Міністерство освіти і науки України
Харківський національний університет імені В.Н. Каразіна
Факультет комп'ютерних наук
Спеціальність 125 «Кібербезпека»
Освітня програма «Кібербезпека»

«Допущено до захисту»
В.о. завідувача кафедри БІСТ
Мелкозьорова О.М.
_____
«___» червня 2024 р.

Пояснювальна записка

до кваліфікаційної роботи бакалавра

на тему: «Інструменти автоматизованого тестування безпеки додатків»

оцінка « »
Голова ЕК
Лемешко О. В. _____

Керівник к.т.н.
Мелкозьорова О. М.

Рецензент д.т.н. Краснобаєв В.А.
Виконавець: студент групи КБ-41

_____ Пожидаєв І.Д.

Харків – 2024

РЕФЕРАТ

Пояснювальна записка до дипломного проекту містить 42 сторінки, 18 рисунків, 1 додаток та 12 посилань на джерела.

Метою даної дипломної роботи є огляд принципу роботи технології розпізнавання відбитків пальців, що використовується у багатьох сьогоденних додатках та створення програмного рішення на мові програмування Java для автоматизованого отримання потрібних метрик для аналізу алгоритму, що використовується.

Об'єктом роботи є технології та алгоритми розпізнавання відбитків пальців для автентифікації та ідентифікації користувачів.

У результаті дослідження виявлено, що значення EER для однієї з найпопулярніших відкритих бібліотек SourceAFIS для Java становить 1.04%, що свідчить про відносно високу точність системи. Однак, було також відзначено, що система розпізнавання відбитків пальців має свої обмеження і не є ідеальною, оскільки вона все ще може робити помилки за певних умов, що може ставити певні системи автентифікації, та як наслідок і конфіденційну інформацію бід загрозу.

Результати дослідження можуть бути використані для подальшого вдосконалення рівня безпеки додатків, що використовують алгоритми розпізнавання відбитків пальців. Крім того, ці результати можуть бути корисними для розробників систем автентифікації, забезпечуючи основу для покращення безпеки та ефективності біометричних систем.

Ключові слова: КІБЕРБЕЗПЕКА, БІОМЕТРІЯ, АВТЕНТИФІКАЦІЯ, ІДЕНТИФІКАЦІЯ, РОЗПІЗНАВАННЯ ВІДБИТКІВ ПАЛЬЦІВ, ВРАЗЛИВІСТЬ, ПРОГРАМУВАННЯ, АНАЛІЗ, FAR, FRR, EER, JAVA, APACHE MAVEN

ABSTRACT

The explanatory note to the diploma project contains 42 pages, 18 figures, 1 appendix and 12 references.

The purpose of this thesis is to review the principle of operation of fingerprint recognition technology used in many of today's applications and to create a software solution in the Java programming language to automatically obtain the necessary metrics for analyzing the algorithm used.

The object of the work is technologies and algorithms for fingerprint recognition for user authentication and identification.

As a result of the study, it was found that the EER value for one of the most popular open source libraries SourceAFIS for Java is 1.04%, which indicates a relatively high accuracy of the system. However, it was also noted that the fingerprint recognition system has its limitations and is not perfect, as it can still make mistakes under certain conditions, which can put certain authentication systems, and as a result, confidential information, at risk.

The results of the study can be used to further improve the security of applications that use fingerprint recognition algorithms. In addition, these results can be useful for developers of authentication systems by providing a framework for improving the security and efficiency of biometric systems.

Keywords: CYBERSECURITY, BIOMETRICS, AUTHENTICATION, IDENTIFICATION, FINGERPRINT RECOGNITION, VULNERABILITY, PROGRAMMING, ANALYSIS, FAR, FRR, EER, JAVA, APACHE MAVEN

ЗМІСТ

ПЕРЕЛІК ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

SQL – Structured Query Language

MFA – Multi-Factor Authentication

2FA – Two-Factor Authentication

FAR – False Acceptance Rate

FRR – False Rejection Rate

EER – Equal Error Rate

FVC2004 – Fingerprint Verification Competition 2004

Maven POM – Maven Project Object Model

ВСТУП

Біометричні технології, зокрема розпізнавання відбитків пальців, широко використовуються у сучасних додатках для автентифікації та ідентифікації користувачів. Незважаючи на свою популярність та поширеність, ці технології все ще мають певні обмеження та недоліки, які можуть ставити під загрозу конфіденційність та безпеку інформаційних систем. Тому важливим питанням є ретельний аналіз та оцінка ефективності алгоритмів розпізнавання відбитків пальців.

Дана дипломна робота присвячена огляду принципів роботи технології розпізнавання відбитків пальців та створенню програмного рішення на мові програмування Java для автоматизованого отримання метрик аналізу використовуваного алгоритму. Метою роботи є дослідження точності та надійності однієї з найпопулярніших відкритих бібліотек SourceAFIS для Java, що реалізує алгоритми розпізнавання відбитків пальців.

Об'єктом дослідження є технології та алгоритми розпізнавання відбитків пальців, які застосовуються для автентифікації та ідентифікації користувачів у різноманітних додатках. У ході роботи буде проаналізовано показники ефективності, такі як EER (Equal Error Rate), FAR (False Acceptance Rate) та FRR (False Rejection Rate), що дозволяють оцінити точність та надійність системи розпізнавання відбитків пальців.

Результати дослідження можуть бути корисними для подальшого вдосконалення рівня безпеки додатків, що використовують біометричні технології, а також для розробників систем автентифікації, забезпечуючи основу для покращення безпеки та ефективності біометричних систем.

1 ТЕОРЕТИЧНІ МАТЕРІАЛИ

1.1 Кібербезпека додатків

З розвитком цифрових технологій та швидким зростанням кількості веб- та мобільних додатків, питання кібербезпеки стає все більш актуальним та важливим. Але що таке кібербезпека і чому вона важлива?

Кібербезпека - це набір заходів, технологій і процесів, спрямованих на захист комп'ютерних систем, мереж і даних від несанкціонованого доступу, атак, крадіжок та пошкоджень.

Кібербезпека додатків (Application Security) є підрозділом загальної кібербезпеки, що фокусується на захисті програмного забезпечення від загроз та вразливостей. Це включає заходи, які розробники та користувачі використовують для захисту програмних додатків на етапах їх розробки, розгортання та експлуатації.

Для отримання доступу до чутливих або конфіденційних даних у додатках, зловмисники можуть використовувати різноманітні методи, такі як:

- SQL-ін'єкції: Зловмисники можуть використовувати SQL-ін'єкції для виклику вразливостей у веб-додатках, які використовують бази даних. Це може дозволити їм витягувати, змінювати або видаляти дані з бази даних. [2]
- Атаки на аутентифікацію: Зловмисники можуть використовувати різні методи атаки на аутентифікацію, такі як перехоплення сесійних файлів, перебор паролів або фішингові атаки, щоб отримати доступ до облікових записів користувачів.
- Переповнення буфера: Атаки на переповнення буфера можуть бути використані для виклику вразливостей у програмах, які приймають введення від користувачів, таких як веб-додатки або мобільні додатки. Це може призвести до виконання зловмисного коду або отримання доступу до системних ресурсів.

• Використання слабкостей у коді: Зловмисники можуть шукати та експлуатувати слабкості у програмному коді додатка, такі як недостатньо захищені точки входу, вразливості у вбудованих бібліотеках або недостатньо захищені API.

1.2 Методи автентифікації у сьогоденних додатках

Ідентифікація, автентифікація і авторизація - це три важливі концепції в області кібербезпеки та управління доступом до ресурсів. Вони використовуються для визначення та контролю доступу користувачів до систем, мереж, додатків та інших ресурсів. Ось їхні визначення:

Ідентифікація: це процес встановлення ідентичності суб'єкта, наприклад, користувача, комп'ютера або програми. У контексті користувачів це може бути, наприклад, ім'я користувача, електронна адреса або номер користувача. В ідеальному випадку кожен користувач має унікальну ідентичність в системі.

Автентифікація - це процес перевірки ідентичності суб'єкта. У реальному світі це еквівалентно пред'явленню документів для перевірки особи. У цифровому світі це може бути введення пароля, біометричних даних, таких як відбитки пальців чи розпізнавання обличчя, або використання інших механізмів перевірки.

Авторизація - це процес визначення прав доступу суб'єкта після успішної автентифікації. Після того як користувач був ідентифікований та автентифікований, авторизація визначає, які дії він може виконувати та до яких ресурсів він має доступ. Наприклад, один користувач може мати доступ лише для читання до певного файлу, тоді як інший може мати право на його зміну або видалення.

Саме методи автентифікації є ключовим елементом у захисті додатків від несанкціонованого доступу. Серед найпоширеніших методів автентифікації можна назвати:

• Логін та пароль: Найбільш поширений метод, який вимагає від користувачів введення ідентифікатора (логіну) та пароля для аутентифікації.

• Двофакторна автентифікація (2FA): Використовує додатковий рівень безпеки, де після введення логіну та пароля користувач також повинен підтвердити

свою особу за допомогою додаткового механізму, такого як одноразовий код або біометричні дані.

- Мультифакторна аутентифікація (MFA): Працює подібно до двофакторної автентифікації, але використовує більше ніж два фактори для підтвердження ідентичності користувача.
- Автентифікація на основі біометричних даних: Використовує унікальні фізичні характеристики користувача, такі як відбитки пальців, розпізнавання обличчя або сканування радужки для підтвердження особи. [1]

1.3 Типи біометричної автентифікації

У світі швидко зростає інтерес саме до біометричних технологій як засобу автентифікації, тобто ідентифікації та верифікації особи на основі її фізіологічних чи поведінкових характеристик. Цей підхід до перевірки особи є ефективним і зручним, оскільки він виключає необхідність запам'ятовування паролів або використання карток доступу. Серед найпоширеніших методів біометричної автентифікації можна виділити наступні:

1) Розпізнавання відбитків пальців - технологія, що використовує унікальні візерунки на пальцях кожної людини для підтвердження її особи. Система розпізнавання відбитків пальців може використовуватися як для верифікації, так і для ідентифікації. При верифікації система порівнює вхідний відбиток пальця з «зареєстрованим» відбитком пальця конкретного користувача, щоб визначити, чи належать вони одному і тому ж пальцю (збіг 1:1). Під час ідентифікації система порівнює вхідний відбиток пальця з відбитками всіх зареєстрованих користувачів у базі даних, щоб визначити, чи не є ця особа вже відомою під дублікатом або фальшивим ім'ям (збіг 1:N). Виявлення багаторазових реєстрацій, коли одна й та сама особа отримує кілька облікових даних, наприклад паспорт на різні імена, вимагає негативної ідентифікаційної функціональності відбитків пальців.[3] Також ця технологія є найпоширенішою від усіх і використовується майже у всіх сучасних мобільних додатках та пристроях, у правоохоронних органах, у банках та інших галузях.

2) Розпізнавання обличчя – технологія, що використовує алгоритми для аналізу унікальних характеристик обличчя людини, таких як форма, розташування очей та рота і порівнює їх зі збереженими базами даних, щоб ідентифікувати або верифікувати особу. Ця технологія широко використовується в системах безпеки та розпізнавання на відеокамерах в аеропортах для покращення безпеки та у багатьох мобільних додатках. Серед відомих українських додатків, що використовують цю технологію можна згадати: Дія, ПриватБанк та МоноБанк.

3) Сканування райдужної оболонки ока – технологія, що базується на аналізі унікальних особливостей структури райдужної оболонки ока людини, таких як її колір і текстура. Оскільки райдужна оболонка ока не змінюється протягом усього життя людини, ця технологія може вважатись надійним методом автентифікації.

4) Розпізнавання голосу – технологія, що використовує тембр, частоту та інтонацію людини для підтвердження особи. Ця технологія також широко застосовується в різних галузях, включаючи усі зазначені раніше.

5) Розпізнавання вен – одна з найновіших технологій, яка використовує унікальний для кожної людини візерунок вен на пальцях або долоні. Зараз ця технологія широко розповсюджується у Китаї і навіть використовується для сплати проїзду у метро Шанхая [11]. Так як вона надає дуже високий рівень запобігання хибним спрацьовуванням, то вже зараз можна прогнозувати і подальше поширення цієї технології у різноманітні сфери безпеки. [4]

1.4 Використані мова програмування та бібліотеки

1.4.1 Мова програмування Java

Обравши мову програмування Java для розробки своєї програми, я керувався кількома ключовими факторами, що підтримують моє рішення:

1) Велика кількість доступних бібліотек: Java має вражаючий екосистему бібліотек і фреймворків, які сприяють розробці широкого спектру програмних продуктів. Це дозволяє значно скоротити час розробки, оскільки багато функціональності можна легко і швидко реалізувати, використовуючи готові рішення.

2) Популярність: Java є однією з найпопулярніших мов програмування у світі. Це означає, що вона має велику спільноту розробників, що забезпечує доступ до великої кількості досвіду, порад та підтримки. Крім того, популярність Java також означає наявність великої кількості документації, уроків та інших ресурсів для вивчення мови та розробки програм.

3) Переносимість: Java відома своєю високою переносимістю коду, оскільки програми, написані на Java, можуть запускатися на будь-якому пристрої, який підтримує відповідний віртуальний машину Java (JVM). Це робить Java привабливим вибором для розробки програм, які мають бути розгорнуті на різних платформах та пристроях.

Загалом, обираючи Java для розробки програми, я врахував її широкі можливості та велику підтримку, що відповідає потребам мого проекту.

1.4.2 Бібліотека machinezoo.sourceafis

У моїй дипломній роботі для отримання потрібних метрик була використана бібліотека SourceAFIS, яка є алгоритмом розпізнавання відбитків пальців людини. Цей алгоритм здатний порівнювати два відбитки пальців 1:1 або здійснювати пошук збігів у великій базі даних 1:N. На вході він приймає зображення відбитків пальців, а на виході надає оцінку схожості, яку порівнюють з порогом збігу, що налаштовується. [5]

Ця бібліотека має дві майже ідентичні реалізації з відкритим вихідним кодом на чистій Java та чистій .NET. API SourceAFIS був розроблений для максимальної простоти та зручності використання розробниками додатків. Його точність та швидкість зіставлення виявляються достатніми для більшості додатків.

Варто зауважити, що алгоритм SourceAFIS є результатом незалежної розробки, хоча він в значній мірі запозичує з інших алгоритмів з відкритим вихідним кодом. Це забезпечує пристойну точність і високу швидкість співставлення.

Завдяки своєму відкритому вихідному коду SourceAFIS надає рідкісну можливість прозорості алгоритму, що відкриває шлях для цікавих застосувань, які раніше були неможливі з комерційними біометричними рішеннями.

1.4.3 Бібліотека aspose.cells

Також для взаємодії Java та MS Excel у своєму проекті я використовував бібліотеку Aspose.Cells.

Aspose.Cells для Java - це високопродуктивна бібліотека, спеціально створена для роботи з електронними таблицями в Java-додатках. Ця бібліотека надає розширені можливості для зчитування, запису, редагування та маніпулювання даними у форматах Excel, CSV та інших. [6]

Основні функції Aspose.Cells включають:

1) Читання та запис даних: За допомогою Aspose.Cells ви можете легко зчитувати дані з Excel-файлів та записувати їх у різних форматах, включаючи XLS, XLSX, CSV та HTML.

2) Редагування та форматування: Ви маєте можливість редагувати та форматувати дані в електронних таблицях, виконуючи операції, такі як додавання та видалення аркушів, стовпців, рядків, копіювання, вставка, об'єднання та розділення комірок, налаштування стилів, шрифтів та колірного форматування.

3) Робота з формулами: Aspose.Cells підтримує широкий спектр вбудованих Excel-формул, що дозволяє виконувати обчислення в електронних таблицях. Ви також можете вставляти, оновлювати та видаляти формули в комірках.

4) Обробка діаграм та графіків: Бібліотека дозволяє створювати, редагувати та маніпулювати діаграмами та графіками безпосередньо з коду Java.

5) Розширені функції: Aspose.Cells підтримує такі функції, як захист аркушів паролем, заборона редагування комірок, робота з макросами та інші розширені опції.

6) Експорт та імпорт даних: Ви можете легко експортувати дані з Excel у форматах PDF, XML, TIFF, SVG, PNG, JPEG, BMP та імпортувати дані з цих форматів.

2 ОГЛЯД АЛГОРИТМУ РОБОТИ СИСТЕМ РОЗПІЗНАВАННЯ ВІДБИТКІВ ПАЛЬЦІВ

2.1 Алгоритм роботи системи розпізнавання відбитків пальців

На рисунку 2.1 зображено схему роботи типової автоматизованої системи розпізнавання відбитків пальців. [10]

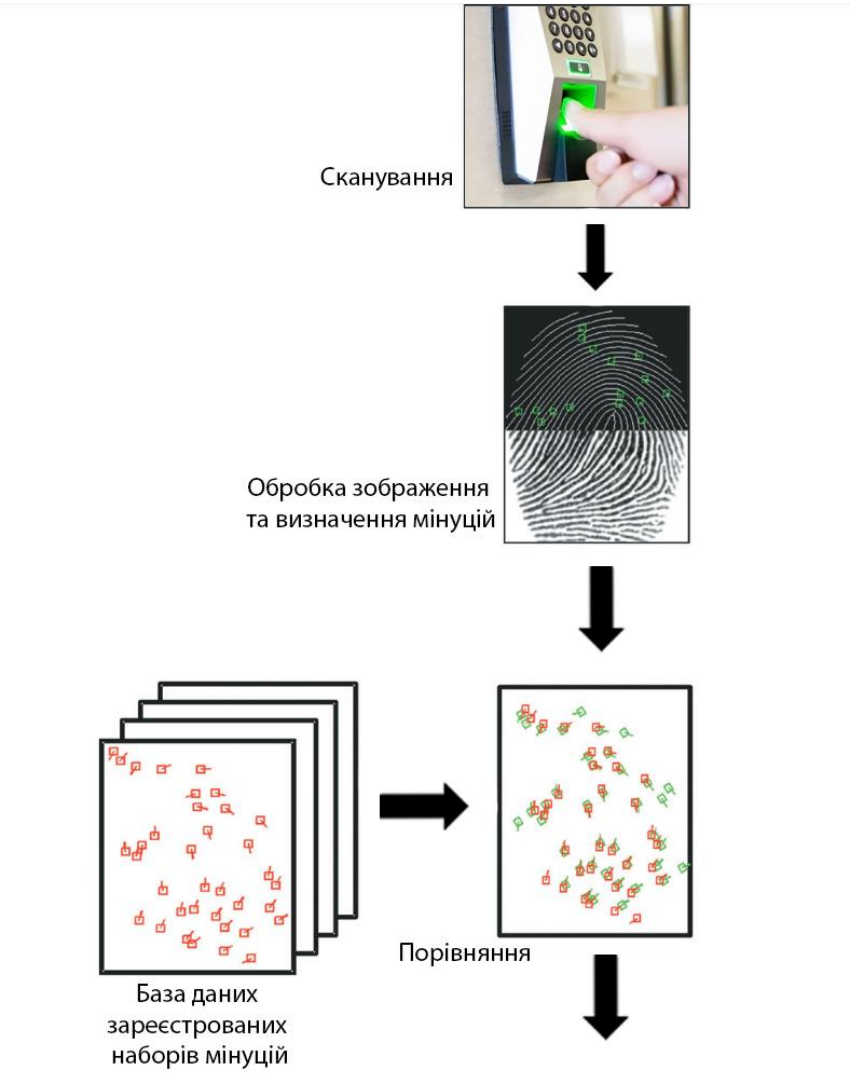


Рисунок 2.1 – Схема роботи автоматизованої системи розпізнавання відбитків пальців.

Під час реєстрації датчик сканує відбиток пальця користувача та перетворює його на цифрове зображення. Екстрактор мінуцій аналізує це зображення для виявлення характерних особливостей, які дозволяють розрізняти користувачів. Мінуції - це точки, де гребінці тертя різко закінчуються або розгалужуються.

Зображення відбитка пальця хорошої якості зазвичай містить близько 20-70 таких точок; їх кількість залежить від розміру датчика і того, як користувач прикладає палець до нього. Система зберігає інформацію про виявлені мінуції - їх розташування та напрямок - разом з даними про користувача у базі даних реєстрації. [10]

Під час ідентифікації користувач знову торкається того ж датчика, створюючи нове зображення відбитка, яке називається відбитком запиту. З цього відбитка мінуції також витягуються за допомогою того самого алгоритму, і модуль зіставлення порівнює набір мінуцій запиту зі збереженими шаблонами в базі даних, щоб знайти спільні точки.

Через різницю у розташуванні пальця і тиску на сенсор, мінуції з шаблону і відбитка запиту потрібно вирівняти перед порівнянням. Після вирівнювання система визначає кількість пар точок, що збігаються за розташуванням та напрямком. Користувач ідентифікується, якщо кількість збігів перевищує порогове значення, встановлене адміністратором.

2.1.1 Обробка зображення та визначення мінуцій

Елементи, витягнуті з зображення відбитків пальців, поділяються на три рівні. Ознаки першого рівня включають макродеталі, такі як потік гребеня, тип візерунка та особливі точки. Ознаки другого рівня стосуються дрібних деталей, таких як біфуркації та закінчення гребеня. Ознаки третього рівня включають всі розмірні атрибути гребеня, такі як відхилення траєкторії, ширина, форма, пори, контур країв та інші деталі, включаючи початкові гребені, складки та рубці.

Ознаки першого рівня дозволяють класифікувати відбитки пальців за основними типами візерунків, такими як арка, петля або завиток; ознаки другого і третього рівнів використовуються для встановлення індивідуальності або унікальності відбитків пальців. Елементи вищих рівнів можна вилучити лише з високоякісних зображень відбитків пальців. Наприклад, для вилучення ознак третього рівня потрібна роздільна здатність понад 500 точок на дюйм.

На рисунку 2.2 представлено блок-схему типового алгоритму вилучення мінуцій. [10]

Рисунок 2.2 – Блок-схема типового алгоритму вилучення мінуцій

Спочатку алгоритм оцінює орієнтацію та частоту гребенів на зображенні. На основі цих даних він виконує контекстну фільтрацію для покращення якості зображення та виділення гребенів. Потім алгоритм створює бінарні скелети гребенів з покращеного зображення, відстежуючи їхні лінії. Кінці гребенів і точки біфуркації виділяються зі скелету гребенів і називаються мінуціями. Алгоритм використовує евристичні правила для виявлення та видалення помилкових мінуцій, які можуть виникнути через недосконаліть зображення скелета. [10]

2.1.2 Порівняння відповідності отриманих мінуцій

Модуль зіставлення відбитків пальців обчислює коефіцієнт відповідності між двома відбитками, який повинен бути високим для відбитків того самого пальця і низьким для відбитків різних пальців. Зіставлення відбитків пальців є складною задачею розпізнавання шаблонів через значні внутрішньокласові варіації (різниця в зображеннях відбитків того самого пальця) та значну міжкласову схожість (схожість між зображеннями відбитків різних пальців). Внутрішньокласові варіації виникають через тиск пальця та його розміщення — обертання, зміщення та площу контакту зі сенсором, а також стан шкіри, наприклад, сухість або порізи. Міжкласова схожість може бути великою, оскільки існує лише три основних типи візерунків відбитків пальців (арка, петля та завиток).

Сучасний підхід у зіставленні мінуцій полягає у використанні локальних структур мінуцій для швидкого визначення грубого вирівнювання між двома відбитками, а потім консолідації результатів локального зіставлення на глобальному рівні. Такий алгоритм зазвичай включає чотири етапи:

Спершу алгоритм обчислює парну схожість між мінуціями двох відбитків, порівнюючи описи мінуцій, які залишаються незмінними при обертанні та зміщенні. Потім він вирівнює два відбитки відповідно до найбільш схожої пари мінуцій. Далі алгоритм встановлює відповідність мінуцій, де мінуції, які достатньо близькі за розташуванням та напрямком, вважаються спареними. І нарешті, алгоритм обчислює коефіцієнт схожості, щоб відобразити ступінь відповідності між двома відбитками на основі таких факторів, як кількість збігшихся мінуцій, їх відсоток в області перекриття двох відбитків та узгодженість кількості гребенів між мінуціями, що збіглись. [9]

$$\text{Matching score } = \frac{100N_{\text{pair}}}{\max\{M, N\}}$$

2.2 Основні показники точності для біометричних систем

Система зіставлення відбитків пальців може допускати два типи помилок: хибне прийняття, коли вона оголошує збіг між зображеннями двох різних пальців, і хибна відмова, коли вона не ідентифікує зображення одного і того ж пальця як збіг. Частота хибного прийняття (FAR) і частота хибних відмов системи (FRR) системи на пряму залежать від робочого порогу значень, велике значення якого призводить до низького FAR за рахунок високого FRR (Рисунок 2.3). [9]

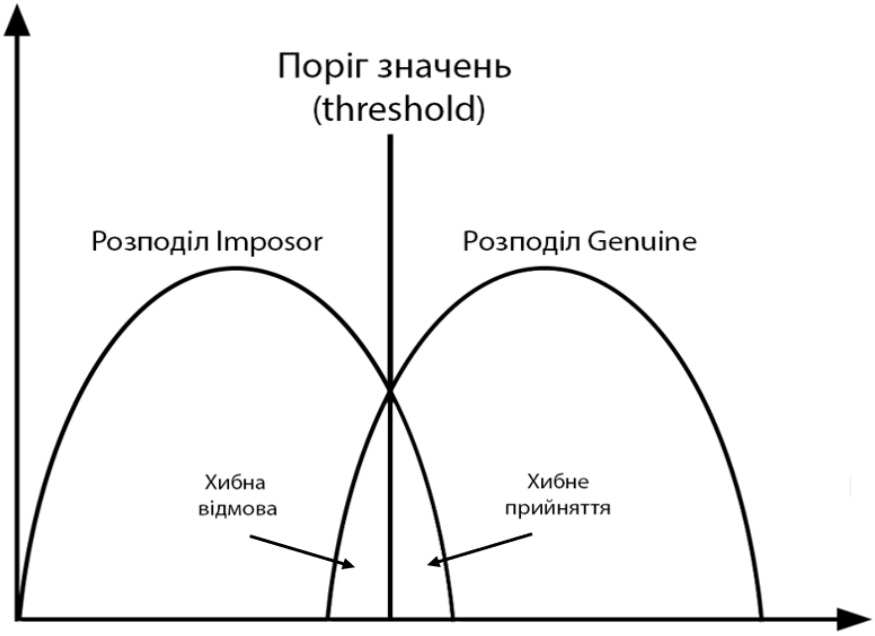


Рисунок 2.3 – Розподіл значень Impostor та Genuine для типової біометричної системи

У біометричних системах ідентифікації та верифікації особистості важливо оцінювати точність та надійність методів, що використовуються для розпізнавання. Основними показниками точності є False Acceptance Rate (FAR), False Rejection Rate (FRR) та Equal Error Rate (EER) (Рисунок 2.4).

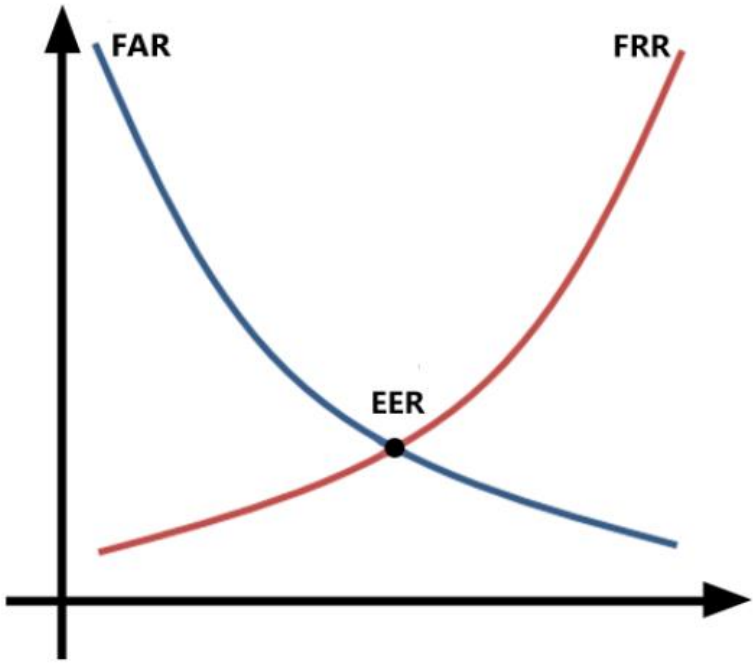


Рисунок 2.4 – Ілюстрація залежності між значеннями FAR та FRR

False Acceptance Rate (FAR), або рівень хибного прийняття, визначає відсоток випадків, коли біометрична система помилково ідентифікує або верифікує неавторизованого користувача як авторизованого. Іншими словами, FAR відображає ймовірність того, що система допускає доступ користувачеві, який не має на це прав. [12]

$$FAR(\%) = \frac{\text{Impostor accepted as a Genuine user}}{\text{all Impostor}} * 100$$

False Rejection Rate (FRR), або рівень хибного відхилення, визначає відсоток випадків, коли біометрична система помилково відхиляє доступ авторизованого користувача. Це показує ймовірність того, що система не визнає користувача, який має право доступу. Високе значення FRR може призводити до незручностей для користувачів, які повинні будуть повторно проходити процедуру ідентифікації або верифікації. [12]

$$FRR(\%) = \frac{\text{Genuine user treated as impostor}}{\text{all Genuine users}} * 100$$

Equal Error Rate (EER) є важливим показником для оцінки загальної ефективності біометричної системи. Це точка, в якій графіки FAR і FRR перетинаються. EER використовується як узагальнений показник продуктивності системи: чим нижче значення EER, тим вищою є точність системи. Він дозволяє легко порівнювати різні біометричні системи між собою, незалежно від налаштувань порогових значень.

FAR і FRR взаємопов’язані: зниження одного показника зазвичай призводить до збільшення іншого. Це означає, що необхідно знайти оптимальний баланс між цими показниками для досягнення найкращої продуктивності системи. Налаштування порогу розпізнавання (threshold) дозволяє регулювати рівні FAR і FRR. При підвищенні порогу знижується FAR, але підвищується FRR, і навпаки. EER є корисним індикатором для знаходження цього балансу, оскільки він відображає точку, де обидва показники рівні.

EER системи може бути використаний для незалежної від порогу оцінки ефективності. Чим нижчий EER, тим краща продуктивність системи, оскільки загальний рівень помилок, який є сумою FAR і FRR в точці EER, зменшується.

Теоретично це працює добре, якщо EER системи розраховується за допомогою нескінченного та репрезентативного тестового набору, що, звісно, неможливо в реальних умовах. Щоб отримати порівняльні результати, необхідно, щоб EER, які порівнюються, були розраховані на основі одних і тих самих тестових даних, використовуючи один і той самий тестовий протокол. [9]

$$EER(\%) = \frac{\text{FAR} + \text{FRR}}{2}$$

2.3 Обрана база даних для дослідження

2.3.1 Опис бази даних відбитків пальців FVC2004

База даних FVC2004 (Fingerprint Verification Competition 2004) є всеосяжною колекцією зображень відбитків пальців, спеціально створеною для бенчмаркінгу алгоритмів верифікації відбитків пальців. Вона була представлена в рамках конкурсу FVC2004, метою якого було оцінити та порівняти продуктивність різних систем розпізнавання відбитків пальців. База даних складається з чотирьох окремих наборів даних, кожен з яких містить відбитки пальців, зібрані за різних умов, щоб імітувати реальні сценарії.

Основні характеристики бази даних FVC2004:

1) Набори даних:

- DB1: Захоплені за допомогою оптичного сенсора.
- DB2: Захоплені за допомогою недорогого оптичного сенсора.
- DB3: Захоплені за допомогою теплового сенсора.
- DB4: Синтетично згенеровані відбитки пальців.

2) Склад наборів даних:

- Кожен набір містить відбитки 100 різних пальців.
- Для кожного пальця доступно по 8 зображень.
- Це призводить до загальної кількості в 800 зображень відбитків пальців на кожен набір.

3) Специфікації зображень:

- Роздільна здатність: Зображення мають роздільну здатність 500 dpi.

- Розмір: Стандартний розмір 388x374 пікселів, хоча він може дещо варіюватися між різними наборами даних.

4) Варіабельність:

- Зображення включають варіації в якості відбитків через такі фактори, як сухість, вологість та зміни тиску під час захоплення.
- Ця варіабельність робить базу даних придатною для тестування стійкості алгоритмів верифікації відбитків пальців.

5) Призначення:

- Створена для полегшення оцінки та порівняння систем верифікації відбитків пальців.
- Допомагає у розробці більш точних та надійних технологій розпізнавання відбитків пальців.

6) Доступність:

- База даних є публічно доступною для дослідницьких цілей.
- Дослідники можуть отримати базу даних, дотримуючись вказівок, наданих організаційним комітетом FVC.

Використання:

База даних FVC2004 широко використовується дослідниками та розробниками у сфері біометричної безпеки для:

- Розробки та тестування алгоритмів.
- Бенчмаркінгу продуктивності систем розпізнавання відбитків пальців.
- Академічних досліджень та публікацій.

Надаючи стандартизований набір зображень відбитків пальців, FVC2004 забезпечує справедливе та послідовне порівняння різних технік верифікації відбитків пальців, сприяючи прогресу в технологіях біометричної аутентифікації.

2.3.2 Опис набору даних FVC2004 DB4_B

Саме у роботі був використаний набір даних FVC2004 DB4_B. Цей набір є частиною бази даних FVC2004, яка призначена для бенчмаркінгу алгоритмів верифікації відбитків пальців.

Основні характеристики:

1) Тип даних:

- DB4_B: Синтетично згенеровані відбитки пальців.
- Цей набір створений для тестування алгоритмів у контрольованих умовах, що дозволяє точніше оцінити їх ефективність.

2) Склад набору даних:

- Набір включає відбитки 10 пальців, номери яких з 101 до 110.
- Для кожного пальця доступно по 8 зображень, що дозволяє перевірити стійкість алгоритмів до різних втілень одного й того самого відбитка.

3) Специфікації зображень:

- Роздільна здатність: 500 dpi.
- Розмір зображень: Стандартний розмір 388x374 пікселів.

Використання у роботі:

Використання саме набору DB4_B було обрано через його синтетичну природу, яка забезпечує однорідність та контрольованість умов. Це дозволяє більш точно оцінити продуктивність алгоритмів верифікації відбитків пальців без впливу зовнішніх факторів, таких як зміни в середовищі або умови захоплення зображення. Таким чином, результати тестування є більш повторюваними і надійними, що сприяє об'єктивному порівнянню різних підходів та методів.

Переваги використання DB4_B:

- Контрольованість умов: Завдяки синтетичному походженню відбитків, набір забезпечує високий рівень контрольованості тестових умов.
- Повторюваність результатів: Відсутність випадкових факторів, які можуть впливати на якість зображень, дозволяє отримати більш стабільні результати тестування.
- Різноманітність втілень: Наявність кількох зображень для кожного пальця дозволяє перевірити алгоритми на стійкість до різних втілень одного й того самого відбитка.

3 ПРАКТИЧНЕ ДОСЛІДЖЕННЯ ВРАЗЛИВОСТІ

3.1 Apache Maven

Для розробки мого проекту на Java я обрав використання Apache Maven, оскільки цей інструмент забезпечує численні переваги, які значно сприяють ефективності, стандартизації та керованості процесу розробки. Нижче наведено основні причини, чому було прийнято таке рішення.

1) Управління залежностями. Однією з ключових причин вибору Maven є його потужна система управління залежностями. Завдяки Maven repository, розробники мають доступ до величезного сховища загальнодоступних бібліотек та фреймворків, які можна легко інтегрувати у проект. Maven автоматично завантажує та включає необхідні залежності, враховуючи їх сумісність та конфлікти версій, що значно спрощує процес управління бібліотеками та залежностями.

2) Стандартизація структури проекту. Maven сприяє стандартизації структури проекту, що забезпечує однаковість та передбачуваність в організації файлів та каталогів. Завдяки чітко визначеній структурі проекту, нові члени команди можуть швидше орієнтуватися у кодовій базі, що сприяє підвищенню продуктивності та зменшенню часу на навчання. Це також полегшує інтеграцію з іншими інструментами та середовищами розробки.

3) Автоматизація збірки. Maven забезпечує автоматизацію процесу збірки проекту, включаючи компіляцію коду, виконання тестів, створення документації та розгортання. Це зменшує ризик помилок, пов'язаних з ручними процесами, та підвищує ефективність розробки. Автоматизація також дозволяє легко інтегрувати проект з системами безперервної інтеграції (CI/CD), що є важливим аспектом сучасної розробки програмного забезпечення.

4) Управління версіями та залежностями. Однією з важливих переваг Maven є його здатність керувати версіями та залежностями проекту. Maven POM (Project Object Model) файл дозволяє чітко визначити версії бібліотек, що використовуються, та їхні взаємозалежності. Це зменшує ризик конфліктів версій

та полегшує оновлення залежностей, забезпечуючи стабільність та надійність проекту.

5) Підтримка спільноти та документація. Maven має велику спільноту користувачів та розробників, а також обширну документацію, що полегшує вирішення проблем та пошук відповідей на питання, які можуть виникати під час розробки. Наявність великої кількості плагінів та інтеграцій також розширює можливості Maven, дозволяючи адаптувати його для різних потреб та вимог проекту.

Загалом, вибір Maven для розробки проекту на Java був зумовлений його потужними можливостями управління залежностями через Maven repository, стандартизацією структури проекту, автоматизацією процесу збірки, ефективним управлінням версіями та широкою підтримкою спільноти. Усі ці фактори сприяють підвищенню продуктивності, надійності та ефективності процесу розробки програмного забезпечення.

3.2 Розроблене програмне рішення

3.2.1 Імпорт бібліотек та оголошення класу

Перший сегмент коду (Рисунок 3.1) виконує ключову роль у підготовці середовища для подальшої обробки відбитків пальців. Імпорти забезпечують доступ до необхідних бібліотек для роботи з файлами, відбитками пальців та Excel документами. Оголошення класу Main і константи FilePath задають структуру програми та налаштовують основні параметри, необхідні для подальшого виконання завдань.

```
1   import com.machinezoo.sourceafis.*;
2   import java.io.IOException;
3   import java.nio.file.Files;
4   import java.nio.file.Paths;
5   import java.util.*;
6   import com.aspose.cells.*;
7
8   public class Main {  int9p *
9       private static final String FilePath = "resources/DB4_B/";  4 usages
```

Рисунок 3.1 – Імпорт бібліотек та оголошення класу

1) Імпорт бібліотек:

- com.machinezoo.sourceafis: пакет, що містить класи та методи для роботи з біометричними даними, зокрема з відбитками пальців.
- java.io.IOException: Ця бібліотека використовується для обробки виключень, пов'язаних з введенням та виведенням даних. Винятки типу IOException можуть виникати під час читання файлів відбитків пальців.
- java.nio.file.Files та java.nio.file.Paths: Ці класи використовуються для роботи з файлами та шляхами до файлів у файловій системі. Вони забезпечують прості та ефективні способи читання файлів у байтові масиви, що необхідно для подальшої обробки відбитків пальців.
- java.util.*: Цей імпорт включає різні утиліти, зокрема колекції (наприклад, Map, HashMap, List, ArrayList, TreeMap), які використовуються для зберігання та обробки даних у програмі.
- com.aspose.cells: Бібліотека Aspose.Cells використовується для роботи з Excel файлами. Вона дозволяє створювати, редагувати та зберігати Excel файли програмно, що необхідно для збереження результатів тестування у форматі Excel.

2) Оголошення класу Main:

- public class Main: Оголошення основного класу програми. Клас Main містить метод main, який є точкою входу в програму.
- private static final String FilePath: Оголошення константи FilePath, яка містить шлях до директорії з базою даних відбитків пальців. Використання константи полегшує зміну шляху до файлів, якщо це необхідно, без потреби змінювати код у багатьох місцях.

3.2.2 Оголошення методу main та основних змінних

Другий сегмент коду (Рисунок 3.2) виконує важливу функцію підготовки змінних та середовища для подальшої обробки бази даних відбитків пальців. Змінні dictionary, metricsGenuine і metricsImpostor забезпечують структури для зберігання результатів порівняння. Налаштування опцій зображень відбитків пальців гарантує коректну обробку даних з відповідною роздільною здатністю. Встановлення

порогового значення схожості визначає критерій, за яким два відбитки пальців вважаються відповідними. Ініціалізація допоміжних змінних забезпечує основу для порівняльного процесу в подальшому коді.

```
public static void main(String[] args) throws IOException {  int9p *
    // Initialization
    Map<String, Integer> dictionary = new HashMap<>();
    List<Double> metricsGenuine = new ArrayList<>();
    List<Double> metricsImpostor = new ArrayList<>();
    var options = new FingerprintImageOptions().dpi(500);
    double threshold = 40;
    boolean matches = false;
    int count = 0;
```

Рисунок 3.2 – Оголошення методу main та основних змінних

1) Оголошення методу main:

- public static void main(String[] args): Це стандартний метод входу для Java-програм. Він є статичним методом, який приймає масив рядків як аргументи командного рядка.
- throws IOException: Метод може викидати виключення типу IOException, яке виникає при помилках введення-виведення. Це необхідно для обробки потенційних помилок, пов'язаних з читанням файлів відбитків пальців.

2) Ініціалізація словника для збереження результатів:

- Map<String, Integer> dictionary: Оголошення змінної dictionary, яка є хеш-картою (HashMap). Ця колекція буде використовуватися для зберігання кількості збігів для кожного файлу відбитків пальців. Ключем є рядок, що представляє ім'я файлу відбитка пальця, а значенням - кількість збігів.
- new HashMap<>(): Створення нового об'єкта HashMap.

3) Ініціалізація списків для метрик схожості:

- List<Double> metricsGenuine: Оголошення списку metricsGenuine, який буде зберігати метрики схожості для тесту справжності відбитків пальців (Genuine Test).
- List<Double> metricsImpostor: Оголошення списку metricsImpostor, який буде зберігати метрики схожості для тесту імпосторів (Impostor Test).

- new ArrayList<>(): Створення нових об'єктів ArrayList.

4) Налаштування опцій для зображень відбитків пальців:

- var options: Оголошення змінної options з автоматичним визначенням типу (інтерпретується як FingerprintImageOptions).
- new FingerprintImageOptions().dpi(500): Створення об'єкта FingerprintImageOptions та встановлення роздільної здатності зображень відбитків пальців у 500 точок на дюйм (dpi). Це необхідно для правильного масштабу та обробки зображень.

5) Встановлення порогового значення схожості:

- double threshold: Оголошення змінної threshold, яка зберігає порогове значення схожості для визначення, чи є два відбитки пальців відповідними (matches). Значення 40 було вибрано як умовний поріг для тестування.

6) Ініціалізація допоміжних змінних:

- boolean matches: Оголошення змінної matches, яка буде використовуватися для зберігання результату порівняння двох відбитків пальців. Вона приймає значення true, якщо схожість між відбитками перевищує поріг, і false в іншому випадку.
- int count: Оголошення змінної count, яка використовується для підрахунку кількості порівнянь. У поточному сегменті вона не використовується, але може бути застосована у подальших розрахунках або для відладки.

3.2.3 Ініціалізація зразків відбитків пальців та порівняння

Третій сегмент коду (Рисунок 3.3) виконує критично важливу функцію ініціалізації зразків відбитків пальців та їх порівняння. На цьому етапі:

- Зчитуються зображення відбитків пальців з файлів та перетворюються у відповідні об'єкти FingerprintTemplate.
- Створюється об'єкт FingerprintMatcher, ініціалізований зразком відбитка пальця.
- Виконується порівняння зразка та кандидата, результатом якого є числове значення схожості.

Цей процес є основою для подальших операцій з аналізу метрик схожості та визначення збігів між відбитками пальців.

```
// Load probe and candidate fingerprints
var probe = new FingerprintTemplate(
        new FingerprintImage(Files.readAllBytes(Paths.get(
                first: "resources/DB1_B/101_1.tif"))));
var candidate = new FingerprintTemplate(
        new FingerprintImage(Files.readAllBytes(Paths.get(
                first: "resources/DB1_B/101_2.tif"))));

var matcher = new FingerprintMatcher(probe);
double similarity = matcher.match(candidate);
```

Рисунок 3.3 – Ініціалізація зразків відбитків пальців та порівняння

1) Ініціалізація зразка відбитка пальця (probe):

- var probe: Оголошення змінної probe з автоматичним визначенням типу (інтерпретується як FingerprintTemplate).
- new FingerprintTemplate(...): Створення нового об'єкта FingerprintTemplate, який представляє шаблон відбитка пальця. Цей шаблон містить екстраговані риси відбитка, які будуть використовуватися для порівняння.
- new FingerprintImage(...): Створення нового об'єкта FingerprintImage, який представляє зображення відбитка пальця.
- Files.readAllBytes(Paths.get("resources/DB1_B/101_1.tif")): Читання байтів зображення відбитка пальця з файлу 101_1.tif у директорії resources/DB1_B. Метод Files.readAllBytes зчитує всі байти файлу і повертає їх у вигляді масиву байтів. Метод Paths.get перетворює рядок, що представляє шлях до файлу, у об'єкт Path.

2) Ініціалізація кандидатського відбитка пальця (candidate):

- var candidate: Оголошення змінної candidate з автоматичним визначенням типу (інтерпретується як FingerprintTemplate).
- new FingerprintTemplate(...): Створення нового об'єкта FingerprintTemplate, що представляє шаблон кандидатського відбитка пальця.

- new FingerprintImage(...): Створення нового об'єкта FingerprintImage, що представляє зображення кандидатського відбитка пальця.
- Files.readAllBytes(Paths.get("resources/DB1_B/101_2.tif")): Читання байтів зображення відбитка пальця з файлу 101_2.tif у директорії resources/DB1_B.

3) Ініціалізація об'єкта для порівняння відбитків пальців (matcher):

- var matcher: Оголошення змінної matcher з автоматичним визначенням типу (інтерпретується як FingerprintMatcher).
- new FingerprintMatcher(probe): Створення нового об'єкта FingerprintMatcher, що ініціалізується зразком відбитка пальця (probe). Цей об'єкт використовуватиметься для порівняння зразка з іншими відбитками пальців.

4) Порівняння відбитків пальців:

- double similarity: Оголошення змінної similarity, яка зберігатиме значення схожості між зразком (probe) та кандидатським відбитком пальця (candidate).
- matcher.match(candidate): Виклик методу match об'єкта matcher, який порівнює шаблон зразка з шаблоном кандидата. Результатом є числове значення схожості, яке відображає, наскільки подібні два відбитки пальців. Чим вище це значення, тим більше схожі відбитки.

3.2.4 Проведення тестів Genuine

Четвертий сегмент коду (Рисунок 3.4) відповідає за виконання тесту Genuine, який перевіряє ефективність алгоритму розпізнавання відбитків пальців при порівнянні відбитків одного і того ж пальця (тобто справжніх пар відбитків). Після виконання тесту результати зберігаються в словнику та виводяться в консоль.

```
// 1. Genuine Test
for (int pi = 101; pi <= 110; pi++) {
    for (int pj = 1; pj <= 8; pj++) {
        dictionary.put(pi + "_" + pj, 0);
        probe = new FingerprintTemplate(new FingerprintImage(Files.readAllBytes(Paths.get(
                first: FilePath + pi + "_" + pj + ".tif"))));

        for (int cj = 1; cj <= 8; cj++) {
            if (cj == pj) {
                continue;
            }
            candidate = new FingerprintTemplate(new FingerprintImage(Files.readAllBytes(Paths.get(
                    first: FilePath + pi + "_" + cj + ".tif"))));
            matcher = new FingerprintMatcher(probe);
            similarity = matcher.match(candidate);
            if (similarity != 0) {
                metricsGenuine.add(similarity);
            }
            matches = similarity >= threshold;
            System.out.println("Matching " + pi + "_" + pj + ".tif with " + pi + "_" + cj + ".tif");
            if (matches) {
                System.out.println(similarity + "\nResult : Fingerprints matched!\n");
                dictionary.put(pi + "_" + pj, dictionary.get(pi + "_" + pj) + 1);
            } else {
                System.out.println(similarity + "\nResult : No match!\n");
            }
        }
    }
}

// Print Genuine Test Results
System.out.println("\nGenuine test with threshold " + threshold + ":");
TreeMap<String, Integer> sortedDictionaryGenuine = new TreeMap<>(dictionary);
System.out.println("[FP name]\t[Matches]");
System.out.println("---------------------");
for (Map.Entry<String, Integer> entry : sortedDictionaryGenuine.entrySet()) {
    System.out.println(" " + entry.getKey() + "\t\t\t" + entry.getValue());
    System.out.println("---------------------");
}
```

Рисунок 3.4 – Проведення тестів Genuine

1) Початок тесту Genuine:

- Цикли for: Два вкладених цикли for ітеруються через діапазони значень pi від 101 до 110 та pj від 1 до 8. Ці значення представляють номери файлів відбитків пальців.

- dictionary.put(pi + "_" + pj, 0): Ініціалізуємо словник dictionary ключем, що складається з номерів pi та pj, з початковим значенням 0. Цей словник буде

використовуватись для підрахунку кількості збігів для кожного файлу відбитка пальця.

- probe: Створюємо шаблон відбитка пальця probe з файлу pi + "_" + pj +".tif".

2) Внутрішній цикл для порівняння зразків:

- int ci = pi: Встановлюємо ci рівним pi, оскільки тест на справжність передбачає порівняння зразків відбитків пальців одного і того ж індивідуума.
- if (cj == pj): Пропускаємо порівняння, якщо індекси зразків збігаються (уникнення порівняння самого з собою).
- candidate: Створюємо шаблон кандидатського відбитка пальця candidate з файлу ci + "_" + cj +".tif".
- matcher = new FingerprintMatcher(probe): Створюємо новий об'єкт FingerprintMatcher, ініціалізований зразком probe.
- similarity = matcher.match(candidate): Виконуємо порівняння зразка probe з кандидатом candidate і отримуємо значення схожості.
- if (similarity != 0): Додаємо значення схожості до списку metricsGenuine, якщо воно не дорівнює нулю.
- matches = similarity >= threshold: Визначаємо, чи є схожість між зразками достатньою для визнання їх відповідними (порівняння з порогом).
- System.out.println(...): Виводимо результати порівняння.
- Оновлення словника: Якщо зразки відповідають (matches == true), оновлюємо значення у словнику для відповідного ключа, збільшуючи його на 1.

3) Вивід результатів тесту Genuine:

- TreeMap<String, Integer> sortedDictionaryGenuine = new TreeMap<>(dictionary): Створюємо об'єкт TreeMap для сортування словника за ключами.
- Вивід результатів: Виводимо на екран результати тесту на справжність, включаючи імена файлів відбитків пальців та кількість збігів для кожного файлу.

3.2.5 Проведення тестів Impostor

Цей сегмент коду (Рисунок 3.5) забезпечує проведення тесту Impostor, який оцінює здатність алгоритму правильно відхиляти відбитки пальців різних осіб. Кожен відбиток пальця порівнюється з відбитками всіх інших осіб. Результати тесту включають значення схожості для кожного порівняння, кількість успішних збігів для кожного відбитка та виводяться в консоль для подальшого аналізу.

```
// 2. Impostor Test
for (int pi = 101; pi <= 110; pi++) {
    for (int pj = 1; pj <= 8; pj++) {
        dictionary.put(pi + "_" + pj, 0);
        probe = new FingerprintTemplate(new FingerprintImage(Files.readAllBytes(Paths.get(
                first: FilePath + pi + "_" + pj + ".tif"))));

        for (int ci = 101; ci <= 110; ci++) {
            if (ci == pi) {
                continue;
            }
            for (int cj = 1; cj <= 8; cj++) {
                candidate = new FingerprintTemplate(new FingerprintImage(Files.readAllBytes(Paths.get(
                        first: FilePath + ci + "_" + cj + ".tif"))));
                matcher = new FingerprintMatcher(probe);
                similarity = matcher.match(candidate);
                if (similarity != 0) {
                    metricsImpostor.add(similarity);
                }
                matches = similarity >= threshold;
                System.out.println("Matching " + pi + "_" + pj + ".tif with " + ci + "_" + cj + ".tif");
                if (matches) {
                    System.out.println(similarity + "\nResult : Fingerprints matched!\n");
                    dictionary.put(pi + "_" + pj, dictionary.get(pi + "_" + pj) + 1);
                } else {
                    System.out.println(similarity + "\nResult : No match!\n");
                }
            }
        }
    }
}
// Print Impostor Test Results
System.out.println("\nImpostor test with threshold " + threshold + ":");
TreeMap<String, Integer> sortedDictionaryImpostor = new TreeMap<>(dictionary);
System.out.println("[FP name]\t[Matches]");
System.out.println("---------------------");
for (Map.Entry<String, Integer> entry : sortedDictionaryImpostor.entrySet()) {
    System.out.println(" " + entry.getKey() + "\t\t\t" + entry.getValue());
    System.out.println("---------------------");
}
```

Рисунок 3.5 – Проведення тестів Impostor

1) Початок тесту Impostor:

- Цикли for: Два вкладених цикли for ітеруються через діапазони значень pi від 101 до 110 та pj від 1 до 8. Ці значення представляють номери файлів відбитків пальців.
- dictionary.put(pi + "_" + pj, 0): Ініціалізуємо словник dictionary ключем, що складається з номерів pi та pj, з початковим значенням 0. Цей словник буде використовуватись для підрахунку кількості збігів для кожного файлу відбитка пальця.
- probe: Створюємо шаблон відбитка пальця probe з файлу pi + "_" + pj +".tif".

2) Внутрішній цикл для порівняння зразків різних осіб:

- for(int ci = 101; ci <= 110; ci++): Ітеруємо по всім можливим індексам осіб, щоб забезпечити порівняння між різними особами.
- if (ci == pi): Пропускаємо порівняння, якщо індекси осіб збігаються (уникнення порівняння зразків однієї і тієї ж особи).
- candidate: Створюємо шаблон кандидатського відбитка пальця candidate з файлу ci + "_" + cj +".tif".
- matcher = new FingerprintMatcher(probe): Створюємо новий об'єкт FingerprintMatcher, ініціалізований зразком probe.
- similarity = matcher.match(candidate): Виконуємо порівняння зразка probe з кандидатом candidate і отримуємо значення схожості.
- if (similarity != 0): Додаємо значення схожості до списку metricsImpostor, якщо воно не дорівнює нулю.
- matches = similarity >= threshold: Визначаємо, чи є схожість між зразками достатньою для визнання їх відповідними (порівняння з порогом).
- System.out.println(...): Виводимо результати порівняння.
- Оновлення словника: Якщо зразки відповідають (matches == true), оновлюємо значення у словнику для відповідного ключа, збільшуючи його на 1.

3) Вивід результатів тесту Impostor:

- TreeMap<String, Integer> sortedDictionaryImpostor = new TreeMap<>(dictionary): Створюємо об'єкт TreeMap для сортування словника за ключами.
- Вивід результатів: Виводимо на екран результати тесту на імітацію, включаючи імена файлів відбитків пальців та кількість збігів для кожного файлу.

3.2.6 Вивід отриманих метрик

Цей сегмент коду (Рисунок 3.1) відповідає за друк метрик схожості для справжніх та імітаційних відбитків пальців. Шляхом знаходження мінімального та максимального значень, він надає загальний огляд діапазону цих метрик. Після цього він виводить кожне значення метрик з точністю до п'ятиох знаків після коми та об'єднує мінімальне та максимальне значення для кожного типу відбитка. Це дозволяє аналізувати та порівнювати розподіл схожості між справжніми та імітаційними відбитками.

```
// Print Genuine Metrics
double minMetricsGenuine = Collections.min(metricsGenuine);
double maxMetricsGenuine = Collections.max(metricsGenuine);
System.out.println("\nGenuine:");
for (double i : metricsGenuine) {
    System.out.printf("%.5f ", i);
}
System.out.println("\nmin:" + minMetricsGenuine);
System.out.println("max:" + maxMetricsGenuine);

// Print Impostor Metrics
double minMetricsImpostor = Collections.min(metricsImpostor);
double maxMetricsImpostor = Collections.max(metricsImpostor);
System.out.println("\nImpostor:");
for (double i : metricsImpostor) {
    System.out.printf("%.5f ", i);
}
System.out.println("\nmin:" + minMetricsImpostor);
System.out.println("max:" + maxMetricsImpostor);
```

Рисунок 3.6 – Вивід отриманих метрик

1) Визначення мінімальних та максимальних значень метрик:

- Для обох тестів (Genuine та Impostor) використовуються методи Collections.min() та Collections.max() для знаходження найменшого та найбільшого значень метрик відповідно.

2) Виведення метрик для тестів Genuine та Impostor:

- За допомогою циклу for виводяться значення метрик для кожного випадку тесту.
- Для кожного значення виводиться з точністю до 5 знаків після коми за допомогою System.out.printf("%.5f ", i);.

3) Виведення мінімальних та максимальних значень:

- Для обох тестів виводяться мінімальні та максимальні значення метрик.
- Ці значення виводяться наступно: "min:" + minMetricsGenuine та "max:" + maxMetricsGenuine.

3.2.7 Експорт отриманих даних у MS Excel

Останній сегмент коду (Рисунок 3.7) виконує функцію експорту результатів аналізу в Excel-файл. Це дозволяє зберігати результати для подальшого аналізу, звітності та документування. Excel-файл містить:

- Значення схожості для справжніх відбитків у стовпці A.
- Мінімальне та максимальне значення для справжніх відбитків у комірках J1 та J2.
- Значення схожості для імітаційних відбитків у стовпці F.
- Мінімальне та максимальне значення для імітаційних відбитків у комірках J4 та J5.

Завдяки цьому підходу, результати тестування легко інтерпретувати та використовувати для подальших досліджень або оптимізації алгоритмів.

```
        // Export Metrics to Excel
        Workbook ExcelWorkbook = new Workbook();
        Worksheet ExcelWorksheet = ExcelWorkbook.getWorksheets().get(0);
        Cells WorksheetCells = ExcelWorksheet.getCells();

        int ExcelIndex = 1;
        for (double i : metricsGenuine) {
            WorksheetCells.get("A" + ExcelIndex).putValue(i);
            ExcelIndex++;
        }
        WorksheetCells.get("J1").putValue( stringValue: "GMin: " + minMetricsGenuine);
        WorksheetCells.get("J2").putValue( stringValue: "GMax: " + maxMetricsGenuine);

        ExcelIndex = 1;
        for (double i : metricsImpostor) {
            WorksheetCells.get("F" + ExcelIndex).putValue(i);
            ExcelIndex++;
        }
        WorksheetCells.get("J4").putValue( stringValue: "IMin: " + minMetricsImpostor);
        WorksheetCells.get("J5").putValue( stringValue: "IMax: " + maxMetricsImpostor);

        try {
            ExcelWorkbook.save( fileName: "C:\\Users\\int9p\\IdeaProjects\\" +
                    "Fingerprint\\resources\\output\\FP_stat.xlsx");
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }
}
```

Рисунок 3.7 – Експорт отриманих даних у MS Excel

1) Ініціалізація робочої книги Excel:

- Створення об'єкта Workbook: Ініціалізується нова робоча книга Excel.
- Отримання першого робочого листа: Отримується перший лист у робочій книзі.
- Отримання клітинок листа: Отримується об'єкт Cells, що представляє колекцію клітинок на робочому листі.

2) Запис метрик Genuine у файл Excel:

- Ініціалізація індексу: ExcelIndex встановлюється на 1, що означає перший рядок у стовпці A.

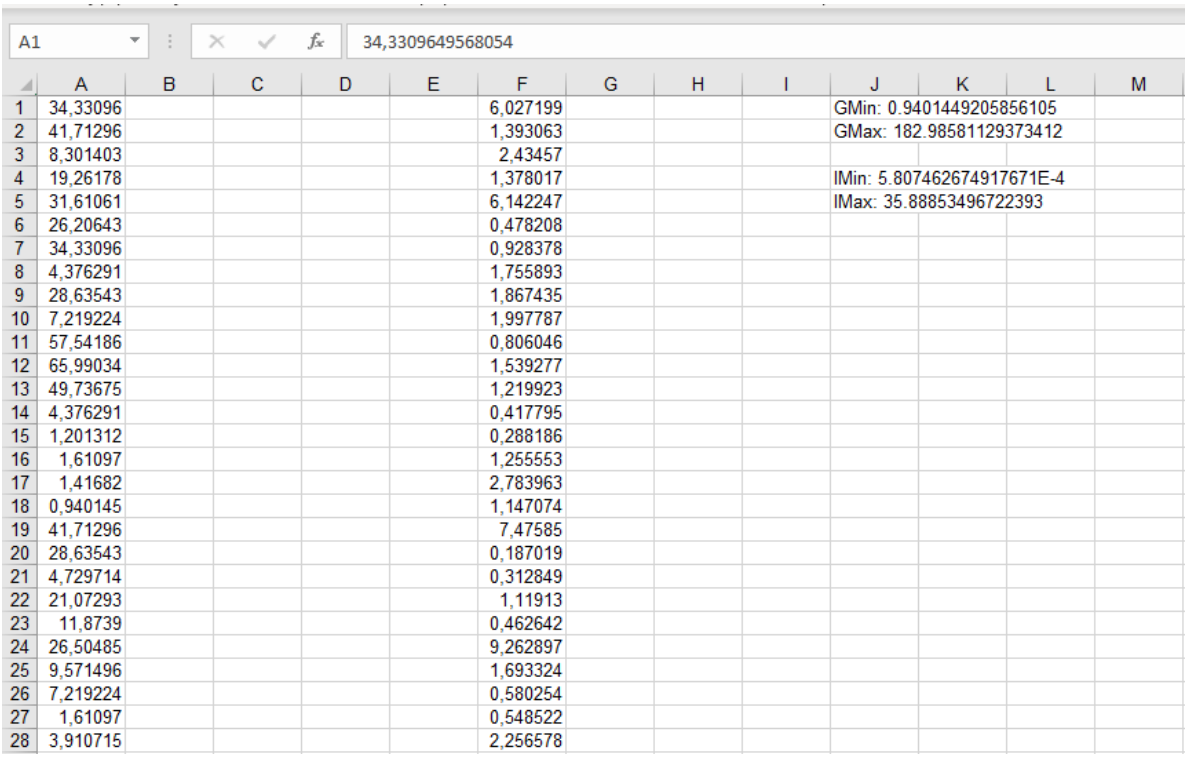
- Запис метрик Genuine: Цикл for проходить по всіх значеннях у списку metricsGenuine і записує кожне значення в послідовні клітинки стовпця A.
- Запис мінімального та максимального значень: Мінімальні та максимальні значення метрик Genuine записуються у клітинки J1 та J2 відповідно.

3) Запис метрик Impostor у файл Excel:

- Ініціалізація індексу: ExcelIndex знову встановлюється на 1 для запису в стовпець F.
- Запис метрик Impostor: Цикл for проходить по всіх значеннях у списку metricsImpostor і записує кожне значення в послідовні клітинки стовпця F.
- Запис мінімального та максимального значень: Мінімальні та максимальні значення метрик Impostor записуються у клітинки J4 та J5 відповідно. (Рисунок 3.8)

4) Збереження файлу Excel:

- Спроба збереження файлу: Робоча книга зберігається за вказаним шляхом.
- Обробка винятків: Якщо виникає помилка при збереженні файлу, вона обробляється за допомогою блоку catch, який генерує RuntimeException з відповідним повідомленням про помилку.



A1 | 34,3309649568054

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	34,33096					6,027199				GMin: 0.9401449205856105			
2	41,71296					1,393063				GMax: 182.98581129373412			
3	8,301403					2,43457							
4	19,26178					1,378017				IMin: 5.807462674917671E-4			
5	31,61061					6,142247				IMax: 35.88853496722393			
6	26,20643					0,478208							
7	34,33096					0,928378							
8	4,376291					1,755893							
9	28,63543					1,867435							
10	7,219224					1,997787							
11	57,54186					0,806046							
12	65,99034					1,539277							
13	49,73675					1,219923							
14	4,376291					0,417795							
15	1,201312					0,288186							
16	1,61097					1,255553							
17	1,41682					2,783963							
18	0,940145					1,147074							
19	41,71296					7,47585							
20	28,63543					0,187019							
21	4,729714					0,312849							
22	21,07293					1,11913							
23	11,8739					0,462642							
24	26,50485					9,262897							
25	9,571496					1,693324							
26	7,219224					0,580254							
27	1,61097					0,548522							
28	3,910715					2,256578							

Рисунок 3.8 – Експортовані дані у MS Excel

3.3 Аналіз отриманих результатів

Після отримання метрик Genuine та Impostor (загалом 555 і 5287 значень відповідно, виключаючи повні нулі), наступним важливим кроком став їх детальний аналіз у програмі Excel. Цей етап включав кілька ключових підходів для визначення розподілу значень метрик і їх подальшої інтерпретації.

Спочатку були визначені проміжки (інтервали) для розподілу значень метрик Genuine та Impostor. Це дозволяє краще зрозуміти, як часто певні значення зустрічаються в отриманих даних. Для розрахунку розподілу значень метрик за визначеними проміжками використовувалася вбудована функція Excel «=FREQUENCY()». Ця функція дозволяє автоматично підрахувати кількість значень, які потрапляють у кожен із зазначених інтервалів. Використання цієї функції включало наступні кроки:

- Визначення діапазону даних: Спочатку були виділені стовпці з метриками Genuine та Impostor.
- Задання проміжків: Далі були створені відповідні проміжки, за якими проводився підрахунок частот.
- Застосування функції «=FREQUENCY»: Ця функція була застосована до діапазонів даних та проміжків, що дозволило автоматично отримати частотний розподіл для кожного інтервалу.

На рисунку 3.9 можна побачити зображення цих розподілів, що дає можливість наочно побачити різницю між розподілами метрик для Genuine та Impostor.

					FAR
ranges	im	g	FRR	0	5287
0,2	153	0	0	153	5134
0,4	198	0	0	351	4936
0,6	228	0	0	579	4708
0,8	212	0	0	791	4496
1	215	2	2	1006	4281
1,2	195	0	2	1201	4086
1,4	240	1	3	1441	3846
1,6	239	2	5	1680	3607
1,8	257	2	7	1937	3350
2	246	2	9	2183	3104
2,2	243	0	9	2426	2861
2,4	223	0	9	2649	2638
2,6	198	2	11	2847	2440
2,8	188	0	11	3035	2252
3	171	0	11	3206	2081
3,2	70	0	11	3276	2011
3,4	70	0	11	3346	1941
3,6	113	0	11	3459	1828
3,8	73	0	11	3532	1755
4	74	1	12	3606	1681
4,2	65	2	14	3671	1616
4,4	49	2	16	3720	1567
4,6	53	0	16	3773	1514
4,8	46	1	17	3819	1468
5	53	0	17	3872	1415
5,2	57	0	17	3929	1358
5,4	47	0	17	3976	1311
5,6	64	0	17	4040	1247
5,8	50	0	17	4090	1197

Рисунок 3.9 – Експортовані дані у MS Excel

Отримані результати були використані для подальшого аналізу розподілу значень метрик Genuine та Impostor. Це дало змогу:

Ідентифікувати аномалії: Аналіз розподілу дозволяє виявити можливі аномалії або незвичні патерни в даних.

Побудувати графіки: На основі отриманих даних були побудовані графіки розподілів для подальшої візуалізації та інтерпретації. (Рисунки 3.10, 3.11 та 3.12)

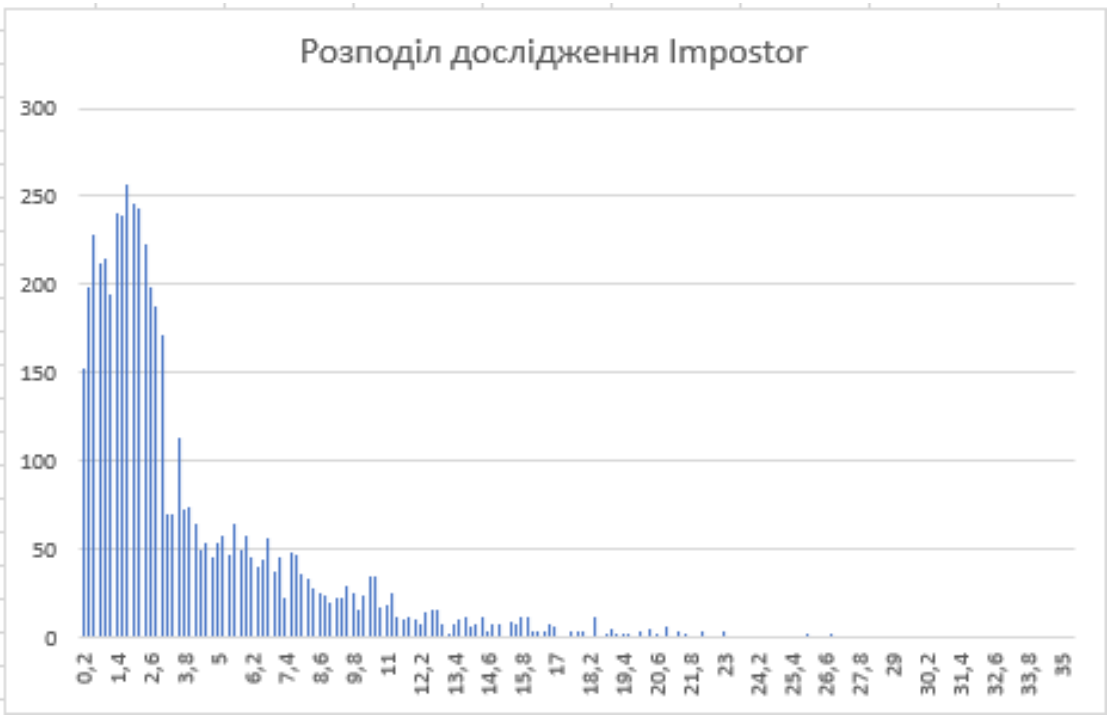


Рисунок 3.10 – Розподіл значень Impostor

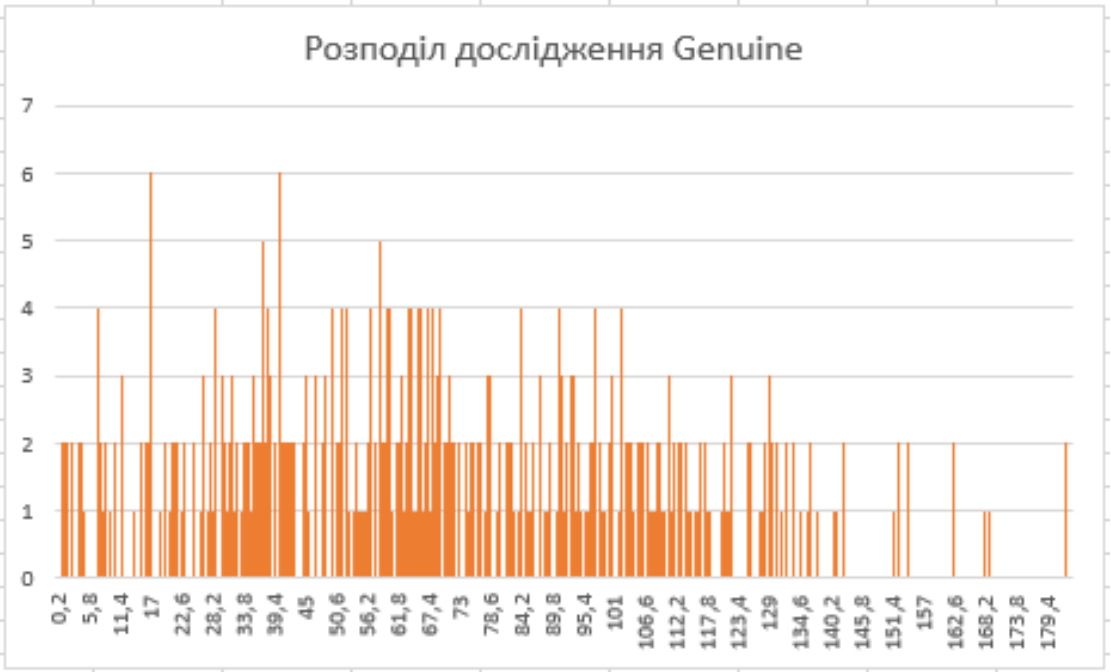


Рисунок 3.11 – Розподіл значень Genuine

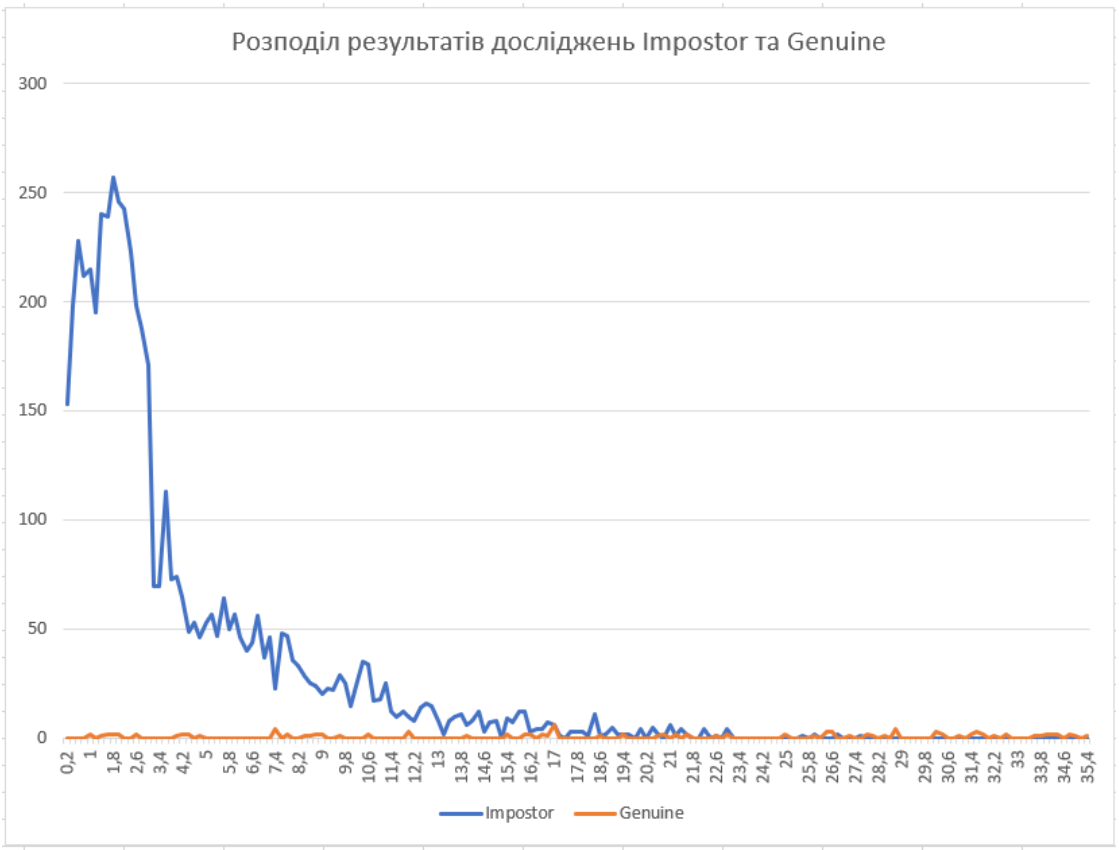


Рисунок 3.12 – Розподіли значень Impostor та Genuine на одному графіку

Після створення графіків розподілу метрик Genuine та Impostor був побудований стовпець даних FRR (False Rejection Rate). Це було зроблено шляхом сумування результатів розподілу Genuine на кожному з представлених інтервалів. Далі був створений стовпець FAR (False Acceptance Rate), який обчислювався відніманням чисел зі стовпця з результатами розподілу Impostor від загальної кількості випробувань Impostor.

Отримані дані були відображені на графіку, який можна побачити на рисунку 3.13. На цьому графіку видно два основні криві - FRR та FAR. Проаналізувавши точку перетину цих кривих на графіку зображеному на рисунку 3.14, вдалося визначити значення EER (Equal Error Rate). Ця точка перетину показує, де частота хибних відмов дорівнює частоті хибних приймань. В результаті аналізу було встановлено, що значення EER становить 1.04% (55/5287).

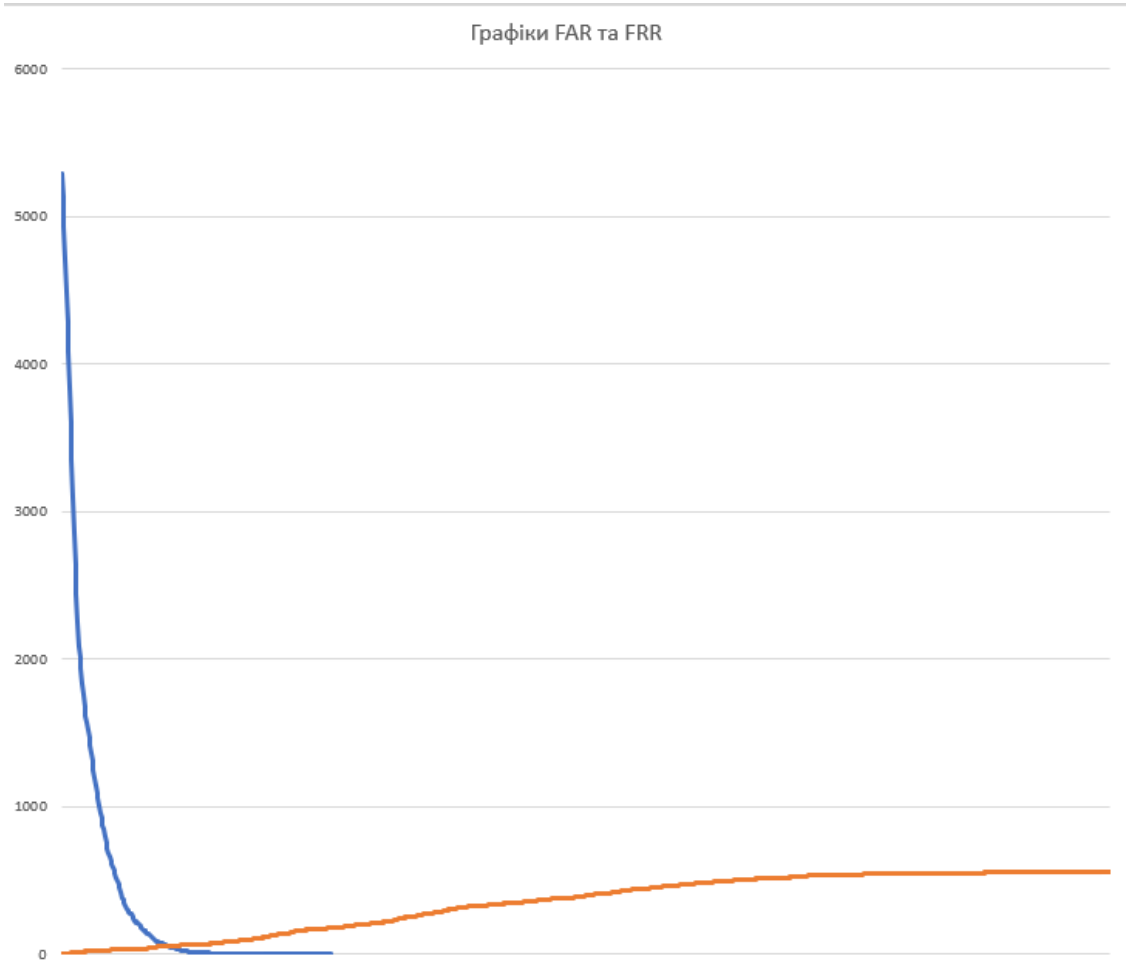


Рисунок 3.13 – Графіки FAR та FRR

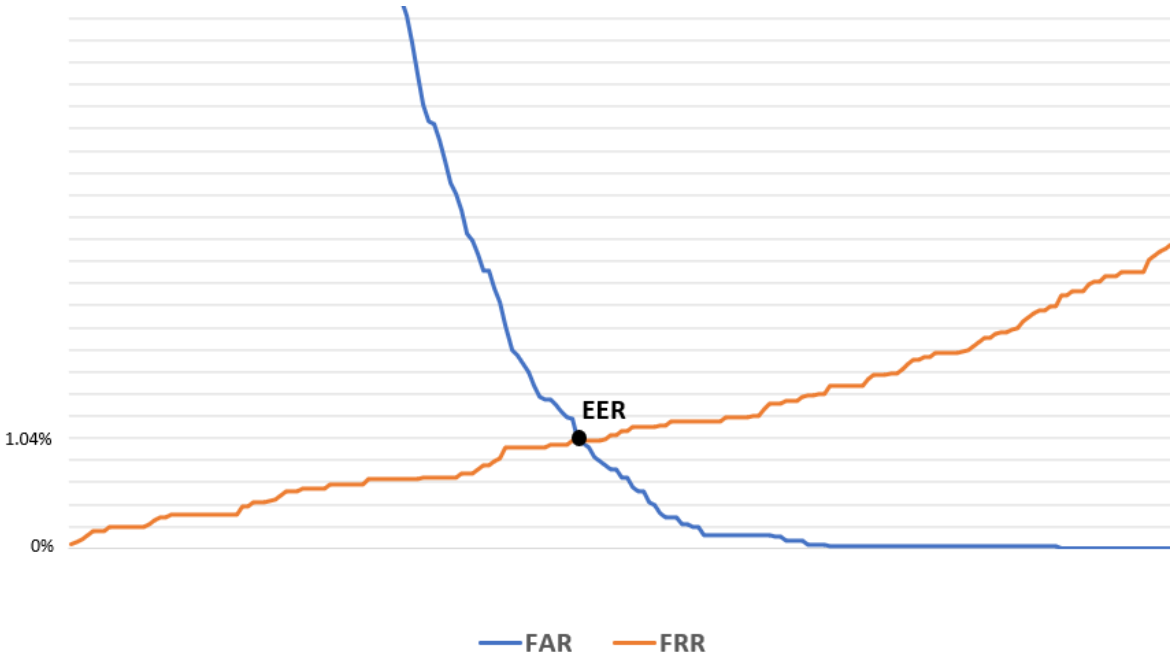


Рисунок 3.14 – FAR, FRR та EER

Цей аналіз є важливим, оскільки EER є одним із ключових показників ефективності біометричної системи. Низьке значення EER свідчить про високу точність системи і її здатність мінімізувати помилки як хибних відмов, так і хибних приймань.

Однак, попри це, технологія розпізнавання відбитків пальців не є ідеальною. Система все ще може робити помилки, і існують обставини, за яких її продуктивність може бути піддана впливу. Наприклад, пошкодження або забруднення пальців, а також певні фізіологічні зміни можуть впливати на точність розпізнавання. Крім того, безпека біометричних даних залишається важливим питанням, оскільки злам або компрометація відбитків пальців може мати серйозні наслідки.

Отже, хоча значення EER вказує на високу точність, необхідно враховувати, що система розпізнавання відбитків пальців має свої обмеження і ризики, які потребують подальшого вдосконалення та врахування у контексті її використання.

ВИСНОВКИ

У цій дипломній роботі було проведено огляд принципів роботи технології розпізнавання відбитків пальців та розроблено програмне рішення на мові програмування Java для автоматизованого отримання метрик аналізу використовуваного алгоритму. Зокрема, була оцінена ефективність відкритої біометричної бібліотеки SourceAFIS шляхом розрахунку основних показників точності, таких як FAR (False Acceptance Rate), FRR (False Rejection Rate) та EER (Equal Error Rate).

У результаті практичного дослідження було виявлено, що значення EER для SourceAFIS становить 1.04%. Це вказує на досить високу точність та ефективність даної системи розпізнавання відбитків пальців. Водночас, аналіз продемонстрував, що технологія все ще має певні обмеження та не є ідеальною, оскільки може робити помилки за певних обставин.

Отримані результати можуть бути використані для подальшого вдосконалення рівня безпеки додатків, які використовують алгоритми розпізнавання відбитків пальців, а також для покращення безпеки та ефективності біометричних систем загалом. Розроблене програмне рішення забезпечує автоматизований підхід до аналізу алгоритмів розпізнавання відбитків пальців, що полегшує процес оцінки та порівняння різних методів.

Незважаючи на високу точність, важливо враховувати, що системи розпізнавання відбитків пальців не є абсолютно досконалими та мають певні ризики та обмеження, пов'язані з забезпеченням безпеки біометричних даних, впливом фізіологічних факторів та можливими пошкодженнями відбитків. Тому для підвищення надійності та ефективності таких систем необхідно продовжувати дослідження та вдосконалення алгоритмів, а також ретельно розглядати потенційні вразливості та ризики при їх застосуванні.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Types of Current Biometric Identification Technologies and Biometric Authentication Technologies [Електронний ресурс]. – Режим доступу: https://www.linkedin.com/pulse/types-current-biometric-identification-technologies/

2. Application Security: The Complete Guide [Електронний ресурс]. – Режим доступу: https://www.imperva.com/learn/application-security/application-security/

3. Types of Biometrics Authentication [Електронний ресурс]. – Режим доступу: https://optimalidm.com/resources/blog/types-of-biometrics-sensors/

4. 5 Popular Biometric Authentication Methods [Електронний ресурс]. – Режим доступу: https://www.descope.com/blog/post/biometric-auth-methods

5. MachineZoo SourceAFIS [Електронний ресурс]. – Режим доступу: https://sourceafis.machinezoo.com/

6. Java Excel Spreadsheet Processing API [Електронний ресурс]. – Режим доступу: https://products.aspose.com/cells/java/

7. Awad, Ali Ismail & Hassanien, Aboul Ella. (2014). Impact of Some Biometric Modalities on Forensic Science. Studies in Computational Intelligence. 555. 47-62. 10.1007/978-3-319-05885-6-3. [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/publication/288801929_Impact_of_Some_Biometric_Modalities_on_Forensic_Science

8. M. M. H. Ali, V. H. Mahale, P. Yannawar and A. T. Gaikwad, "Fingerprint Recognition for Person Identification and Verification Based on Minutiae Matching," 2016 IEEE 6th International Conference on Advanced Computing (IACC), Bhimavaram, India, 2016, pp. 332-339, doi: 10.1109/IACC.2016.69. [Електронний ресурс]. – Режим доступу: https://ieeexplore.ieee.org/abstract/document/7544858

9. Anzar, S M & P S, Sathidevi. (2014). On combining multi-normalization and ancillary measures for the optimal score level fusion of fingerprint and voice biometrics. EURASIP Journal on Advances in Signal Processing. 2014. 10. 10.1186/1687-6180-2014-10. [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/publication/269588602_On_combining_multi-

normalization_and_ancillary_measures_for_the_optimal_score_level_fusion_of_fingerprint_and_voice_biometrics

10. Jain, A. K., Feng, J., & Nandakumar, K. (2010). Fingerprint Matching. Computer, 43(2), 36–44. doi:10.1109/mc.2010.38 [Електронний ресурс]. – Режим доступу: https://biometrics.cse.msu.edu/Publications/Fingerprint/JainFpMatching_IEEEComp10.pdf

11. Palm-scanning machines installed at 2 Shanghai metro stations. Shine news [Електронний ресурс]. – Режим доступу: https://www.shine.cn/news/metro/2402292596/

12. Raval, Mehul & Joshi, Vaibhav. (2019). Adaptive Threshold for Fingerprint Recognition System Based on Threat Level and System Load. 10.31219/osf.io/9usp4. [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/publication/336102017_Adaptive_Threshold_for_Fingerprint_Recognition_System_Based_on_Threat_Level_and_System_Load

ДОДАТОК А

ЛІСТИНГ РОЗРОБЛЕНОЇ ПРОГРАМИ ДЛЯ АНАЛІЗУ

```
import com.machinezoo.sourceafis.*;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.*;
import com.aspose.cells.*;

public class Main {
    private static final String FilePath = "resources/DB4_B/";

    public static void main(String[] args) throws IOException {
        // Initialization
        Map<String, Integer> dictionary = new HashMap<>();
        List<Double> metricsGenuine = new ArrayList<>();
        List<Double> metricsImpostor = new ArrayList<>();
        var options = new FingerprintImageOptions().dpi(500);
        double threshold = 40;
        boolean matches = false;
        int count = 0;



        // Load probe and candidate fingerprints
        var probe = new FingerprintTemplate(
                new FingerprintImage(Files.readAllBytes(Paths.get(
                        "resources/DB1_B/101_1.tif"))));
        var candidate = new FingerprintTemplate(
                new FingerprintImage(Files.readAllBytes(Paths.get(
                        "resources/DB1_B/101_2.tif"))));

        var matcher = new FingerprintMatcher(probe);
        double similarity = matcher.match(candidate);
```

```
        // 1. Genuine Test
        for (int pi = 101; pi <= 110; pi++) {
            for (int pj = 1; pj <= 8; pj++) {
                dictionary.put(pi + "_" + pj, 0);
                probe = new FingerprintTemplate(new
FingerprintImage(Files.readAllBytes(Paths.get(
                        FilePath + pi + "_" + pj + ".tif"))));

                for (int cj = 1; cj <= 8; cj++) {
                    if (cj == pj) {
                        continue;
                    }
                    candidate = new FingerprintTemplate(new
FingerprintImage(Files.readAllBytes(Paths.get(
                            FilePath + pi + "_" + cj + ".tif"))));
                    matcher = new FingerprintMatcher(probe);
                    similarity = matcher.match(candidate);
                    if (similarity != 0) {
                        metricsGenuine.add(similarity);
                    }
                    matches = similarity >= threshold;
                    System.out.println("Matching " + pi + "_" + pj +
".tif with " + pi + "_" + cj + ".tif");
                    if (matches) {
                        System.out.println(similarity + "\nResult :
Fingerprints matched!\n");
                        dictionary.put(pi + "_" + pj,
dictionary.get(pi + "_" + pj) + 1);
                    } else {
                        System.out.println(similarity + "\nResult :
```

```
No match!\n");
                    }
                }
            }
        }

        // Print Genuine Test Results
        System.out.println("\nGenuine test with threshold " +
threshold + ":");
        TreeMap<String, Integer> sortedDictionaryGenuine = new
TreeMap<>(dictionary);
        System.out.println("[FP name]\t[Matches]");
        System.out.println("---------------------");
        for (Map.Entry<String, Integer> entry :
sortedDictionaryGenuine.entrySet()) {
            System.out.println(" " + entry.getKey() + "\t\t\t" +
entry.getValue());
            System.out.println("---------------------");
        }

        // 2. Impostor Test
        for (int pi = 101; pi <= 110; pi++) {
            for (int pj = 1; pj <= 8; pj++) {
                dictionary.put(pi + "_" + pj, 0);
                probe = new FingerprintTemplate(new
FingerprintImage(Files.readAllBytes(Paths.get(
                        FilePath + pi + "_" + pj + ".tif"))));

                for (int ci = 101; ci <= 110; ci++) {
                    if (ci == pi) {
                        continue;
                    }
                    for (int cj = 1; cj <= 8; cj++) {
                        candidate = new FingerprintTemplate(new
FingerprintImage(Files.readAllBytes(Paths.get(
                                FilePath + ci + "_" + cj +
```

```
".tif"))));
                        matcher = new FingerprintMatcher(probe);
                        similarity = matcher.match(candidate);
                        if (similarity != 0) {
                            metricsImpostor.add(similarity);
                        }
                        matches = similarity >= threshold;
                        System.out.println("Matching " + pi + "_" +
pj + ".tif with " + ci + "_" + cj + ".tif");
                        if (matches) {
                            System.out.println(similarity +
"\nResult : Fingerprints matched!\n");
                            dictionary.put(pi + "_" + pj,
dictionary.get(pi + "_" + pj) + 1);
                        } else {
                            System.out.println(similarity +
"\nResult : No match!\n");
                        }
                    }
                }
            }
        }
        // Print Impostor Test Results
        System.out.println("\nImpostor test with threshold " +
threshold + ":");
        TreeMap<String, Integer> sortedDictionaryImpostor = new
TreeMap<>(dictionary);
        System.out.println("[FP name]\t[Matches]");
        System.out.println("---------------------");
        for (Map.Entry<String, Integer> entry :
sortedDictionaryImpostor.entrySet()) {
            System.out.println(" " + entry.getKey() + "\t\t\t" +
entry.getValue());
            System.out.println("---------------------");
        }
```

```
        // Print Genuine Metrics
        double minMetricsGenuine = Collections.min(metricsGenuine);
        double maxMetricsGenuine = Collections.max(metricsGenuine);
        System.out.println("\nGenuine:");
        for (double i : metricsGenuine) {
            System.out.printf("%.5f ", i);
        }
        System.out.println("\nmin:" + minMetricsGenuine);
        System.out.println("max:" + maxMetricsGenuine);

        // Print Impostor Metrics
        double minMetricsImpostor =
Collections.min(metricsImpostor);
        double maxMetricsImpostor =
Collections.max(metricsImpostor);
        System.out.println("\nImpostor:");
        for (double i : metricsImpostor) {
            System.out.printf("%.5f ", i);
        }
        System.out.println("\nmin:" + minMetricsImpostor);
        System.out.println("max:" + maxMetricsImpostor);

        // Export Metrics to Excel
        Workbook ExcelWorkbook = new Workbook();
        Worksheet ExcelWorksheet =
ExcelWorkbook.getWorksheets().get(0);
        Cells WorksheetCells = ExcelWorksheet.getCells();

        int ExcelIndex = 1;
        for (double i : metricsGenuine) {
            WorksheetCells.get("A" + ExcelIndex).putValue(i);
            ExcelIndex++;
        }
        WorksheetCells.get("J1").putValue("GMin: " +
minMetricsGenuine);
        WorksheetCells.get("J2").putValue("GMax: " +
```

```
maxMetricsGenuine);

        ExcelIndex = 1;
        for (double i : metricsImpostor) {
            WorksheetCells.get("F" + ExcelIndex).putValue(i);
            ExcelIndex++;
        }
        WorksheetCells.get("J4").putValue("IMin: " +
minMetricsImpostor);
        WorksheetCells.get("J5").putValue("IMax: " +
maxMetricsImpostor);

        try {
            ExcelWorkbook.save("C:\\Users\\int9p\\IdeaProjects\\" +
                    "Fingerprint\\resources\\output\\FP_stat.xlsx");
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }
}
```