підписів кожного підписанта = 50. Важливо, що цей фрагмент БД не містить повторюваних зображень підписів, тобто кожен підпис був отриманий за унікальний сеанс. Перевірено, що вхідні дані є однаковими за розміром та форматом.



Рисунок 6.1 – Зразки рукописних підписів з бази даних

* + 1. **Тестування єдиної нейронної мережі**

Тестування було проведене запуском нейронної мережі на тестових даних з аналізом співвідношення кількості правильно розпізнаних векторів та неправильно розпізнаних векторів.

Точність класифікації на тестових векторах склала від 49.92% до 53.11% в залежності від тестових даних.

Підхід з використанням єдиної нейронної мережі, що був розглянутий, мав декілька суттєвих недоліків. Це вимусило відмовитися від подальшої його реалізації. Серед них:

1. низька точність класифікації, що неприпустимо у випадках аутентифікації підпису;
2. стабільність моделі залежить від багатьох налаштувань, які нетривіально вибрати оптимально та які залежать від характеристик тренувальної вибірки;
3. немає можливості детектувати, що клас не належить до жодного з відомих моделі;
4. висока ймовірність перенавчання нейронної мережі через її складність.
   * 1. **Тестування глибокого дерева рішень**

Модель добре проявила свої властивості на тестових даних, що були згенеровані так само, як і дані для тестування єдиної мережі. Точність розпізнавання склала 64.83% на тестових даних, що є середнім результатом.

Однак, не зважаючи на таку точність алгоритму, він має декілька суттєвих недоліків.

З боку навчання:

1. Висока обчислювальна складність повного перебору, що унеможливить   
   побудову моделі для великої кількості даних в класах.
2. Неможливість коригування окремих підкласифікаторів без повторного навчання усіх підкласифікаторів моделі.

З боку виконання:

1. Нестабільність моделі: побудова моделі залежить від якості навчання кожного з обраних класифікаторів.
   * 1. **Тестування широкого дерева рішень**

Для тестування було обрано той самий набір даних, що і для двох інших типів класифікаторів. Точність розпізнавання склала 93.28%, тому у роботі програми було прийнято рішення використовувати саме цей підход з обов’язковими попередніми обробками зображень – бінарізацією, фільтрацією та ін.

Основною перевагою широкого дерева рішень є те, що воно не може «пропустити» деякий клас через те, що якийсь клас у дереві до нього отримав результат, більший за пороговий.

**6.2 Тестування системи з використанням функціональних діаграм**

Після реалізації програмного модулю було проведено тестування основних прецедентів. Мета функціонального [тестування](https://uk.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) – виявлення того, наскільки робота реалізованих функцій не відповідає очікуваній поведінці, визначеній згідно до вимог та специфікації.

Перший рівень тестування, який зазвичай виконується розробниками – це юніт (модульне) тестування. Воно допомагає переконатися, що окремі компоненти ПЗ функціонують на рівні коду та поводяться очікувано. Такі тести можна проводити вручну, однак автоматизація процесу завжди економить час на розгортання та розширює тестове покриття.

Наступний шаг тестування – інтеграційне тестування. Інтеграційне тестування перевіряє, чи правильно працюють незалежні компоненти ПЗ, коли об'єднуються. Це можливість знайти такі помилки, як несумісність у форматах повідомлень чи даних, неприпустимі вхідні чи вихідні параметри тощо.

Останній крок – системне тестування. Це тестування методом «чорного ящика», яке оцінює повний та інтегрований програмний продукт. Проводиться якщо треба перевірити, чи відповідає система заданим вимогам.

Для усіх функцій потрібно оцінити найбільш ймовірні помилки. Розроблені тести повинні охоплювати усі такі помилки. Для наскрізної перевірки сукупності тестів їх поєднують у так звані тестові сценарії. Кожний сценарій має мету перевірки певної задачі.

Для створення функціональних тестів визначаються та аналізуються зовнішні специфікації функцій та вивчається інформація щодо предметної області. Розроблені тести перевіряються на повноту та несуперечливість.

Розглянемо функціональну діаграму причинно-наслідкових зв’язків для одного з прецедентів – «Identify signature». Функціональними називають діаграми, в першу чергу відображають взаємозв'язки функцій програмного забезпечення. Сучасні методи структурного аналізу і проектування складових частин проектованої системи надають розробникові певні синтаксичні та графічні засоби проектування функціональних діаграм інформаційних систем.

Виділимо причини і наслідки для діаграми причинно-наслідкових зв’язків прецеденту «Identify signature»:

Причини:

1. Користувач бажає автентифікувати підпис.
2. Рукописний підпис створено.

Наслідки:

1. Відображення інформації щодо результатів автентифікації підпису.
2. Помилка «Визначення неможливо. Введіть підпис».

Для наочності графа використовуються проміжні вершини 10 і 11 (рис. 6.2).

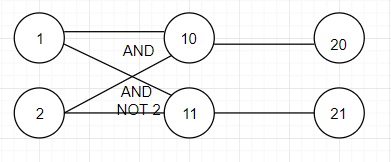


Рисунок 6.2– Функціональна діаграма причинно-наслідкових зв’язків для прецеденту «Identify signature»

Будуємо таблицю рішень по графу (табл.. 6.1). Для цього фіксуємо стані «1» по черзі всі наслідки.

Таблиця 6.1 – Таблиця рішень

|  |  |  |
| --- | --- | --- |
| Причина | 1 | 2 |
| 1 | 1 | 1 |
| 2 | 0 | 1 |
| Наслідок |  |  |
| 20 | - | 1 |
| 21 | 1 | - |

Розглянемо два тести, наведені у табл. 6.2.

Таблиця 6.2 – Таблиця тестів

|  |  |
| --- | --- |
| Тест 1 | Якщо користувач поперше завантажив програмний засіб або запустив його без визначення підпису та спробує провести його автентифікацію, то виведеться повідомлення «Визначення неможливо. Введіть підпис» |
| Тест2 | Якщо користувач запустив програмний засіб та ввів підпис або завантажив його з файлу, то виводиться інформація щодо його стану. |

Наслідок 21 буде в стані «1» тільки в тому випадку, якщо причина 2, з якою вона пов'язана, буде знаходиться в стані «0». 20 – в стані «1», якщо проміжна вершина 10 буде в стані «1». Вона, в свою чергу, буде знаходиться в стані «1» тільки в тому випадку, коли причина 1 в стані «1» та причина 2 в стані «1» одночасно, тому що вони пов'язані логічним І. Будуємо тестові набори. Кожен стовпець – це тест.

**6.3 Приклади використання системи Signature Scan Tool**

Розглянемо основні приклади для демонстрації роботи програми.

Для кожного прикладу наведемо вигляд вікна програми, аналіз отриманого зображення та рекомендовану дію щодо можливості автентифікації підпису на основі точності її розпізнавання.

На рис. 6.3 наведено загальний вигляд програмного застосування.



Рисунок 6.3 – Загальний вигляд програми

Початкова авторизація та реєстрація нового користувача виконується за допомогою натискання кнопок «Login» та «Register» відповідно.

Для вводу нового рукописного підпису використовується «Add sample». При вводі підпису з файлу користувачеві надається доступ до проводника файлової системи.

Для перевірки коректності наданого підпису використовується «Check sample». Важливо, що для для навчання та донавчання нейромережі можна використовувати тільки перевірені підписи.

Для автентифікації підпису використовується «Identify signature». Результатом роботи є список «Authenticated as» з ім’ям особи, що є автором підпису. Для додаткової обробки зображення підпису використовується «Filter», який виконує низку попередніх обробок. Тренування класифікатору проводиться натисканням «Train classifier».

У табл. 6.3 наведено приклади введеного рукописного підпису для розпізнавання нейромережею.

Таблиця 6.3 – Приклади роботи програми

|  |  |  |
| --- | --- | --- |
| Приклад | Аналіз підпису | Рекомендована дія |
|  | Низька точність розпізнавання, низька якість зображення | Обробка зображення – порогова бінеризація, видалення фону |

Продовження таблиці 6.3

|  |  |  |
| --- | --- | --- |
|  | Низька точність розпізнавання, низька якість частини зображення | Обробка зображення – отримання вектора ознак, видалення фону |
|  | Орієнтація підпису відрізняється від звичайної, кут відхилення від звичайного горизонтального напряму – до 15 градусів | Обробка зображення – поворот підпису |
|  | Низька роздільна здатність апаратних засобів при отриманні підпису чи скануванні його фото | Обробка зображення – скелетизація з малою кількістю ітерацій |

Продовження таблиці 6.3

|  |  |  |
| --- | --- | --- |
|  | Дуже низька роздільна здатність апаратних засобів при отриманні підпису чи скануванні його фото | Обробка зображення – скелетизація з великою кількістю ітерацій |

1. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

У дипломному проекті розглянуті питання розпізнавання та автентифікації рукописного підпису з високим рівнем точності, вирішення яких здійснюються працівниками компаніі по розробці програмного забезпечення для необслуговуємих приміщень, а саме їх здійснює розробник програмного забезпечення.

Предметна область «Охорона праці» є регульованою, для неї потрібно рекомендувати види та способи проведення заходів, що є найбільш доцільними та необхідними для успішної реалізації кваліфікаційної роботи магістра зі спеціальності «Інженерія програмного забезпечення» [14].

Отже у рамках розділу з охорони праці та безпеки у надзвичайних ситуаціях розглянемо особливості підходів з безпеки для робочого місця розробника програмного забезпечення, яке розташоване у багатоповерховому офісному будинку на 7 поверсі.

Наведемо перелік основних нормативних документів для ефективної організація заходів із забезпечення безпечних умов праці та поведінки у надзвичайних ситуаціях:

* ДСТУ 2293-99. Охорона праці. Терміни та визначення [15];
* НАПБ А.01.001-2015. Правила пожежної безпеки в Україні [16];
* НПАОП 0.00-1.28-10. Правила охорони праці при експлуатації ЕОМ;
* НРБУ-97/Д-2000. Норми радіаційної безпеки України; доповнення: Радіаційний захист від джерел потенційного опромінення [17];
* НАПБ Б.03.001-2204. Типові норми належності вогнегасників;

Визначення заходів з охорони праці та безпеки у надзвичайних ситуаціях для працівників галузі інженерія програмного забезпечення наведено у Додатку В та висвітлює наступні питання:

* організація та управління охороною праці у фірми розробника програмного забезпечення для необслуговуємих приміщень;
* обґрунтування заходів з покращення умов охорони праці;
* індивідуальне завдання – розрахунок захисного занулення;
* надзвичайні ситуації та шляхи їх запобігання.

В ході роботи було проведено аналіз заходів безпеки на робочому місцу розробників програмного забезпечення для приміщень, що не обслуговуються. В процесі аналізу приміщення, що було дано за варіантом були виділені основні параметри робочого приміщення користувача ПЕОМ, була розрахована електробезпека технічної системи та пожежна безпека. Було розраховано занулення для даного приміщення.

Окрім цього, були наведені надзвичайні ситуації, що можуть виникнути на виробництві та шляхи їх запобігання.Було розраховано дозу опромінення в надзвичайної ситуації радиоактивного забруднення за вихідними даними згідно з варінтом. Також були запропоновані заходи з охорони праці та безпеки у надзвичайних ситуаціях.

ВИСНОВКИ

У дипломній роботі розроблено мобільний програмний Android-засіб для розпізнавання рукописного підпису з використанням нейромережі. Засіб призначено для автентифікації підпису з високою точністю. Точність розпізнавання склала 93.28% при при застосуванні широкого дерева рішень.

У рамках роботи було виконано наступне.

У першому розділі виконано аналіз існуючих рішень щодо розпізнавання рукописного підпису. Огляд існуючих програмних засобів показав, що існує необхідність у спеціалізованому програмному продукті, який дозволяє розпізнавання та автентифікації рукописного підпису з високим рівнем точності. У другому розділі розглянуто та обрано алгоритми збору та розпізнавання підписів, можливості застосування нейромережі для розпізнавання та визначено принципи верифікації результатів роботи нейромережі. У третьому розділі сформовано завдання на розробку, в тому числі вимоги до функціональних характеристик розроблюваної програми. Четвертий розділ присвячено проєктуванню програмного засобу: розроблено архітектуру, побудовано основні алгоритми для навчання та класифікації за допомогою нейромережі, визначено структуру і організацію програмних класів. П’ятий розділ містить програмну реалізацію мобільного засобу: особливості створення програмних модулів з урахуванням мови програмування, розробку класифікаторів на основі нейромереж, структури даних. У шостому розділі визначено властивості програми та проведено тестування її роботи, а також наведено приклади використання. Сьомий розділ містить заходи з охорони праці та безпеки у надзвичайних ситуаціях.

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Элементарное введение в технологию нейронных сетей с примерами программ [Текст] / Р. Тадеусевич, Б. Боровик, Т. Гончаж, Б. Леппер, переклад: Рудинський Д.В. — М.: Горячая линия, 2011. — 408 с.
2. Паклин Н.Б. Бизнес–аналитика: от данных к знаниям: Учебное пособие. [Текст] / Н.Б. Паклин, В.И. Орешков — СПб: Питер, 2013. — с. 428–472.
3. Хайкин С. Нейронные сети: полный курс [Текст] / С.С. Хайкін — М.: Вильямс, 2006. — 1104 с.
4. Komleva N.O., Nevzorov V. D. Handwritten signature authentication program using a neural network // XIII Міжнародна науково-практична конференція «Інформаційні технології і автоматизація – 2020», Одеса, 22-23 жовтня 2020. С. 270-272.
5. Фомин Я. А. Распознавание образов: теория и применения [Текст] / Я.А. Фомин — М.: ФАЗИС, 2012. — 429 с.
6. Голубев Ю. Ф. Нейросетевые методы в мехатронике [Текст] / Голубев Ю.Ф. — М.: Изд–во Моск. ун-та, 2007. — 157 с.
7. Абраменко А. Принципы распознавания / А. Абраменко – K:.Компьютер–пресс, 2005 – 123 с.
8. Кулик С. Д., Никонец Д. А. Примеры использования нейросетевого алгоритма в методиках дляэксперта-почерковеда // Нейрокомпьютеры: разработка и применение. 2009. № 9. С.61–85.
9. Черенков А. М. Возможности определения особенностей характера человека по признакам почерка //Экспертная практика. 1995. № 37. С. 38–39.
10. Ren G. Recognition of Online Handwriting with Variability on Smart Devices / G. Ren, V. Ganapathy. // ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). – 2019. – Pp. 7605–7609.
11. Feuz S. RNN-Based Handwriting Recognition in Gboard [Електронний ресурс] / S. Feuz, P. Gonnet // Google AI Blog. – 2019. – Режим доступу до ресурсу: https://ai.googleblog.com/2019/03/rnn-based-handwriting-recognition-in.html.
12. Doetsch P. Fast and robust training of recurrent neural networks for offline handwriting recognition / P. Doetsch, M. Kozielski, H. Ney. // 2014 14th International Conference on Frontiers in Handwriting Recognition. – 2014. – №14. – pp. 279–284.
13. Дэрси Л. Android за 24 часа. Программирование приложений/ Дэрси Л., Кондер Ш. — М.: Рид Групп, 2011. — 464 с.
14. Гандзюк М.П. Основи охорони праці: Підручник/ М.П. Гандзюк, Є.П. Желібо, М. О. Халімовський. - Львів: Новий світ-2000, 2003. – 408 с.
15. Гогіташвілі Г.Г. Основи охорони праці: Навчальний посібник/ Г.Г. Гогіташвілі, В.М. Лапін. – Львів: Новий світ-2000, 2006. – 232 с.
16. Про затвердження Правил пожежної безпеки в Україні : наказ Міністерства внутрішніх справ України від 30.12.2014 р. № 1417. Офіційний вісник України. 2015 р. № 26. С. 91, Ст. 767.
17. Норми радіаційної безпеки України; доповнення: Радіаційний захист від джерел потенційного опромінення (НРБУ-97/Д-2000) : Постанова Головного державного санітарного лікаря Укаїни від 12.07.2000 р. № 116 — Режим доступа: \WWW/ URL: https://zakon.rada.gov.ua/rada/show/v0116488-00.