Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему *Спосіб нейромережевого виявлення фейкових зображень документів для систем ідентифікації особистості*

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

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

Освітня програма *Комп'ютерні науки*
Назва освітньої програми

Виконав:	*студент 4 курсу, група КН-19-1* (Курс, група виконавця)	(Підпис)	*О.В. Жарновський* (Ініціали, прізвище)
Керівник:	*викладач кафедри КН* (Науковий ступінь, посада)	(Підпис)	*О.В. Собко* (Ініціали, прізвище)
Нормоконтроль:	*к.т.н., доцент кафедри КН* (Науковий ступінь, посада)	(Підпис)	*Р.О. Багрій* (Ініціали, прізвище)

До захисту допускаю:
Зав. кафедри КН, д.т.н., професор (Підпис) *О.В. Бармак* (Ініціали, прізвище)

05 06 2023 р.

Хмельницький 2023

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет *інформаційних технологій*
Кафедра *комп'ютерних наук*
Освітній ступінь *бакалавр*
Галузь знань *12 – Інформаційні технології*
Спеціальність *122 – Комп'ютерні науки*
Освітня програма *освітньо-професійна програма підготовки бакалавра*

ЗАТВЕРДЖУЮ
Завідувач кафедри *комп'ютерних наук*

_____
(підпис)
д.т.н., професор О.В. Бармак

«*06*» *03* *2023* року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

1. Тема кваліфікаційної роботи бакалавра: *«Спосіб нейромережевого виявлення фейкових зображень документів для систем ідентифікації особистості»*

2. Завдання видано студенту *Жарновському Олександру Володимировичу*
(прізвище, ім'я, по батькові)

3. Керівник роботи *викладач кафедри КН Собко Олена Віталіївна*
(посада, прізвище, ім'я, по батькові)

4. Затверджено наказом університету від «*01*» *03* *2023* р. № *5*

5. Дата видачі завдання студенту: «*03*» *03* *2023* р.

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

Провести аналіз предметної області, визначити особливості застосування згорткових нейронних мереж для задач класифікації при роботі з зображеннями документів. Виконати аналіз існуючих рішень щодо подібних задач. Розробити спосіб нейромережевого виявлення фейкових зображень документів. Розробити архітектуру нейронної мережі для визначення рівня достовірності зображень документів. Спроєктувати структуру інформаційної системи ідентифікації особистості й структуру відповідної бази даних. Спроєктувати та створити застосунок що використовує розроблений спосіб, виконати його тестування.

7. Календарний план виконання кваліфікаційної роботи бакалавра:

№	Назва етапів (розділів) кваліфікаційної роботи бакалавра	Термін виконання	Примітка
1	*Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи бакалавра з керівником*	*грудень 2022*	виконано
2	*Ознайомлення з предметною областю, формулювання мети та задач дослідження, визначення об'єкта та предмета дослідження*	*січень 2023*	виконано
3	*Робота над розділом 1 – Характеристика предметної області та постановка задачі*	*січень 2023*	виконано
4	*Робота над розділом 2 – Спосіб нейромережевого виявлення фейкових зображень документів для систем ідентифікації особистості*	*березень 2023*	виконано
5	*Робота над розділом 3 – Програмна реалізація інформаційної системи ідентифікації особистості*	*квітень 2023*	виконано
6	*Оформлення пояснювальної записки згідно вимог*	*травень 2023*	виконано
7	*Попередній захист кваліфікаційної роботи бакалавра*	*травень 2023*	виконано
8	*Захист кваліфікаційної роботи бакалавра на засіданні Екзаменаційної комісії*	*червень 2023*	виконано

Виконавець: *студент 4 курсу, група КН-19-1* (Курс, група виконавця) _____ (Підпис) *О.В.Жарновський* (Ініціали, прізвище)

Керівник: *викладач кафедри КН* (Науковий ступінь, посада) _____ (Підпис) *О.В. Собко* (Ініціали, прізвище)

Анотація

Тема кваліфікаційної роботи бакалавра: *Спосіб нейромережевого виявлення фейкових зображень документів для систем ідентифікації особистості*

Виконавець кваліфікаційної роботи бакалавра: *студент групи КН-19-1 Жарновський Олександр Володимирович*

Керівник кваліфікаційної роботи бакалавра: *викладач кафедри КН Собко Олена Віталіївна*

Кваліфікаційна робота бакалавра містить:

Пояснювальна записка				Кількість додатків
Сторінок	Рисунків	Таблиць	Джерел інформації	
66	63	11	25	6

Метою кваліфікаційної роботи бакалавра є розробка та програмна реалізація способу нейромережевого виявлення фейкових зображень документів для систем ідентифікації особистості, яка дозволяє за відсканованим чи сфотографованим зображеннями документу що посвідчує особу автоматизовано визначати рівень його достовірності за допомогою згорткової нейронної мережі.

Результатом виконання кваліфікаційної роботи бакалавра є створений спосіб нейромережевого виявлення фейкових зображень документів та інформаційна система ідентифікації особистості, яка використовує розроблений спосіб й дозволяє за відсканованим чи сфотографованим зображеннями документу що посвідчує особу автоматизовано визначати рівень його достовірності за допомогою згорткової нейронної мережі.

Ключові слова: ідентифікація особистості, фейкові зображення, згорткова нейронна мережа, зображення документу, нейромережа, система ідентифікації особистості.

Виконавець: *студент 4 курсу, група КН-19-1* (Курс, група виконавця) _____ (Підпис) *О.В. Жарновський* (Ініціали, прізвище)

Зміст

Перелік скорочень

Скорочення, термін, позначення	Пояснення
БД	База даних
КРБ	Кваліфікаційна робота бакалавра
КН	Комп’ютерні науки
НМ	Нейронна мережа
AI	Artificial intelligence / Штучний інтелект
ELA	Error level analysis / Аналіз рівня помилок
CNN	Convolutional neural network / Згорткова нейронна мережа
UI	User Interface / Користувацький інтерфейс

Вступ

Кваліфікаційна робота бакалавра присвячена розробці та програмній реалізації способу нейромережевого виявлення фейкових зображень документів для систем ідентифікації особистості, яка дозволяє за відсканованим чи сфотографованим зображеннями документу що посвідчує особу автоматизовано визначати рівень його достовірності за допомогою згорткової нейронної мережі.

Актуальність – Одним із багатьох способів ідентифікації особистості є офіційний документ, що містять набір ключових параметрів, за допомогою яких можливо встановити особу, такі як фотокартка, прізвище/ім’я/по-батькові. Способи створення фейкових зображень стали доволі розповсюдженими та загальнодоступними – різноманітні редактори, нейромережі, тощо. Це створює труднощі для процесу перевірки автентичності та потребує покращення. Крім того, розробка програмного забезпечення з використанням нейронних мереж є актуальним методом для часткової автоматизації процесу перевірки зображень документів.

Об’єкт дослідження – процес виявлення фейкових зображень документів для систем ідентифікації особистості.

Предмет дослідження – моделі, методи, алгоритми та засоби для визначення рівня достовірності відсканованих та сфотографованих зображень документів що посвідчують особу.

Мета кваліфікаційної роботи бакалавра – розробка та програмна реалізація способу нейромережевого виявлення фейкових зображень документів для систем ідентифікації особистості.

Завдання кваліфікаційної роботи бакалавра – Провести аналіз предметної області, визначити особливості застосування згорткових нейронних мереж для задач класифікації при роботі з зображеннями документів. Виконати аналіз існуючих рішень щодо подібних задач. Розробити спосіб

нейромережевого виявлення фейкових зображень документів. Розробити архітектуру нейронної мережі для визначення рівня достовірності зображень документів. Спроєктувати структуру інформаційної системи ідентифікації особистості й структуру відповідної бази даних. Спроєктувати та створити застосунок що використовує розроблений спосіб, виконати його тестування.

Розділ 1 Характеристика предметної області та постановка задачі

1.1 Аналіз предметної області

Одним із багатьох способів ідентифікації особистості є офіційний документ, що містять набір ключових параметрів, за допомогою яких можливо встановити особу, такі як фотокартка, прізвище/ім’я/по-батькові. Такі посвідчення поділяються на дві основних категорії [1]:

– посвідчують особу та підтверджують громадянство;

– посвідчують особу та підтверджують її спеціальний статус.

До них належать, але не обмежені ними:

– паспорт громадянина України;

– паспорт громадянина України для виїзду закордон;

– дипломатичний паспорт України;

– службовий паспорт України;

– посвідчення водія;

– посвідчення біженця.

Обіг документів в будь-якому форматі створює потенційні загрози. Такі загрози різняться від володінням різних копій одного документа багатьма офіційними установами до повноцінної фальсифікації.

Самими простими прикладами підробки є підчистка (видалення частин документу механічно), додрукування (нанесення нових слів або друк поверх старих) заміна аркушів (в випадку, коли документ містить декілька аркушів). Якщо сліди від тертя чи додрукування на місцях з реквізитами можливо визначити і без «експертизи», то заміна чи підробка документа в цілому потребує більш детальної уваги. Для запобігання фальсифікації використовують декілька основних типів захисту [2].

Елементи, що можливо розпізнати візуально. Різноманітні печатки, штампи, захисні сітки, стрічки чи водяні знаки (рисунок 1.1). Сюди також можливо віднести різноманітні елементи ідентифікації, такі як серійні номери.

Рисунок 1.1 – Стоковий приклад печаті [24]

Захист по видам чи способам друку [3]. Спеціальні шрифти, матеріал, нестандартна фарба, елементи що можливо відчути на дотик, штрих коди чи лазерне гравіювання (рисунок 1.2) – всі ці елементи неможливо підробити без спеціального обладнання, а невміла підробка буде очевидна.

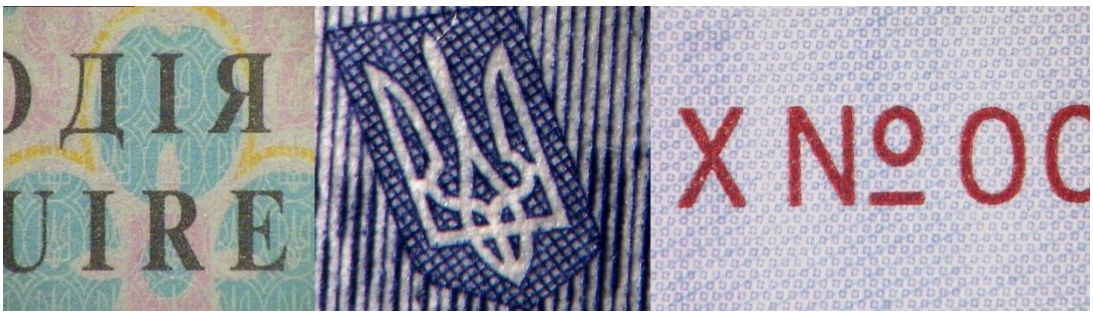


Рисунок 1.2 – Різноманітні методи друку [3]

Елементи, що можливо розпізнати приладами. Це різноманітні фрагменти, що людське око не може розпізнати саме по собі – в декілька разів зменшений текст, елементи які невидимі без дії інфрачервоного чи ультрафіолетового опромінювання, зони машинного зчитування (рисунок 1.3).

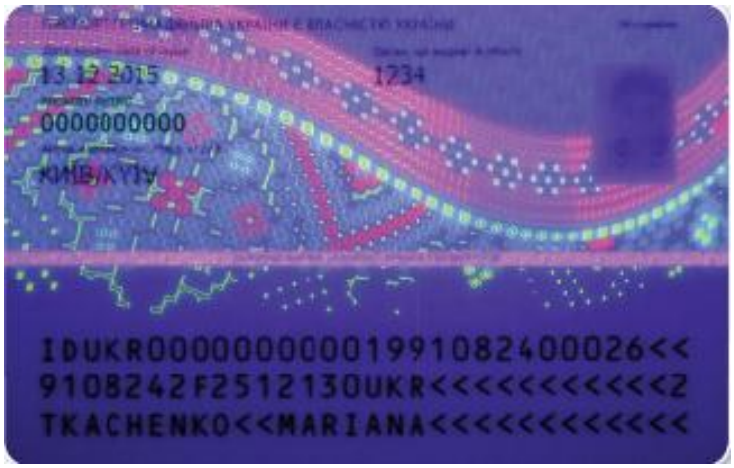


Рисунок 1.3 – Зворотна сторона паспорту під дією ультрафіолету [25]

Будь-який документ, що має хоча б якусь бюрократичну цінність буде використовувати комбінацію з цих методів, документи ідентифікації особистості не виключення. Обов’язковий документ кожного громадянина України – паспорт це один із найкращих прикладів розповсюдженого документу з величезною кількістю захисних мір (рисунок 1.4).

Рисунок 1.4 – Захисні міри на паспорті громадянина України у формі картки [25]

Способів захисту документів є величезна кількість, але і стільки ж способів їх підробити. Головний закон ринку – де попит там і пропозиція. Завжди будуть люди, що хочуть сфальсифікувати документи. При створенні нових методів захисту будуть виникати нові методи їх підробки.

Частину таких підробок знайдуть ще на етапі подання заявки з використанням фейку, але частина все ж таки потрапить в реєстр. Це може бути як і через неякісний процес валідації, так і просто через величезну кількість документів, що і з доданням людського фактору створює певний відсоток фальсифікату що пройшов перевірку.

Через ті ж самі причини зробити повторну перевірку не завжди ефективно або можливо. Використання технологій автоматизації за допомогою нейронних мереж дозволить значно зменшити відсоток пропущених підробок та дозволить протидіяти аналогічним методам зі сторони злочинців.

Отже, в результаті аналізу, для задачі побудови способу нейромережевого виявлення фейкових зображень документів для систем ідентифікації особистості, існує велика кількість параметрів для зображення сторінок ідентифікаційних документів.

1.2 Особливості застосування згорткових нейромереж для задач класифікації

Повсякденне використання нейронних мереж набуває все більшої популярності в наш час.

Нейронні мережі (neural network) – комплексні алгоритми машинного навчання подібні до біологічної структури мереж що складають мозок тварин. Такий алгоритм, складається з багатьох шарів – вхідний/вихідний та один чи більше шарів між ними. (рисунок 1.5) [4].

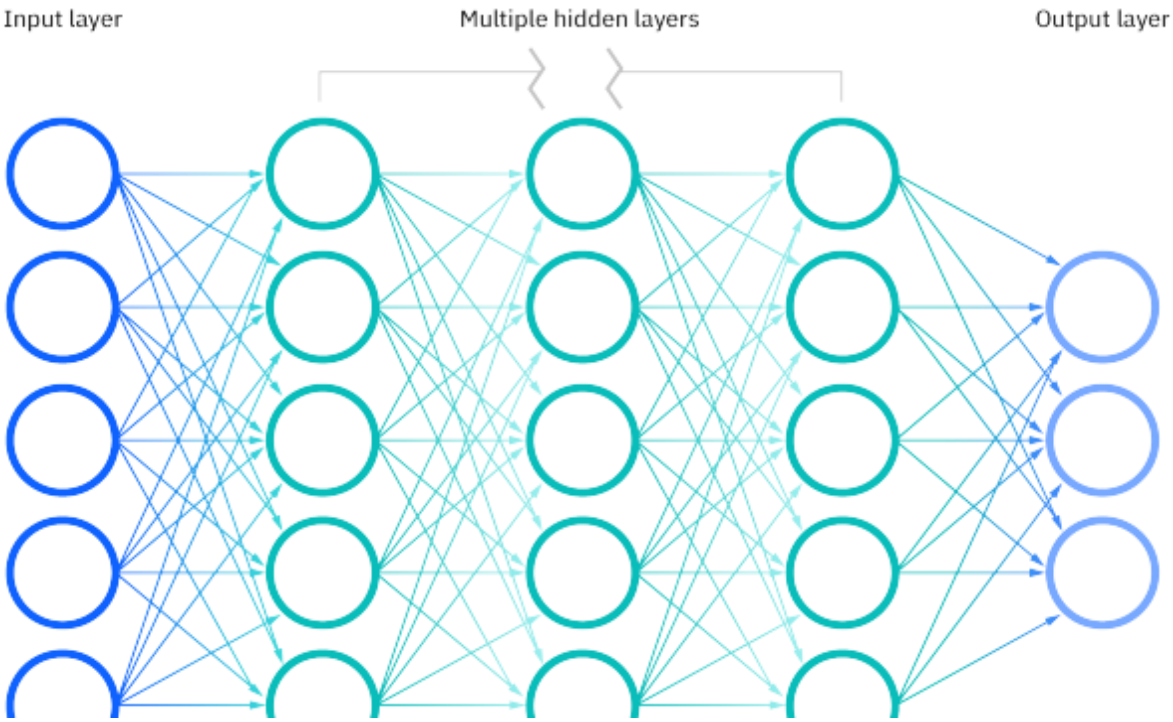


Рисунок 1.5 – Схематичне зображення нейронної мережі [4]

В свою чергу шари поділені на вузли – штучні нейрони, що з’єднані між собою складною сіткою ваг та лімітів. При надходженні інформації на кожен окремий нейрон, вона обчислюється та результат цієї операції передається на наступний шар для подальшого обчислення.

Ліміти та ваги це певні числові коефіцієнти, що були закладені в алгоритм з самого початку та змінюються з часом при навчанні мережі. Для початку, мережу «тренують» тестовим «дата сетом», де чітко розставлені «так» чи «не так». Якщо результат обчислення вірний – добре, якщо ні – алгоритм змінює ваги на нейронах для отримання правильного результату. Коректність складеного навчального сету даних має дуже важливу роль в програмуванні нейронної мережі – завжди існує можливість як і не достатньо навчити, так і перенавчити, що в зменшить точність мережі в цілому

Спеціалізованих типів нейронних мереж є велика кількість, одним із них є згорткові нейронні мережі (convolutional neural networks) – мережі створені для завдань по розпізнаванню залежностей, аудіо чи відеосигналу [5]. В випадку з зображеннями, шари спочатку можуть розділяти окремо взяте зображення на

ребра, грані чи контури, виділяючи ключові ознаки. З кожним новим шаром підвищується алгоритмічна складність і комплексність взятих фрагментів зображення, чи навіть повноцінних об’єктів (рисунок 1.6).

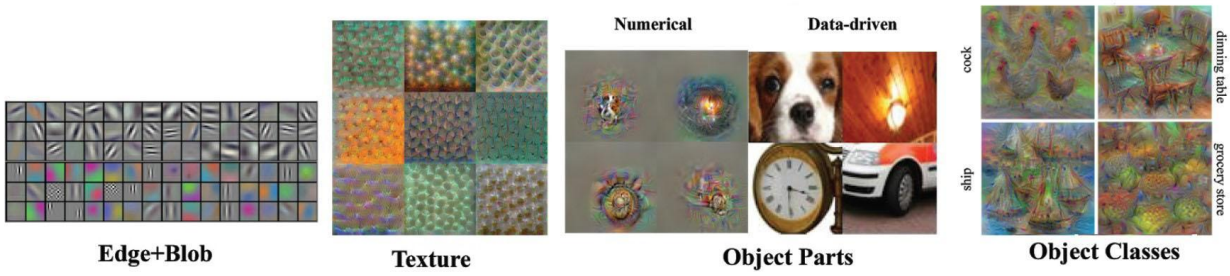


Рисунок 1.6 – Приклад розбиття зображення [5]

Feed Forward Neural Network (нейронна мережа прямого поширення). Складається з нейронів, що передають дані тільки в одному напрямку (напрямок вихідного шару) застосовується для розпізнавання обличчя чи мови, може протидіяти «шумам» та легка в підтримці через свою доволі просту структуру.

Recurrent Neural Network (рекурентна нейронна мережа) – на відміну від інших, ця мережа запам’ятовує частину інформації після передачі її в наступні шари, що дозволяє їй зробити кращий висновок при допущенні помилок в наступному шарі. Через свою структуру вона краще всього підходить для використання в автоматизованих системах перевірки тексту (наприклад граматика чи пошукові системи), де інформація розділена на чіткі сегменти – слова.

Як і самих мов програмування є велика кількість, так і для кожної окремої існує безліч бібліотек різного спрямування, включно і для розробки нейронних мереж. Таке різноманіття добре тим, що завжди є вибір, але воно настільки широке, що може збити с пантелику питанням «з чого почати?».

Мова програмування Python виділяється своєю кількістю бібліотек. Простота Python дозволяє розробнику сфокусуватися на написанні самого

алгоритму, з використанням цих бібліотек, а не зав’язнути в технічній частині мови програмування [6].

Розглянемо декілька бібліотек:

– NeuroLab – комплексна бібліотека, використовується для створення та тренування різних алгоритмів нейронних мереж, написана на чистому Python [7];

– TensorFlow – open-source бібліотека з використанням графів для числових обчислень. Крім Python підтримує і C++, Java/JS [8];

– FFnet (feed forward neural network) – бібліотека спрямована на тренування і тестування нейронних мереж прямого типу [9].

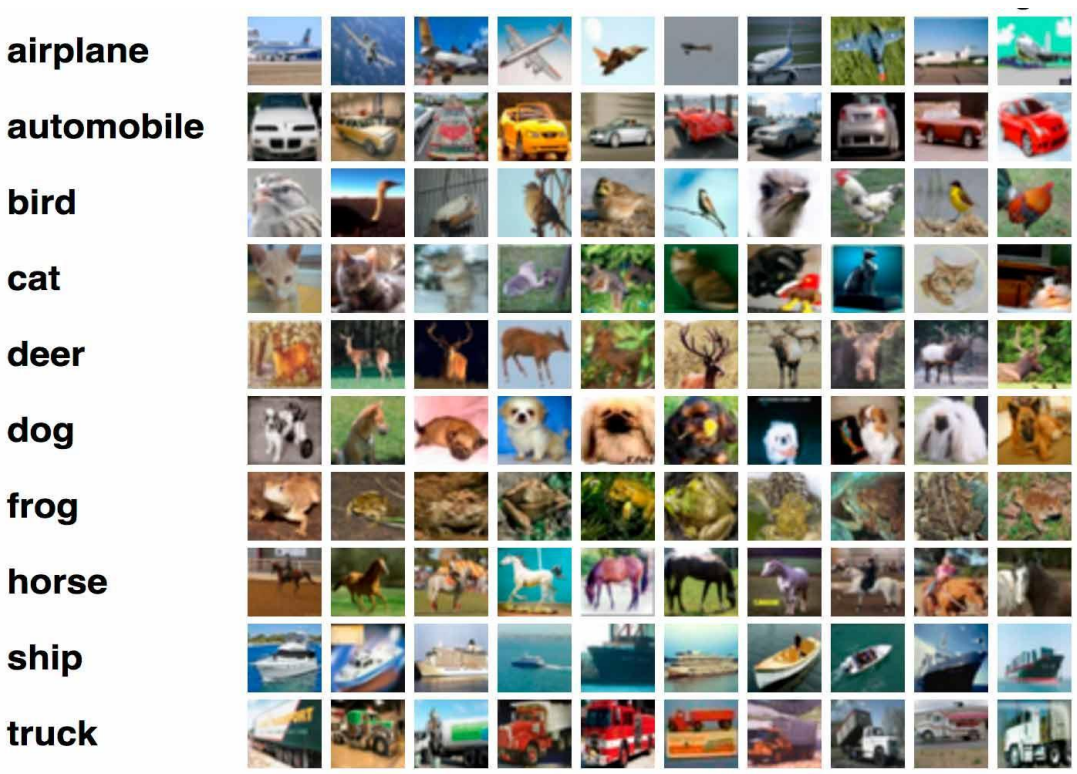


Рисунок 1.7 – Приклад завдання для класифікації [5]

Головними перевагами таких алгоритмів є автоматизація та швидкість обчислення. База даних під класифікацію (рисунок 1.7), може складається з десятків чи сотень тисяч зображень, якщо не більше. Для людини виконання цієї роботи буде коштувати набагато більше ресурсів, чим правильно налаштований алгоритм. Далеко не кожен працівник зможе виконувати монотонну роботу на

великих проміжках часу, а алгоритм обмежений тільки обладнанням. Крім того, згорткові нейронні мережі дозволяють зменшувати складність зображень, що потрібно обробити без втрати важливих деталей, що ще краще оптимізує необхідні характеристики.

Таким чином, напрямок розробки програмного забезпечення з використанням нейронних мереж чи штучного інтелекту для задач класифікації зображень активно розвивається та має свої специфічні риси. Тому розробка способу ідентифікації фейкових зображень документів для систем ідентифікації особистості є актуальною.

1.3 Аналіз існуючих рішень для подібних задач

Інформаційні технології у сфері розпізнавання зображень досить розвинені на сьогоднішній день. Розгляд та аналіз подібних рішень допоможе встановити потрібний функціонал для майбутньої інформаційної системи, а також можливі труднощі при його реалізації, що вже були описані.

До прикладу була взята публікація «Neural Network-based System for Automatic Passport Stamp Classification» [10]. В ній розглядається проблематика класифікації штампу на закордонному паспорті та розробляється прикладне рішення з використанням нейронної мережі.

Головне завдання – забезпечити швидке, гнучке та надійне рішення перевірки штампів паспортів. Для цього алгоритм автоматизації на базі нейронної мережі буде перевіряти паспорт, що забезпечує швидкість. Гнучкість же потрібна через величезне різноманіття штампів із різних країн, що можуть відрізнятися по формі, кольорам і не тільки, тому мережа має розрізняти не тільки окремо взятий тип штампу.

Перший етап – підготування фотографії до аналізу було розділено на три основних частини:

– конверсія – фотографія з формату RGB переводиться в чорно-білу;

– зміна розміру – стандартизація розміру всієї картинки для подальшого аналізу;

– фільтрація – відсіювання «шумів» (спотворень).

В результаті обробки, з початкового зображення (рисунок 1.8) отримуємо чорно-білий зліпок з контурами печаті, що далі буде передаватися самій нейронній мережі для подальшої обробки.

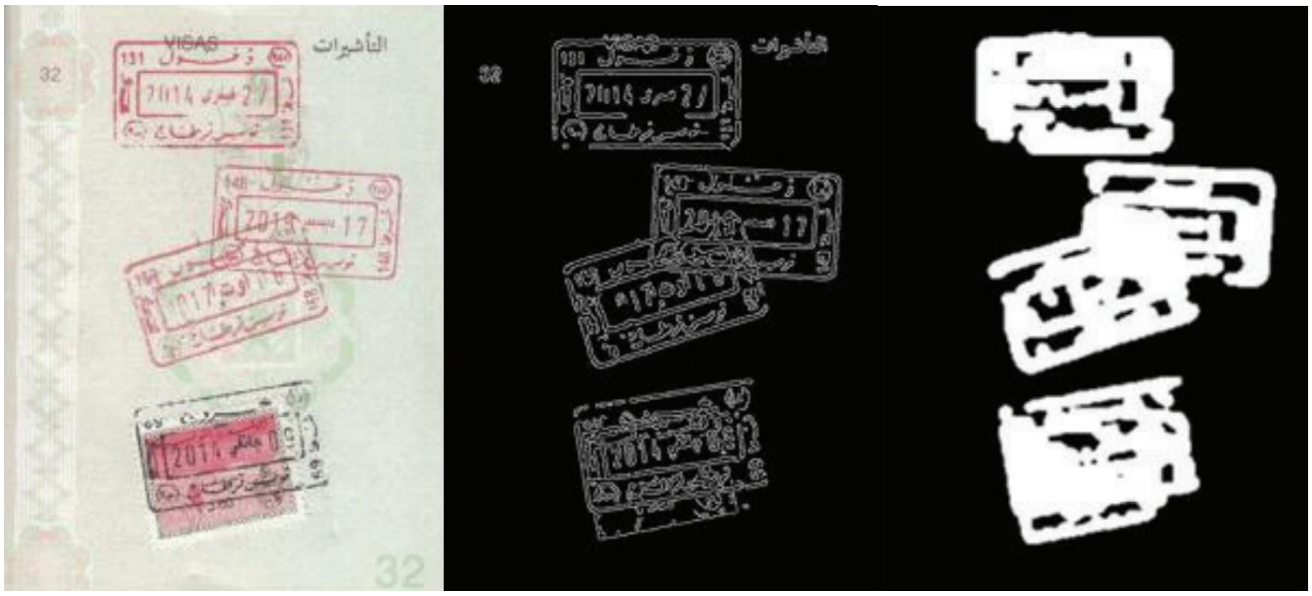


Рисунок 1.8 – Приклад обробки фоторгафії, наведеної в публікації [10]

Другим етапом буде саму отримати марку з обробленого зображення. Це досягається в два кроки:

– знаходження зовнішніх контурів;

– фільтрування всіх об’єктів що не є маркою.

Результат роботи – розділення вхідного зображення на декілька зображень з марками (рисунок 1.9).

Третій етап – сама класифікація за допомогою багато-шарового перцептрону (рисунок 1.10).

Крім того, третій етап створення системи це не тільки алгоритм класифікації, але і його навчання за допомогою великої кількості самплів. Для

чого був створений датасет із ~9120 марок та випадковим чином розподілений на декілька копій навчальної мережі використовуючи різні алгоритми.

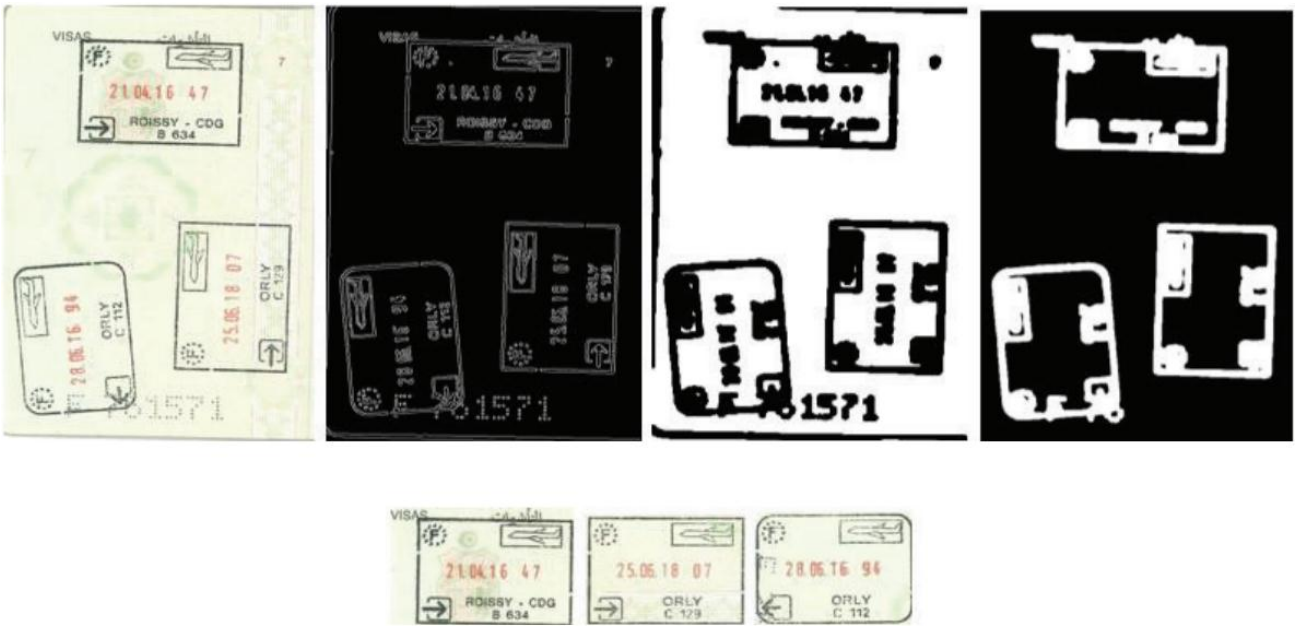


Рисунок 1.9 – Результат на момент третього етапу [10]

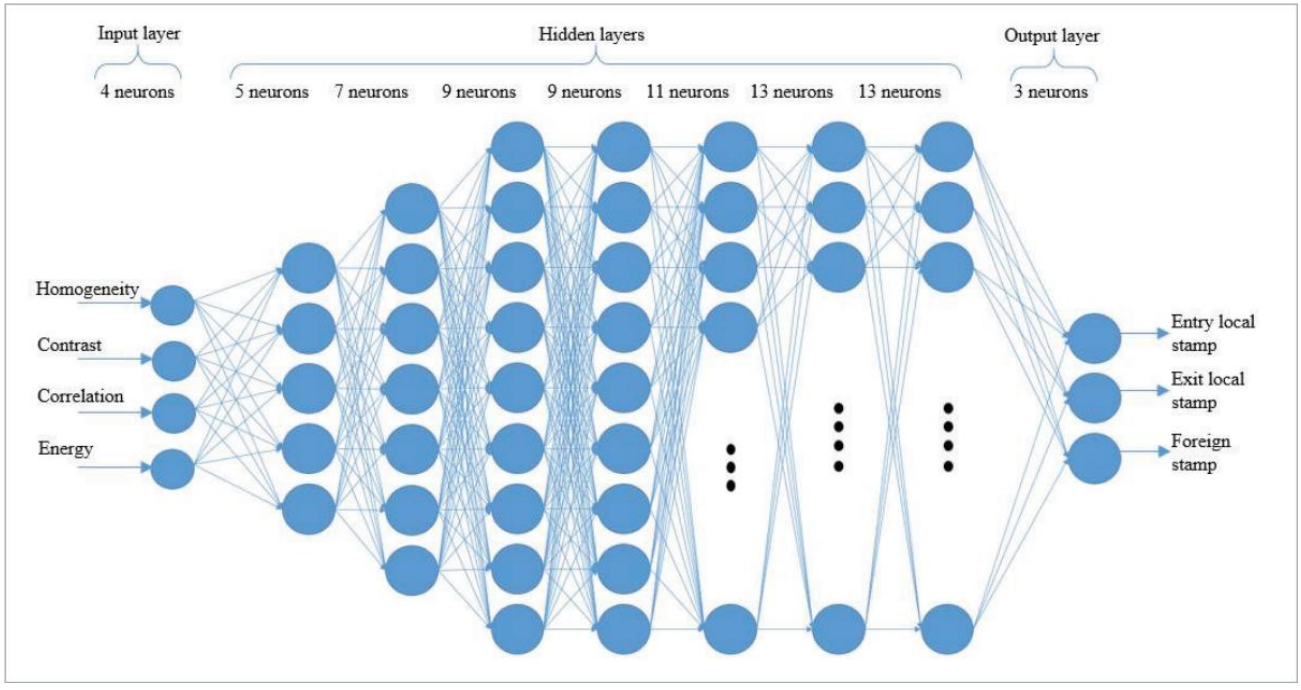


Рисунок 1.10 – Схема перцептрону, представленого в публікації [10]

В кінці перебору 15 разів за метод класифікації отримали приблизний граф (рисунок 1.11) з найвищим відсотком в 96.23% для перцептрону.

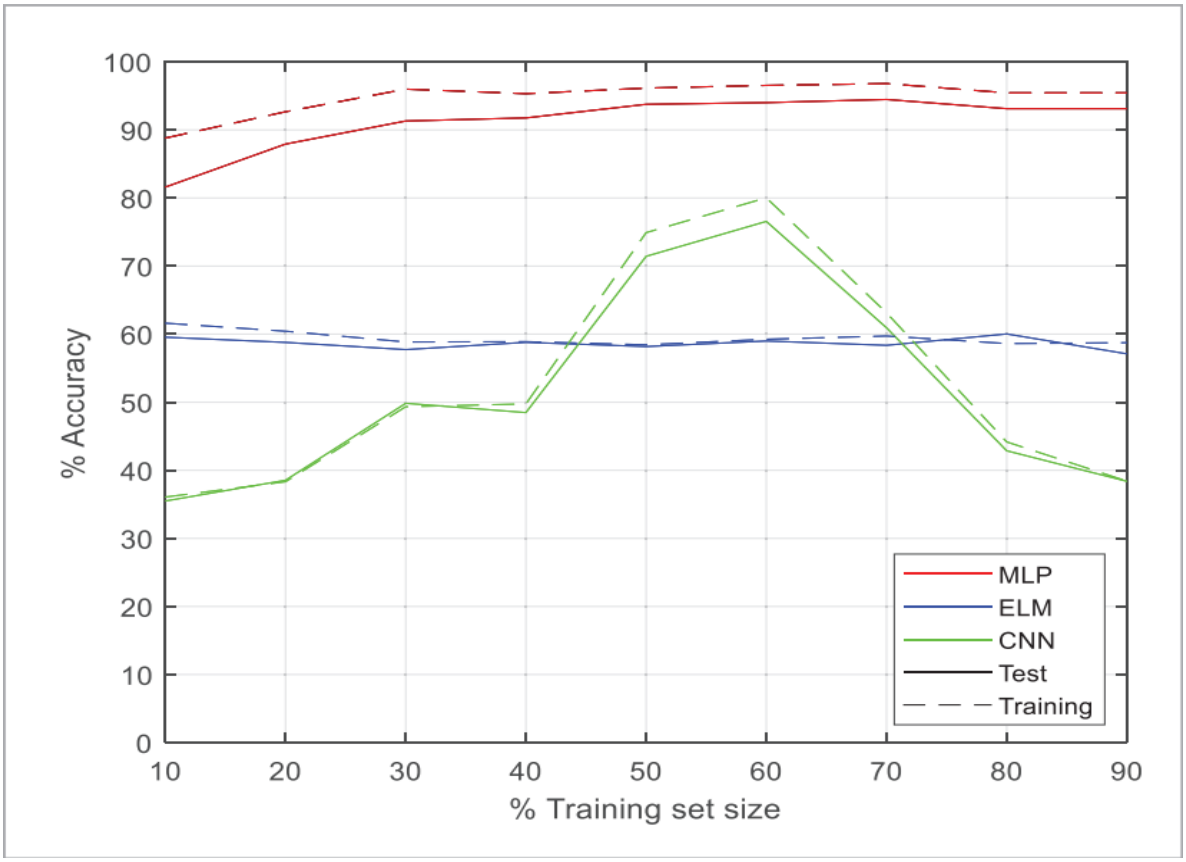


Рисунок 1.11 – Граф точності [10]

Зазначається, що для CNN алгоритму датасет був занадто малим, що потребувало б доробки. Крім того при подальшому покращенні нейронна мережа може проводити не тільки класифікацію, але і повний аналіз достовірності.

Схожу проблему також розбирають в статті «Deep learning for automated forgery detection in hyperspectral document images» [11]. Метою даної публікації є розбір можливих підробок в підписах документів на основі різних типів чорнил.

Датасет складається із блоків по п’ять речень написаних різними синіми чи чорними чорнилами одним із семи авторів (рисунок 1.12).

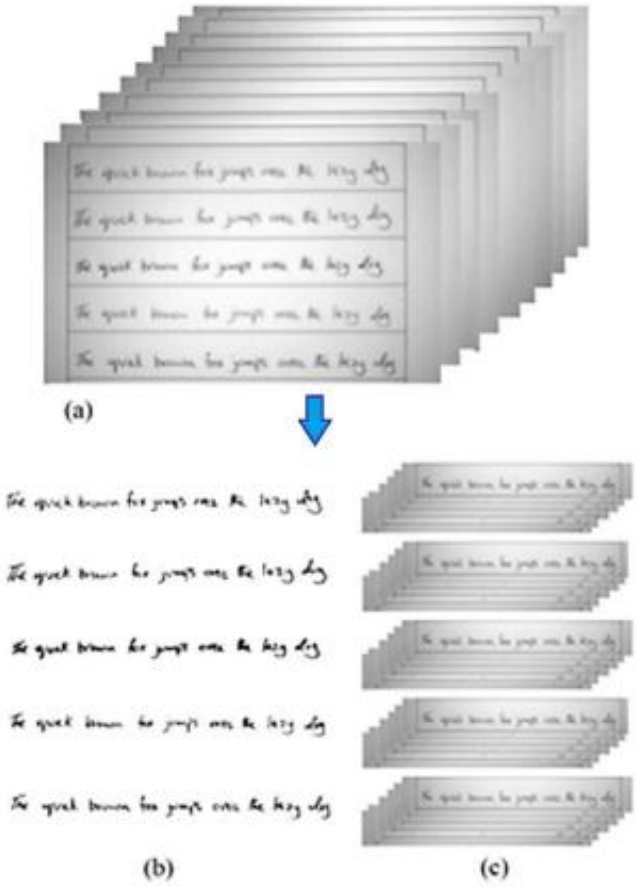


Рисунок 1.12 – Приклад із датасету [11]

Для тестування обраного рішення в різних ситуаціях частини датасету були обрані для сплайсів – розділення двох самплів з різними чорнилами на частини та комбінацію в один (рисунок 1.13).

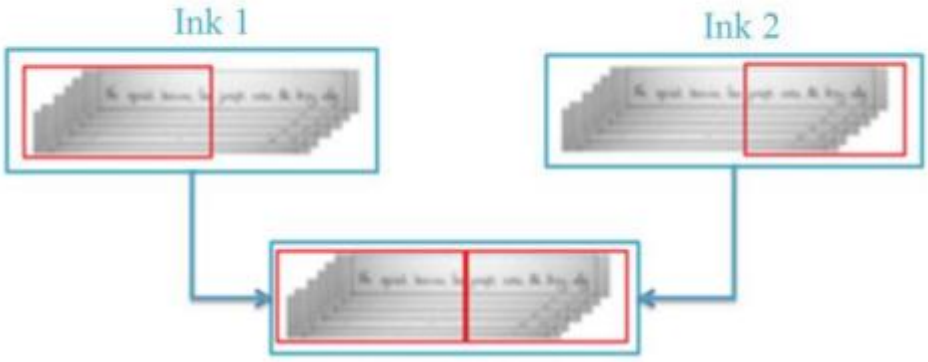


Рисунок 1.13 – Розширення датасету [11]

Було розроблено декілька різних структур CNN та проведено тестування індивідуально на кожній із них для порівняння результатів. Обирали між двома типами фільтрів 3x3 чи 5x5. Для кожного фільтру обрано три комбінації шарів нейронів – 2/4/6 з різною комбінацією внутрішніх фільтрів. Крім того був застосований шар для запобігання перенавчання. Фінальний шар складаються з 5 нейронів для кожного із п’яти типів чорнил (рисунок 1.14).

CNN - 1	**CNN - 2**	**CNN - 3**	**CNN - 4**	**CNN - 5**	**CNN - 6**
Network Layers					
InputLayer ConvLayer1(3x3, 6) ReLU-Layer ConvLayer2(3x3, 18) ReLU-Layer MaxPooling1(2x2) Dropout-Layer Fully-Connected(5) Softmax-Layer	InputLayer ConvLayer1(5x5, 6) ReLU-Layer ConvLayer2(5x5, 18) ReLU-Layer MaxPooling1(2x2) Dropout-Layer Fully-Connected(5) Softmax-Layer	**InputLayer** **ConvLayer1(3x3, 6)** **ReLU-Layer** **ConvLayer2(3x3, 18)** **ReLU-Layer** **MaxPooling1(2x2)** **ConvLayer3(3x3, 36)** **ReLU-Layer** **ConvLayer4(3x3, 54)** **ReLU-Layer** **MaxPooling2(2x2)** **Dropout-Layer** **Fully-Connected(5)** **Softmax-Layer**	InputLayer ConvLayer1(5x5, 6) ReLU-Layer ConvLayer2(5x5, 18) ReLU-Layer MaxPooling1(2x2) ConvLayer3(5x5, 6) ReLU-Layer ConvLayer4(5x5, 18) ReLU-Layer MaxPooling2(2x2) Dropout-Layer Fully-Connected(5) Softmax-Layer	InputLayer ConvLayer1(3x3, 6) ReLU-Layer ConvLayer2(3x3, 18) ReLU-Layer MaxPooling1(2x2) ConvLayer3(3x3, 36) ReLU-Layer ConvLayer4(3x3, 54) ReLU-Layer MaxPooling2(2x2) ConvLayer5(3x3, 72) ReLU-Layer ConvLayer6(3x3, 90) ReLU-Layer MaxPooling3(2x2) Dropout-Layer Fully-Connected(5) Softmax-Layer	InputLayer ConvLayer1(5x5, 6) ReLU-Layer ConvLayer2(5x5, 18) ReLU-Layer MaxPooling1(2x2) ConvLayer3(5x5, 6) ReLU-Layer ConvLayer4(5x5, 18) ReLU-Layer MaxPooling2(2x2) ConvLayer5(5x5, 72) ReLU-Layer ConvLayer6(5x5, 90) ReLU-Layer MaxPooling3(2x2) Dropout-Layer Fully-Connected(5) Softmax-Layer
Number of Train Parameters					
2,346	4,188	**37,500**	80,796	163,464	372,648
Average Accuracy (Blue, Black)					
(96.8, 85.1)	(95.4, 85)	**(98.2, 88)**	(97.1, 86.2)	(65.6, 60.3)	(59.8, 51.8)

Рисунок 1.14 – Тестування різних варіацій структури CNN [11]

Процес тренування складається із спектрального аналізу картинки відсортованих по шести із семи авторів та типу чорнила, сампли написаним сьомим автором будуть використані для подальшого тестування (рисунок 1.15).

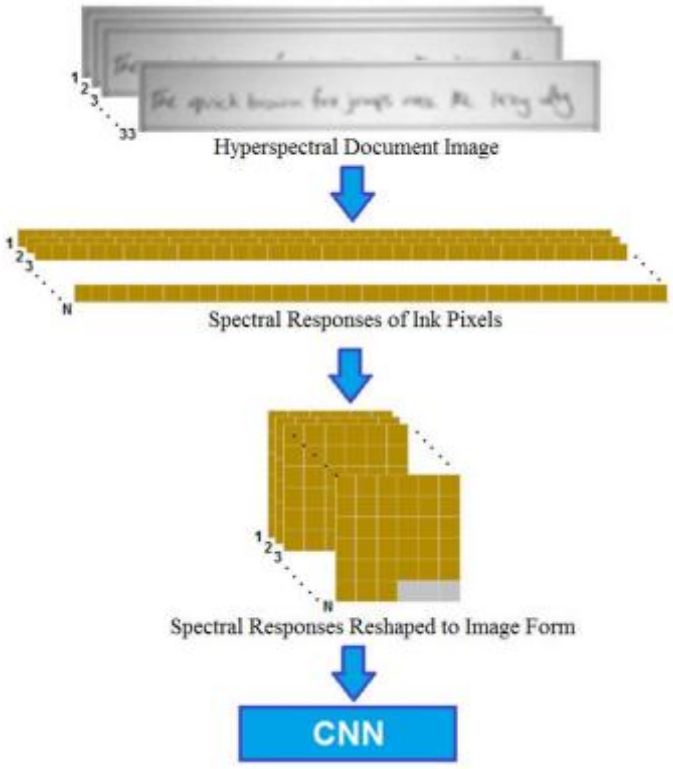


Рисунок 1.15 – Стратегія для процесу класифікації [11]

В результаті тестування було помічено, що використання меншої кількості вхідних фільтрів, але більша кількість шарів на подальших шарах дали кращий результат (рисунок 1.16).

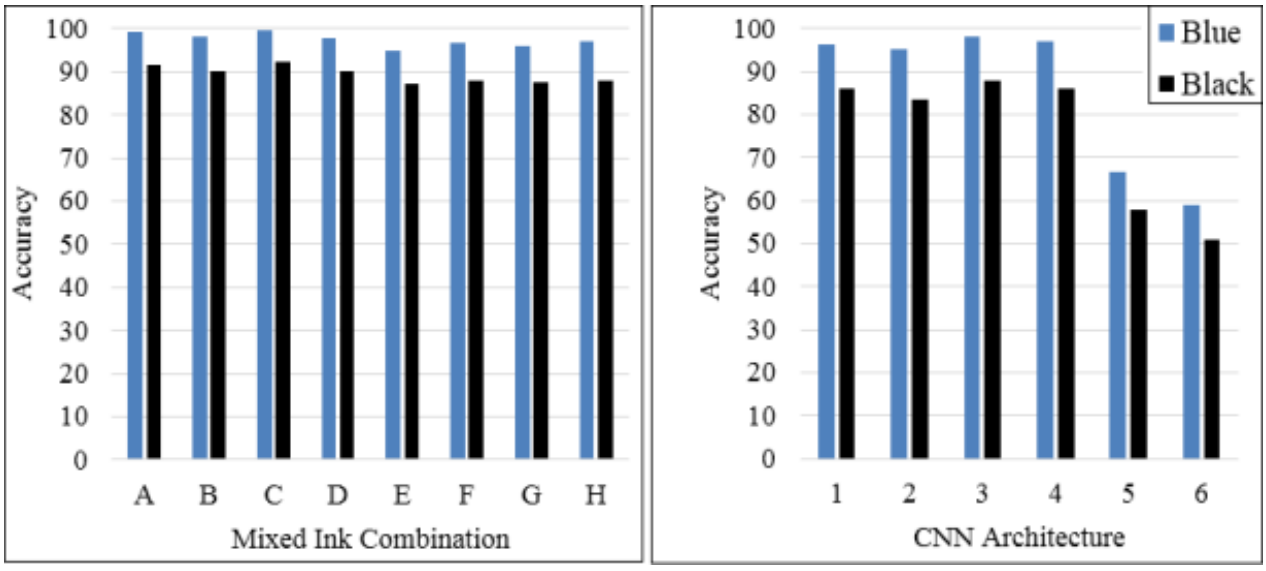


Рисунок 1.16 – Результат тестування CNN [11]

Як підсумок, були проаналізовані практичні реалізації CNN в сфері ідентифікації зображень. Під час аналізу було виявлено переваги та недоліки використаних методів, які потрібно враховувати під час власної реалізації програмного застосунку.

1.4 Мета, задачі та вимоги до реалізації програмної системи

Метою роботи є розробка та програмна реалізація способу нейромережевого виявлення фейкових зображень документів для систем ідентифікації особистості, для чого слід вирішити наступні задачі:

– розробити спосіб нейромережевого виявлення фейкових зображень документів для систем ідентифікації особистості;

– спроєктувати структуру інформаційної системи ідентифікації особистості, що використовує спосіб нейромережевого виявлення фейкових зображень документів;

– розробити архітектуру згорткової нейронної мережі для виявлення фейкових зображень документів;

– виконати програмну реалізацію інформаційної системи;

– провести тестування розробленої інформаційної системи.

Розділ 2 Спосіб нейромережевого виявлення фейкових зображень документів для систем ідентифікації особистості

2.1 Опис способу нейромережевого виявлення фейкових зображень документів

Загалом, способи створення фейкових зображень документів значно не відрізняються від створення звичайних фейкових зображень за допомогою графічних редакторів, чи іншого подібного інструментарію.

Одні із самих розповсюджених способів модифікації зображень це:

– Image retouching (ретуш зображення) – зміна деяких характеристик зображення (рисунок 2.1);

Рисунок 2.1 – Ретуш зображення

– Copy-move (копіювання) – копіюється одна частина картинки поверх іншої (рисунок 2.2);

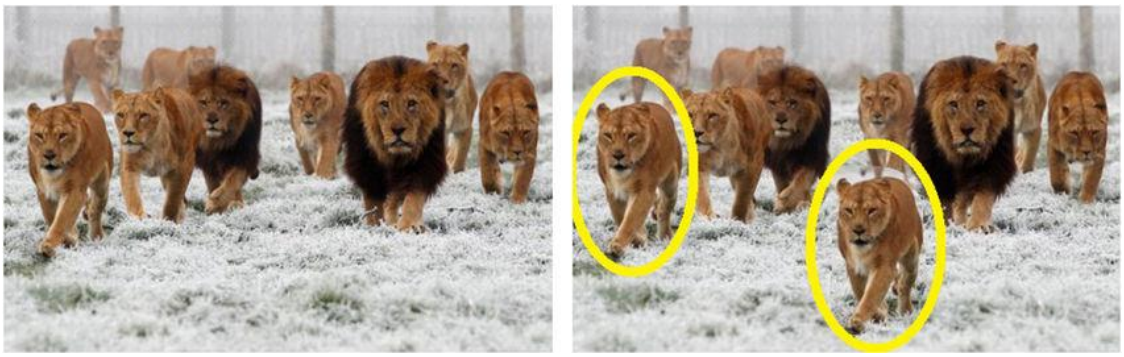


Рисунок 2.2 – Копіювання зображення

– Splicing (сплетення) – з частин двох чи більше зображень створюється одне (рисунок 2.3);

Рисунок 2.3 – Сплайс зображення

– Processing (обробка) – зображення проходить через зміну кольорів та фільтри (рисунок 2.4).

Рисунок 2.4 – Використання фільтрів для редагування зображення

При маніпуляціях з картинкою різного характеру, наприклад в Photoshop чи Paint.net, зазвичай буде допущена похибка в місці редагування і/або в метадаті.

Метадата – це додаткова інформація що містить файл зображення. Вона може включати в себе тип камери, інформацію про позицію кольорі, тощо.

Такі похибки можуть бути використані для аналізу зображення на предмет модифікацій за допомогою комбінації алгоритмів ELA та CNN в функціональних методах системи для ідентифікації фейкових зображень документів (рисунок 2.5).

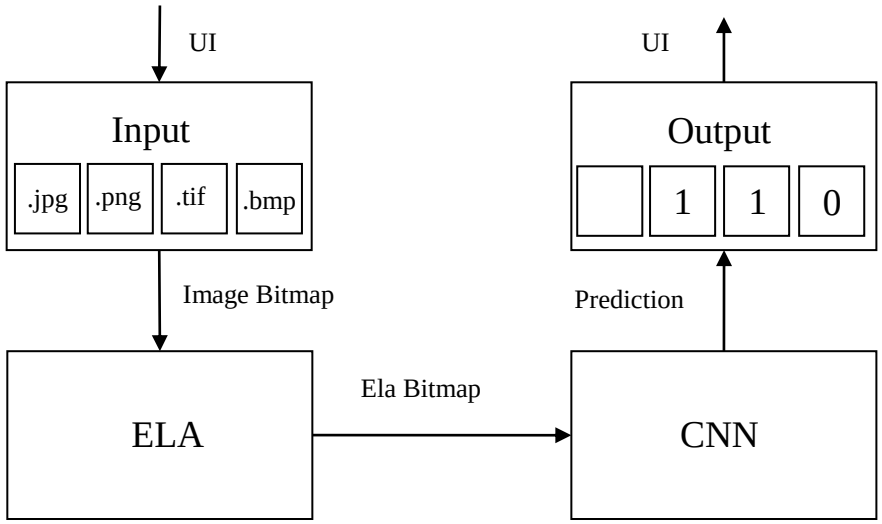


Рисунок 2.5 – Узагальнена схема способу нейромережевого виявлення фейкових зображень документів

Input – метод взаємодії з користувачем що дає змогу завантажувати вибране зображення через UI та вказувати відповідні параметри.

ELA / Error Level Analysis (Аналіз рівняння помилок) – один із способів ідентифікації змін в зображенні за допомогою компресії на заданому рівні якості та порівняння його з із оригіналом. Якщо картинку не редагували, помилка має бути однакова на кожному окремому сегменті (рисунок 2.6). В іншому випадку редаговане місце буде виділятися (рисунок 2.7).

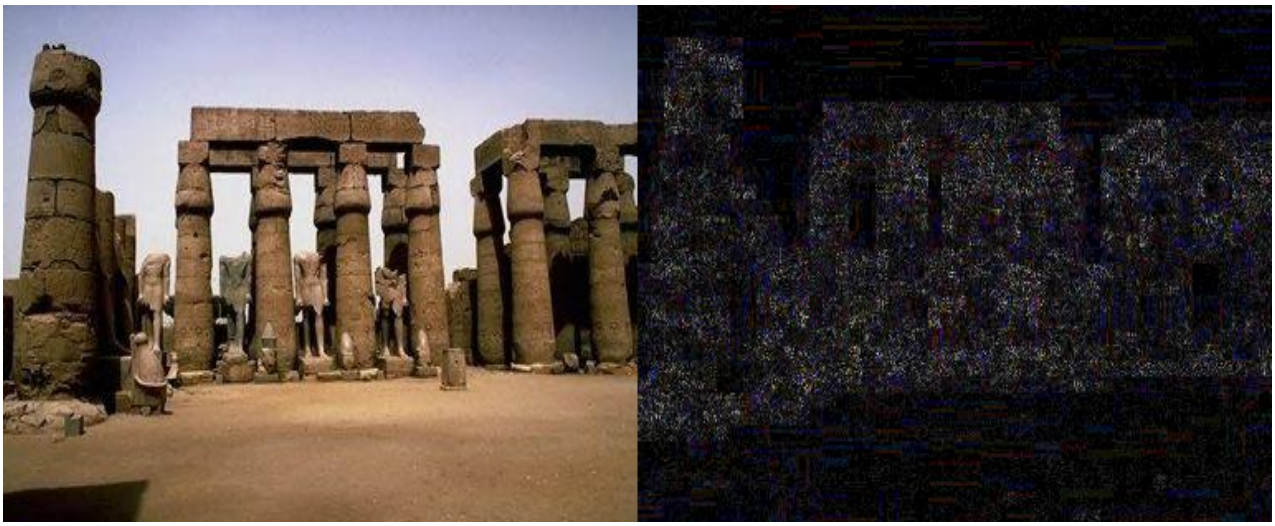


Рисунок 2.6 – Приклад ELA для реального зображення

Рисунок 2.7 – Приклад ELA для фейкового зображення з виділеною зоною сплайсу

Завдання функціонального методу це зберегти вхідне зображення з заданою користувачем якістю, порівняти їх та створити ELA що надалі буде використовуватися нейронною мережею (рисунок 2.8).

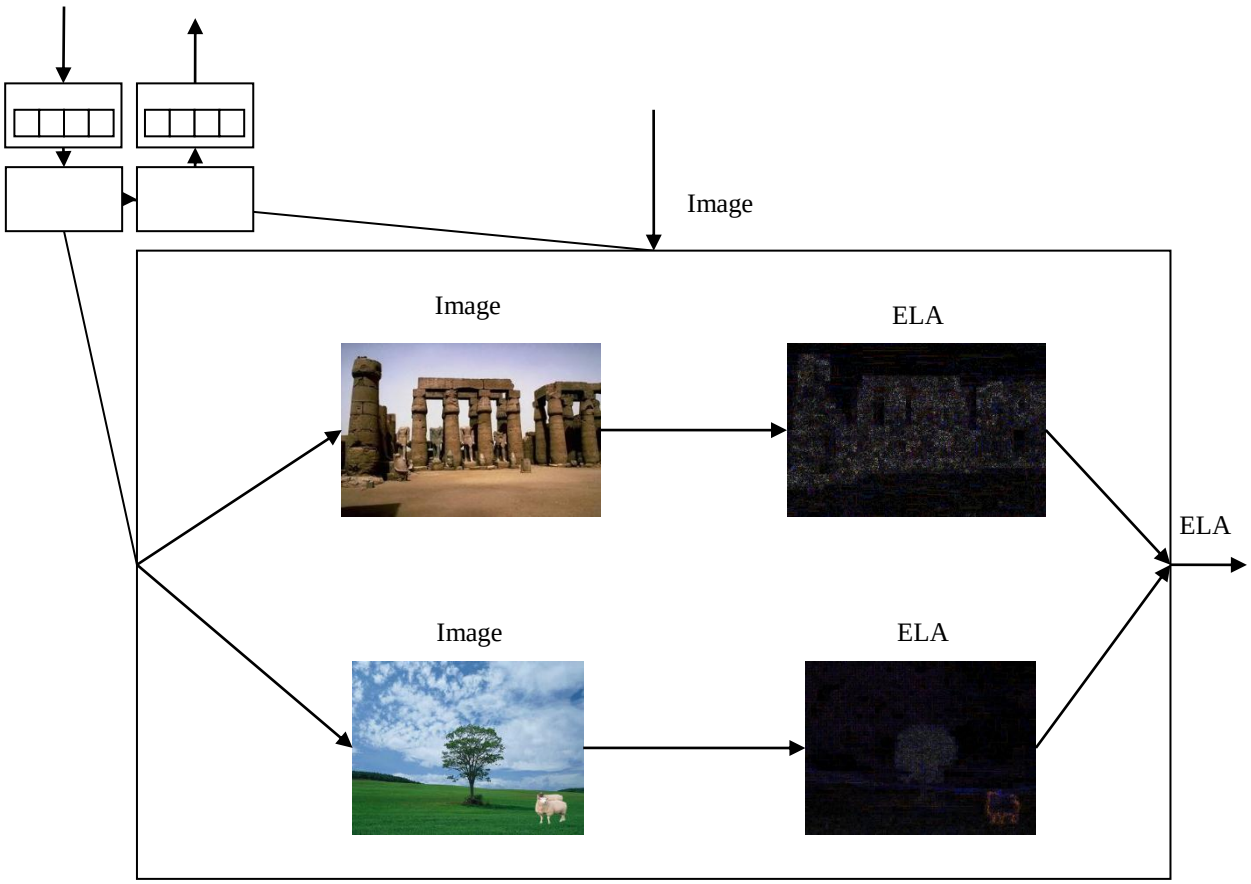


Рисунок 2.8 – Функціонал модуля ELA

CNN/ConVnet – Convolutional neural network (Згорткова нейронна мережа) – тип нейронних мереж, що розроблений для вимоги мінімального обсягу попередньої обробки. Складається з шарів входу, виходу та декількох прихованих шарів. Приховані шари можуть бути згорткові, агрегувальні, повноз’єдні та шари нормалізації.

Завдання методу CNN – за допомогою зображень навчитися класифікувати отримане зображення на справжні та відредаговані, після процесу навчання мережа буде має класифікувати зображення.

Output – модуль інтерпретації та візуалізації результату роботи CNN в зручній формі для користувача.

Таким чином, було розроблено спосіб нейромережевого виявлення фейкових зображень документів для систем ідентифікації особистості.

2.2 Проєктування структури інформаційної системи ідентифікації особистості

Важливою частиною перевірки достовірності зображення документів є ідентифікація особистості, тому подальша розробка програмного застосунку має враховувати можливість взаємодії підсистем ідентифікації фейкових зображень документів із системою ідентифікації особистості та відповідною базою даних (рисунок 2.10).

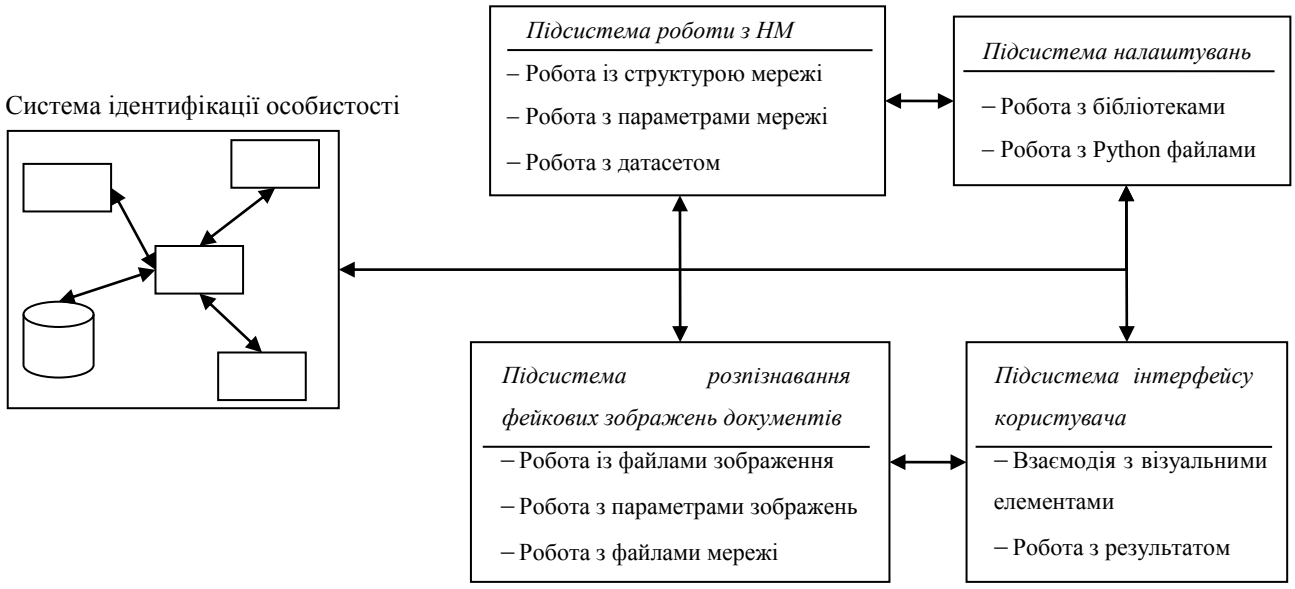


Рисунок 2.9 – Підсистеми програмного застосунку

Підсистема роботи з НМ:

– робота із структурою мережі – налаштування функціональних шарів нейронної мережі, їх параметрів таких як вхідний тип чи розширення;

– робота з параметрами мережі – налаштування загальних параметрів мережі, таких як кількість епох, розмір груп чи якість конвертування зображень;

– робота з датасетом – вибір файлу з посиланнями на зображення для навчання з відповідно вказаними класами.

Підсистема розпізнавання фейкових зображень документів:

– робота із файлами зображення – завантаження зображення для класифікації з використанням нейронної мережі;

– робота з параметрами зображень – зміна якості конвертування зображення;

– робота з файлами мережі – вибір відповідних файлів мережі для подальшого її використання при класифікації.

Підсистема налаштувань:

– робота з бібліотеками – вибір чи зміна python.dll файлу;

– робота з Python файлами – зміна шляху до файлів Python.

Підсистема інтерфейсу користувача:

– взаємодія з візуальними елементами – використання елементів UI для взаємодії користувача з іншими функціональними модулями;

– робота з результатом – візуалізація результату з інших функціональних модулів для користувача.

Таким чином, була спроєктована структура інформаційної системи ідентифікації особистості.

2.3 Розробка архітектури нейронної мережі для ідентифікації фейкових зображень документів

Для коректної роботи НМ обов’язковим та надзвичайно важливим етапом є навчання мережі. Навчання дає мережі можливість класифікувати вхідне зображення згідно «згодованих» прикладів в процесі навчання, та в залежності від коректного налаштування самого процесу, мережа буде робити правильний чи не правильний виснови.

Існує два основних алгоритми навчання для нейронних мереж – з вчителем та без.

При навчанні без вчителя, тобто «кластеризація», НМ не знає правильних відповідей і сама відносить дані до певних класів – кластерів. Цей метод має значні втрати по точності і не підходить для даного завдання.

При навчанні з вчителем вхідні дані потрібно розділити на певні класи – класифікація (в нашому випадку автентичність/фейк). Після навчання мережі за допомогою функції втрат рахується відсоток помилок, ваги корегуються та навчання проходить знов, до тих пір коли відсоток помилок буде не мінімізований або досягне оптимальних значень (рисунок 2.10).

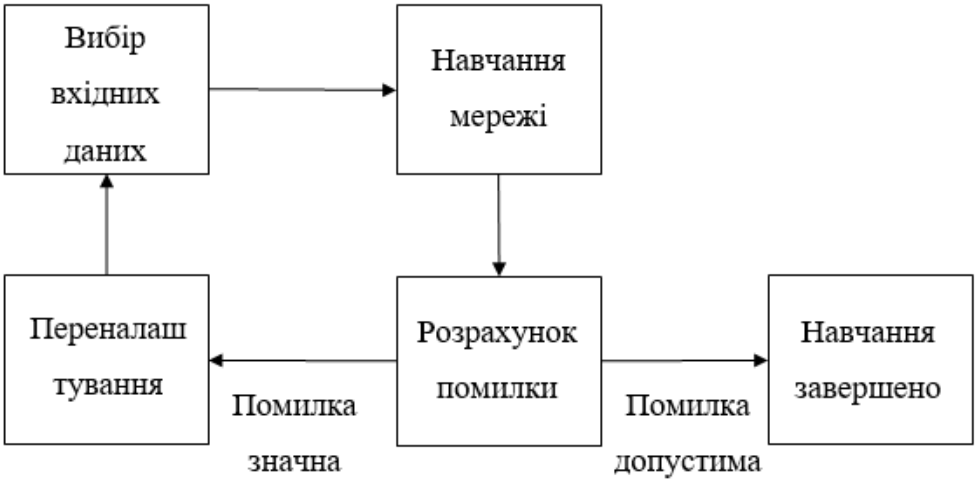


Рисунок 2.10 – Процес навчання

Структура самої мережі розділяється на основні функціональні шари:

– Convlution layer (згортка) – перший шар що використовується для збору ознак (feature extraction) із зображення, виводить карту ознак (feature map) для подальшої обробки іншими шарами мережі;

– Pooling layer (агрегувальний шар) – зазвичай використовується після згортки, його функція це зменшення розмірів карти ознак для зменшення необхідних обчислень, що пришвидшує роботу та зменшує необхідні вимоги до комп’ютера;

– Fully connected layer (повзноз’єднаний шар) – існує для з’єднання між собою різних шарів нейронної мережі, де зображення сплючують (flattned) та передають на наступний шар, зазвичай використовуються перед шаром виводу;

– Dropout – для запобігання перенавчання використовується цей шар що видаляє частину нейронів із мережі, зменшуючи її розмір.

Таким чином за допомогою комбінації даних шарів була створена архітектура мережі для ідентифікації фейкових зображень документів (рисунок 2.11).

Іншим важливим параметром мережі є кількість епох (epoch) – кількість циклів. Ця кількість визначається за об’ємом даних та об’ємом кожного окремої групи (batch-size) зображень що передається в мережу (рисунок 2.12).

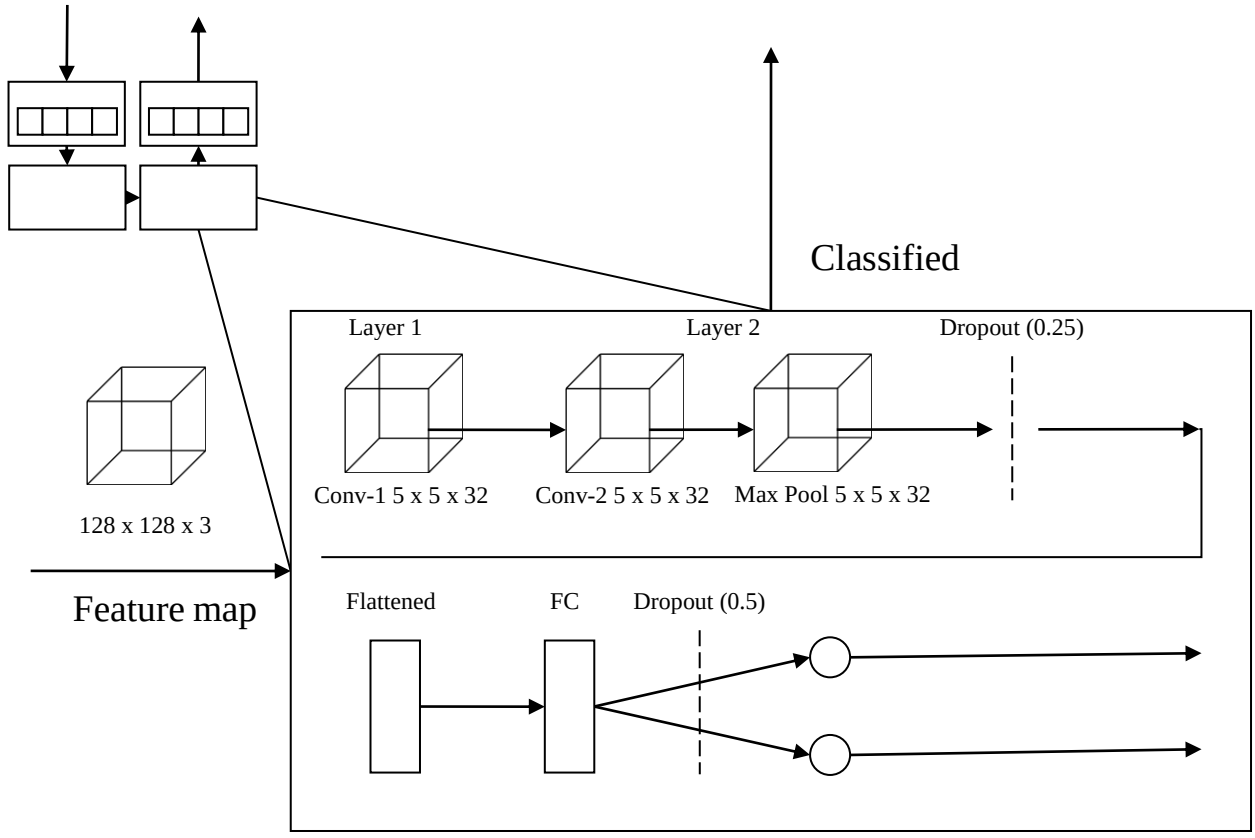


Рисунок 2.11 – Архітектура нейромережі

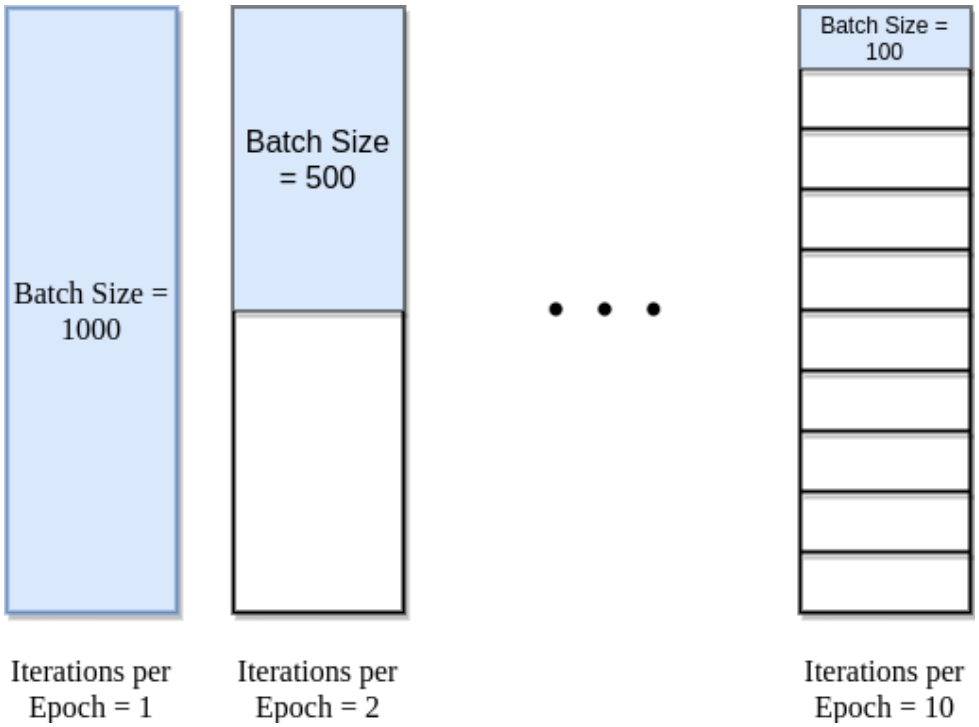


Рисунок 2.12 – Епохи в нейромережі

В випадку, якщо розмір груп не розподілений рівномірно відносно загальної кількості зображень, в такому випадку одні і ті ж самі дані будуть використані більше одного разу.

Оптимальна кількість епох буде визначатися за допомогою тестування. Крім того вважається можливим налаштування як і структури та і епох за допомогою користувацького інтерфейсу.

Таким чином була створена архітектура НМ для ідентифікації фейкових зображень документів.

2.4 Підготовка робочих вхідних даних для системи

Для коректної роботи НМ важливим етапом розробки є підбір коректного набору даних, на яких і буде навчатися мережа – датасету.

Для навчання та тестування мережі були взяти датасети CASSIA2 [12] та MIDV [13].

CASSIA2 складається з більше чим 12 000 різноманітних автентичних (рисунок 2.32) та фейкових зображень (рисунок 2.14).

Рисунок 2.13 – Автентичне зображення [12]

Рисунок 2.14 – Фейк [12]

MIDV складається з 500 фотографій та кліпів різноманітних документів для ідентифікації особистості (рисунок 2.15).

Рисунок 2.15 – Приклад датасету [25]

З нього потрібно відсортувати та вирізати зображення що будуть використані для тестування (рисунок 2.16).

Рисунок 2.16 – Вирізане зображення із датасету [25]

Отриманий датасет потрібно обробити – всі файли мають різне розширення, формат та якість, їх потрібно звести до одного стандарту та класифікувати для подальшого використання нейронною мережею

Класифікація це процес розділення отриманих зображень на класи, що потім будуть передаватися на входи мережі. В даному випадку ці класи відповідно «справжній» та «фейк».

Крім того, датасет потрібно випадковим чином розділяти при кожній ітерації моделі на три частини – тренувальну, валідаційну та тестову (рисунок 2.17).

Рисунок 2.17 – Приклад розділення датасету

Тренувальна частина – основна частина що складається з найбільшої (~70-80%) кількості зображень та буде використовуватись для тренування моделі.

Валідаційна частина – другий по розміру (~20-10%) сегмент для тестування ітерацій моделі та зміни її параметрів.

Для запобігання перенавчання та коректного тестування моделі потрібен третій сегмент – тестовий. Складається з найменшої кількості даних (~10%) та слугує для фінальної перевірки.

В випадку, коли від відсутній, модель може видавати хороші результати, але ці результати будуть специфічні для конкретно цього датасету і можуть сильно різнитися при тестуванні на інших даних.

Якщо результати тестування моделі на валідаційному сегменті досягли бажаних – модель можна вважати навченою. Якщо результат значно відрізняються потрібно перенавчити модель аналізуючи допущені помилки та уникаючи їх при повторному навчанні.

2.5 Вибір засобів розробки інформаційної системи

2.5.1 Вибір мови програмування

Для створення додатку по нейромережевому виявленню фейкових зображень документів були обрані мови програмування C# та Python, а також середовища розробки Visual studio [14] та Visual Studio Code [15].

C# – об’єктно-орієнтована мова програмування, що дозволяє створювати кросплатформенні додатки будь-якого призначення. Важливою особливістю є статична типізація [16].

Основні переваги C#:

– легка структура для розуміння;

– підтримка ООП;

– підтримка поліморфізму;

– статична типізація.

Ця мова програмування була обрана для реалізації UI – візуальної частини додатку з якою буде взаємодіяти користувач.

Python – об’єктно-орієнтована мова програмування, на відміну від C# має динамічну типізацію [17].

Основні переваги Python:

– логічний та простий синтаксис;

– гнучкість та масштабованість;

– інтерпретованість дозволяє створювати та редагувати код без використання спеціалізованого середовища розробки;

– постійна підтримка як і самої мови програмування так і спільноти в цілому.

Python була обрана через велику кількість спеціалізованих бібліотек машинного навчання та простоту інтеграції. Вона буде використовуватися для реалізації функціональної частини.

2.5.2 Вибір бібліотек

Для реалізації функціональної складової додатку були обрані наступні бібліотеки:

– Tensorflow – відкрита програмна бібліотека для тренування та навчання нейронних мереж, здатних виявляти та розшифровувати образи і кореляції [8];

– Keras – відкрита нейромережна бібліотека спеціалізована під python що здатна працювати поверх tensorflow [18];

– Numpy – python розширення з розширеною підтримкою багатовимірних масивів та високорівневих математичних функцій [19];

– Pandas – бібліотека для python спрямована на маніпулювання та аналіз даних, включаючи таблиці числові ряди та графіки [20];

– PIL – (Python Image Library) – бібліотека призначена для роботи з растровою графікою [21];

– MathPlotLib – інструментарій для графічної візуалізації написаний на Python [22];

– Scikit-learn – бібліотека надає інструменти для ефективного аналізу даних, включно з аналізом результатів навчання мереж [23].

Використання цих бібліотек значно спростить та оптимізує роботу додатку в порівнянні з написанням власних функцій.

Розділ 3 Програмна реалізація інформаційної системи ідентифікації особистості

3.1 Структура модулів інформаційної системи та їх взаємозв'язок

В попередньому розділі було розроблено спосіб для ідентифікації зображення фейкових документів та обрано датасети, що є важливою частиною для вирішення поставленого завдання.

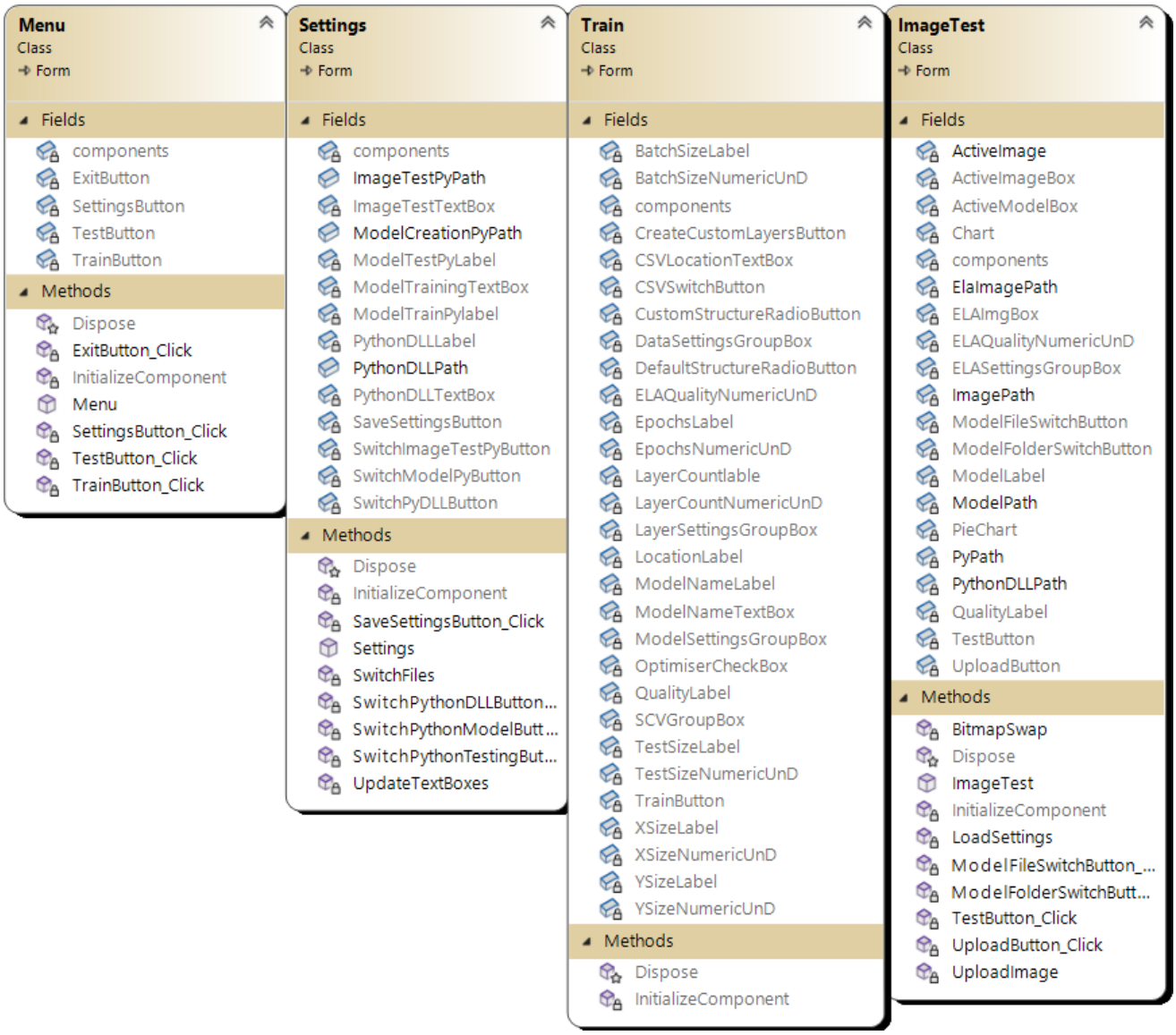


Рисунок 3.1 – Діаграма класів програмного застосунку по ідентифікації фейкових документів

Крім того, була розроблена інформаційна модель та архітектура нейронної мережі. Далі має бути розроблена інформаційна система ідентифікації особистості, яка використовує розроблений спосіб й дозволяє за відсканованим чи сфотографованим зображеннями документу що посвідчує особу автоматизовано визначати рівень його достовірності за допомогою згорткової нейронної мережі.

Для прикладної реалізації способу ідентифікації фейкових зображень документів було створено діаграмі класів відповідно до очікуваного функціоналу (рисунок 3.1).

Робота додатку починається із класу Menu. На формі реалізований графічний інтерфейс для переходу на інші форми, з якими буде працювати користувач (рисунок 3.2).

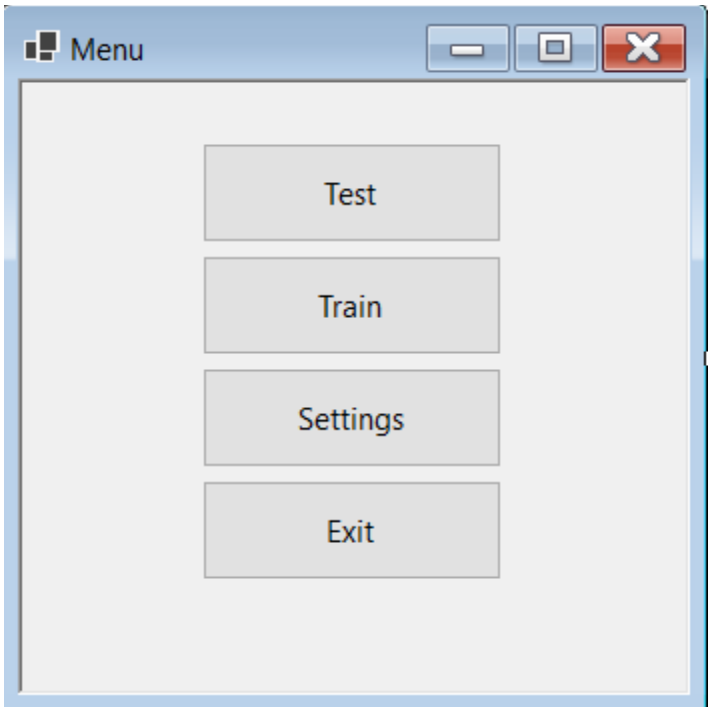


Рисунок 3.2 – Інтерфейс модулю класу Menu

Методи SettingsButton_Click, TestButton_Click та TrainButton_Click дозволяють перейти на обрану форму при натисненні відповідної кнопки.

Клас Settings містить методи для зміни шляхів до Python класів та DLL файлу, що необхідний для їх роботи. Зміни зберігаються в форматі txt файлу, що потім буде завантажуватись іншими модулями при подальшій роботі, та будуть збережені в випадку повторного завантаження програми (рисунок 3.3).

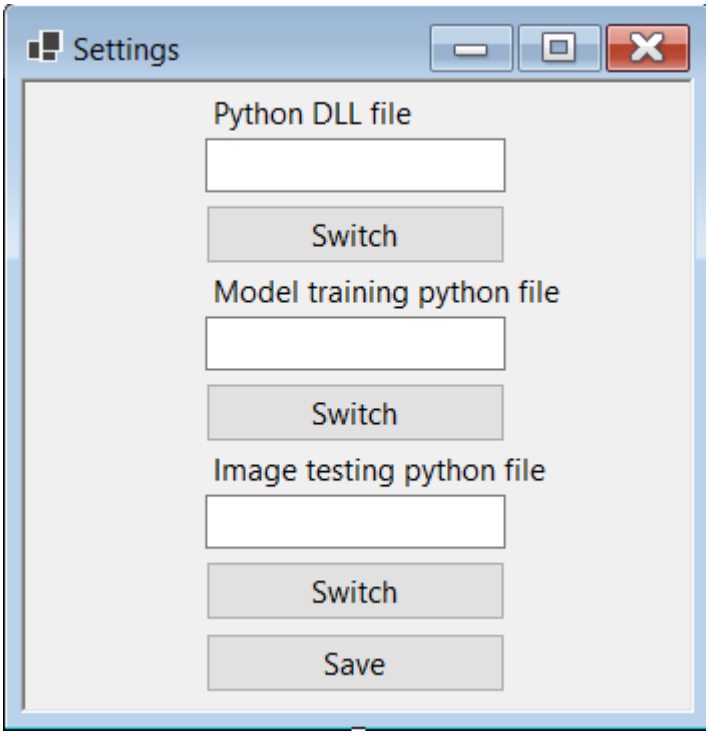


Рисунок 3.3 – Інтерфейс класу Settings

Методи SwitchPythonDLLButton_Click, SwitchPythonModelButton_Click та SwitchPythonTestingButton_Click викликають інтерфейс для зміни відповідних файлів. SaveSettingsButton_Click зберігає внесені зміни в Settings.txt файл з шляхами до файлів та коментарями до них.

У класі Train реалізована логіка налаштування, створення, тренування CNN та її збереження для подальшого використання (рисунок 3.4).

Метод LoadSettings реалізує завантаження файлу з налаштуваннями, при його існуванні, в іншому випадку використовуються шляхи по замовчуванню. Перед викликом методу Train користувачу потрібно вказати відповідні параметри нейронної мережі та CSV файл з розподілом зображень по класах (фейк/не фейк). Train завантажує .py файл, написаний на Python, з методами

навчання нейронної мережі та викликає їх використовуючи дані з форми що ввів користувач. Після завершення навчання модель буде збережена в папці SavedModels/ з відповідною назвою папки самої моделі, що ввели до цього, виведено вікно з повідомленням про завершення навчання та буде доступна можливість змінити параметри та навчити іншу мережу.

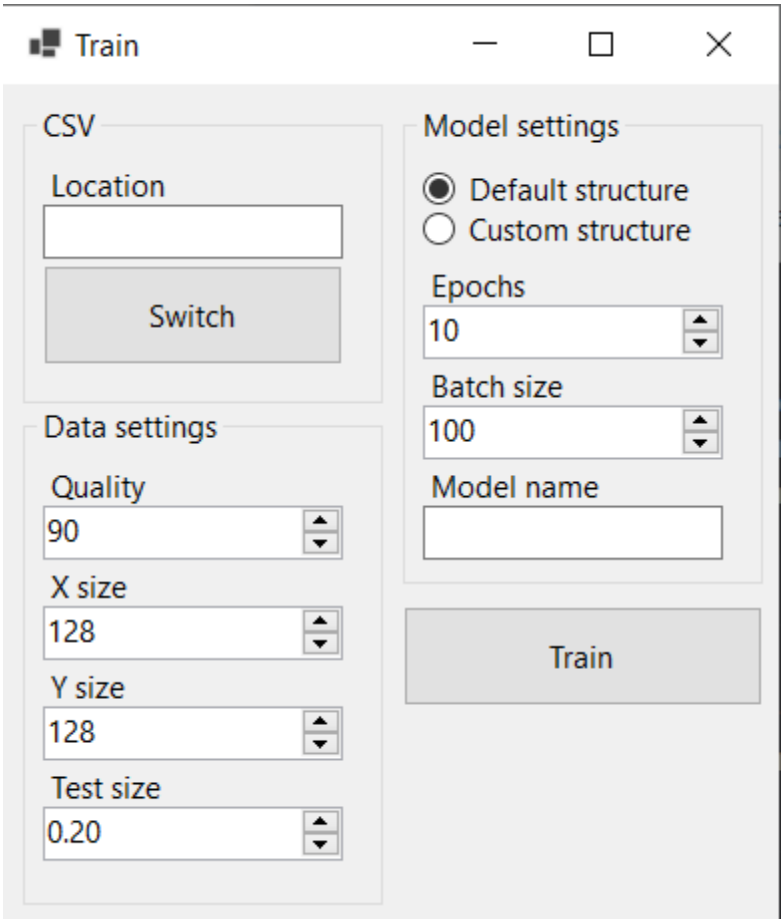


Рисунок 3.4 – Інтерфейс класу Train

ImageTest це клас, що слугує для перевірки достовірності вибраного зображення документу за допомогою навченої мережі (рисунок 3.5).

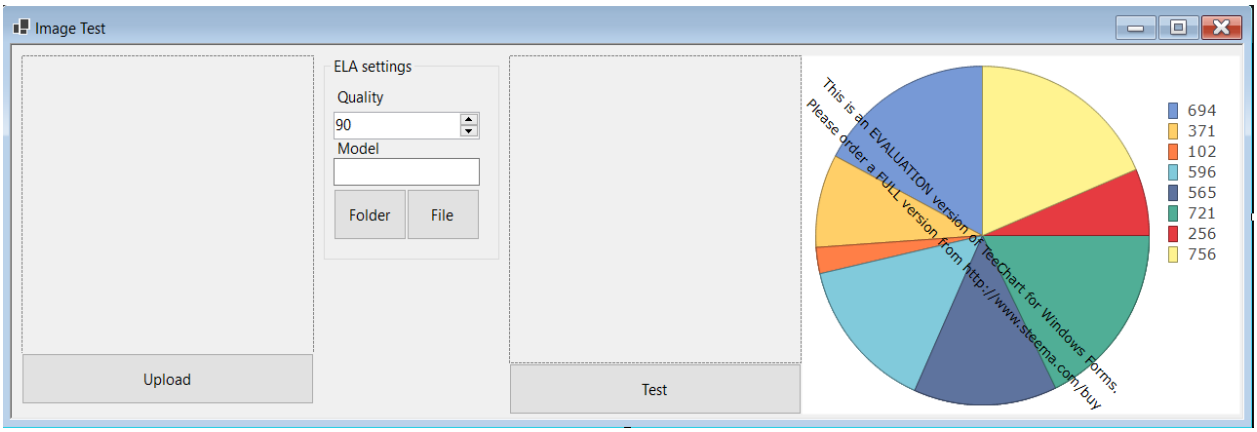


Рисунок 3.5 – Інтерфейс класу ImageTest

LoadSettings, аналогічно до відповідного методу класу Train, завантажує налаштування шляхів до потрібних файлів. UploadImageButton_Click викликає метод UploadImage, що в свою чергу викликає інтерфейс для завантаження зображення для подальшого використання. BitmapSwap дозволяє паралельно з роботою додатку перемістити, видалити чи змінити використаний файл зображення без впливу на роботу програми. Методи ModelFileSwitchButton_Click та ModelFolderSwitchButton_Click дозволяють вибрати відповідно файл чи папку з моделлю, що буде використана для перевірки зображення. ImageTestButton_Click завантажує налаштований .py файл з методом ImageTest для перевірки зображення, викликає його з використанням введених на формі параметрів та вибраної нейронної мережі та відображує отриманий результат в форматі кругового графіку.

Таким чином, під час розробки способу ідентифікації фейкових зображень документів для системи ідентифікації особистості були отримані вище-описані модулі системи, з якими буде вести роботу користувач.

3.2 Особливості реалізації інформаційної системи

Важливою частиною розробки програмного забезпечення є чітка структура та розділ програмного застосунку на функціональні під-частини – класи, модулі, блоки тощо.

Перший модуль програмного коду, з якого розпочинає роботу додаток – меню, де реалізовані методи переходу на інші інтерфейси (рисунок 3.6).

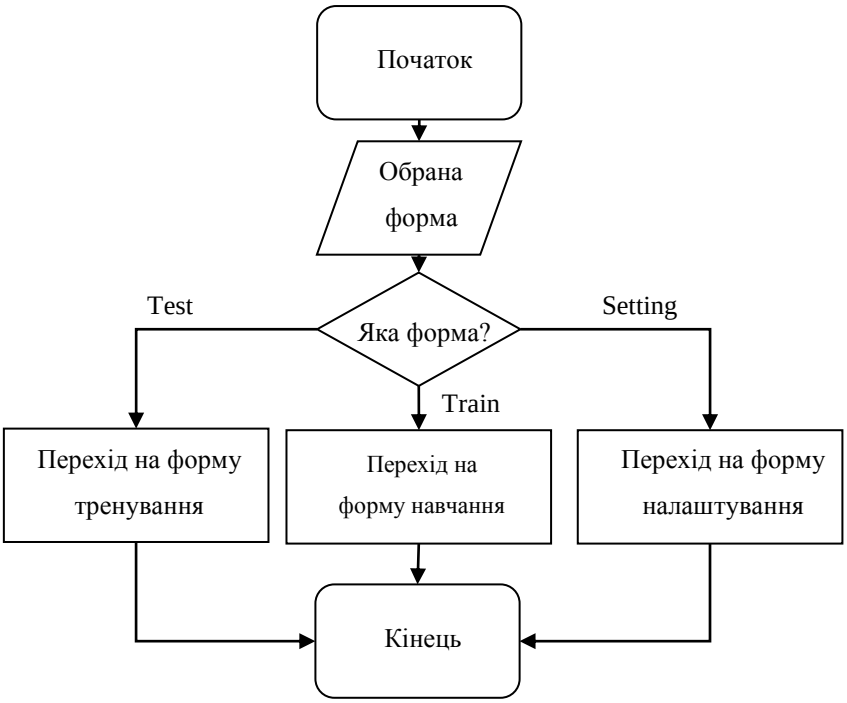


Рисунок 3.6 – Реалізація зміни форм

Розроблена система використовує додаткові DLL та .py файли, що завантажуються програмно, шлях до них може змінитися що буде викликати помилку при спробі їх завантаження.

Щоб запобігти це, була реалізована можливість налаштування шляху до відповідних файлів з використанням фільтру по типу (рисунок 3.7).

Крім того, внесені зміни потрібно зберігати, щоб не було потреби повторювати процес налаштування при кожному завантажені програми. Для цього був реалізований метод запису змін до текстового файлу (рисунок 3.8).

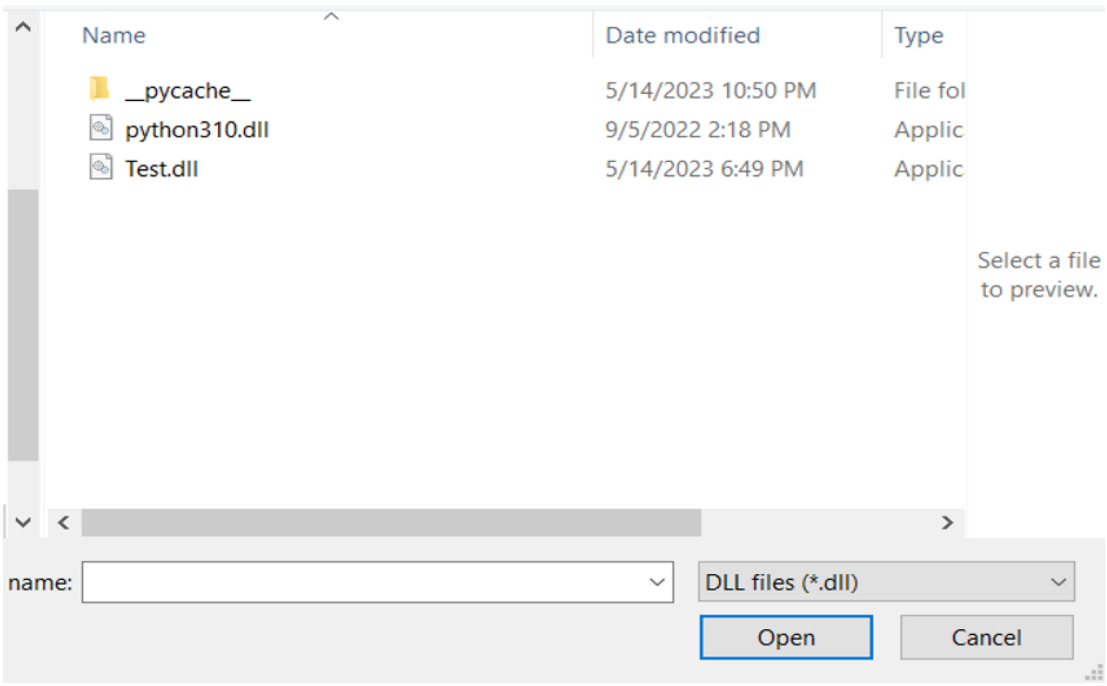


Рисунок 3.7 – Інтерфейс вибору шляху до відповідного файлу

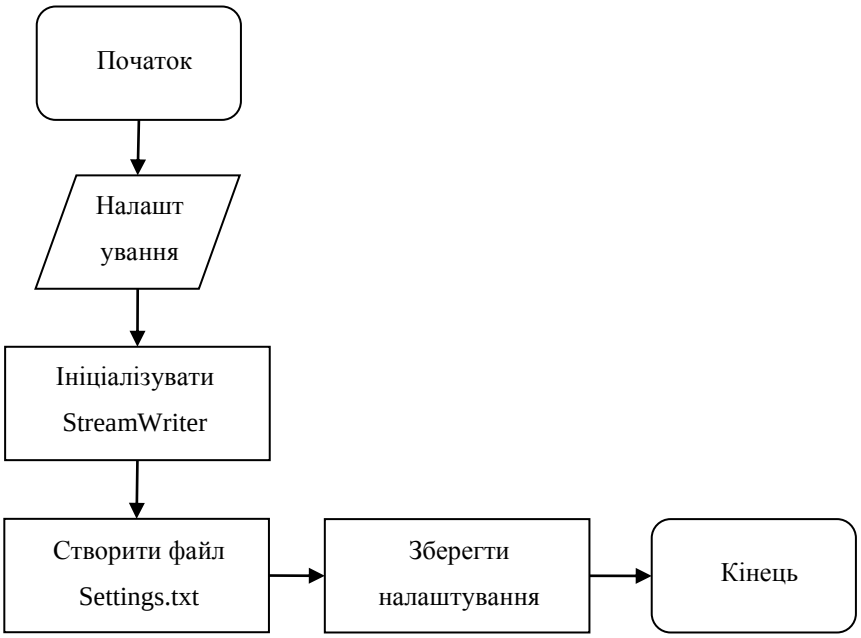


Рисунок 3.8 – Реалізація збереження налаштувань

Налаштування потрібно завантажувати при використанні інших модулів, для чого був розроблений відповідний метод для класів Train та ImageTest. В випадку, якщо файл не існує буде виведене повідомлення та модуль буде використовувати налаштування за замовчуванням.

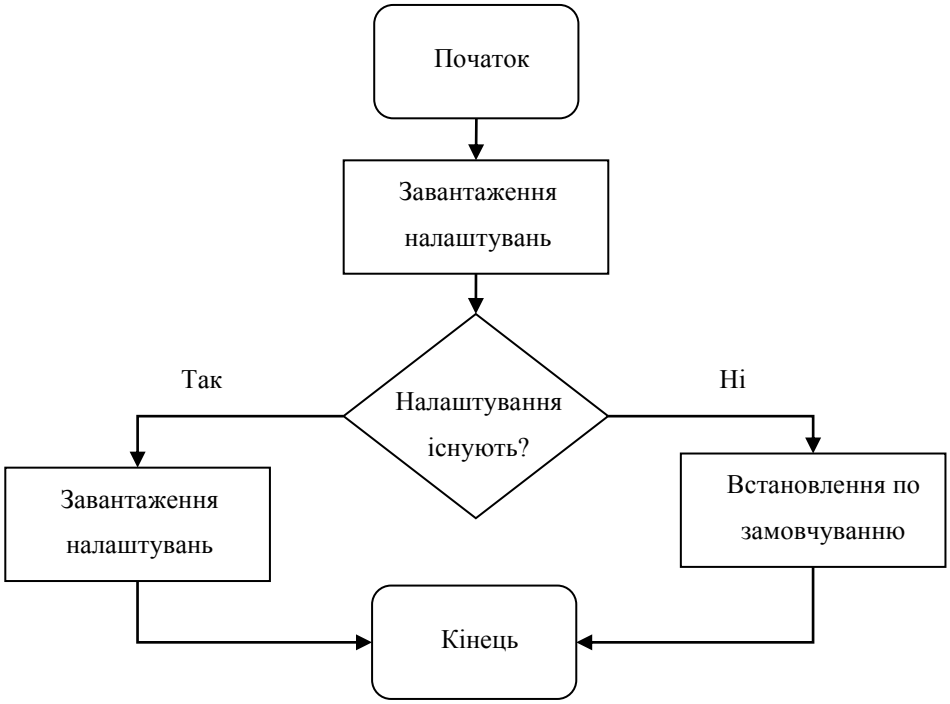


Рисунок 3.9 – Реалізація завантаження налаштувань

При завантаження файлу зображення для роботи програми, цей файл буде заблокований, система виведе відповідну помилку якщо будуть здійснені спроби перенесення чи редагування, як самого зображення, так і його параметрів (рисунок 3.10).

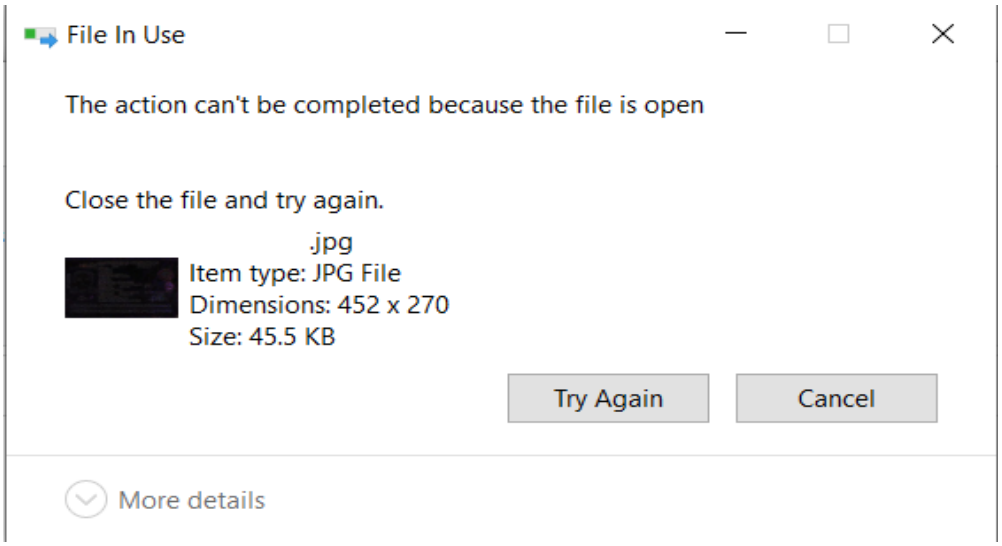


Рисунок 3.10 – Помилка при спробі редагування файлу у використанні

Щоб розблокувати файл для використання під час його використання програмою був розроблений відповідний метод.

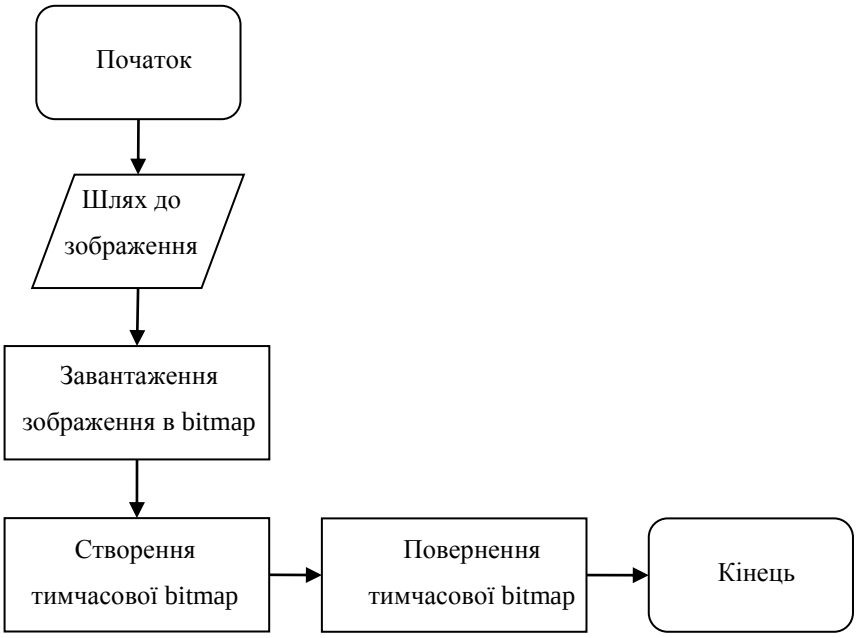


Рисунок 3.11 – Реалізація методу заміни Bitmap для розблокування зображення

Щоб реалізувати можливість гнучкого налаштування структури CNN потрібно змінювати інтерфейс форми ModelTrain (рисунок 3.4) в залежності від вибору користувача. Якщо буде використана структура по замовчуванню, величезна частина форми залишиться не використаною, тому її потрібно розширяти динамічно (рисунок 3.12).

Крім того, кожен шар має свої вхідні параметри, що відрізняються від інших. Тому в залежності від вибраного типу шару потрібно також змінювати елементи інтерфейсу відповідної групи (рисунок 3.13).

Для функціональної взаємодії модулів написаних на C# та Python була використана бібліотека PythonTet. Це стороннє розширення забезпечує можливість завантаження .py файлів, що містять методи для роботи з CNN , написаних на Python з використанням відповідних бібліотех, виклик відповідних методів із вказаними параметрами та повернення можливого результату (рисунок 3.14).

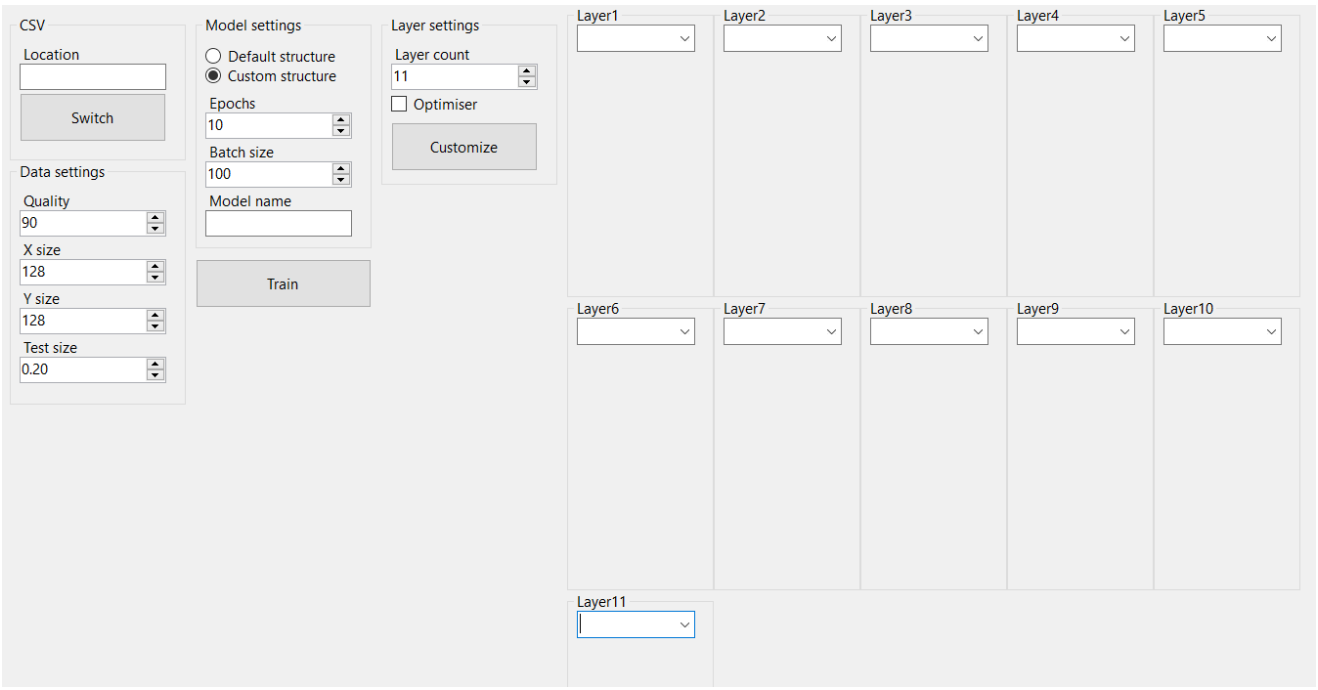


Рисунок 3.12 – Динамічно розширена частина форми ModelTrain

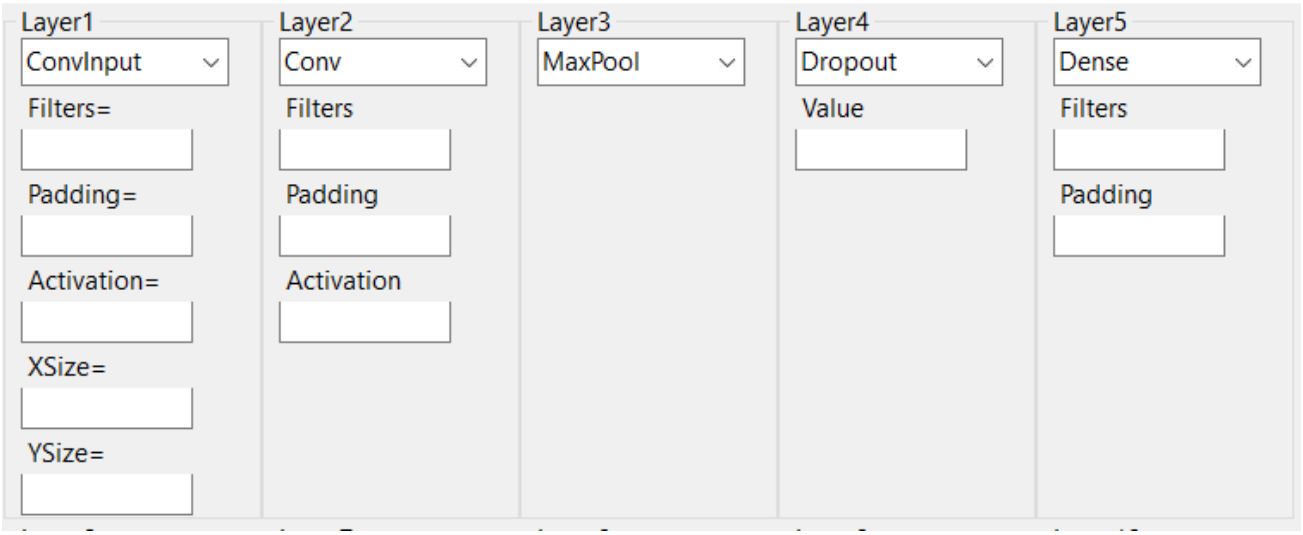


Рисунок 3.13 – Різні типи шарів з відповідними вхідними даними

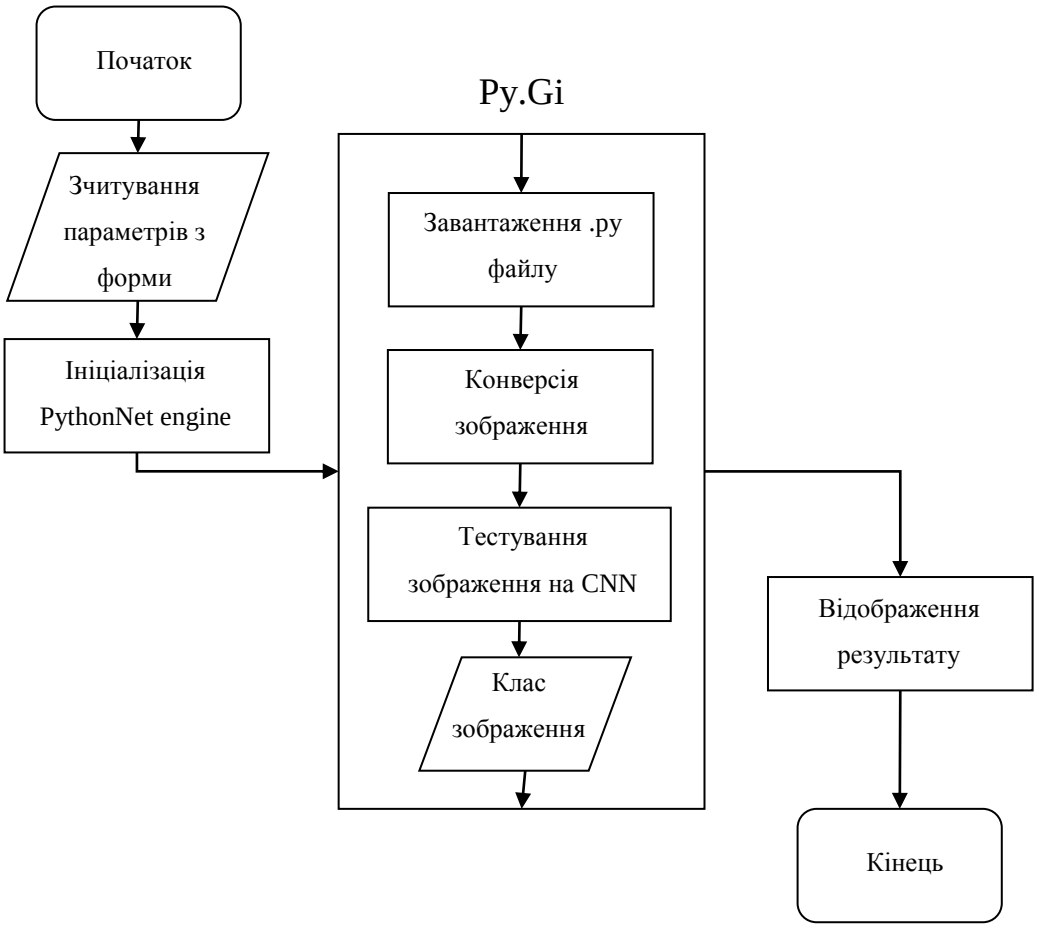


Рисунок 3.14 – Реалізація взаємодії з Python

Таким чином був реалізований програмний продукт з використанням способу ідентифікації фейкових зображень документів для системи ідентифікації особистості.

3.3 Опис функціональних можливостей інформаційної системи ідентифікації зображень

Для перевірки коректної роботи застосунку способу ідентифікації фейкових зображень документів для системи ідентифікації особистості було проведено функціональне тестування та на базі результатів створені тест-кейси для функціональних модулів ModelTest та Train.

Потрібно перевірити існування файлу налаштувань та коректно завантажити шляхи до вказаних файлів, в випадку відсутності цього файлу

потрібно використовувати налаштування по замовчуванням (таблиця 3.1) (рисунок 3.15).

Таблиця 3.1 – Тест-кейс 0.1.1

Тест-Кейс ID: 0.1.1	**Пріоритет:** 1	**Створено:** 23.04.2023
Назва: Перевірка коректного завантаження файлу налаштувань **Вхідні дані:** Нічого		
Кроки		**Очікуваний результат**
1. Відкрити застосунок 2. Обрати опцію ImageTest із стартового меню		Поява вікна із попередженням про відсутність файлу налаштувань.
Результат виконання тест-кейсу: пройдено успішно		

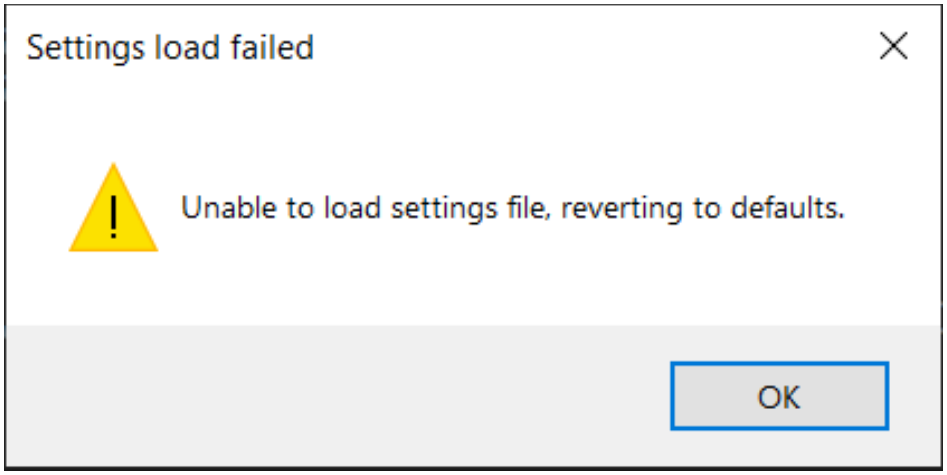


Рисунок 3.15 – Результат тестування

Нехай користувач при взаємодії з інтерфейсом закрив вікно не вказавши модель для тестування. Потрібно вивести повідомлення про некоретне завантаження моделі (таблиця 3.2) (рисунок 3.16).

Таблиця 3.2 – Тест-кейс 0.1.2

Тест-Кейс ID: 0.1.2	**Пріоритет:** 1	**Створено:** 23.04.2023
Назва: Перевірка коректності вводу моделі для тестування зображення **Вхідні дані:** Нічого		
Кроки		**Очікуваний результат**
1. Завантажити додаток та обрати форму ImageTest 2. Натиснути на кнопку завантаження моделі 3. Вийти із інтерфейсу завантаження без вказання моделі		Поява вікна із попередженням про некоректне завантаження моделі.
Результат виконання тест-кейсу: пройдено успішно		

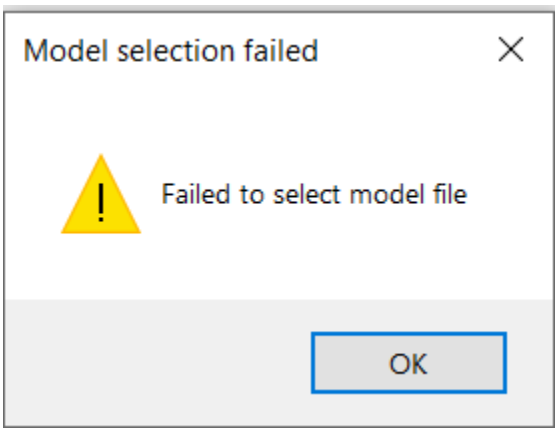


Рисунок 3.16 – Результат тестування

Крім можливості не завантажити модель в цілому, є можливість завантаження файлу чи папки що не містить модель (таблиця 3.3) (рисунок 3.17).

Таблиця 3.3 – Тест-кейс 0.1.3

Тест-Кейс ID: 0.1.3	**Пріоритет:** 2	**Створено:** 23.04.2023
Назва: Перевірка коректності завантаженого файлу моделі **Вхідні дані:** Тестовий файл зображення, пустий txt файл		
Кроки		**Очікуваний результат**
1. Завантажити додаток та обрати форму ImageTest 2. Натиснути на кнопку завантаження моделі 3. В інтерфейсі вказати потрібний txt файл 4. Почати тестування зображення		Поява вікна із помилкою завантаження моделі.
Результат виконання тест-кейсу: пройдено успішно		

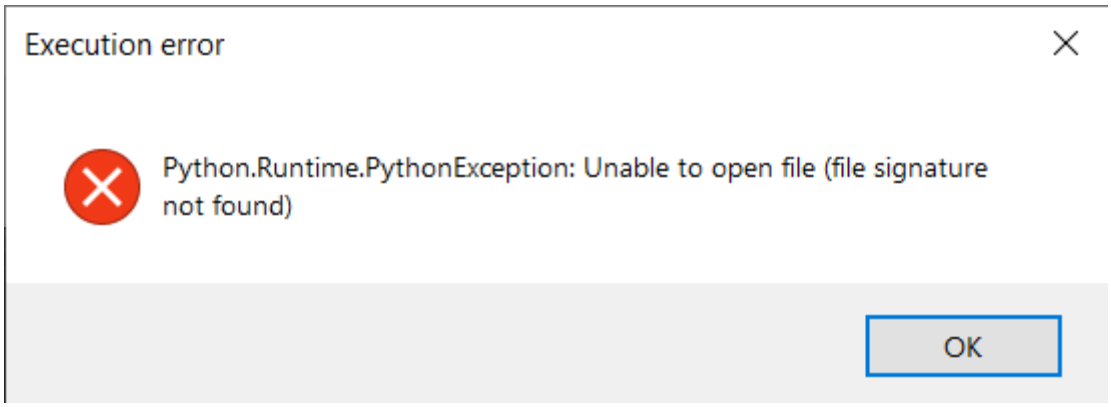


Рисунок 3.17 – Результат тестування

При завантаженні зображення можливо допущена помилка в параметрі якості конвертування, коли число буде виходити за допустимі межі. В такому випадку це значення потрібно скорегувати на максимально чи мінімально допустиме (таблиця 3.4)(рисунок 3.18).

Таблиця 3.4 – Тест-кейс 0.1.4

<table>
<tr><td>Тест-Кейс ID: 0.1.4</td><td>Пріоритет: 3</td><td>Створено: 23.04.2023</td></tr>
<tr><td colspan="3">Назва: Корегування максимального порогу якості зображення
Вхідні дані: Нічого</td></tr>
<tr><td colspan="2">Кроки</td><td>Очікуваний результат</td></tr>
<tr><td colspan="2">1. Завантажити додаток та обрати форму ImageTest
2. Натиснути на кнопку завантаження зображення та в інтерфейсі вибрати тестове зображення
3. Натиснути на кнопку завантаження мережі та в інтерфейсі вибрати папку з навченою моделлю
4. Змінити параметр якості зображення на 100000
5. Почати тестування зображення</td><td>Автоматичне корегування параметру якості зображення на максимально допустиме.</td></tr>
<tr><td colspan="3">Результат виконання тест-кейсу: пройдено успішно</td></tr>
</table>

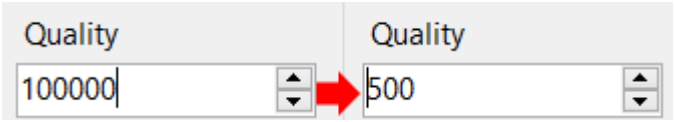


Рисунок 3.18 – Результат тестування

При тестуванні зображення та коректних вхідних параметрах можливо бути допущення помилка в налаштуванні шляху до системних файлів і файл вказаний некоректно, або не містити методу з потрібним ідентифікатором (таблиця 3.5)(рисунок 3.19).

Таблиця 3.5 – Тест-кейс 0.1.5

Тест-Кейс ID: 0.1.5	**Пріоритет:** 1	**Створено:** 23.04.2023
Назва: Перевірка коректності методів Python модулю **Вхідні дані:** Тестовий файл зображення, навчена модель, пустий .py файл вказаний в налаштуваннях		
Кроки		**Очікуваний результат**
1. Завантажити додаток та обрати форму ImageTest 2. Натиснути на кнопку завантаження зображення та в інтерфейсі вибрати тестове зображення 3. Натиснути на кнопку завантаження мережі та в інтерфейсі вибрати папку з навченою моделлю 4. Почати тестування зображення		Поява вікна із помилкою завантаження модуля Python
Результат виконання тест-кейсу: пройдено успішно		

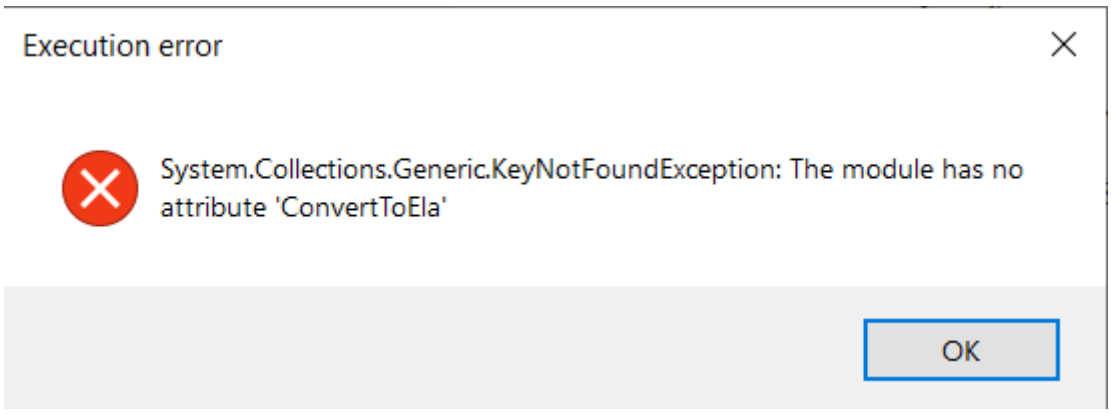


Рисунок 3.19 – Результат тестування

Вказаний файл також може містити метод з аналогічним ідентифікатором, але не повертати коректний результат, це також потрібно врахувати (таблиця 3.6)(рисунок 3.20).

Таблиця 3.6 – Тест-кейс 0.1.6

<table>
<tr><td>Тест-Кейс ID: 0.1.6</td><td>Пріоритет: 1</td><td>Створено: 23.04.2023</td></tr>
<tr><td colspan="3">Назва: Перевірка коректності повернення результату Python модулю
Вхідні дані: Тестовий файл зображення, навчена модель, .py файл вказаний в налаштуваннях що містить аналогічний метод</td></tr>
<tr><td colspan="2">Кроки</td><td>Очікуваний результат</td></tr>
<tr><td colspan="2">1. Завантажити додаток та обрати форму ImageTest
2. Натиснути на кнопку завантаження зображення та в інтерфейсі вибрати тестове зображення
3. Натиснути на кнопку завантаження мережі та в інтерфейсі вибрати папку з навченою моделлю
4. Почати тестування зображення</td><td>Поява вікна із помилкою завантаження модуля Python</td></tr>
<tr><td colspan="3">Результат виконання тест-кейсу: пройдено успішно</td></tr>
</table>

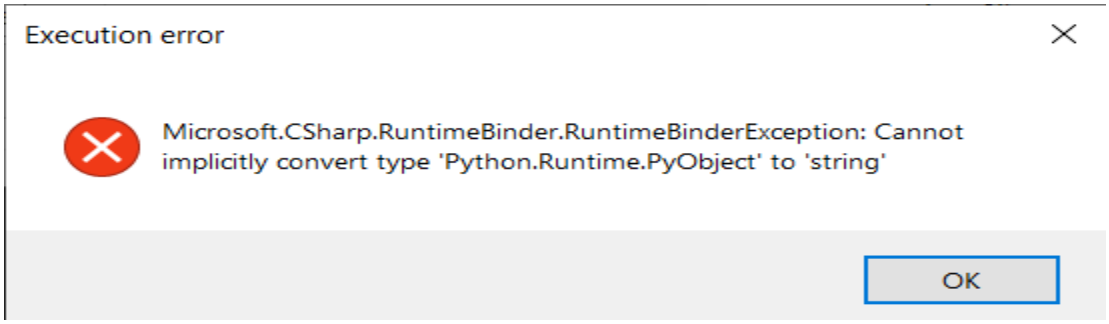


Рисунок 3.20 – Результат тестування

На цьому тестування функціонального модуля ImageTest було завершено та розпочато тестування модуля ModelTrain.

При тренуванні моделі потрібно мати датасет та файл коректно створений .csv, з шляхом та класифікацією зображень. Можливо, що під час наступного тренування моделі частина зображень буде видаленна, але .csv файл не буде відповідно змінений, система має про це повідомляти (таблиця 3.7) (рисунок 3.21).

Таблиця 3.7 – Тест-кейс 0.2.1

Тест-Кейс ID: 0.2.1	**Пріоритет:** 1	**Створено:** 23.04.2023
Назва: Перевірка коректності завантаженого датасету **Вхідні дані:** cvs файл з класифікованими зображеннями, частковий датасет		
Кроки		**Очікуваний результат**
1. Завантажити додаток та обрати форму Train 2. Натиснути на кнопку завантаження зображення csv файлу та в інтерфейсі вибрати цей файл 3. Натиснути на кнопку тренування		Поява вікна із помилкою завантаження зображення
Результат виконання тест-кейсу: пройдено успішно		

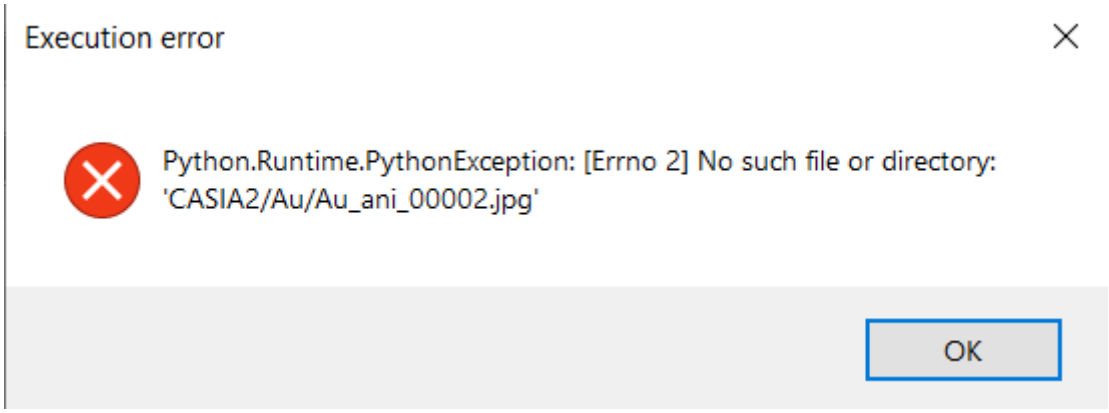


Рисунок 3.21 – Результат тестування

Аналогічно модулю ModelTest, користувач може спробувати ввести дані, що виходять за пороги мінімально чи максимально можливих чи ввести текст, що також потрібно коригувати автоматично (таблиця 3.8)(рисунок 3.22).

Таблиця 3.8 – Тест-кейс 0.2.2

Тест-Кейс ID: 0.2.2	**Пріоритет:** 3	**Створено:** 23.04.2023
Назва: Корегування максимального та мінімального порогу вхідних даних, а також їх формат **Вхідні дані:** Нічого		
Кроки		**Очікуваний результат**
1. Завантажити додаток та обрати форму Train 2. Натиснути на кнопку завантаження зображення csv файлу та в інтерфейсі вибрати його 3. Змінити параметри розширення зображень на 10000 4. Змінити розмір тестового розподілу на -20 5. Змінити кількість епох на «one» (текст)		Автоматичне корегування параметрів на допустимі
Результат виконання тест-кейсу: пройдено успішно		

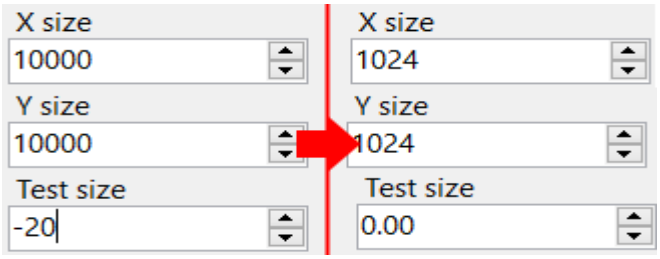


Рисунок 3.22 – Результат тестування

При виборі кількості шарів для власного налаштування структури моделі користувач може передумати та змінити їх кількість вже після генерації відповідних елементів, тому потрібно коректно видаляти зайві шари з форми при повторній генерації(таблиця 3.9)(рисунок 3.23).

Таблиця 3.9 – Тест-кейс 0.2.3

Тест-Кейс ID: 0.2.3	**Пріоритет:** 3	**Створено:** 23.04.2023
Назва: Динамічна зміна шарів на формі **Вхідні дані:** Нічого		
Кроки		**Очікуваний результат**
1. Завантажити додаток та обрати форму ModelTrain 2. Натиснути на Checkbox для зміни типу структури моделі із «по замовчування» на свою 3. В полі «кількість шарів» ввести 10 та натиснути кнопку генерації 4. В полі «кількість шарів» ввести 8 та натиснути кнопку генерації		Видалення непотрібних шарів з форми
Результат виконання тест-кейсу: пройдено успішно		

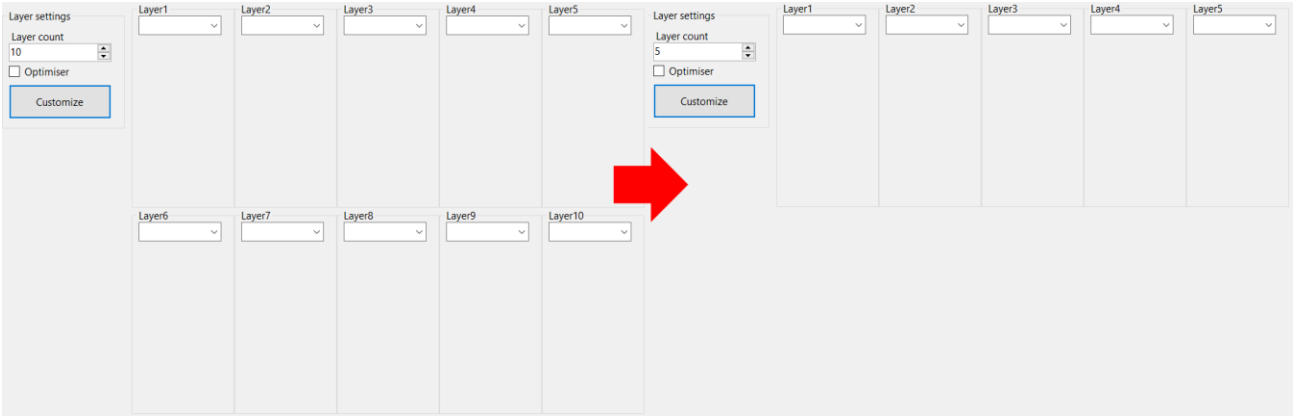


Рисунок 3.23 – Результат тестування

Таким чином було проведено тестування функціональних модулів системи ідентифікації особистості та доведено працездатність, коректне виконання поставленого завдання та можливість реагування додатку на нестандартні дії зі сторони користувача.

3.4 Дослідження ефективності застосування способу нейромережевого виявлення фейкових зображень документів

Для дослідження ефективності розробленого способу нейромережевого виявлення зображень документів було навчено та протестовано нейромережу із обраною структурою та зміною основних її параметрів перед навчанням (таблиця 3.10).

Таблиця 3.10 – Параметри навчання нейромережі

		Параметри навчання мережі		
		Epochs	Batch size	Train-Test Ratio
Ідентифікатор	CNN1	10	100	0,2
	CNN2	20	125	0,2
	CNN3	30	130	0,2
	CNN4	10	100	0,35
	CNN5	20	125	0,35
	CNN6	30	150	0,35

Тестування моделі CNN1 з наступними вхідними параметрами:

– Кількість епох – 10

– Розмір групи – 100

– Розділ тестових та валідаційних даних 80:20

Результат тренування – рисунок 3.14.

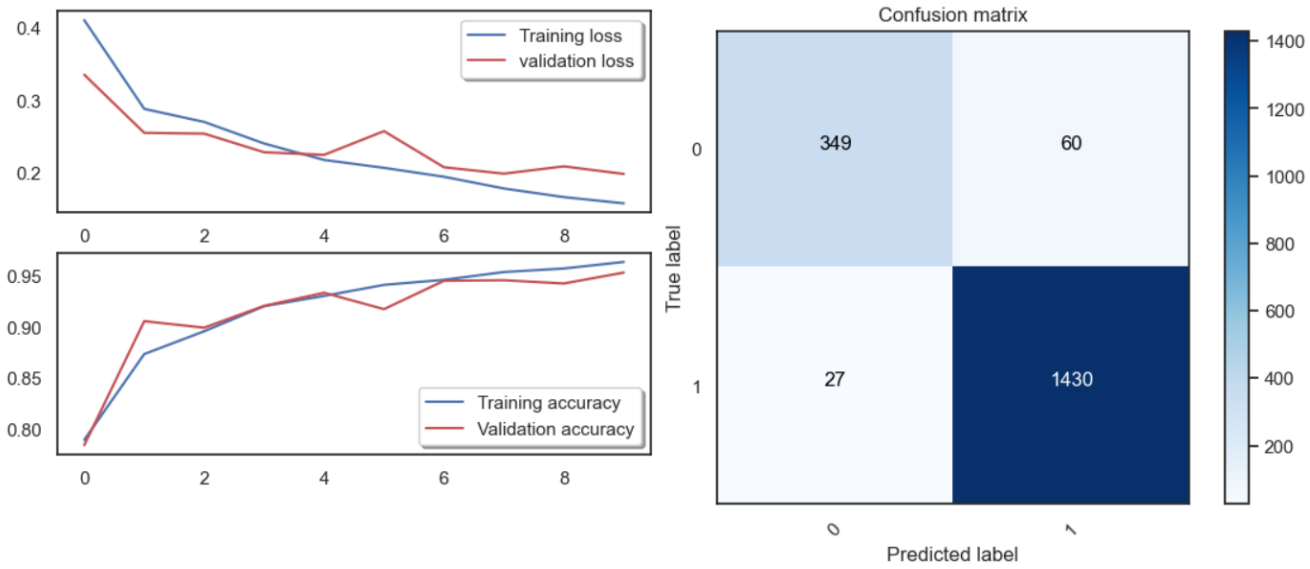


Рисунок 3.24 – Графіки та матриця сплутаності отриманої мережі

В результаті тренування кінцева точність мережі при тестуванні становить 0.9638 (~96%), при валідації 0.9534 (~95%).

Помічено незначне відхилення параметрів на одній з епох, відхилення в допустимих межах.

Матриця ілюструє незначну кількість FN – FalseNegative’s (помилковий негатив, тобто частину фейкових зображень не було ідентифіковано) та більш значну частину FP – FalsePositives (помилковий позитив, тобто частина реальних зображень була ідентифікована як фейкова).

Тестування моделі CNN2 з наступними вхідними параметрами:

– Кількість епох – 20

– Розмір групи – 125

– Розділ тестових та валідаційних даних 80:20

Результат тренування – рисунок 3.25.

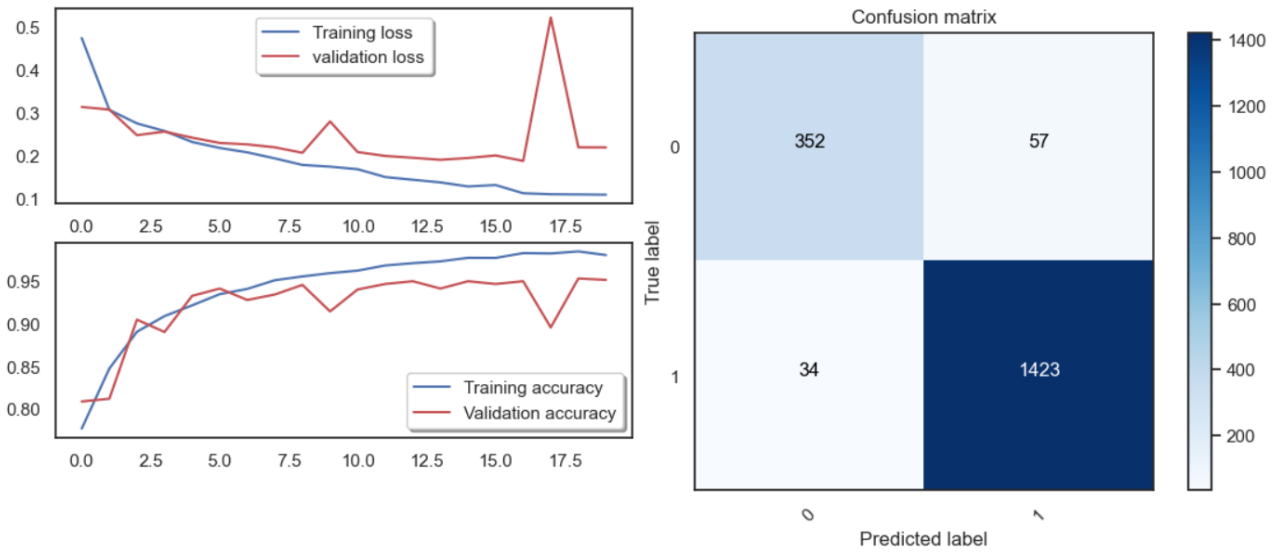


Рисунок 3.25 – Графіки та матриця сплутаності отриманої мережі

Кінцева точність:

– При тестуванні – 0.9803 (~98%);

– При валідації – 0.9512 (~95%).

На графіку видно значне відхилення для двох епох та незначне загальне відхилення результатів тестування та валідації.

Кількість FN та FP матриці зросла в допустимих маштабах.

Збільшення кількості епох та розмірів групи покращило точність тестування з 96% до 98% в порівнянні з попередньою версією мережі, але при цьому точність валідації залишилась незмінною.

Тестування моделі CNN3 з наступними вхідними параметрами:

– Кількість епох – 30

– Розмір групи – 150

– Розділ тестових та валідаційних даних 80:20

Результат тренування – рисунок 3.26.

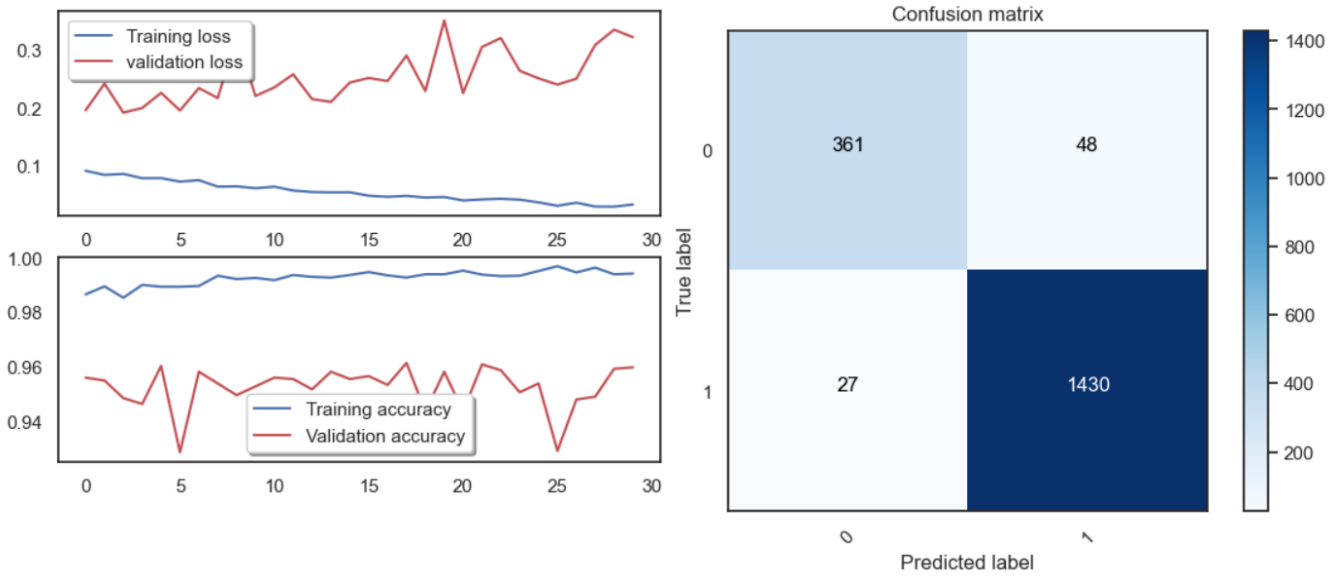


Рисунок 3.26 – Графіки та матриця сплутаності отриманої мережі

Кінцева точність:

– При тестуванні – 0.9941 (~99%);

– При валідації – 0.9598 (~96%).

Графік демонструє значне відхилення точності тестування та валідації по всім епохам.

Збільшення кількості епох та розмірів групи значно покращило точність тестування з 96%/98% до 99% в порівнянні з попередніми версіями мережі, але при цьому точність валідації збільшилась відносно мало. При цьому значне відхилення результатів тестування та валідації демонструє часткове перенавчання, тому на цьому збільшення параметрів епох та груп було завершено.

Тестування моделі CNN4 з наступними вхідними параметрами:

– Кількість епох – 10

– Розмір групи – 100

– Розділ тестових та валідаційних даних 65:35

Результат тренування – рисунок 3.27.

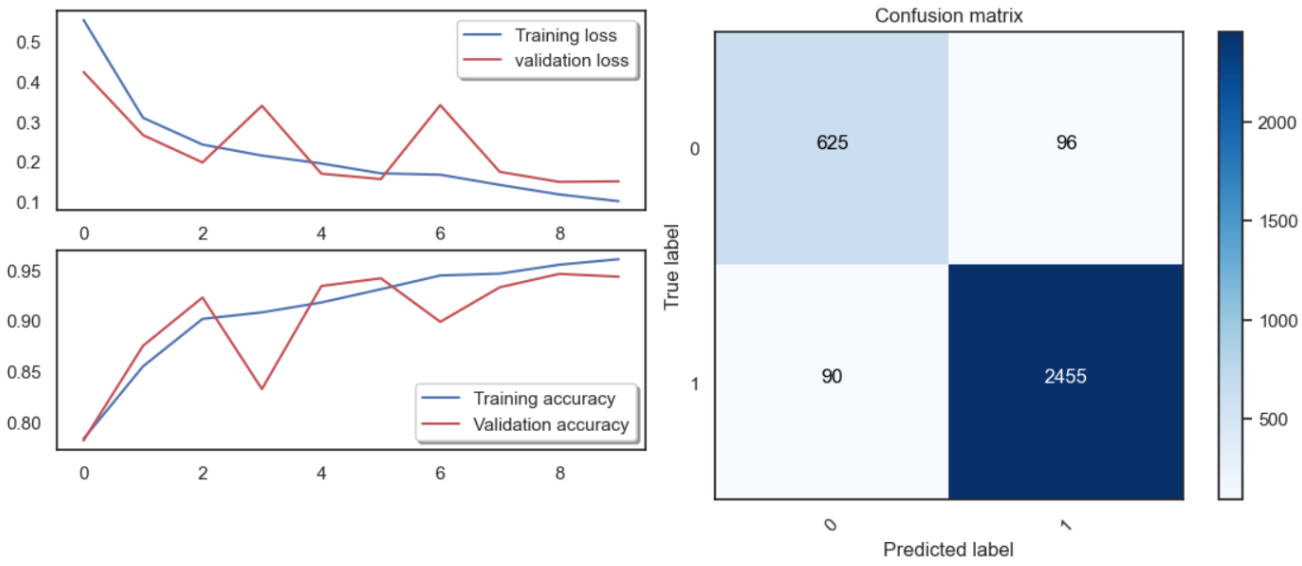


Рисунок 3.27 – Графіки та матриця сплутаності отриманої мережі

Кінцева точність:

– При тестуванні – 0.9603 (~96%);

– При валідації – 0.9430 (~94%).

Підхід по збільшенню кількості епох та розмірів групи продемонстрував свої недоліки, тому далі було змінено на відсоток розділу зображень для тестування та валідації.

Помічено незначне відхилення параметрів на двох епохах, загальне відхилення в допустимих межах.

Зміна відносного розділу валідаційних та тестових частин зображень мала незначний вплив на навчання мережі даної ітерації.

Тестування моделі CNN5 з наступними вхідними параметрами:

– Кількість епох – 20

– Розмір групи – 125

– Розділ тестових та валідаційних даних 65:35

Результат тренування – рисунок 3.28.

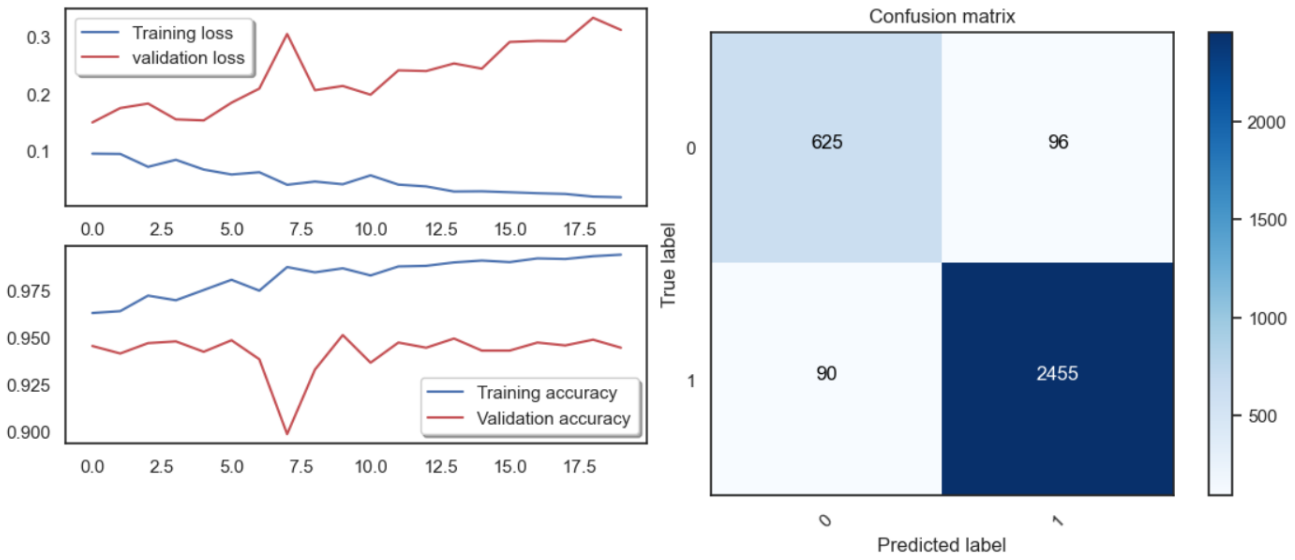


Рисунок 3.28 – Графіки та матриця сплутаності отриманої мережі

Кінцева точність:

– При тестуванні – 0.9841 (~98%);

– При валідації – 0.9446 (~94%).

В результаті на графіку видно значне відхилення результатів тестування та валідації, значно більше чим аналогічної моделі без зміни розділу.

Тестування моделі CNN6 з наступними вхідними параметрами:

– Кількість епох – 20

– Розмір групи – 150

– Розділ тестових та валідаційних даних 65:35

Результат тренування – рисунок 3.29.

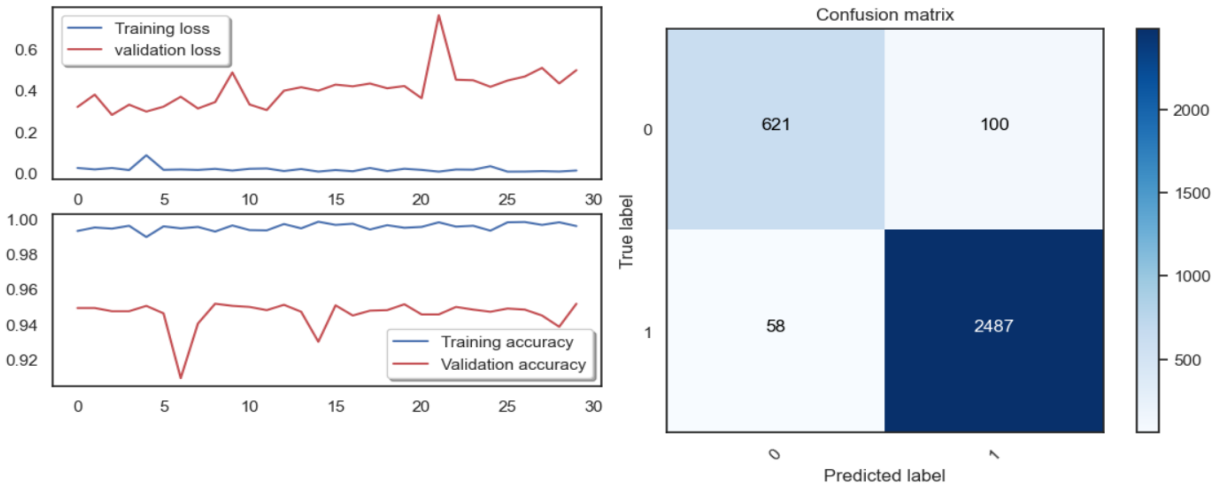


Рисунок 3.29 – Графіки та матриця сплутаності отриманої мережі

Кінцева точність:

– При тестуванні – 0.9960 (~99%);

– При валідації – 0.9516 (~95%).

Графік демонструє, аналогічно CNN3, значне відхилення точності валідації та тестування.

При порівнянні з CNN3, ця ітерація демонструє незначне зменшення кінцевої валідаційної точності з 96% до 95%.

В результаті отримуємо таблицю (таблиця 3.11) та графік (рисунок 3.30) результатів тестування

Таблиця 3.11 – Отримані кінцеві результати

		Результат		
		Accuracy	Val_Accuracy	Diff
Ідентифікатор	CNN1	0.9638	0.9534	0.0104
	CNN2	0.9803	0.9515	0.0288
	CNN3	0.9941	0.9598	0.0343
	CNN4	0.9603	0.9430	0.0173
	CNN5	0.9841	0.9446	0.0395
	CNN6	0.9960	0.9516	0.0444

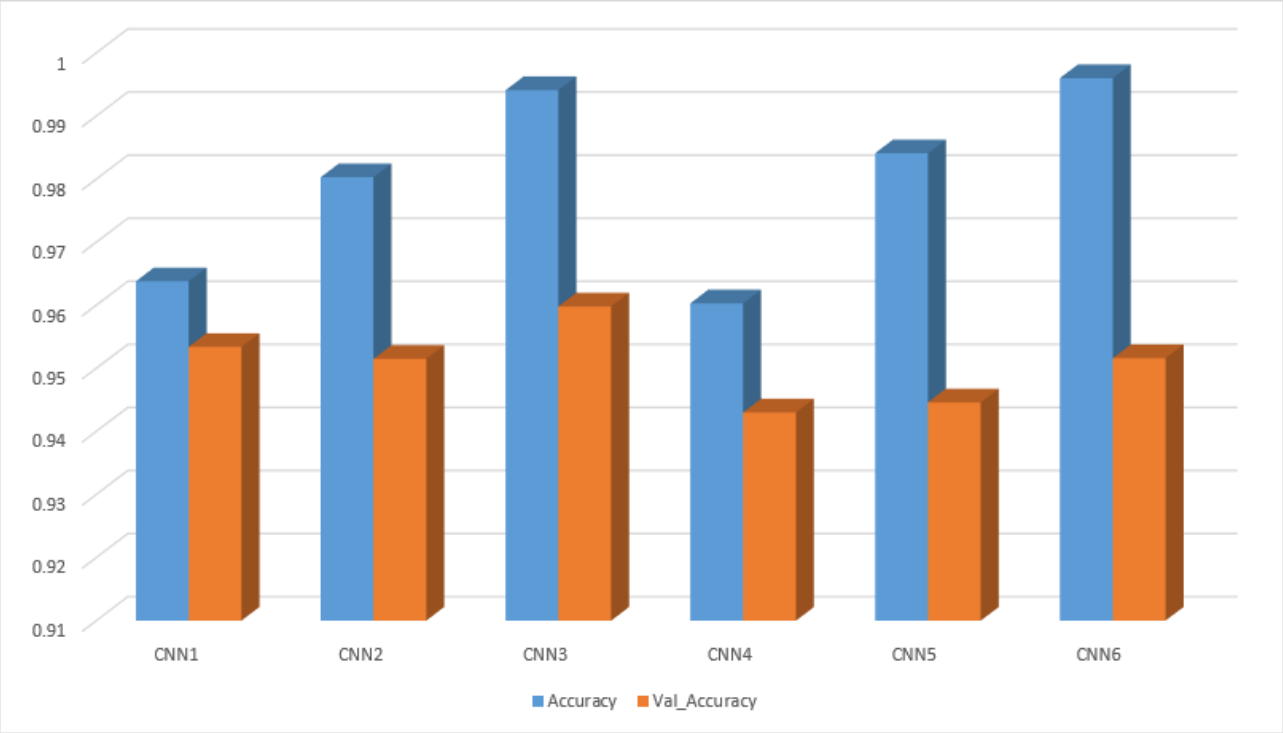


Рисунок 3.30 – Графік результатів тестування

Збільшення кількості епох та розміру групи дало відносно позитивний результат, збільшивши кінцеву точність, але також і збільшила різницю між тестовим та валідаційними параметрами. В той же час зміна відносного розділу зображень дала більше негативний результат, значно збільшивши різницю при незначній зміні точності.

Для подальшого покращення результатів мережі потрібно змінювати саму архітектуру мережі.

Найкращий результат отримала CNN3 з точністю в ~96%, але при цьому отримана різниця більша в 3 рази при порівнянні з CNN1, коли результат покращився лише на 0.005.

Отже, було розглянуто ефективність моделей CNN зі зміною параметрів розміру, кількості епох та частки розділу зображень.

Висновки

В результаті виконання кваліфікаційної роботи бакалавра був розроблений та програмно реалізований спосіб ідентифікації фейкових зображень документів для системи ідентифікації особистості. Для розробки модулів відповідної системи були використані мови програмування C# та Python, а також бібліотеки для них.

Для досягнення мети були виконані наступні етапи: аналіз предметної області та існуючих прикладних рішень, проєктування моделі CNN з аналізом відповідних програмного та алгоритмічного забезпечення, програмна реалізація розробленого методу та дослідження дієспроможності програмного додатку, дослідження ефективності застосування способу нейромережевого виявлення фейкових зображень документів.

Результатом виконання кваліфікаційної роботи бакалавра є створений спосіб нейромережевого виявлення фейкових зображень документів та інформаційна система ідентифікації особистості, яка використовує розроблений спосіб й дозволяє за відсканованим чи сфотографованим зображеннями документу що посвідчує особу автоматизовано визначати рівень його достовірності за допомогою згорткової нейронної мережі.

Реалізований додаток виконує наступні функції:

– можливість тестування завантаженого зображення на вибраній навченій моделі з можливостю зміни вхідного параметру, візуалізація результату;

– тренування моделі на вибраному датасеті з можливістю зміни та налаштування параметрів моделі, збереження в вигляді файлу чи папки;

– можливість налаштування функціональних модулів.

Перелік посилань

1. Сайт «Legalclinics». URL: https://legalclinics.in.ua/consult/consultation-26-06-2020-5/

2. Сайт «Core.ac», стаття «Поняття документів, їх види, способи розпізнання та захисту». URL: https://core.ac.uk/download/pdf/12241561.pdf

3. Сайт «Studfile», робота «Виявлення підробних документів». URL: https://studfile.net/preview/5536152/

4. Сайт «ibm» URL: https://www.ibm.com/topics/neural-networks

5. Сайт «Evergreens», стаття «Згорткова нейронна мережа – просте пояснення CNN та її застосування» URL: https://evergreens.com.ua/ua/articles/cnn.html

6. Сайт «Steelwiki», стаття «Why Use Python for AI and Machine Learning». URL: https://steelkiwi.com/blog/python-for-ai-and-machine-learning/

7. Сайт «Neurolab». URL: https://pypi.org/project/neurolab/

8. Сайт «Tensorflow». URL: https://www.tensorflow.org/

9. Сайт «FFnet». URL: https://pypi.org/project/ffnet/

10. Сайт «Imt-Atlantique», стаття «Neural Network-based System for Automatic Passport Stamp Classification» URL: https://imt-atlantique.hal.science/hal-03101199/document

11. Сайт «Researchgate», Стаття «Deep Learning for Automated Forgery Detection in Hyperspectral Document Images». URL: https://www.researchgate.net/publication/327069500_Deep_Learning_for_Automated_Forgery_Detection_in_Hyperspectral_Document_Images

12. Сайт «Kaggle», датасет «Cassia2». URL: https://www.kaggle.com/datasets/sophatvathana/casia-dataset

13. Сайт «PapersWithCode», датасет «MIDV-500». URL: https://paperswithcode.com/dataset/midv-500

14. Visual studio. URL: https://visualstudio.microsoft.com/

15. Visual studio Code: https://code.visualstudio.com/

16. Сайт «Learn.microsoft» . URL: https://learn.microsoft.com/en-us/dotnet/csharp/

17. Сайт «Python» . URL: https://www.python.org/

18. Сайт «Keras». URL: https://keras.io/

19. Сайт «Numpy». URL: https://numpy.org/

20. Сайт «Pandas». URL: https://pandas.pydata.org/

21. Сайт «Pillow». URL: https://pypi.org/project/Pillow/

22. Сайт «MathPlotLib». URL:https://matplotlib.org/

23. Сайт «Scikit-learn». URL: https://scikit-learn.org/stable/

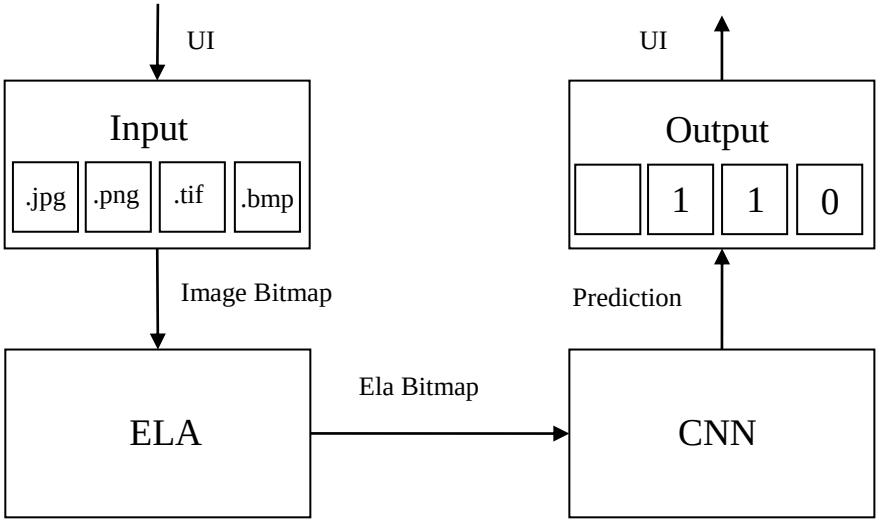
24. Ресурс. URL: https://shtamp.com.ua/assets/cache_image/assets/image/official/250_400x400_051.jpg

25. Ресурс. URL: https://dmsu.gov.ua/faq/biometrichni-dokumenti-v-ukrajni/yaka-stupin-zaxistu-u-biometrichnix-dokumentiv.html

Додатки

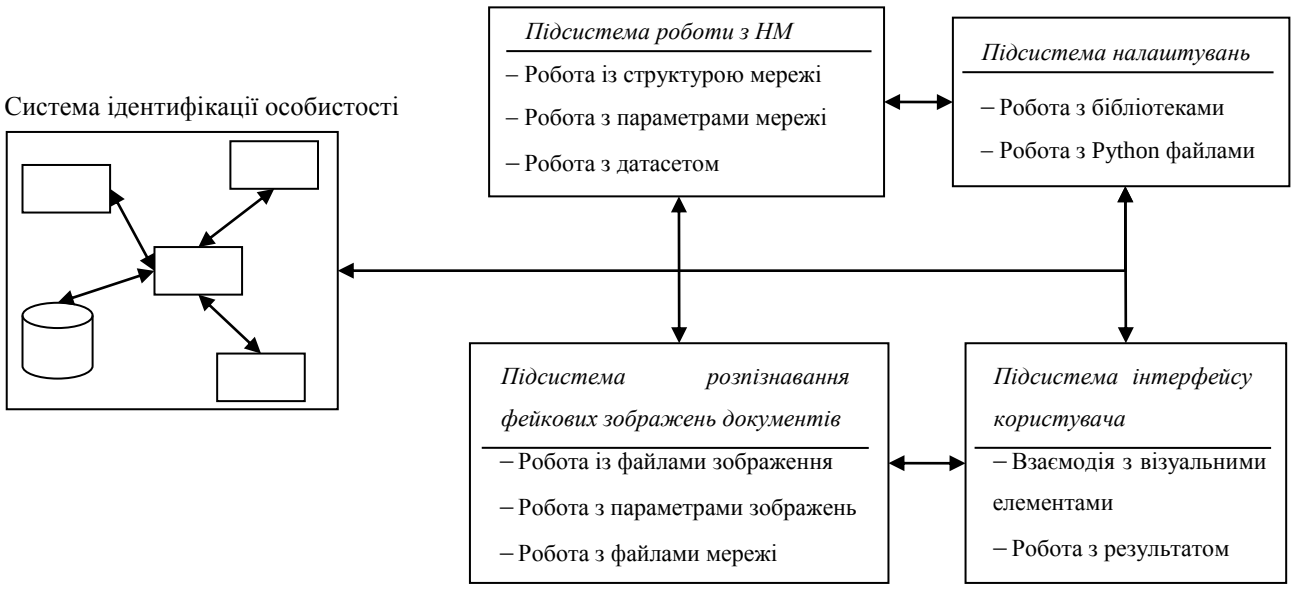
Додаток А

Узагальнена схема способу нейромережевого виявлення фейкових зображень документів



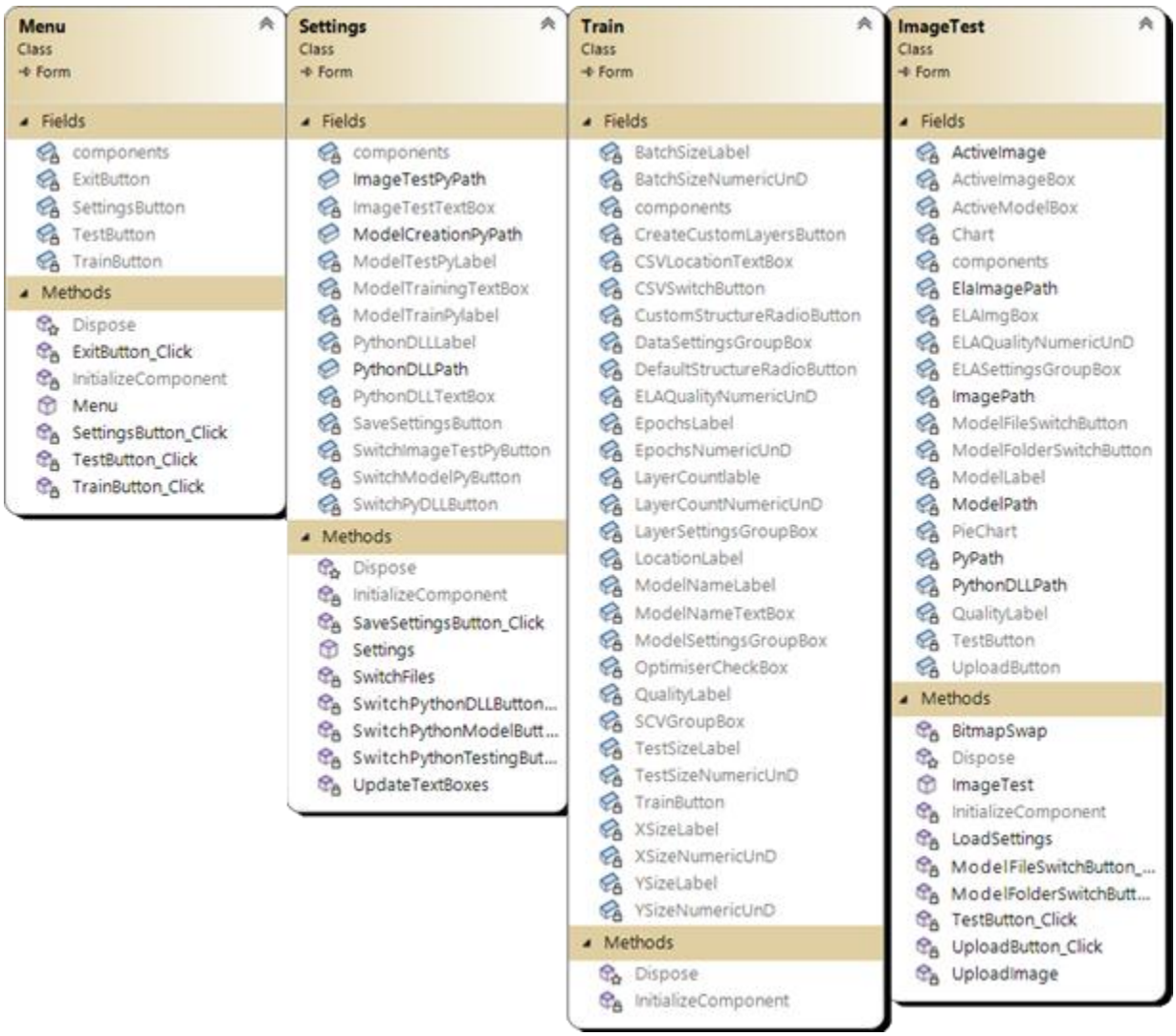
Додаток Б

Підсистеми інформаційної системи ідентифікації особистості



Додаток В

Діаграма класів інформаційної системи ідентифікації особистості



Додаток Г

Програмні коди функціональної частини реалізованої на мові програмування Python

Лістинг ModelTrain.py

```
import pandas as pd

import numpy as np
np.random.seed(2)

from sklearn.model_selection import
train_test_split
from sklearn.metrics import
confusion_matrix

from keras.utils.np_utils import
to_categorical
from keras.models import Sequential
from keras.layers import Dense, Dropout,
Flatten, Conv2D, MaxPool2D
from keras.optimizers import RMSprop
from keras.callbacks import EarlyStopping

from PIL import Image, ImageChops,
ImageEnhance

from pylab import *

import os

#Converting Image to ela and returnign it
def ConvertToEla(ImagePath,Quality):
    FileName = ImagePath
    ReFileName =
'../../../TrainChache/'+os.path.basename(Fi
leName.split('.')[0])+ '.resaved.jpg'

    Img =
Image.open(FileName).convert('RGB')
    Img.save(ReFileName, 'JPEG',
quality=Quality)
    ReImage = Image.open(ReFileName)

    ElaImg = ImageChops.difference(Img,
ReImage)

    Extrema = ElaImg.getextrema()
    MaxDiff = max([ex[1] for ex in
Extrema])
    if MaxDiff == 0:
        MaxDiff = 1
    Scale = 255.0 / MaxDiff

    ElaImg =
ImageEnhance.Brightness(ElaImg).enhance(Sca
le)
    #os.remove(ReFileName)
    return ElaImg

#Read CVS file with image path and class
value
def ReadDataset(CSVPath):
    Dataset = pd.read_csv(CSVPath)

    return Dataset

#Converting CVS data in to model values
def
ConvertData(Dataset,Quality,XSize,YSize,Tes
tSize):
    X = []
    Y = []

    for index, row in Dataset.iterrows():
        X.append(array(ConvertToEla(row[0],
Quality).resize((XSize, YSize))).flatten()
/ 255.0)
        Y.append(row[1])

    X = np.array(X)
    Y = to_categorical(Y, 2)
    X = X.reshape(-1, XSize, YSize, 3)

    XTrain, XVal, YTrain, YVal =
train_test_split(X, Y, test_size =
TestSize, random_state=5)
    Values = [XTrain,YTrain,XVal,YVal]

    return Values

#Building default model structure
def InitiateDefaultModel():
    Model = Sequential()

    Model.add(Conv2D(filters = 32,
kernel_size = (5,5),padding = 'valid',
activation ='relu', input_shape =
(128,128,3)))
    Model.add(Conv2D(filters = 32,
kernel_size = (5,5),padding = 'valid',
activation ='relu'))
    Model.add(MaxPool2D(pool_size=(2,2)))
    Model.add(Dropout(0.25))
    Model.add(Flatten())
    Model.add(Dense(256, activation =
'relu'))
    Model.add(Dropout(0.5))
    Model.add(Dense(2, activation =
'softmax'))
```

```
    Optimizer =
RMSprop(learning_rate=0.0005, rho=0.9,
epsilon=1e-08, decay=0.0)

    Model.compile(optimizer = Optimizer ,
loss = 'categorical_crossentropy',
metrics=['accuracy'])

    return Model

#Early stopping
def DefineStopping():
    Early_Stopping =
EarlyStopping(monitor='val_accuracy',min_de
lta=0,patience=2,verbose=2, mode='auto')

    return Early_Stopping

#Training and saving process
def
ModelTrain(Model,Epochs,Batch_Size,X_train,
Y_train,X_val,Y_val,Early_Stopping,ModelNam
e):
    Model.fit(X_train, Y_train, batch_size
= Batch_Size, epochs = Epochs,
validation_data = (X_val, Y_val), verbose =
0, callbacks=[Early_Stopping])
    Model.save('../../../SavedModels/' +
ModelName)

def
Run(CSVPath,Quality,XSize,YSize,TestSize,Mo
delType,Epochs,BatchSize,ModelName,Optimise
r):
    Dataset = ReadDataset(CSVPath)
    Values =
ConvertData(Dataset,Quality,XSize,YSize,Tes
tSize)
    if ModelType:
        Model = InitiateDefaultModel()
    else: Model =
CompileCustomModel(Optimiser)
    Eearly_Stopping = DefineStopping()
    ModelTrain(Model, Epochs, BatchSize,
Values[0], Values[1], Values[2], Values[3],
Eearly_Stopping, ModelName)

#Creating custom model structure
CustomModel = None

def BuildCustomModel():
    global CustomModel
    CustomModel = Sequential()

def
AddConvInputLayer(Filters,Padding,Activatio
n,XSize,YSize):
    CustomModel.add(Conv2D(filters =
Filters, kernel_size = (5,5),padding =
Padding, activation = Activation,
input_shape = (XSize,YSize,3)))

def
AddConvLayer(Filters,Padding,Activation):
    CustomModel.add(Conv2D(filters =
Filters, kernel_size = (5,5),padding =
Padding, activation = Activation))

def AddMaxPoolLayer():

CustomModel.add(MaxPool2D(pool_size=(2,2)))

def AddDropoutLayer(DropoutVal):
    CustomModel.add(Dropout(DropoutVal))

def AddFlatten():
    CustomModel.add(Flatten())

def AddDenseLayer(Density,Activation):
    CustomModel.add(Dense(Density,
activation = Activation))

def CompileCustomModel(Optimiser):
    if Optimiser:
        CustomOptimiser =
RMSprop(learning_rate=0.0005, rho=0.9,
epsilon=1e-08, decay=0.0)
        CustomModel.compile(optimizer =
CustomOptimiser , loss =
'categorical_crossentropy',
metrics=['accuracy'])
    else: CustomModel.compile(loss =
'categorical_crossentropy',
metrics=['accuracy'])
    return CustomModel
```

Лістинг ModelTest.py

```
from keras.models import load_model
from PIL import Image, ImageChops,
ImageEnhance
import numpy as np
from skimage import transform
import os

#Load ELA file in numpy format
def Load(FileName):
   NpImg = Image.open(FileName)
   NpImg =
np.array(NpImg).astype('float32')/255

   NpImg = transform.resize(NpImg, (128,
128, 3))
   NpImg = np.expand_dims(NpImg, axis=0)

   return NpImg

#Single image Prediction using model
def ImageTest(ImagePath,ModelPath):
   Img = Load(ImagePath)
   Model = load_model(ModelPath)

   Prediction =
Model.predict(Img,verbose=0)
```

```
   Prediction = Prediction[0].flatten()
   Prediction = [Array.tolist() for Array
in Prediction]

   return Prediction

#Converting Image to ela and saving it
def ConvertToEla(ImagePath,Quality):
    FileName = ImagePath
    ReFileName =
'../../../Images/'+os.path.basename(FileNam
e.split('.')[0])+ '.resaved.jpg'
    ElaFileName =
'../../../Images/'+os.path.basename(FileNam
e.split('.')[0])+'_ELA.jpg'

    Img =
Image.open(FileName).convert('RGB')
    Img.save(ReFileName, 'JPEG',
quality=Quality)
    ReImage = Image.open(ReFileName)

    ElaImg = ImageChops.difference(Img,
ReImage)

    Extrema = ElaImg.getextrema()
    MaxDiff = max([ex[1] for ex in
Extrema])
    if MaxDiff == 0:
        MaxDiff = 1
    Scale = 255.0 / MaxDiff

    ElaImg =
ImageEnhance.Brightness(ElaImg).enhance(Sca
le)
    ElaImg.save(ElaFileName, 'JPEG',
quality=Quality)
    os.remove(ReFileName)

    return ElaFileName
```

Лістинг CSVCreation.py

```
import os
path_orig = 'CASIA2/Au/'
path_modif = 'CASIA2/Tp/'

folder_orig = os.listdir()
folder_modif = os.listdir()

strings = []

for file in os.listdir(path_orig):
  try:
    if file.endswith('jpg'):
      if int(os.stat(path_orig +
file).st_size) > 10000:
        line =  path_orig + file  + ',1\n'
        strings.append(line)
  except:
    print(path_orig+file)

for file in os.listdir(path_modif):
    try:
      if file.endswith('jpg'):
        if int(os.stat(path_modif +
file).st_size) > 10000:
            line =  path_modif + file +
',0\n'
            strings.append(line)
    except:
      print(path_modif+file)

for line in strings:
      with open('dataset.csv', 'a') as f:
        f.write(line)
```

Додаток Д

Програмні коди візуальної частини реалізованої на мові програмування C#

Menu.cs

```
public partial class Menu : Form
    {
        public Menu()
        {
            InitializeComponent();
        }

        private void
TrainButton_Click(object sender, EventArgs
e)
        {
            Train TrainForm = new Train();
            TrainForm.Show();
        }

        private void
TestButton_Click(object sender, EventArgs
e)
        {
            ImageTest ImageTestForm = new
ImageTest();
            ImageTestForm.Show();
        }

        private void
SettingsButton_Click(object sender,
EventArgs e)
        {
            Settings SettingstForm = new
Settings();
            SettingstForm.Show();
        }

        private void
ExitButton_Click(object sender, EventArgs
e)
        {
            Application.Exit();
        }
    }
```

Settings.cs

```
public partial class Settings : Form
    {

        //Default file links
        public string PythonDLLPath =
@"..\..\..\Python\python310.dll",
            ImageTestPyPath =
@"..\..\..\Python\ModelTest.py",
            ModelCreationPyPath =
@"..\..\..\Python\ModelTrain.py";

        public Settings()
        {
            InitializeComponent();
            UpdateTextBoxes();
        }

        private void UpdateTextBoxes()
        {
            PythonDLLTextBox.Text =
PythonDLLPath;
            PythonDLLTextBox.SelectionStart
= PythonDLLPath.Length - 1;
            ModelTrainingTextBox.Text =
ModelCreationPyPath;

ModelTrainingTextBox.SelectionStart =
ModelCreationPyPath.Length - 1;
            ImageTestTextBox.Text =
ImageTestPyPath;
            ImageTestTextBox.SelectionStart
= ImageTestPyPath.Length - 1;
        }

        private void SwitchFiles(int Line)
        {
            OpenFileDialog DialogWindow =
new OpenFileDialog();

            switch (Line)
            {
                case 1:
                    DialogWindow.Title =
"Link Python DLL file";
                    DialogWindow.Filter =
"DLL files (*.dll)|*.dll";
                    break;
                case 2:
                case 3:
                    DialogWindow.Title =
"Link Py file";
                    DialogWindow.Filter =
"Python files (*.py)|*.py";
                    break;
            }
            if (DialogWindow.ShowDialog()
== DialogResult.OK)
            {
                switch (Line)
                {
                    case 1:
                        PythonDLLPath =
DialogWindow.FileName;
                        UpdateTextBoxes();
                        break;
                    case 2:
```

```
                        ModelCreationPyPath
= DialogWindow.FileName;
                        UpdateTextBoxes();
                        break;
                    case 3:
                        ImageTestPyPath =
DialogWindow.FileName;
                        UpdateTextBoxes();
                        break;
                }
            }
            else
            {
                MessageBox.Show("Unable to
link file, try again.", "File linking
failed", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
            }
        }

        private void
SwitchPythonDLLButton_Click(object sender,
EventArgs e)
        {
            SwitchFiles(1);
        }

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

ModelTrain.cs

```
public partial class Train : Form
    {
        //Used values
        string PythonDLLPath, PyPath,
            CSVPath;
        public Train()
        {
            InitializeComponent();
            FormBuild();
            LoadSettings();
        }

        //Default form element values
        int FormWidth = 410, FormHeight =
470;

        //Procedual form building
        public void FormBuild()
        {
            this.Size = new Size(FormWidth,
FormHeight);
            LayerCountNumericUnD.Maximum =
MaxLayers;
        }
        private void LoadSettings()
        {
            if
(File.Exists(@"..\..\..\Settings.txt"))
            {
                string[] Lines =
File.ReadAllLines(@"..\..\..\Settings.txt")
;
                PythonDLLPath = @Lines[1];
                PyPath = @Lines[3];
            }
            else
            {
                MessageBox.Show("Unable to
load settings file, reverting to
defaults.", "Settings load failed",
MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                PythonDLLPath =
@"..\..\..\Python\python310.dll";
                PyPath =
@"..\..\..\Python\ModelTrain.py";
            }
        }

        private void
CSVSwitchButton_Click(object sender,
EventArgs e)
        {
            OpenFileDialog DialogWindow =
new OpenFileDialog();

            DialogWindow.Title = "Link CSV
dataset file";
            DialogWindow.Filter = "CSV
files (*.CSV)|*.CSV";
        {
            SwitchFiles(2);
        }

        private void
SwitchPythonTestingButton_Click(object
sender, EventArgs e)
        {
            SwitchFiles(3);
        }

        private void
SaveSettingsButton_Click(object sender,
EventArgs e)
        {
            using (StreamWriter writer =
new StreamWriter(@"..\..\..\Settings.txt",
false))
            {

writer.WriteLineAsync("Python dll:\n" +
PythonDLLPath +
                    "\nModel training
py:\n" + ModelCreationPyPath +
                    "\nImage testing py:\n"
+ ImageTestPyPath);
            }
        }
```

```
            if (DialogWindow.ShowDialog()
== DialogResult.OK)
            {
                CSVPath =
DialogWindow.FileName;
                CSVLocationTextBox.Text =
CSVPath;
            }
            else
            {
                MessageBox.Show("Failed to
select CSV folder", "File selection
failed", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
            }
        }

        private void
TrainButton_Click(object sender, EventArgs
e)
        {
            try
            {
                TrainButton.Enabled =
false;

                //PythonNet engine
initialization
                Runtime.PythonDLL =
PythonDLLPath;
                PythonEngine.Initialize();

                //Getting values
                int ELAQuality =
(int)ELAQualityNumericUnD.Value,
                    XSize =
(int)XSizeNumericUnD.Value,
                    YSize =
(int)YSizeNumericUnD.Value,
                    Epochs =
(int)EpochsNumericUnD.Value,
                    BatchSize =
(int)BatchSizeNumericUnD.Value;
                double TestSize =
(double)TestSizeNumericUnD.Value;
                string ModelName =
ModelNameTextBox.Text;
                bool ModelType = true,
Optimiser = false;
                if
(OptimiserCheckBox.Checked) Optimiser =
true;

                using (Py.GIL())
                {
                    //Building
                    dynamic os =
Py.Import("os");
                    dynamic sys =
Py.Import("sys");
sys.path.append(os.path.dirname(os.path.exp
anduser(PyPath)));
                    dynamic ModelTrain =
Py.Import(Path.GetFileNameWithoutExtension(
PyPath));

                    //Custom model
                    if
(CustomStructureRadioButton.Checked)
                    {
                        ModelType =
false;//False = not default


ModelTrain.BuildCustomModel();//Initialise
py.global model value
                        for (int i = 0; i <
LayerCountNumericUnD.Value; i++)
                        {
                            int Filter,
LayerXSize, LayerYSize,Density;
                            string Padding,
Activation;
                            double Dropout;

                            //Layer builder
                            switch
(LayerTypesBox[i].SelectedItem.ToString())
                            {
                                case
"ConvInput":
                                    Filter
= Convert.ToInt32(TextBoxes[i, 0].Text);

LayerXSize = Convert.ToInt32(TextBoxes[i,
3].Text);

LayerYSize = Convert.ToInt32(TextBoxes[i,
4].Text);
                                    Padding
= TextBoxes[i, 1].Text;

Activation = TextBoxes[i, 2].Text;

ModelTrain.AddConvInputLayer(Filter,Padding
,Activation,LayerXSize,LayerYSize);
                                    break;
                                case
"Conv":
                                    Filter
= Convert.ToInt32(TextBoxes[i, 0].Text);
                                    Padding
= TextBoxes[i, 1].Text;

Activation = TextBoxes[i, 2].Text;

ModelTrain.AddConvLayer(Filter, Padding,
Activation);
                                    break;
                                case
"MaxPool":
```

```
ModelTrain.AddMaxPoolLayer();
                                    break;
                                case
"Dropout":
                                    Dropout
= Convert.ToDouble(TextBoxes[i, 0].Text);
ModelTrain.AddDropoutLayer(Dropout);
                                    break;
                                case
"Flatten":
ModelTrain.AddFlatten();
                                    break;
                                case
"Dense":
                                    Density
= Convert.ToInt32(TextBoxes[i, 0].Text);
                                    Padding
= TextBoxes[i, 1].Text;
ModelTrain.AddDenseLayer(Density, Padding);
                                    break;
                            }
                        }
ModelTrain.CompileCustomModel(Optimiser);
                    }

                    //Training
                    ModelTrain.Run(CSVPath,
ELAQuality, XSize, YSize, TestSize,
ModelType, Epochs, BatchSize, ModelName,
Optimiser);
                }
            }
            catch (Exception ex)
            {
MessageBox.Show(ex.ToString(), "Execution
error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
            finally
            {
                PythonEngine.Shutdown();
                TrainButton.Enabled = true;
            }
        }

        //Resizing form for custom layer
menu
        int LayerCustomisationOffset = 190;
        private void
CustomStructureRadioButton_CheckedChanged(o
bject sender, EventArgs e)
        {
            if
(CustomStructureRadioButton.Checked)
            {
                this.Size = new
Size(FormWidth + LayerCustomisationOffset,
FormHeight);
            }
            else
            {
                this.Size = new
Size(FormWidth, FormHeight);
            }
        }

        //Dynamically adding layer settings
to form
        static int MaxLayers = 15;
        static string[] LayerTypes = {
"ConvInput", "Conv", "MaxPool", "Dropout",
"Flatten", "Dense" };
        GroupBox[] LayerGroup = new
GroupBox[MaxLayers];
        ComboBox[] LayerTypesBox = new
ComboBox[MaxLayers];
        Label[,] Labels = new
Label[MaxLayers, 5];
        TextBox[,] TextBoxes = new
TextBox[MaxLayers, 5];
        int GroupBoxWidth = 150,
GroupBoxHeight = 300;
        private void
CreateCustomLayersButton_Click(object
sender, EventArgs e)
        {
            int LayerCount =
(int)LayerCountNumericUnD.Value,
                Layer = 0;

            //Cleanup
            for (int i = 0; i < MaxLayers;
i++)
            {
this.Controls.Remove(LayerGroup[i]);
            }

            for (int Row = 0; Row <
(int)(LayerCount / 5) + 1; Row++)
            {
                for (int Column = 0; Column
< 5; Column++)
                {
                    if (Layer ==
LayerCount)
                    {
                        break;
                    }
                    LayerGroup[Layer] = new
GroupBox();
                    LayerGroup[Layer].Text
= "Layer" + (Layer + 1);
                    LayerGroup[Layer].Size
= new Size(GroupBoxWidth, GroupBoxHeight);
LayerGroup[Layer].Location = new Point(580
```

```
+ Column * GroupBoxWidth, Row *
GroupBoxHeight);

this.Controls.Add(LayerGroup[Layer]);

                    LayerTypesBox[Layer] =
new ComboBox();

LayerTypesBox[Layer].Items.AddRange(LayerTy
pes);

LayerTypesBox[Layer].Location = new
Point(10, 20);

LayerTypesBox[Layer].SelectedIndexChanged
+= GroupBoxIndexChange;

LayerGroup[Layer].Controls.Add(LayerTypesBo
x[Layer]);

                    Layer++;
                }
            }
            this.Size = new Size(FormWidth
+ LayerCustomisationOffset + 5 *
GroupBoxWidth + 20, FormHeight + 2 *
GroupBoxHeight);

//MessageBox.Show(Layer.ToString());
            //Cleanup-bugged
            /*
            for (int i = Layer; i <
MaxLayers; i++)
            {


this.Controls.Remove(LayerGroup[i]);
            }*/

        }
        private void
GroupBoxIndexChange(object sender,
System.EventArgs e)
        {
            int Layer =
Array.IndexOf(LayerTypesBox,
(ComboBox)sender);

            //Cleanup
            for (int i = 0; i < 5; i++)
            {

LayerGroup[Layer].Controls.Remove(Labels[La
yer, i]);

LayerGroup[Layer].Controls.Remove(TextBoxes
[Layer, i]);
            }
            //Getting index of ComboBox
that is being changed
            string[] ConvValueStrings = {
"Filters", "Padding", "Activation",
"XSize", "YSize" },
                DenseValueStrings = {
"Density", "Activation" };
            switch
(LayerTypesBox[Layer].SelectedItem.ToString
()) // todo:replace constant strings with
array refs
            {
                case "ConvInput":
                    for (int i = 0; i < 5;
i++)
                    {
                        Labels[Layer, i] =
new Label();
                        Labels[Layer,
i].Text = ConvValueStrings[i] + "=";
                        Labels[Layer,
i].Location = new Point(10, 50 + 50 * i);

LayerGroup[Layer].Controls.Add(Labels[Layer
, i]);

                        TextBoxes[Layer, i]
= new TextBox();
                        TextBoxes[Layer,
i].Location = new Point(10, 70 + 50 * i);

LayerGroup[Layer].Controls.Add(TextBoxes[La
yer, i]);
                    }
                    break;
                case "Conv":
                    for (int i = 0; i < 3;
i++)
                    {
                        Labels[Layer, i] =
new Label();
                        Labels[Layer,
i].Text = ConvValueStrings[i];
                        Labels[Layer,
i].Location = new Point(10, 50 + 50 * i);

LayerGroup[Layer].Controls.Add(Labels[Layer
, i]);

                        TextBoxes[Layer, i]
= new TextBox();
                        TextBoxes[Layer,
i].Location = new Point(10, 70 + 50 * i);

LayerGroup[Layer].Controls.Add(TextBoxes[La
yer, i]);
                    }
                    break;
                case "MaxPool": break;
                case "Dropout":
                    {
                    Labels[Layer, 0] = new
Label();
                    Labels[Layer, 0].Text =
"Value";
                    Labels[Layer,
0].Location = new Point(10, 50);
```

```
LayerGroup[Layer].Controls.Add(Labels[Layer, 0]);

                    TextBoxes[Layer, 0] = new TextBox();
                    TextBoxes[Layer, 0].Location = new Point(10, 70);

LayerGroup[Layer].Controls.Add(TextBoxes[Layer, 0]);
                    }
                    break;
                case "Flatten": break;
                case "Dense":
                    for (int i = 0; i < 2; i++)
                    {
                        Labels[Layer, i] = new Label();
```

ModelTest.cs

```
public partial class ImageTest : Form
    {
        //Used values
        Image ActiveImage;
        string PythonDLLPath, PyPath,
            ModelPath = @"..\..\..\SavedModels\Model 10epoch-128-128-folder",//default model
            ImagePath, ElaImagePath;

        public ImageTest()
        {
            InitializeComponent();

            LoadSettings();

            Chart.Series[0].Clear();
            ActiveModelBox.Text = ModelPath;
            ActiveModelBox.SelectionStart = ModelPath.Length - 1;
        }

        //Loading settings from txt file
        private void LoadSettings()
        {
            if (File.Exists(@"..\..\..\Settings.txt"))
            {
                string[] Lines = File.ReadAllLines(@"..\..\..\Settings.txt");
                PythonDLLPath = @Lines[1];
                PyPath = @Lines[5];
            }
            else
            {
                MessageBox.Show("Unable to load settings file, reverting to defaults.", "Settings load failed",
                    Labels[Layer, i].Text = DenseValueStrings[i];
                    Labels[Layer, i].Location = new Point(10, 50 + 50 * i);

LayerGroup[Layer].Controls.Add(Labels[Layer, i]);

                    TextBoxes[Layer, i] = new TextBox();
                    TextBoxes[Layer, i].Location = new Point(10, 70 + 50 * i);

LayerGroup[Layer].Controls.Add(TextBoxes[Layer, i]);
                }
                break;
            }
        }
    }


MessageBoxButtons.OK, MessageBoxIcon.Warning);
                PythonDLLPath = @"..\..\..\Python\python310.dll";
                PyPath = @"..\..\..\Python\ModelTest.py";
            }
        }

        private Image UploadImage()
        {
            OpenFileDialog DialogWindow = new OpenFileDialog();

            DialogWindow.Title = "Image upload";
            DialogWindow.Filter = "Image Files(*.jpg; *.jpeg; *.png; *.bmp)|*.jpg; *.jpeg; *.png; *.bmp";
            if (DialogWindow.ShowDialog() == DialogResult.OK)
            {
                ImagePath = DialogWindow.FileName;
                Image Image = BitmapSwap(ImagePath);
                return Image;
            }
            else return null;
        }

        //Trick to unlock image file that is being used
        private Image BitmapSwap(string ImagePath)
        {
            Image Image;
            using (var TempBitmap = new Bitmap(ImagePath))
            {
```

```
                Image = new
Bitmap(TempBitmap);
            }
            return Image;
        }

        private void
UploadButton_Click(object sender, EventArgs
e)
        {
            ActiveImage = UploadImage();
            if (ActiveImage is null)
            {
                MessageBox.Show("Unable to
upload file, try again.", "Image upload
failed", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
            }
            else
            {
                ActiveImageBox.Image =
ActiveImage;
            }
        }

        private void
TestButton_Click(object sender, EventArgs
e)
        {
            try
            {
                TestButton.Enabled = false;
                Chart.Series[0].Clear();
                ELAImgBox.Image = null;

                //PythonNet engine
initialization
                Runtime.PythonDLL =
PythonDLLPath;
                PythonEngine.Initialize();

                using (Py.GIL())
                {
                    //Building
                    dynamic os =
Py.Import("os");
                    dynamic sys =
Py.Import("sys");

sys.path.append(os.path.dirname(os.path.exp
anduser(PyPath)));
                    dynamic ImageTest =
Py.Import(Path.GetFileNameWithoutExtension(
PyPath));

                    //Ela script
                    int ELAQuality =
Convert.ToInt16(ELAQualityNumericUnD.Value)
;
                    ElaImagePath =
ImageTest.ConvertToEla(ImagePath,
ELAQuality);
                    ELAImgBox.Image =
BitmapSwap(ElaImagePath);

                    //Model testing
                    double[] result =
ImageTest.ImageTest(ElaImagePath,
ModelPath);

                    //Chart filling

Chart.Series[0].Add(result[0] * 100,
"Fake");

Chart.Series[0].Add(result[1] * 100,
"Real");
                }
            }
            catch (Exception ex)
            {

MessageBox.Show(ex.ToString(), "Execution
error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
            finally
            {
                PythonEngine.Shutdown();
                TestButton.Enabled = true;
            }
        }

        //Model folder type selection
        private void
ModelFolderSwitchButton_Click(object
sender, EventArgs e)
        {
            FolderBrowserDialog
FolderBrowser = new FolderBrowserDialog();
            FolderBrowser.Description =
"Select folder containing model";
            if (FolderBrowser.ShowDialog()
== DialogResult.OK)
            {
                ModelPath =
FolderBrowser.SelectedPath;
                ActiveModelBox.Text =
ModelPath;

ActiveModelBox.SelectionStart =
ModelPath.Length - 1;
            }
            else
            {
                MessageBox.Show("Failed to
select model folder", "Model selection
failed", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
            }
        }
        //Model file type selection
        private void
ModelFileSwitchButton_Click(object sender,
EventArgs e)
```

```
        {
            OpenFileDialog DialogWindow =
new OpenFileDialog();

            DialogWindow.Title = "Model
file selection";
            if (DialogWindow.ShowDialog()
== DialogResult.OK)
            {
                ModelPath =
DialogWindow.FileName;
                ActiveModelBox.Text =
ModelPath;
            }
            else
            {
                MessageBox.Show("Failed to
select model file", "Model selection
failed", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
            }
        }
    }
```

Додаток Е

Презентаційний матеріал

Завдання

Мета роботи

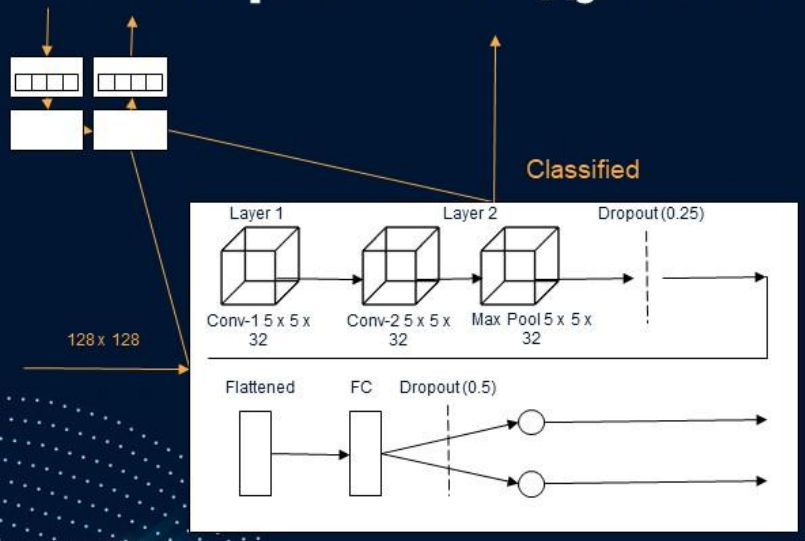
Розробка та програмна реалізація способу нейромережевого виявлення фейкових зображень документів для систем ідентифікації особистості.

- Провести аналіз предметної області, визначити особливості застосування згорткових нейронних мереж для задач класифікації при роботі з зображеннями документів.
- Виконати аналіз існуючих рішень щодо подібних задач.
- Розробити спосіб нейромережевого виявлення фейкових зображень документів.
- Розробити архітектуру нейронної мережі для визначення рівня достовірності зображень документів.
- Спроєктувати структуру інформаційної системи ідентифікації особистості й структуру відповідної бази даних.
- Спроєктувати та створити застосунок що використовує розроблений спосіб, виконати його тестування.

Схема способу нейромережевого виявлення фейкових зображень документів

Схема роботи модуля ELA

Схема роботи модуля CNN



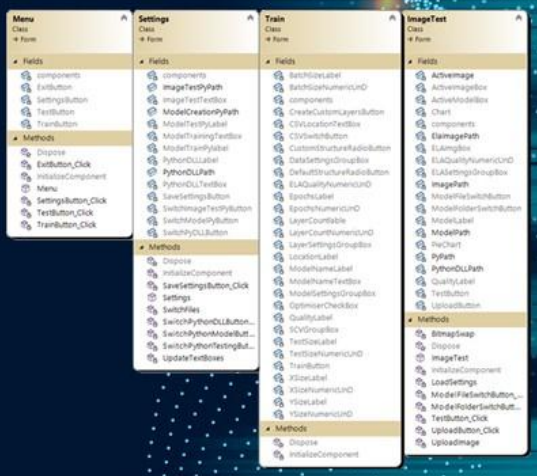
Структура компонентів способу нейромережевого виявлення фейкових зображень документів

Підсистема роботи з НМ
- Робота із структурою мережі
- Робота з параметрами мережі
- Робота з датасетом

Підсистема налаштувань
- Робота з бібліотеками
- Робота з Python файлами

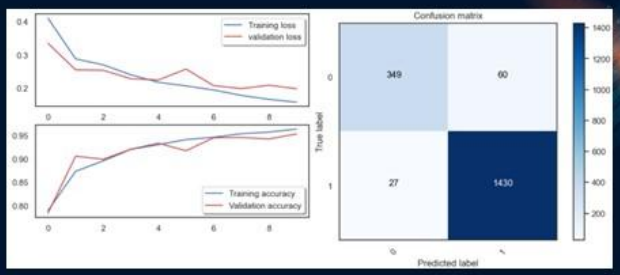
Підсистема розпізнавання фейкових зображень документів
- Робота із файлами зображення
- Робота з параметрами зображень
- Робота з файлами мережі

Підсистема інтерфейсу користувача
- Взаємодія з візуальними елементами
- Робота з результатом

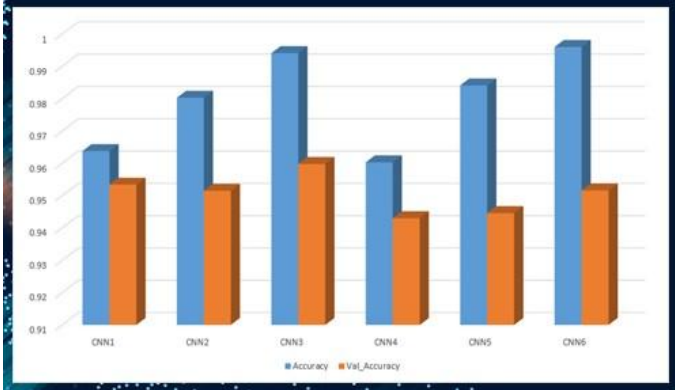


Дослідження ефективності - тестування

		Параметри Навчання мережі		
		Epochs	Batch size	Train-test Ratio
Ідентифікатор	CNN1	10	100	0.2
	CNN2	20	125	0.2
	CNN3	30	150	0.2
	CNN4	10	100	0.35
	CNN5	20	125	0.35
	CNN6	30	150	0.35



Дослідження ефективності - результат



Ідентифікатор		Результат		
		Accuracy	Val_Accuracy	Diff
	CNN1	0,9638	0,9534	0,0104
	CNN2	0,9803	0,9515	0,0288
	CNN3	0,9941	0,9598	0,0343
	CNN4	0,9603	0,9430	0,0173
	CNN5	0,941	0,9446	0,0395
	CNN6	0,9960	0,9516	0,0444

Висновки

В результаті виконання кваліфікаційної роботи бакалавра був розроблений спосіб ідентифікації фейкових зображень документів для системи ідентифікації особистості.

Для розробки модулів системи були використані мови програмування C# та Python, а також бібліотеки для них.

Реалізований додаток виконує наступні функції:

- Можливість тестування завантаженого зображення на вибраній навченій моделі з можливостю зміни вхідного параметру, візуалізація результату;
- Тренування моделі на вибраному датасеті з можливістю зміни та налаштування параметрів моделі, збереження в вигляді файлу чи папки;
- Можливість налаштування функціональних модулів.

Крім того для досягнення мети були виконані наступні етапи:

Аналіз предметної області та існуючих прикладних рішень, проєктування моделі CNN з аналізом відповідних програмного та алгоритмічного забезпечення, програмна реалізація розробленого методу та дослідження дієспроможності програмного додатку.

UNICHECK

Ім'я користувача:
Кафедра КН

ID перевірки:
1015409229

Дата перевірки:
03.06.2023 22:52:48 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
03.06.2023 22:53:32 EEST

ID користувача:
100005671

Назва документа: **КН-19-1 Жарновський**

Кількість сторінок: **64** Кількість слів: **7110** Кількість символів: **54479** Розмір файлу: **3.15 MB** ID файлу: **1015072628**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

4.39%
Схожість

Найбільша схожість: **2.03%** з джерелом з Бібліотеки (ID файлу: **1015071849**)

3.45% Джерела з Інтернету	276	Сторінка 66
2.42% Джерела з Бібліотеки	96	Сторінка 67

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

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

Замінені символи	1
Підозріле форматування	11 сторінок

Sat Jun 03 22:00:26 EEST 2023, Петровський Сергій Степанович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 4.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 11%**

ID: 114636 Назва: КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА Додано в БД: 2023-06-03 Автора: О.В. Жарновський Керівники: О.В. Собко Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	39867	602	2963 (7%)	46 (8%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ КАФЕДРИ КОМП'ЮТЕРНИХ НАУК
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

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

Назва: *Спосіб нейромережевого виявлення фейкових зображень документів для систем ідентифікації особистості*

Автор: *студент групи КН-19-1 Жарновський Олександр Володимирович*

Спеціальність: *122 – Комп'ютерні науки*

Освітня програма: *освітньо-професійна*

Науковий керівник: *викладач Собко О.В.*

Після аналізу звіту подібності зроблено такий висновок:

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

Підтвердження:

Запозичення, виявлені в роботі Жарновського О.В., не є плагіатом, оскільки: запозичення розміщені в розділі огляду існуючих підходів, не описують безпосередньо авторську роботу і не стосуються її результатів; усі запозичення фрагментарні; серед запозичень знаходяться загальновідомі терміни, скорочення та матеріали статей.

Обсяг запозичень, визначений системами виявлення збігів/ ідентичності/схожості, складає:

- за системою Anti-Plagiarism: 4%;

- за системою Unicheck: 4.39 %.

Керівник роботи _____ *Олена СОБКО*

Гарант ОП _____ *Олександр МАЗУРЕЦЬ*

Завідувач кафедри КН _____ *Олександр БАРМАК*

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

Кафедра комп'ютерних наук

ВІДГУК НАУКОВОГО КЕРІВНИКА

на кваліфікаційну роботу бакалавра

студента *гр. КН-19-1 Жарновського Олександра Володимировича*
за темою *Спосіб нейромережевого виявлення фейкових зображень документів для систем ідентифікації особистості*

1. Актуальність теми

Актуальність теми "Спосіб нейромережевого виявлення фейкових зображень документів для систем ідентифікації особистості" обумовлена зростаючою потребою впровадження надійних методів ідентифікації особистості та боротьби з шахрайством та підробкою документів. За допомогою нейромережевих методів, системи ідентифікації особистості спеціалісти можуть ефективно аналізувати зображення документів, виявляючи ознаки фальсифікації, такі як змінені деталі, неспівпадіння шрифтів або піксельних маніпуляцій. Це дозволяє забезпечити високу точність виявлення фейків і забезпечити надійну ідентифікацію особи.

2. Відповідність роботи предметній області Стандарту спеціальності 122 Комп'ютерні науки

За описом предметної області об'єктом дослідження є процес виявлення фейкових зображень документів для систем ідентифікації особистості. Предметом дослідження є моделі, методи, алгоритми та засоби для визначення рівня достовірності відсканованих та сфотографованих зображень документів що посвідчують особу. Під час вирішення даного завдання були використані математичні моделі, методи та алгоритми для розв'язання теоретичних і практичних проблем, які виникають при розробці інформаційних технологій. Виконана кваліфікаційна робота бакалавра відповідає стандарту бакалавра спеціальності 122 – Комп'ютерні науки.

3. Професійні та особистісні якості бакалавра

Жарновський Олександр Володимирович виявився обізнаним та компетентним студентом під час написання пояснювальної записки та розробки прикладного програмного забезпечення. Він продемонстрував неабиякі вміння та знання, а його професійні навички були на високому рівні. Свої знання у галузі розробки програмного забезпечення Жарновський Олександр Володимирович ефективно використовував у роботі, забезпечуючи успішне виконання поставлених завдань. Загалом, Жарновський Олександр Володимирович успішно впорався з усіма поставленими завданнями своєї кваліфікаційної роботи бакалавра і показав високу професійну компетентність.

4. Ступінь самостійності під час виконання кваліфікаційної роботи

Студент Жарновський О.В. продемонстрував високий рівень самостійності під час виконання всіх поставлених завдань у своїй кваліфікаційній роботі бакалавра. Він виявився студентом, що має вміння самостійно вирішувати труднощі, знаходити та реалізовувати ефективні рішення проблем, а також досягати поставлених цілей.

5. Ступінь оволодіння методами дослідження

Студент виявив глибоке розуміння теоретичних та практичних аспектів своєї теми. Застосування необхідних інструментів та технологій було виконано професійно та ефективно. Студент продемонстрував достатній рівень компетентності та майстерності у використанні інструментів, методів та технологій комп'ютерних наук, що сприяло успішному виконанню його кваліфікаційної роботи.

6. Повнота та якість розкриття теми роботи

Студент проявив високий рівень дослідницької роботи, аналітичних навичок та здатність до системного підходу. Аналіз предметної області був здійснений з достатньою деталізацією, що дозволило глибше розібратись у сутності та важливих аспектах виявлення фейкових зображень документів для систем ідентифікації особистості, тема в роботі була розкрита повністю та якісно.

7. Логічність, послідовність, аргументованість, літературна грамотність викладення матеріалу

Матеріал в роботі викладено логічно, послідовно та аргументовано. Літературна грамотність тексту роботи на високому рівні.

8. Можливість практичного застосування кваліфікаційної роботи бакалавра, окремих її частин

Практичне застосування способу нейромережевого виявлення фейкових зображень документів для систем ідентифікації особистості є надзвичайно актуальним і значущим. У сучасному цифровому світі, де маніпуляція зображеннями та підробка документів стають все поширенішими, необхідні надійні та ефективні методи виявлення фальсифікацій з метою забезпечення безпеки та достовірності ідентифікації щодо особистості.

9. Висновок про можливість допуску кваліфікаційної роботи бакалавра до захисту, на яку оцінку заслуговує робота

*Враховуючи високий рівень виконання та забезпечення усіх необхідних вимог, робота може бути допущена до захисту. Рекомендована оцінка «**відмінно**» .*

Керівник _____ *викладач каф. КН Олена СОБКО*

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

Кафедра комп'ютерних наук

РЕЦЕНЗІЯ

на кваліфікаційну роботу бакалавра

студента *гр. КН-19-1 Жарновського Олександра Володимировича*

за темою: *Спосіб нейромережевого виявлення фейкових зображень документів для систем ідентифікації особистості*

1. Актуальність обраної теми

Актуальність теми "Спосіб нейромережевого виявлення фейкових зображень документів для систем ідентифікації особистості" полягає в необхідності боротьби з шахрайством і зловживаннями в системах ідентифікації особистості. Використання нейромережевих методів дозволяє автоматично виявляти фейкові зображення документів, такі як підробки паспортів або прав водія, що допомагає запобігти незаконним діям та забезпечити надійну ідентифікацію особистості.

2. Повнота розкриття мети та завдань роботи

Мета та завдання роботи за темою "Спосіб нейромережевого виявлення фейкових зображень документів для систем ідентифікації особистості" розкриті повністю.

3. Зміст кожного розділу роботи

Розділ 1 включає детальний огляд теоретичних та практичних аспектів, пов'язаних з розпізнаванням підроблених документів, розглянуто існуючі програмні рішення, що дозволяють розпізнавати фейкові документи, а також проаналізовано ряд наукових робіт, що присвячені цій темі. У розділі 2 описано спосіб нейронного виявлення фейкових документів: надано схеми та їх детальний опис. Також наведено архітектуру нейронної мережі для реалізації способу, описано інформаційну структуру системи. Наведено перелік тестових даних. У розділі 3 описано детально структуру модулів інформаційної системи, подано у вигляді блок-схем особливості реалізації та детально описано їх. Також в даному розділі проаналізовано ефективність розробленого способу нейромережевого виявлення фейкових зображень документів.

4. Оцінка розробленої інформаційної системи, її практична цінність

Розроблена інформаційна система має велику практичну цінність. Дана система, що використовує нейромережеві методи, дозволяє автоматично виявляти фейкові зображення документів і попереджати можливі шахрайства та зловживання в системах ідентифікації особистості. Це забезпечує більш високу надійність і безпеку процесу ідентифікації, допомагає запобігти шахрайствам, використанню підроблених документів і покращує загальну ефективність системи ідентифікації особистості.

5. Якість оформлення кваліфікаційної роботи бакалавра

Кваліфікаційна робота студента Жарновського Олександра Володимировича оформлена згідно вимог до структури та нормоконтролю.

6. Недоліки кваліфікаційної роботи бакалавра

Суттєвих недоліків немає. Для тестування програмної системи використано лише один вид тестування, а інтерфейс виконано на англійській мові. Було б доцільно використати різні види тестування, а інтерфейс зробити українським або двомовним.

7. Загальний висновок (допускається чи не допускається до захисту), та оцінка на яку заслуговує кваліфікаційна робота.

Враховуючи рівень виконання та забезпечення усіх необхідних вимог, робота може бути допущена до захисту. Рекомендована оцінка « відмінно *».*

Рецензент Яцишин О.М. доц. каф. ІПТ