МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**

**НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЗАХИСТУ ІНФОРМАЦІЇ
КАФЕДРА УПРАВЛІННЯ ІНФОРМАЦІЙНОЮ ТА КІБЕРНЕТИЧНОЮ БЕЗПЕКОЮ**

КВАЛІФІКАЦІЙНА РОБОТА

на тему: “РОЗРОБКА ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ДЛЯ ВИЯВЛЕННЯ ФЕЙКІВ У ЗОБРАЖЕННЯХ ТА ВІДЕО”

на здобуття освітнього ступеня бакалавра
зі спеціальності 125 Кібербезпека
освітньої програми Управління інформаційною та кібернетичною безпекою

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Андрій ШВЕЦЬ
(підпис) *Ім’я, ПРІЗВИЩЕ здобувача*

Виконав(ла): здобувач(ка) вищої освіти гр. УБД-42

Ім’я, ПРІЗВИЩЕ

Керівник: Віталій ТИЩЕНКО
Д.т.н., професор Ім’я, ПРІЗВИЩЕ

Рецензент:

Ім’я, ПРІЗВИЩЕ

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Захисту інформації

Кафедра Управління інформаційною та кібернетичною безпекою

Ступінь вищої освіти бакалавр

Спеціальність 125 Кібербезпека

Освітня програма Управління інформаційною та кібернетичною безпекою

ЗАТВЕРДЖУЮ
Завідувач кафедрою УІКБ
_____ Світлана ЛЕГОМІНОВА
"____" _____2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Швецю Андрію Костянтиновичу
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи " Розробка інтелектуальної системи для виявлення фейків у зображеннях та відео",
керівник кваліфікаційної роботи ТИЩЕНКО Віталій ,
(ІПРІЗВИЩЕ, Ім'я,, науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологій від "27" лютого 2024 р. №36.

2. Строк подання кваліфікаційної роботи "25" травня 2024р.
3. Вихідні дані до кваліфікаційної роботи: *інтелектуальна система, машинне навчання, методи та засоби розробки інтелектуальної системи, наукова та технічна література.*
4. Перелік питань, які мають бути розроблені:
   1. Проаналізувати основні характеристики інтелектуальної системи для виявлення фейків.
   2. Дослідити моделі інтелектуальної системи для виявлення фейків зокрема представити код.
   3. Визначити напрями та методи розробка інтелектуальної системи для виявлення фейків
5. Перелік ілюстративного матеріалу: *презентація PowerPoint*
6. Дата видачі завдання "22" лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Етапи кваліфікаційної роботи	Термін виконання Етапів роботи	Примітка
1.	Визначення об’єкту, предмету, мети та завдань дослідження.	13.03.2024	
2.	Збір та аналіз літератури.	30.03.2024	
3.	Аналіз основних характеристик інтелектуальної системи для виявлення фейків	17.04.2024	
4.	Дослідження особливостей та моделі інтелектуальної системи для виявлення фейків	01.05.2024	
5.	Розробка інтелектуальної системи для виявлення фейків	15.05.2024	
6.	Формулювання висновків за результатами проведеного дослідження.	22.05.2024	
7.	Оформлення роботи.	24.05.2024	
8.	Оформлення презентації.	01.06.2024	
9.	Отримання рецензії на роботу.	04.06.2024	
10.	Захист в ДЕК.	__.06.2024	

Здобувач(ка) вищої освіти _____ Андрій ШВЕЦЬ
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Віталій ТИЩЕНКО
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЗАХИСТУ ІНФОРМАЦІЇ

ПОДАННЯ
ГОЛОВІ ЕКЗАМЕНАЦІЙНОЇ КОМІСІЇ
ЩОДО ЗАХИСТУ КВАЛІФІКАЦІЙНОЇ РОБОТИ

на здобуття освітнього ступеня бакалавра

Направляється здобувач Швець А.К. до захисту кваліфікаційної роботи
(прізвище та ініціали)
за спеціальністю 125 Кібербезпека
(код, найменування спеціальності)
освітньої програми Управління інформаційною та кібернетичною безпекою
(назва)
на тему: “ Розробка інтелектуальної системи для виявлення фейків у зображеннях та відео ”
Кваліфікаційна робота і рецензія додаються.

Директор ННІЗІ _____ Віталій САВЧЕНКО
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

Висновок керівника кваліфікаційної роботи

Здобувач ШВЕЦЬ Андрій у кваліфікаційній роботі дослідив ефективність та ризики використання штучного інтелекту для виявлення фейків у зображеннях та відео. Він проаналізував основні характеристики технологій штучного інтелекту, які використовуються для виявлення та класифікації фейкових зображень і відеоматеріалів, а також розглянув інструменти та методи, що сприяють оптимізації цих процесів. На основі проведених досліджень були розроблені практичні рекомендації щодо вдосконалення алгоритмів та підходів для підвищення точності виявлення фейків.

ШВЕЦЬ Андрій продемонстрував глибоке розуміння проблематики дослідження та основних теоретичних і практичних аспектів її вирішення. Він показав здатність використовувати наукові методи дослідження, а також проявив себе як організований та відповідальний виконавець. Результати дослідження були апробовані на одній конференції.

Все це дозволяє оцінити кваліфікаційну роботу здобувача ШВЕЦЯ Андрія на оцінку “_____” та присвоїти йому кваліфікацію бакалавра з кібербезпеки за освітньою програмою Управління інформаційною та кібернетичною безпекою.

Керівник кваліфікаційної роботи ____ Віталій ТИЩЕНКО
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

“_____“____2024 року

Висновок кафедри про кваліфікаційну роботу

Кваліфікаційна робота розглянута. Здобувач Швець А.К. допускається до захисту даної роботи в Екзаменаційній комісії.

Завідувач кафедри
управління інформаційною
та кібернетичною безпекою _____ Світлана ЛЕГОМІНОВА
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

ВІДГУК РЕЦЕНЗЕНТА
на кваліфікаційну бакалаврську роботу

здобувача вищої освіти Швеця Андрія

на тему “Розробка інтелектуальної системи для виявлення фейків у зображеннях та відео ”

Актуальність. У світі, де інформаційна безпека стає дедалі важливішою через зростання кількості фейкових новин та маніпуляцій у зображеннях та відео, важливо забезпечити надійні методи для виявлення таких фейків. Розробка інтелектуальних систем для виявлення фейків у зображеннях та відео є критично важливим завданням для захисту громадськості та підтримки довіри до інформації. Дослідження у цій сфері є актуальним і відповідає сучасним потребам суспільства у забезпеченні інформаційної безпеки.

Позитивні сторони.

1. У роботі досліджено особливості використання штучного інтелекту для виявлення фейків у зображеннях та відео.
2. Кваліфікаційна робота оформлена відповідно до вимог. Виклад матеріалу здійснено відповідно до плану, зроблено логічні висновки. Ключові положення роботи представлено у вигляді рисунків.
3. Автор опрацював значну джерельну базу: близько 42 публікацій, в тому числі 30 англомовних.
4. За результатами дослідження запропоновано рекомендації щодо оптимізації процесів виявлення фейків у зображеннях та відео за допомогою штучного інтелекту.

Недоліки.

Доцільно було б приділити більше уваги вивченню і класифікації програмних інструментів для оцінки ефективності процесів виявлення фейків у зображеннях та відео.

Однак, вищезгадані зауваження не впливають на загальну позитивну оцінку кваліфікаційної роботи.

Висновок Кваліфікаційна робота виконана на належному науково-методичному рівні і заслуговує оцінки “_____”, а здобувач Швець Андрій заслуговує присвоєння кваліфікації бакалавра з кібербезпеки за освітньою програмою Управління інформаційною та кібернетичною безпекою.

Рецензент: _____ _____

підпис Ім’я, ПРІЗВИЩЕ

РЕФЕРАТ

Кваліфікаційна робота присвячена розробці системи виявлення підробки зображень з використанням передових методів машинного навчання, зокрема, згорткових нейронних мереж (CNN). Робота складається зі вступу, трьох розділів, що містять 20 рисунків, висновківта списку використаних джерел, що містить 42 найменування. Загальний обсяг роботи становить 66 аркушів, з яких 15 аркушів займають перелік умовних скорочень, список використаних джерел та додатки.

Метою роботи є дослідження комплексне дослідження з розробки системи виявлення підробки зображень Для цього у роботі використовуються передовиі методи машинного навчання, зокрема, згорткові нейронні мережі (CNN).

Об'єктом дослідження є інтелектуальна система виявлення фейків.

Предмет дослідження - особливості розробки інтелектуальної системи для виявлення фейків у зображеннях та відео.

Методи дослідження. Для вирішення означеного вище наукового завдання в роботі використані методи машинного навчання згорткових нейронних мереж (CNN).

Як результат у роботі проведено аналіз основних характеристик інтелектуальної системи для виявлення фейків; досліджено особливості розробки інтелектуальної системи для виявлення фейків, зокрема представлено код програми на приклад системи; визначено напрями та методи розробки та сама інтелектуальна система для виявлення фейків відповідно до запропонованої класифікації загроз.

Галузь застосування. Розроблена інтелектуальна система допоможе виявляти фейки в умовах інформаційного протиборства, пропаганді та фейкових новин. Складні моделі ШНМ і застосування таких методів, як навчання з переносом і ансамблеві методи, для покращення можливостей системи виявлення підробок.

Ключові слова: ВИЯВЛЕННЯ ПІДРОБКИ ЗОБРАЖЕНЬ, ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, МАШИННЕ НАВЧАННЯ, ЦИФРОВА ОБРОБКА ЗОБРАЖЕНЬ, КЛАСИФІКАЦІЯ АВТЕНТИЧНОСТІ, ГЛИБОКЕ НАВЧАННЯ.

ABSTRACT

The qualification work is devoted to the development of an image forgery detection system using advanced machine learning methods, in particular, convolutional neural networks (CNN). The work consists of an introduction, three chapters containing 20 figures, conclusions and a list of used sources containing 42 titles. The total volume of the work is 66 sheets, of which 15 sheets are a list of conventional abbreviations, a list of used sources, and appendices.

The purpose of the work is to conduct a comprehensive study on the development of a system for detecting forgery of images. For this, the work uses advanced methods of machine learning, in particular, convolutional neural networks (CNN).

The object of the study is an intelligent system for detecting fakes.

The subject of the study is the peculiarities of the development of an intelligent system for detecting fakes in images and videos.

Research methods. To solve the above-mentioned scientific task, the methods of machine learning of convolutional neural networks (CNN) were used in the work.

As a result, the paper analyzed the main characteristics of the intelligent system for detecting fakes; the peculiarities of the development of an intelligent system for the detection of fakes are investigated, in particular, the program code is presented as an example of the system; directions and methods of development and the intelligent system for detecting fakes according to the proposed classification of threats are determined.

Field of application. The developed intelligent system will help detect fakes in the conditions of information conflict, propaganda and fake news. Complex ANN models and the application of techniques such as transfer learning and ensemble methods to improve the capabilities of the counterfeit detection system.

Keywords: IMAGE DETECTION, CONVOLUTIONAL NEURAL NETWORKS, MACHINE LEARNING, DIGITAL IMAGE PROCESSING, AUTHENTICITY CLASSIFICATION, DEEP LEARNING.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

1. **AI** - Artificial Intelligence
2. **CNN** - Convolutional Neural Network
3. **DL** - Deep Learning
4. **ML** - Machine Learning
5. **GPU** - Graphics Processing Unit
6. **CPU** - Central Processing Unit
7. **ReLU** - Rectified Linear Unit
8. **SVM** - Support Vector Machine
9. **RGB** - Red, Green, Blue (color model)
10. **HSV** - Hue, Saturation, Value (color model)
11. **API** - Application Programming Interface
12. **SSD** - Solid-State Drive
13. **L2** - Layer 2 (regularization technique)
14. **IoT** - Internet of Things
15. **RNN** - Recurrent Neural Network
16. **GAN** - Generative Adversarial Network
17. **TP** - True Positive
18. **TN** - True Negative
19. **FP** - False Positive
20. **FN** - False Negative
21. **ROC** - Receiver Operating Characteristic
22. **AUC** - Area Under the Curve

ВСТУП

У цифрову епоху поширення інструментів для редагування зображень сприяло широкому розповсюдженню підробки зображень, що створює значні проблеми в різних галузях, зокрема в медіа, судовому доказуванні та інформаційній безпеці.

Це підкреслює актуальність розробки складних систем виявлення, здатних ідентифікувати маніпуляції з зображеннями, забезпечуючи тим самим цілісність візуальної інформації.

Практичне значення такої роботи важко переоцінити, оскільки вона безпосередньо сприяє забезпеченню автентичності та достовірності цифрових медіа, які є невід'ємною складовою функціонування суспільства, заснованого на правді.

Метою цієї роботи є розробка та реалізація надійної системи виявлення підробки зображень з використанням передових методів машинного навчання, зокрема, з використанням можливостей згорткових нейронних мереж (CNN) для автоматизації процесу виявлення з високою точністю та ефективністю. Основним завданням цієї роботи є розробка моделі, яка може точно розрізняти підроблені та справжні зображення, навчання цієї моделі на повному наборі даних та оцінка її продуктивності за допомогою стандартних метрик, таких як точність, достовірність, відтворення та F1-бали.

Об'єктом цієї роботи є цифрові зображення, що піддаються різним поширеним методам маніпуляцій, включаючи, але не обмежуючись ними, склеювання, клонування та ретушування.

Предметом роботи є методологічний підхід із застосуванням методів глибокого навчання для аналізу та класифікації цих зображень на основі їх автентичності.

Методи дослідження. Для вирішення проблеми виявлення підробки зображень у цьому дослідженні буде застосовано методологічний підхід,

заснований на використанні згорткових нейронних мереж (CNN) - класу алгоритмів глибокого навчання, які є особливо ефективними в аналізі візуальних зображень. Дослідження буде проводитися в кілька структурованих етапів:

Тестування результатів. Після того, як модель буде розроблена і навчена, будуть зроблені наступні кроки для перевірки її ефективності та надійності у виявленні підробок зображень.

Цей підхід не лише забезпечує технічне вирішення зростаючої проблеми, але й робить внесок у ширший дискурс про етику цифрових медіа та роль штучного інтелекту у підтримці суспільної довіри до цифрового контенту.

РОЗДІЛ 1. ОГЛЯД ЛІТЕРАТУРИ

1.1 Методи підробки зображень

Підробка зображень є серйозною проблемою в цифрову епоху, яка впливає на такі сектори, як журналістика, судові докази та громадська інформація. У цьому розділі розглядаються основні методи підробки зображень, а саме: копіювання-переміщення, склеювання та ретушування. Кожен метод розглядається з точки зору його техніки, труднощів виявлення та наслідків його використання.

Підробка копіюванням полягає у копіюванні частини одного і того ж зображення та вставці її в інше місце на тому ж самому зображенні. Цей метод часто використовується для того, щоб приховати небажаний елемент або скопіювати існуючий об'єкт для створення оманливого контексту. Процес зазвичай включає такі кроки: вибір області, яку потрібно скопіювати, застосування трансформації, якщо це необхідно (наприклад, обертання або масштабування), і змішування її з областю призначення для забезпечення безшовності [1].

Методи виявлення підробок копіювання-переміщення розвинулися, зосередившись на таких особливостях, як блокові методи, коли зображення ділиться на блоки, що перекриваються, і кожен блок порівнюється на наявність дублікатів. Удосконалені методи використовують методи на основі ключових точок, застосовуючи такі алгоритми, як Scale-Invariant Feature Transform (SIFT) або Speeded Robust Features (SURF) для виявлення ділянок зі схожими ознаками навіть у випадках масштабування або повороту.

Сплайсинг передбачає об'єднання сегментів з двох або більше різних зображень для створення нового, складеного зображення. Цей метод є особливо небезпечним, оскільки він може бути використаний для створення повністю вигаданих сценаріїв шляхом злиття елементів з декількох джерел. Сплайсинг, як правило, складніше виявити, ніж копіювання, оскільки він включає зовнішні елементи, які спочатку не були присутні в зображенні.

Методи виявлення склеювання часто ґрунтуються на розбіжностях в освітленні, рівнях шуму або крайових артефактах на кордонах області, що склеюється. Ці невідповідності іноді можуть бути тонкими і не помітними для людського ока. Обчислювальні методи виявлення зрощування включають використання моделей машинного навчання, навчених розпізнавати типові ознаки зрощених зображень, такі як аномальні гістограми або непослідовні артефакти стиснення.Ретушування - це більш витончена форма підробки, яка маніпулює зовнішнім виглядом зображення без додавання або вилучення ключових елементів. Поширені методи ретуші включають згладжування, підвищення різкості, регулювання колірного балансу та зміну яскравості або контрастності. Хоча ретуш часто використовується з доброзичливою метою, наприклад, для покращення естетичної привабливості фотографій, її також можна зловживати, щоб змінити сприйняття або спотворити реальність[2].

Виявлення ретуші є складним завданням, оскільки зміни часто є мінімальними і не містять явних артефактів, як в інших типах підробок. Методи виявлення включають аналіз гістограм зображення на наявність незвичних патернів, вивчення розподілу шуму та використання частотного аналізу для виявлення неприродних підсилень або придушень.

Цілісність візуальної інформації має вирішальне значення для збереження довіри до цифрових медіа. Розуміння методів підробки зображень і розробка надійних методів виявлення є важливими для боротьби з дезінформацією. Кожен метод підробки створює унікальні виклики і вимагає специфічних стратегій для виявлення. Оскільки технологія обробки цифрових зображень продовжує розвиватися, методології виявлення та пом'якшення наслідків підробки зображень також повинні розвиватися. Ця безперервна боротьба вимагає глибокого розуміння як технічних аспектів маніпуляцій із зображеннями, так і психологічного впливу оманливих зображень.

1.2 Традиційні методи виявлення підробок

Виявлення підробок зображень історично спиралося на кілька традиційних методів, в першу чергу на методику водяних знаків і підписів. Ці методи заклали основу для розуміння і розробки більш досконалих інструментів цифрової експертизи зображень. У цьому розділі розглядаються ці традиційні методи, обговорюються їхні операційні рамки, ефективність та притаманні їм обмеження.

Водяні знаки передбачають вбудовування інформації в зображення під час створення або перед розповсюдженням. Ця вбудована інформація, або водяний знак, розроблена таким чином, щоб бути непомітною за звичайних умов перегляду, але може бути виявлена або вилучена для перевірки автентичності зображення. Водяні знаки можуть бути видимими або невидимими і зазвичай використовуються для підтвердження авторських прав, перевірки цілісності контенту або як засіб передачі іншої важливої інформації про зображення. Математичне представлення процесу вбудовування водяного знаку може бути виражене наступним чином:

$$[I_w = f(I, W, k)], \tag{1.1}$$

де (I_w) - зображення з водяним знаком,

(I) - оригінальне зображення,

(W) представляє водяний знак,

(k) - ключ, який використовується для вбудовування водяного знаку за допомогою функції (f).

Незважаючи на свою корисність, водяні знаки мають обмеження. Основною проблемою є дотримання балансу між невидимістю водяного знаку та його стійкістю до видалення або знищення шляхом стиснення, обрізання або інших форм маніпуляцій. Крім того, водяні знаки не запобігають підробці, а

скоріше допомагають виявити її постфактум, якщо припустити, що водяний знак залишається неушкодженим і його можна виявити після маніпуляцій.

Методи на основі підписів передбачають створення унікального підпису на основі вмісту зображення, який потім використовується для перевірки цілісності зображення. Цей підпис обчислюється з використанням певних атрибутів зображення, таких як інтенсивність пікселів, гістограми або коефіцієнти перетворення, і зберігається окремо від зображення. Коли потрібна перевірка, обчислюється новий підпис і порівнюється з оригіналом. Процес генерації підпису може бути представлений функцією:

$$[S = g(I, k)], \tag{1.2}$$

де (S) - підпис,

(I) - зображення,

(k) - ключ або параметр, що впливає на процес генерації підпису через функцію (g).

Методи, засновані на підписі, мають перевагу завдяки своїй неінтрузивній природі, оскільки вони не змінюють зображення. Однак, вони дуже чутливі до витончених підробок, які не змінюють суттєво характеристики, що використовуються для генерації підпису. Крім того, ці методи можуть вимагати значних обчислювальних витрат, особливо при роботі з великими зображеннями або базами даних, оскільки підпис необхідно перераховувати для кожного процесу перевірки.Традиційні методи виявлення підробок стикаються з кількома проблемами. Високі обчислювальні витрати на обробку та перевірку водяних знаків або підписів роблять їх менш придатними для застосування в реальному часі або в системах з обмеженими обчислювальними ресурсами. Крім того, ефективність цих методів знижується з розвитком методів підробки, особливо з появою методів глибокого навчання, здатних генерувати високореалістичні підробки.

Водяні знаки та методи на основі підписів також мають проблеми з надійністю та стійкістю. Складні атаки, що включають просторові або частотні маніпуляції, можуть зробити ці методи неефективними, або затушовуючи водяний знак, або змінюючи властивості зображення настільки, що оригінальний підпис більше не збігається з ним.Хоча традиційні методи виявлення підробки зображень, такі як водяні знаки та методи на основі підписів, надали фундаментальні знання та інструменти для автентифікації зображень, їхні обмеження підкреслюють необхідність більш досконалих та адаптивних підходів. Еволюція методів підробки зображень, особливо з інтеграцією штучного інтелекту, вимагає постійного вдосконалення технологій виявлення. Цей постійний розвиток має вирішальне значення для збереження цілісності цифрових медіа та достовірності візуальної інформації в різних додатках.

1.3 Машинне навчання та підходи до глибокого навчання

Поява технологій машинного навчання (ML) і глибокого навчання (DL) зробила революцію в галузі виявлення підробок зображень, запровадивши більш складні, ефективні та автоматизовані методи. У цьому розділі розглядаються різні методи ML і DL, включаючи машини опорних векторів (SVM), згорткові нейронні мережі (CNN) і автокодери, обговорюється їх застосування для виявлення підробок зображень, їх порівняльна ефективність і тонкощі роботи. Машини на основі опорних векторів - це набір методів керованого навчання, що використовуються для класифікації, регресії та виявлення викидів. У контексті виявлення підробки зображень, SVM використовуються для класифікації зображень як підроблених або справжніх на основі вилучення ознак. SVM-класифікатор працює шляхом пошуку гіперплощини, яка найкраще розділяє набір даних на класи.

Класифікацію SVM можна представити у вигляді функції:

$$[f(x) = \text{sgn}(w^T x + b)], \quad (1.3)$$

де (x) представляє вхідні ознаки,

(w) - вектор ваги,

(b) - зміщення,

sgn - функція знаку.

Ефективність SVM у виявленні підробок значною мірою залежить від вибору функції ядра та ознак, що використовуються для навчання моделі. Хоча SVM потужні в обробці даних високої розмірності і добре працюють з чіткою межею поділу, вони менш ефективні з великими наборами даних і можуть бути схильні до перенастроювання у дуже складних моделях. CNN є значним досягненням у галузі глибокого навчання для обробки зображень. Ці мережі особливо добре вловлюють ієрархічні патерни в просторових даних, що робить їх винятково придатними для завдань аналізу зображень, включаючи виявлення підробок. CNN автоматично навчається і витягує ознаки із зображень за допомогою декількох шарів згорткових фільтрів і шарів об'єднання, а потім повністю з'єднує шари для класифікації[11].

Робота згорткового шару може бути математично описана наступним чином:

$$\left[f_{ij}^k = \sigma\left(\sum_m \sum_n I_{(i+m)(j+n)} \cdot K_{mn}^k + b_k\right)\right], \quad (1.4)$$

де (I) - вхідне зображення,

(K^k) - k-те ядро фільтра,

(b_k) - член зсуву,

(σ) - нелінійна функція активації,

$\left(f_{ij}^k\right)$ - вихідна карта ознак.

ШНДК є високоефективними у виявленні складних підробок, в тому числі із застосуванням складних маніпуляцій, таких як склеювання та ретушування, завдяки своїй здатності навчатися дискримінативних ознак безпосередньо з

даних без необхідності ручного вилучення ознак. Автокодери - це тип нейронної мережі, що використовується для навчання ефективного кодування немаркованих даних, як правило, з метою зменшення розмірності або вивчення особливостей. У сфері виявлення підробок автокодери можна використовувати для реконструкції зображень, де помилка реконструкції може вказувати на потенційну підробку[8]. Ідея полягає в тому, що автокодер, навчений на автентичних зображеннях, матиме вищий рівень помилок при реконструкції підроблених зображень. Базовий автокодер можна описати двома функціями, кодера і декодера:

$$[h = \sigma(Wx + b)][r = \sigma(W'h + b')], \quad (1.5)$$

де (x) - вхідне зображення,

(h) - закодоване зображення,

(r) - відновлене зображення,

(W) і (W') - ваги,

(b) і (b') – зсуви,

(σ) - функція активації.

Хоча автокодери мають перевагу завдяки здатності навчатися без нагляду і працювати зі складними структурами даних, вони потребують великих обсягів даних для ефективного навчання і можуть вимагати значних обчислювальних витрат. Порівнюючи SVM, CNN та автокодери, очевидно, що кожен алгоритм має свої сильні та слабкі сторони. SVM менш обчислювально інтенсивні і можуть бути ефективними з добре підібраним ядром, але мають проблеми з великими, складними наборами даних. ШНМ відмінно справляються із завданнями, що вимагають виділення ознак, і можуть обробляти складні маніпуляції із зображеннями, але потребують значних обчислювальних ресурсів і даних для навчання. Автокодери забезпечують унікальний підхід завдяки неконтрольованому навчанню і можуть ефективно виявляти аномалії,

але також вимагають значних навчальних даних і обчислювальних потужностей.

Машинне та глибинне навчання глибоко вплинули на сферу виявлення підробок зображень, пропонуючи потужні інструменти, які перевершують традиційні методи як за ефективністю, так і за результативністю. Вибір алгоритму залежить від конкретних вимог завдання, включаючи характер даних, складність підробок і доступні обчислювальні ресурси. Оскільки ці технології продовжують розвиватися, вони обіцяють ще більший прогрес у виявленні та запобіганні підробці зображень.

Висновки до розділу 1

У першому розділі проведено огляд літератури та досліджено існуючі методи підробки зображень. Описано традиційні підходи до виявлення підробок, які включають аналіз метаданих, перевірку узгодженості тіней, відбитків та інших аспектів зображення. Виявлено, що ці методи мають обмежену ефективність у сучасних умовах через розвиток технологій маніпулювання зображеннями. Окрему увагу приділено машинному навчанню та глибокому навчанню як перспективним підходам до виявлення підробок, що демонструють високу точність та ефективність у аналізі візуальних даних.

РОЗДІЛ 2. ЗАПРОПОНОВАНА ІНТЕЛЕКТУАЛЬНА СИСТЕМА

2.1 Архітектура системи

Архітектура запропонованої системи для виявлення підробок зображень призначена для ефективної обробки вхідних зображень, застосування складних алгоритмів машинного навчання для виявлення підробок і виведення результатів у зручний для користувача спосіб. У цьому розділі описано архітектуру системи, детально описано етапи введення, обробки та виведення результатів, а також наведено схематичне зображення робочого процесу для кращого розуміння.

Вхідний етап системи відповідає за отримання та підготовку зображень до аналізу. Цей етап включає кілька ключових процесів:

1. Отримання **зображень**: Зображення можуть бути завантажені користувачами або автоматично отримані з баз даних чи онлайн-платформ.

2. Попередня **обробка**: Для стандартизації вхідних даних і підвищення ефективності алгоритмів виявлення зображення проходять попередню обробку. Це включає зміну розміру зображень до єдиного розміру, нормалізацію значень пікселів і, можливо, перетворення зображень у відтінки сірого для зменшення обчислювальної складності.

Математичне представлення процесу нормалізації може бути виражене наступним чином:

$$\left[I_{norm} = \frac{I-\mu}{\sigma}\right], \tag{2.1}$$

де (I) - вихідне зображення,

(μ) - середнє значення пікселів,

(σ) - стандартне відхилення,

(I_{norm}) - нормалізоване зображення.

Етап обробки

Ядром системи є етап обробки, на якому моделі машинного навчання аналізують попередньо оброблені зображення для виявлення ознак підробки. Цей етап структурований наступним чином:

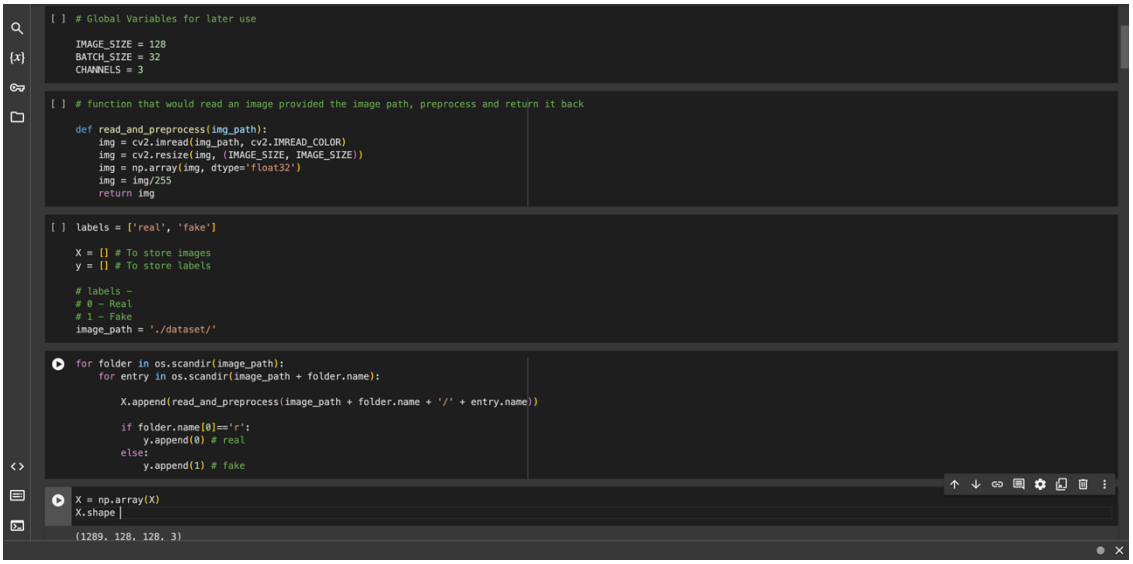


Рис 2.1. Попередня обробка даних

1. **Виділення** ознак: Система виокремлює з зображень відповідні ознаки, які можуть свідчити про потенційну підробку. Залежно від моделі, це можуть бути такі аспекти, як текстура, краї та кольорові гістограми.

2. **Модель виявлення** підробок: Виділені ознаки вводяться в попередньо навчену модель машинного навчання. Система в основному використовує згорткові нейронні мережі (CNN) завдяки їхній здатності обробляти дані зображень. Архітектура CNN призначена для виявлення розбіжностей, які вказують на маніпуляції, наприклад, неприродні кореляції пікселів або невідповідності в освітленні.

Робота CNN може бути описана операцією згортки:

$$\left[f_{ij}^{k} = \sigma\left(\sum_{m} \sum_{n} I_{(i+m)(j+n)} \cdot K_{mn}^{k} + b_{k} \right) \right], \tag{2.2}$$

де (I) - вхідна карта ознак,

(K^k) - ядро k-го фільтра,

(b_k) - член зсуву, (\sigma) - нелінійна функція активації,

$\left(f_{ij}^k\right)$ - вихідна карта ознак k-го фільтра у позиції (i, j).

3. Прийняття **рішення**: На основі даних, отриманих від CNN, рівень прийняття рішень визначає, чи є зображення автентичним, чи підробленим. Цей рівень може використовувати додаткові критерії або порогові значення для підвищення точності.

Вихідний етап передає користувачеві результати процесу виявлення підробки:

1. **Презентація результатів**: Система представляє детальний звіт про аналіз із зазначенням ймовірності підробки зображення та виокремленням областей потенційних маніпуляцій.

2. **Механізм** зворотного зв'язку: Користувачі можуть надавати відгуки про точність розпізнавання, які можуть бути використані для подальшого навчання та вдосконалення моделі.

Робочий процес системи можна візуалізувати на наступній схемі:

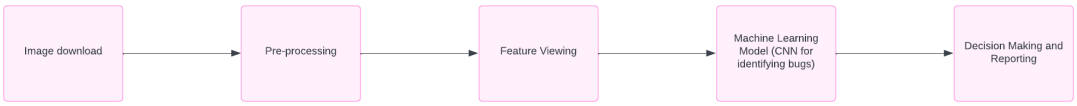


Рис 2.2. Схема робочого процесу системи

Запропонована архітектура системи для виявлення підробки зображень розроблена таким чином, щоб бути надійною, ефективною та зручною для користувача. Інтегруючи передові методи машинного навчання з оптимізованим робочим процесом, система забезпечує високу точність виявлення підробок, зберігаючи при цьому простоту використання для кінцевого користувача. Архітектура підтримує постійне вдосконалення завдяки зворотному зв'язку з користувачами, забезпечуючи розвиток системи у

відповідь на нові виклики у сфері підробки зображень. Цей цілісний підхід не лише відповідає технічним вимогам виявлення підробок, але й враховує практичні потреби користувачів, що робить його комплексним рішенням проблеми маніпуляцій з цифровими зображеннями.

2.2 Попередня обробка та виділення особливостей

Ефективність будь-якої системи виявлення підробок зображень значною мірою залежить від надійності етапів попередньої обробки та вилучення ознак. Ці етапи є критично важливими, оскільки вони готують вхідні дані для подальшого аналізу і впливають на точність і продуктивність алгоритмів виявлення. У цьому розділі детально розглядаються етапи попередньої обробки та методи вилучення ознак, що використовуються в запропонованій системі, з'ясовується їх наукове підґрунтя та інтеграція в загальну архітектуру.

Етапи попередньої обробки

Попередня обробка - це початковий етап робочого процесу аналізу зображень, на якому сирі вхідні зображення перетворюються в стандартизований формат, який може бути ефективно оброблений алгоритмами виявлення. Основними цілями попередньої обробки є зменшення обчислювальної складності та покращення розрізнення об'єктів. Наступні кроки є невід'ємною частиною цього етапу:

Рис 2.3. Зміна розміру зображення

1. Зміна **розміру**: Зображення змінюються до єдиного розміру, щоб забезпечити узгодженість по всьому набору даних. Цей крок має вирішальне значення, оскільки вхідні зображення можуть значно відрізнятися за розміром, і обробка їх у початкових розмірах може бути обчислювально дорогою і

непослідовною. Зміна розміру допомагає зменшити обчислювальне навантаження і стандартизує вхідні дані для етапу вилучення ознак.

2. **Нормалізація**: Цей процес передбачає масштабування значень пікселів зображень до стандартного діапазону, зазвичай [0, 1] або [-1, 1]. Математично нормалізація має такий вигляд:

$$\left[I_{norm} = \frac{I - \min(I)}{\max(I) - \min(I)} \right], \tag{2.3}$$

де (I) - оригінальне зображення, а (I_{norm}) - нормалізоване зображення. Цей крок необхідний для приведення вхідних ознак до спільного масштабу та покращення чисельної стабільності алгоритмів, що використовуються на наступних етапах.

3. Перетворення колірного **простору**: Залежно від вимог методів вилучення ознак, зображення можуть бути перетворені з RGB в інші колірні простори, такі як HSV або відтінки сірого. Таке перетворення може допомогти підкреслити певні особливості, які є більш помітними в цих альтернативних колірних просторах.

```
[ ]  for folder in os.scandir(image_path):
         for entry in os.scandir(image_path + folder.name):

             X.append(read_and_preprocess(image_path + folder.name + '/' + entry.name))

             if folder.name[0]=='r':
                 y.append(0) # real
             else:
                 y.append(1) # fake
```

```
X = np.array(X)
X.shape
```

```
(1289, 128, 128, 3)
```

```
[ ]  y = np.array(y)
     y.shape
```

```
(1289,)
```

Рис 2.4. Методи виділення елементів

Виділення ознак є важливим компонентом процесу виявлення підробки зображень, коли відповідна інформація виділяється з попередньо оброблених зображень. Ця інформація, або ознаки, є тим, що аналізують моделі машинного навчання для визначення автентичності зображення. Система використовує кілька методів вилучення ознак, кожен з яких націлений на різні аспекти зображення:

1. **Особливості текстури**: Аналіз текстури передбачає вивчення характеристик поверхні зображення, які можуть свідчити про фальсифікацію. Для вилучення дескрипторів текстури використовуються такі методи, як матриця співпадіння рівнів сірого (GLCM) або локальні бінарні патерни (LBP). Ці дескриптори фіксують розподіл і співвідношення інтенсивностей пікселів, які можуть бути значно змінені в процесі підробки[5].

2. **Особливості форми**: Аналіз форми має вирішальне значення для виявлення геометричних змін у зображенні. Для вилучення дескрипторів меж застосовуються фільтри виявлення країв, такі як оператори Собела або Кенні. Ці функції допомагають виявити неприродні форми або невідповідності, що виникають внаслідок склеювання зображень або вставки об'єктів.

3. Кольорові **характеристики**: Характеристики кольору витягуються для аналізу узгодженості розподілу кольорів на зображенні. Гістограми колірних каналів обчислюються для відображення глобального розподілу кольорів, тоді як більш локалізовані колірні характеристики можна отримати за допомогою таких методів, як колірні корелограми. Розбіжності в колірних характеристиках можуть вказувати на ділянки зображення, які могли бути піддані маніпуляціям.

Рис 2.5. Векторізація ознаків

Витягнуті ознаки потім векторизуються і стандартизуються, щоб сформувати вектор ознак для кожного зображення. Цей вектор є вхідними даними для моделей машинного навчання, які класифікують зображення як автентичне або підроблене. Вибір ознак та їх представлення має вирішальне значення, оскільки це безпосередньо впливає на здатність до навчання та продуктивність моделей класифікації.

Етапи попередньої обробки та вилучення ознак є основоположними для роботи системи виявлення підроблених зображень. Завдяки стандартизації та вилученню релевантної інформації із зображень, ці етапи гарантують, що моделі машинного навчання працюють в оптимальних умовах і з найкращими вхідними даними. Наукові підходи, що застосовуються в цих процесах, мають вирішальне значення для підвищення точності та ефективності системи, що робить її надійним інструментом у боротьбі з підробкою зображень.

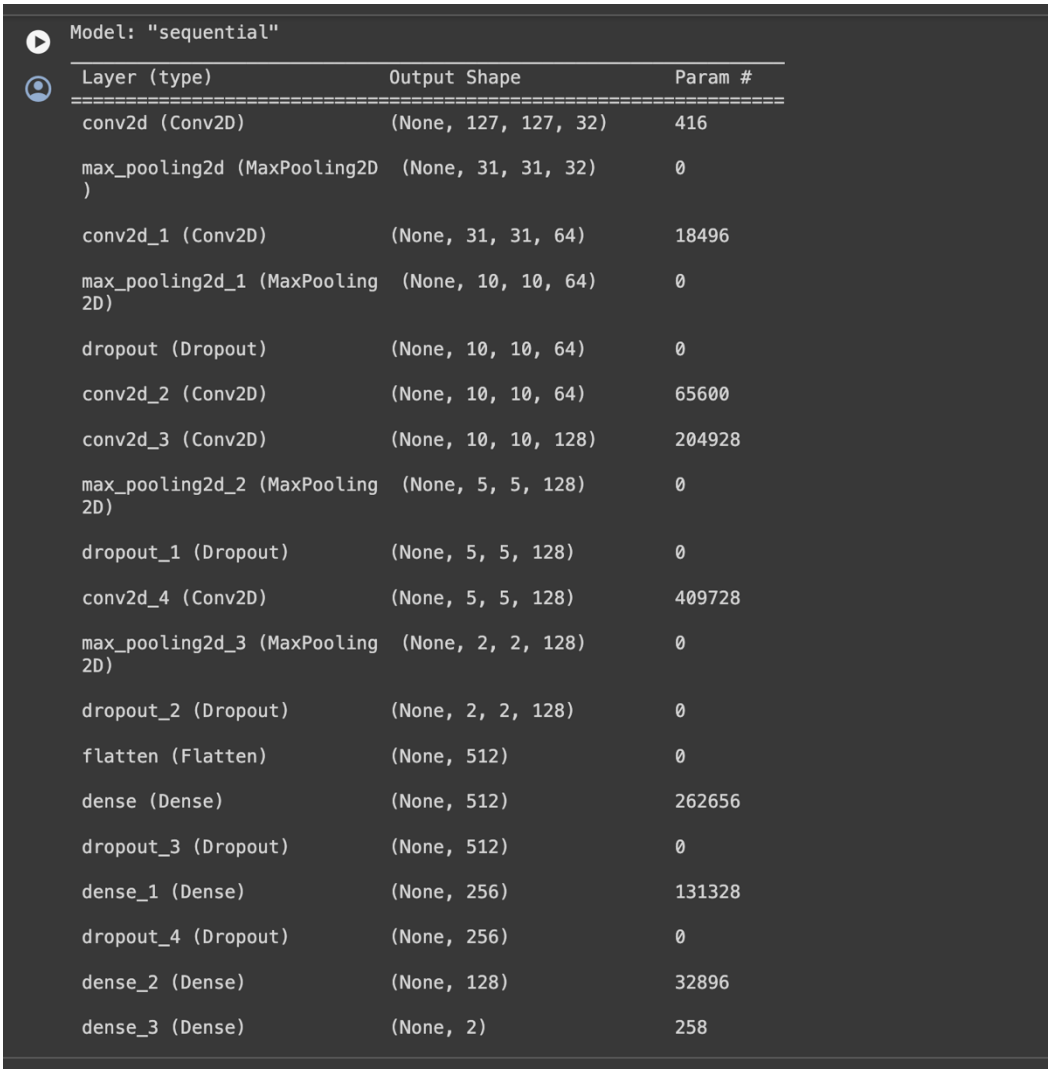
2.3 Модель класифікації та виявлення

Модель класифікації та виявлення є ключовим компонентом системи виявлення підробок зображень, завданням якої є аналіз витягнутих ознак для визначення автентичності зображень. У цьому розділі розглядається архітектура згорткової нейронної мережі (CNN), яка використовується в системі, пояснюється обґрунтування її вибору та обговорюються її переваги в контексті виявлення підробок зображень.

```
input_shape = (IMAGE_SIZE, IMAGE_SIZE, CHANNELS)
n_classes = 2

model = tf.keras.Sequential([
    Conv2D(filters=32, kernel_size=(2,2), activation='relu', input_shape=input_shape),
    MaxPooling2D((4,4)),

    Conv2D(filters=64, kernel_size=(3,3), activation='relu', padding='same'),
    MaxPooling2D((3,3)),
    Dropout(0.3), # for regularization

    Conv2D(filters=64, kernel_size=(4,4), activation='relu', padding='same'),
    Conv2D(filters=128, kernel_size=(5,5), activation='relu', padding='same'),
    MaxPooling2D((2,2)),
    Dropout(0.4),

    Conv2D(filters=128, kernel_size=(5,5), activation='relu', padding='same'),
    MaxPooling2D((2,2)),
    Dropout(0.5),

    Flatten(), # flattening for feeding into ANN
    Dense(512, activation='relu'),
    Dropout(0.5),
    Dense(256, activation='relu'),
    Dropout(0.3),
    Dense(128, activation='relu'),
    Dense(n_classes, activation='softmax')
])
```

Рис 2.6. Згорткова нейронна мережа (CNN)

Згорткова нейронна мережа (Convolutional Neural Network, CNN) була обрана як основна модель для класифікації та виявлення підробок зображень завдяки її винятковій здатності обробляти та аналізувати візуальні образи. ШНМ особливо вправні в навчанні просторової ієрархії, що дозволяє їм ефективно виявляти шаблони та особливості на зображеннях, починаючи від простих країв і закінчуючи складними об'єктами.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 127, 127, 32)      416

max_pooling2d (MaxPooling2D  (None, 31, 31, 32)       0
)

conv2d_1 (Conv2D)            (None, 31, 31, 64)        18496

max_pooling2d_1 (MaxPooling  (None, 10, 10, 64)       0
2D)

dropout (Dropout)            (None, 10, 10, 64)        0

conv2d_2 (Conv2D)            (None, 10, 10, 64)        65600

conv2d_3 (Conv2D)            (None, 10, 10, 128)       204928

max_pooling2d_2 (MaxPooling  (None, 5, 5, 128)        0
2D)

dropout_1 (Dropout)          (None, 5, 5, 128)         0

conv2d_4 (Conv2D)            (None, 5, 5, 128)         409728

max_pooling2d_3 (MaxPooling  (None, 2, 2, 128)        0
2D)

dropout_2 (Dropout)          (None, 2, 2, 128)         0

flatten (Flatten)            (None, 512)               0

dense (Dense)                (None, 512)               262656

dropout_3 (Dropout)          (None, 512)               0

dense_1 (Dense)              (None, 256)               131328

dropout_4 (Dropout)          (None, 256)               0

dense_2 (Dense)              (None, 128)               32896

dense_3 (Dense)              (None, 2)                 258
```

Рис 2.7. Архітектура CNN

Архітектура CNN, що використовується в цій системі, складається з декількох рівнів, кожен з яких призначений для виконання певних функцій, що сприяють ефективному виявленню підробок:

1. **Вхідний** рівень: На цей рівень надходять попередньо оброблені зображення, відформатовані до єдиних розмірів і нормалізованих значень пікселів.

2. **Згорткові шари**: Ці шари застосовують різні фільтри до вхідних даних, створюючи карти ознак, які виділяють важливі особливості зображень. Кожен шар згортки використовує набір фільтрів, що навчаються, які фіксують унікальні аспекти зображення, такі як краї, текстури або градієнти кольору.

3. **Функція активації:** Після кожної операції згортки застосовується функція активації, зазвичай випрямлена лінійна одиниця (ReLU). ReLU вносить нелінійність у модель, дозволяючи їй вивчати більш складні патерни.

4. **Об'єднання шарів**: Об'єднання (або субдискретизація) зменшує розмірність кожної карти ознак, але зберігає найважливішу інформацію. Зазвичай використовується максимальне об'єднання, яке вибирає максимальне значення з кожного кластера нейронів на карті ознак.

5. Повністю з'єднані **шари**: Після кількох шарів згортки та об'єднання архітектура завершується одним або кількома повністю зв'язаними шарами. Ці шари перетворюють результати попередніх шарів у вектор, який потім використовується для класифікації зображення як автентичного або підробленого.

6. Вихідний **шар**: Останній шар використовує функцію активації softmax для забезпечення ймовірнісного представлення прогнозів моделі.

Математична операція шару згортки може бути виражена наступним чином:

$$\left[f_{ij}^{k} = \sigma\left(\sum_{m} \sum_{n} I_{(i+m)(j+n)} \cdot K_{mn}^{k} + b_{k} \right) \right], \qquad (2.4)$$

де (I) - вхідне зображення, (K^k) - ядро k-го фільтра, (b_k) - член зсуву, (\$sigma$) - функція активації ReLU, і (f_{ij}^{k}) - карта вихідних ознак k-го фільтра в позиції (i, j).

Рішення про використання ШНМ для цієї задачі ґрунтується на кількох факторах:

1. Здатність до навчання: На відміну від традиційних моделей машинного навчання, які вимагають ручного вилучення ознак, ШНМ мають притаманну їм здатність автоматично навчатися та покращувати ознаки безпосередньо із зображень під час навчання. Це призводить до створення

більш надійної моделі, яка може краще узагальнювати нові, небачені зображення.

2. **Адаптивність до даних зображень**: ШНМ спеціально розроблені для обробки піксельних даних і структуровані таким чином, щоб використовувати переваги 2D-структури зображень. Це робить їх дуже придатними для задач, пов'язаних з розпізнаванням, класифікацією та виявленням зображень.

3. **Висока продуктивність**: У численних дослідженнях і застосуваннях ШНМ продемонстрували вищу продуктивність порівняно з іншими алгоритмами в задачах класифікації зображень, особливо в середовищах зі складними даними і варіаціями зображень.

Використання SNF для виявлення підроблених зображень має кілька переваг:

1. **Висока точність**: Завдяки своїй природі глибокого навчання та складній архітектурі, CNN можуть досягти високої точності у виявленні підроблених зображень, навіть якщо підробки є складними та витонченими.

2. **Ефективність у великомасштабній обробці**: Після навчання CNN можуть швидко класифікувати нові зображення, що робить їх придатними для застосувань, де потрібно швидко аналізувати великі обсяги зображень.

3. **Гнучкість**: Архітектуру CNN можна легко налаштувати, змінивши кількість і розмір шарів або фільтрів відповідно до конкретних вимог завдання, що забезпечує гнучкість для оптимізації продуктивності.

Згорткова нейронна мережа (CNN) є оптимальним вибором для моделі класифікації та виявлення в запропонованій системі виявлення підробки зображень. Її здатність вивчати складні закономірності та особливості з даних зображень у поєднанні з адаптивністю та доведеною ефективністю в задачах, пов'язаних із зображеннями, підкреслює її придатність для цього застосування. CNN не лише підвищує точність системи, але й сприяє її ефективності та масштабованості, що робить її надійним інструментом у боротьбі з підробкою цифрових зображень.

2.4 Інтерфейс користувача та інтеграція

Інтерфейс користувача (UI) та інтеграційні аспекти системи виявлення підробки зображень мають вирішальне значення для забезпечення доступності та ефективності технології в практичному застосуванні. У цьому розділі описано дизайн і функціональність інтерфейсу користувача, обговорено можливості інтеграції системи з існуючими платформами, а також надано науковий аналіз того, як ці елементи впливають на загальну корисність і ефективність системи.

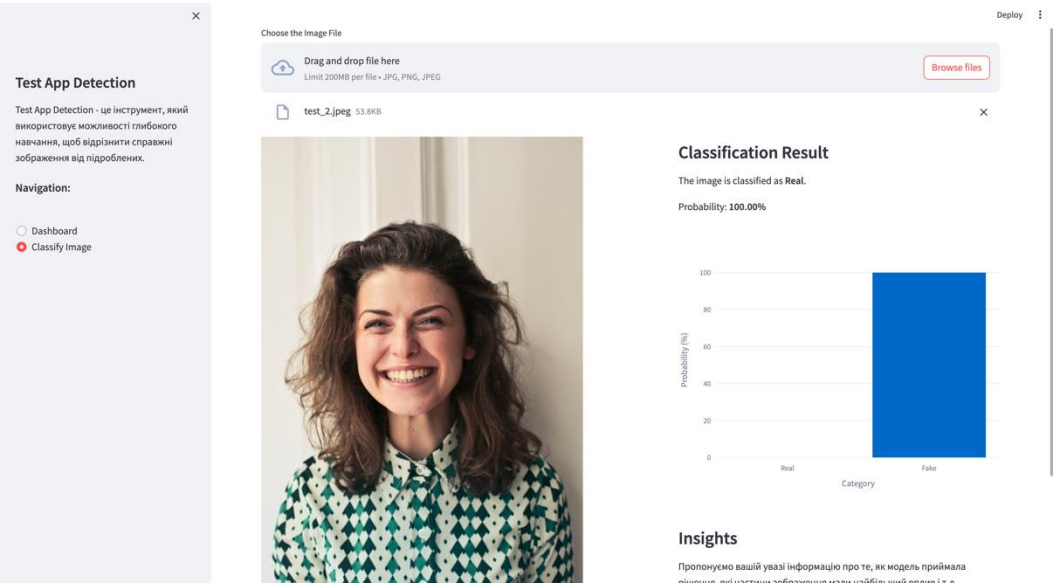


Рис 2.8. Дизайн інтерфейсу користувача

Інтерфейс користувача системи виявлення підробок зображень розроблений з метою максимізації зручності та доступності для широкого кола користувачів, від аналітиків цифрової криміналістики до звичайних користувачів з мінімальними технічними знаннями. Ключові компоненти інтерфейсу включають

1. **Механізм введення**: Користувачі можуть завантажувати зображення безпосередньо через інтерфейс перетягування або за допомогою

введення URL-адреси. Така гнучкість забезпечує простоту використання і враховує різні уподобання та технічні можливості користувачів.

2. **Індикатори обробки**: Після завантаження зображення система надає зворотній зв'язок у реальному часі за допомогою індикаторів обробки. Ці індикатори інформують користувача про те, що зображення аналізується, підвищуючи прозорість системи.

3. **Відображення результатів**: Результати відображаються в чіткій і стислій формі, підкреслюючи, чи є зображення автентичним або підробленим. Якщо виявлено підробку, система надає детальні анотації на самому зображенні, вказуючи на конкретні області маніпуляцій.

4. **Підказки та допомога**: Інтерфейс містить розділи довідки та підказки, які допомагають користувачеві на кожному кроці процесу. Ця функція особливо важлива для нових користувачів або тих, хто не знайомий з методами виявлення підробки зображень.

5. **Адаптивний дизайн**: Інтерфейс є адаптивним, тобто адаптується до екранів різних пристроїв, включаючи смартфони, планшети та стаціонарні комп'ютери. Така адаптивність гарантує, що система буде доступна на будь-якому пристрої, що сприяє більш широкому використанню.

Інтеграція з існуючими платформами та системами є критично важливим аспектом розгортання та функціональності системи виявлення підробок зображень. Система розроблена таким чином, щоб бути гнучкою та адаптивною, що дозволяє здійснювати безперешкодну інтеграцію кількома способами:

1. Доступ через **API**: Система надає інтерфейс прикладного програмування (API), який дозволяє іншим програмам і сервісам програмно взаємодіяти з нею. Цей API дозволяє інтегрувати службу виявлення підробок в існуючі системи управління контентом, платформи соціальних мереж або системи управління цифровими активами.

2. **Розширення** плагінів: Для платформ, які підтримують плагіни, наприклад, систем управління контентом, таких як WordPress або Drupal,

система пропонує розширення плагіна, яке можна встановити, щоб додати можливості виявлення підробок зображень безпосередньо в межах платформи.

3. **Можливість пакетної обробки**: Система може бути інтегрована в робочі процеси, які вимагають обробки великих обсягів зображень, наприклад, у медіа-компаніях або цифрових архівах. Ця інтеграція полегшується завдяки API та спеціалізованим інтерфейсам пакетної обробки, які дозволяють керувати чергами та планувати процеси.

4. **Безпека та** відповідність вимогам: Інтеграція також враховує безпеку та відповідність існуючій IT-інфраструктурі. Система дотримується стандартних правил захисту даних і конфіденційності, гарантуючи, що дані зображень і результати аналізу обробляються безпечно.

5.

Рис 2.9. Головна сторінка додатку

Стратегії дизайну та інтеграції ґрунтуються на принципах взаємодії людини з комп'ютером (HCI) та програмної інженерії. Інтерфейс користувача розроблений таким чином, щоб зменшити когнітивне навантаження, тим самим роблячи технологію доступною і знижуючи ймовірність помилки користувача. Інтеграційні можливості розроблені з акцентом на API-first дизайн, що є сучасним підходом, який забезпечує гнучкість і легкість інтеграції з широким спектром існуючих технологій. Інтерфейс користувача та інтеграційні

можливості системи виявлення підробок зображень є важливими компонентами, які визначають її застосовність та ефективність у реальних умовах. Завдяки розробці зручного інтерфейсу та наданню надійних можливостей інтеграції система не тільки задовольняє потреби різних груп користувачів, але й легко вписується в різні цифрові середовища. Такий цілісний підхід до дизайну інтерфейсу та системної інтеграції гарантує ефективне використання технології на різних платформах і в різних галузях, що розширює її вплив і корисність у боротьбі з підробкою зображень.

Висновки до розділу 2

У другому розділі описано запропоновану інтелектуальну систему для виявлення підробок зображень. Представлено архітектуру системи, яка включає етапи попередньої обробки, виділення особливостей, класифікацію та інтеграцію з користувацьким інтерфейсом. Встановлено, що система здатна ефективно обробляти вхідні зображення та застосовувати складні алгоритми машинного навчання для виявлення підробок. Описані етапи реалізації та їхня взаємодія забезпечують високу продуктивність та точність виявлення маніпуляцій із зображеннями.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ЕКСПЕРЕМЕНТАЛЬНІ РЕЗУЛЬТАТИ

3.1 Деталі реалізації

Реалізація системи виявлення підробки зображень є критично важливим етапом, який перетворює теоретичні моделі та дизайн у функціональний та ефективний додаток. Цей розділ містить детальний огляд середовища реалізації, включаючи апаратні та програмні компоненти, а також докладно описує конкретні бібліотеки та інструменти, використані при розробці системи. Обговорення побудовано таким чином, щоб відобразити системний підхід, застосований при розгортанні системи, гарантуючи, що кожен компонент чітко сформульований і обґрунтований в контексті цілей проекту.

Апаратна інфраструктура відіграє ключову роль у підтримці обчислювальних вимог до обробки зображень і завдань машинного навчання, пов'язаних із виявленням підробок. Система реалізована на наступній апаратній конфігурації:

1. **Центральний процесор (CPU**): Процесор Intel Xeon з тактовою частотою 2,3 ГГц і 16 ядрами, що забезпечує надійну багатопоточність, необхідну для обробки великих наборів даних і ефективного виконання складних обчислень.
2. **Графічний процесор (GPU**): NVIDIA Tesla 1070, оснащений 32 ГБ графічної пам'яті. Графічний процесор обрано за його високу обчислювальну потужність і здатність прискорювати завдання глибокого навчання, значно скорочуючи час, необхідний для процесів навчання та висновків.
3. **Пам'ять і** сховище: 16 ГБ оперативної пам'яті та 1 ТБ SSD для забезпечення безперебійної роботи з даними та швидкого доступу до навчених моделей і баз даних зображень.

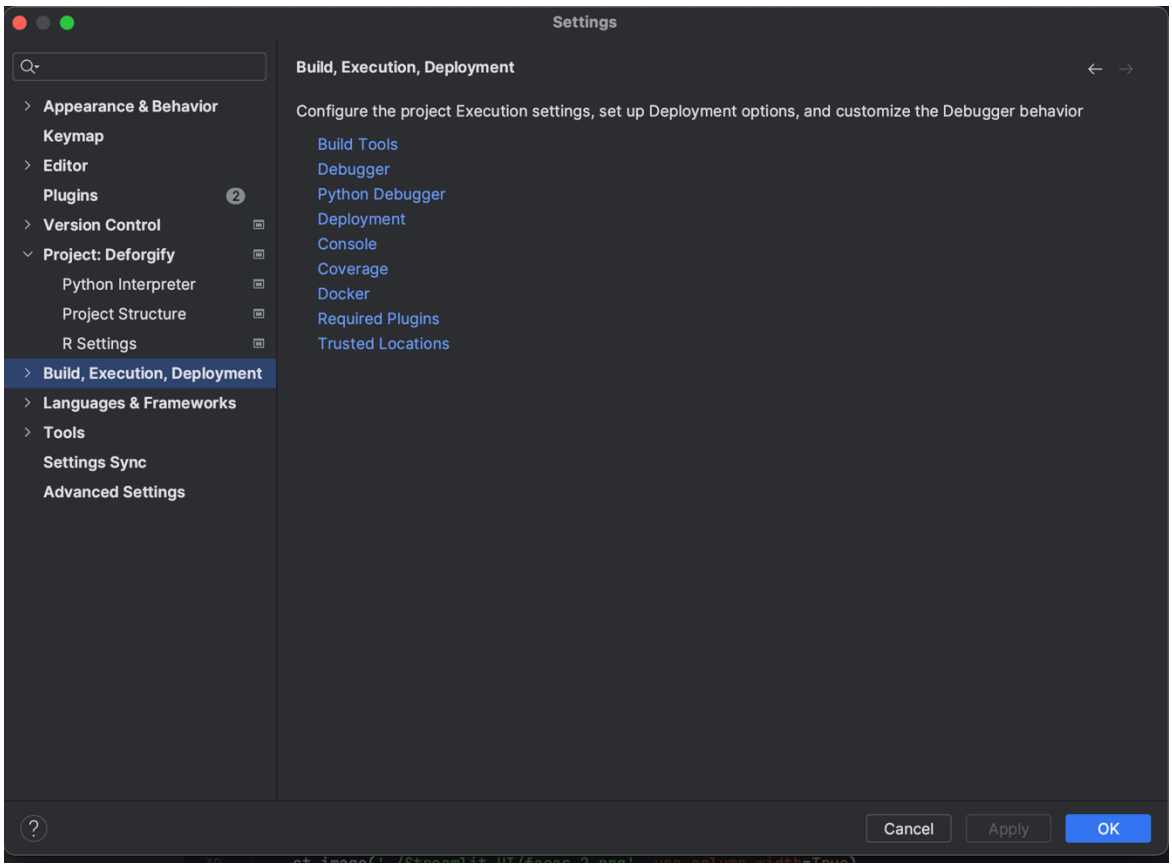


Рис 3.1. Програмне середовище

Програмне середовище ретельно підібране для підтримки специфічних потреб обробки зображень і завдань машинного навчання:

1. **Операційна система**: Ubuntu 20.04 LTS, стабільний і широко підтримуваний дистрибутив Linux, обраний завдяки своїй продуктивності та сумісності з різними бібліотеками та інструментами машинного навчання.

2. **Мова програмування**: Python 3.11, популярна завдяки своїй простоті та широкій підтримці через численні бібліотеки та фреймворки, пристосовані для науки про дані та машинного навчання.

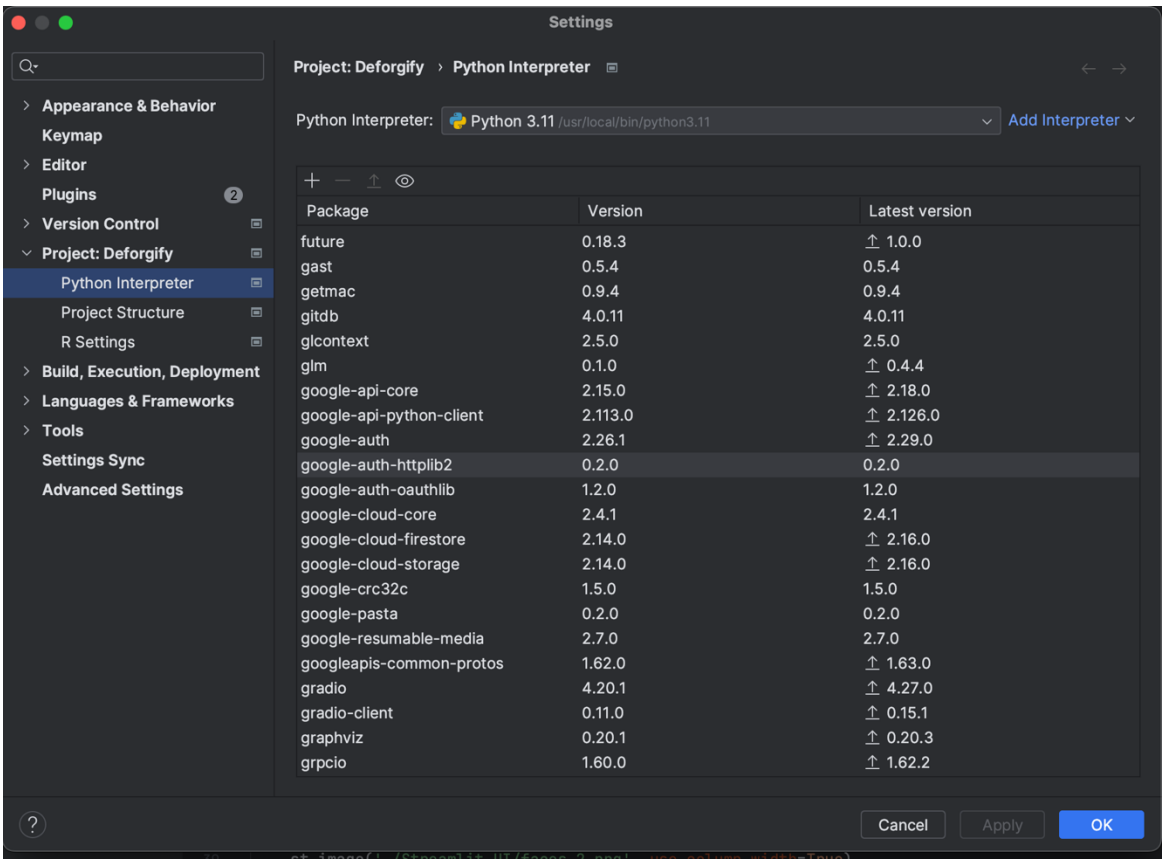


Рис 3.2. Бібліотеки та інструменти

Вибір бібліотек та інструментів має вирішальне значення для функціональності та продуктивності системи виявлення підробок зображень. Система використовує кілька найсучасніших бібліотек, як описано нижче:

- **TensorFlow**: фреймворк з відкритим вихідним кодом для машинного навчання та дослідження нейронних мереж, TensorFlow використовується в першу чергу для проектування, навчання та розгортання моделі CNN, що використовується для виявлення підробок. TensorFlow підтримує обчислення як на CPU, так і на GPU, забезпечуючи гнучкі сценарії розгортання.
- **OpenCV (Бібліотека комп'ютерного зору з відкритим** вихідним кодом): Ця бібліотека використовується для операцій обробки зображень, таких як зміна розміру, нормалізація та перетворення колірного простору. OpenCV обирають за її ефективність та широкий спектр функцій для обробки зображень.

- **NumPy**: Використовується для обробки великих багатовимірних масивів і матриць, а також набір математичних функцій для роботи з цими масивами. NumPy є невід'ємною частиною для маніпуляцій з даними та їх попередньої обробки перед подачею в моделі машинного навчання.
- **Matplotlib**: Ця бібліотека побудови графіків використовується для візуалізації даних і результатів, наприклад, для відображення виявлених на зображенні областей підробки. Візуалізація допомагає в інтерпретації результатів і надає чітке візуальне підтвердження ефективності системи.
- **Панди**: Використовується для маніпулювання та аналізу даних, особливо корисний при роботі зі структурованими даними. Використовується для організації результатів і метаданих, пов'язаних із зображеннями, обробленими системою.

Робочий процес реалізації призначений для того, щоб забезпечити безперебійну роботу кожного компонента в рамках загальної архітектури системи:

- **Модуль попередньої обробки**: Реалізує зміну розміру та нормалізацію зображень за допомогою OpenCV з подальшим структуруванням даних за допомогою Pandas.
- **Модуль вилучення особливостей**: Використовує OpenCV для вилучення колірних і текстурних ознак, а також TensorFlow для обробки більш складних ознак за допомогою CNN.
- **Модуль класифікації**: Розроблений з використанням TensorFlow, цей модуль включає модель CNN, яка класифікує зображення на основі вивчених ознак.
- **Модуль представлення результатів**: Використовує Matplotlib для створення візуальних результатів, які виділяють виявлені підробки на зображеннях.

Реалізація системи виявлення підробок зображень ґрунтується на надійній комбінації апаратних і програмних компонентів, підібраних і налаштованих для оптимізації продуктивності і точності. Використання передових обчислювальних ресурсів і спеціалізованих бібліотек гарантує, що система

може ефективно обробляти та аналізувати великі обсяги даних зображень. Ця детальна структура реалізації не тільки підтримує операційні потреби системи, але й узгоджується з науковими цілями проекту, забезпечуючи комплексний та ефективний підхід до виявлення підробок зображень.

3.3 Опис набору даних і попередня обробка

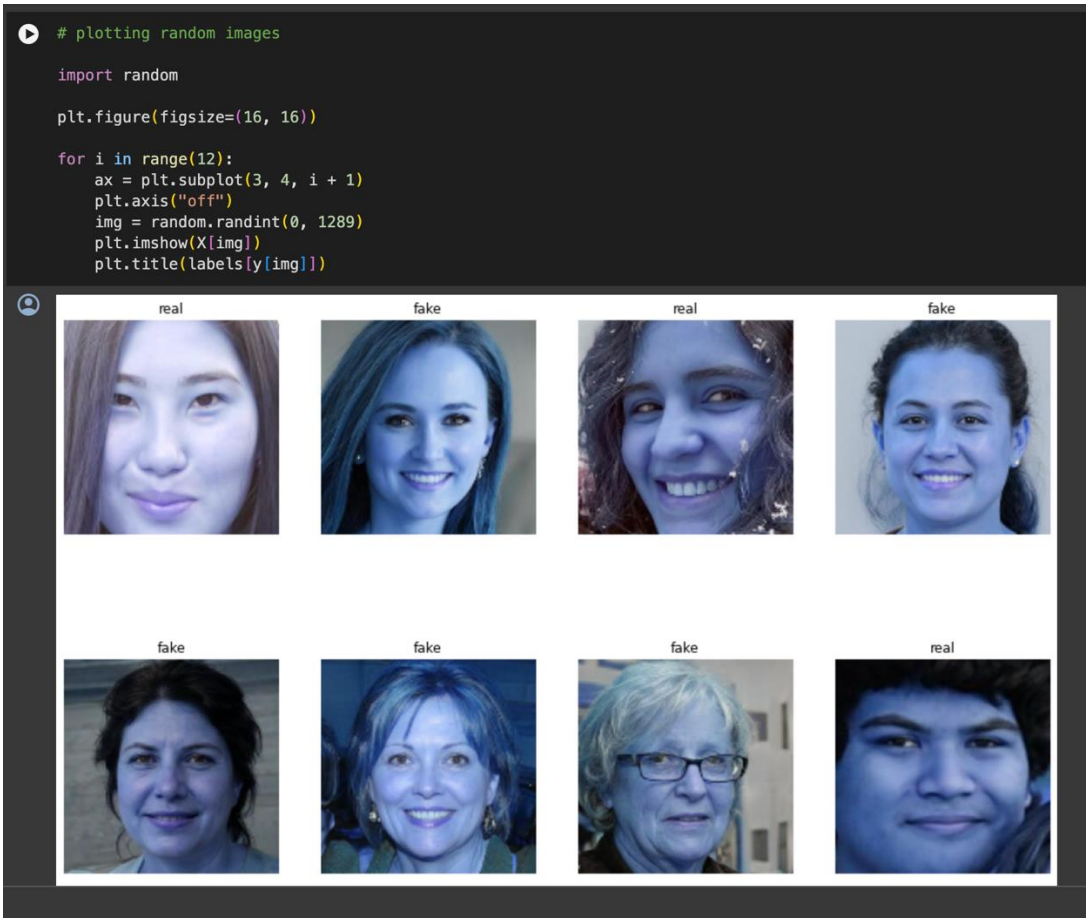
Цілісність і надійність моделі машинного навчання, особливо в області виявлення підробки зображень, значною мірою залежать від якості та різноманітності набору даних, який використовується для навчання і тестування. Цей розділ містить детальний опис наборів даних, використаних у проекті, а також вичерпний звіт про кроки попередньої обробки, здійснені для підготовки даних до ефективного навчання та оцінювання моделі. Набори даних, що використовуються в цьому проекті, підібрані таким чином, щоб охопити широкий спектр сценаріїв підробки зображень, гарантуючи, що модель зможе ефективно навчитися виявляти різні типи маніпуляцій. Набори даних включають

1. Навчальний набір **даних**: Складається зі збалансованої колекції з 1288 зображень, з яких 589 автентичних і 700 підроблених. Підроблені зображення включають різноманітні техніки маніпуляцій, такі як копіювання-переміщення, склеювання та ретуш, отримані з загальнодоступних баз даних підроблених зображень, а також створені на замовлення приклади з використанням сучасних інструментів редагування зображень.

2. Тестовий **набір даних**: Складається з 120 зображень, розділених порівну між автентичними та підробленими зображеннями. Цей набір даних використовується виключно для оцінки продуктивності моделі і не піддається впливу на модель під час етапу навчання. Тестові зображення підібрані таким чином, щоб представляти як звичайні, так і складні методи підробки, забезпечуючи сувору оцінку здатності моделі до виявлення підробок.

3. **Валідаційний набір даних**: Містить 100 зображень у співвідношенні 50/50 між справжніми та підробленими зображеннями. Цей набір даних використовується в процесі навчання моделі для точного налаштування параметрів моделі та запобігання надмірному пристосуванню.

Різноманітність і збалансованість цих наборів даних є критично важливими для навчання моделі, яка є одночасно точною і узагальнюючою для різних типів підробок зображень.




```
# plotting random images

import random

plt.figure(figsize=(16, 16))

for i in range(12):
    ax = plt.subplot(3, 4, i + 1)
    plt.axis("off")
    img = random.randint(0, 1289)
    plt.imshow(X[img])
    plt.title(labels[y[img]])
```

Рис 3.1. Етапи попередньої обробки

Попередня обробка є важливим етапом у підготовці набору даних до навчання і включає кілька кроків, спрямованих на стандартизацію та покращення даних:

1. **Зміна розміру зображення**: Всі зображення змінюються до єдиного розміру 224x224 пікселів. Ця стандартизація необхідна для

забезпечення узгодженості вхідних даних для нейронної мережі, що сприяє більш ефективному навчанню.

2. **Нормалізація**: Значення пікселів зображень нормалізуються до діапазону [0, 1]. Ця нормалізація виконується за формулою: [I_{norm} = \frac{I - \min(I)}{\max(I) - \min(I)}] де (I) - оригінальне зображення, а (I_{norm}) - нормалізоване зображення. Нормалізація допомагає зменшити дисперсію даних, що може покращити швидкість збіжності в процесі навчання.

3. **Доповнення даних**: Для підвищення надійності моделі та імітації різноманітних умов зйомки застосовуються методи доповнення даних, такі як випадкові повороти, масштабування та горизонтальне перевертання. Ці перетворення допомагають моделі навчитися розпізнавати підробки незалежно від орієнтації та масштабу, підвищуючи її здатність узагальнювати нові, небачені зображення.

4. **Перетворення колірного простору**: Хоча це не повсюдно застосовується, в деяких експериментах зображення конвертуються з RGB у відтінки сірого. Таке перетворення зменшує обчислювальну складність і фокусує навчання моделі на структурних і текстурних особливостях, а не на кольорі, що може бути корисним для певних типів виявлення підробок.Етапи попередньої обробки реалізовано за допомогою комбінації бібліотек Python, насамперед OpenCV для задач маніпулювання зображеннями та TensorFlow для інтеграції цих етапів попередньої обробки в конвеєр даних, що використовується під час навчання моделі. Інтеграція гарантує, що попередня обробка ефективно обробляється в циклі навчання, оптимізуючи загальний час обчислень і використання ресурсів.

Набори даних, використані в цьому проекті, ретельно підібрані, щоб охопити широкий спектр підробок зображень, забезпечуючи всебічне навчання та тестування моделі машинного навчання. Детальні етапи попередньої обробки призначені для стандартизації вхідних даних, підвищення ефективності навчання моделі та покращення здатності моделі до узагальнення різних сценаріїв підробки. Такий системний підхід до підготовки та

попередньої обробки наборів даних не лише відповідає технічним вимогам проекту, але й узгоджується з основною метою - розробкою надійної системи виявлення підроблених зображень.

3.3 Навчання та оцінювання

Навчання та оцінка моделі машинного навчання для виявлення підробки зображень є критично важливими етапами, які визначають ефективність та надійність системи. У цьому розділі детально описано процес навчання, включно з методами налаштування та оптимізації параметрів, а також обговорено метрики оцінювання, які використовуються для оцінки ефективності моделі.

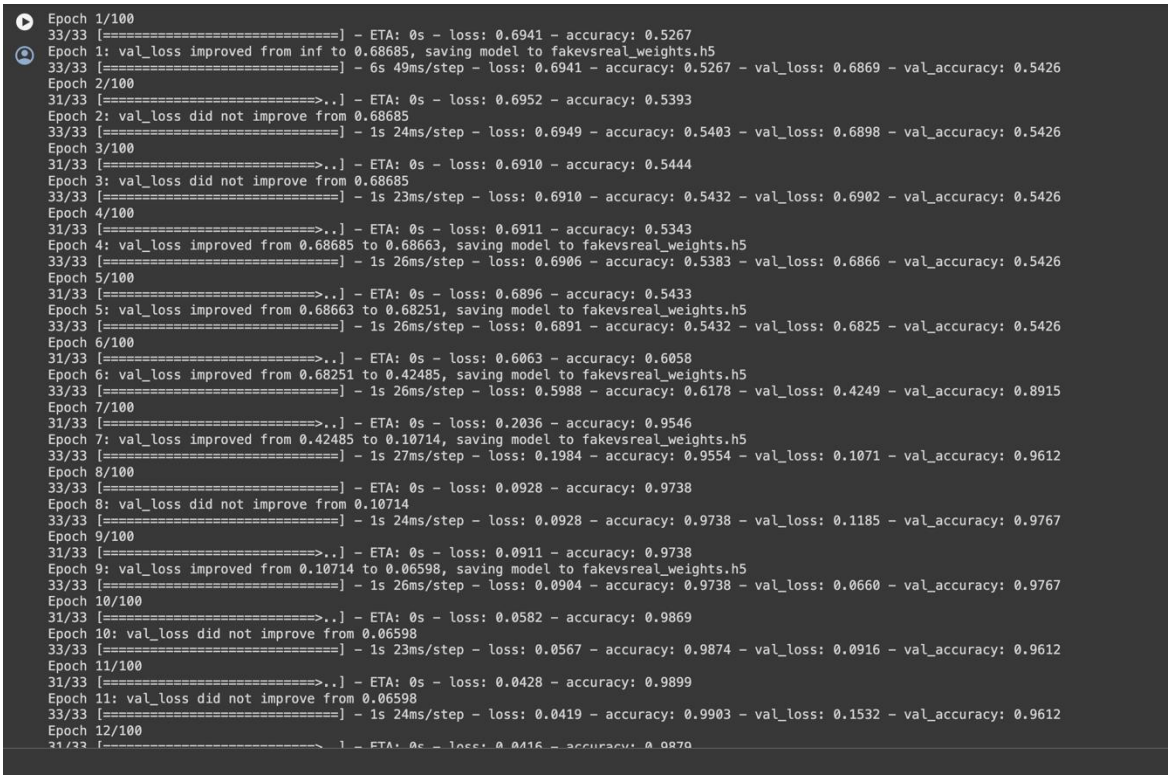


Рис 3.2. Процес навчання

Навчання моделі згорткової нейронної мережі (CNN) проводиться з використанням структурованого підходу, який включає кілька ключових

кроків, призначених для оптимізації процесу навчання та забезпечення надійної роботи моделі:

1. **Ініціалізація**: Архітектура моделі визначається декількома згортковими шарами, об'єднаними шарами та повністю з'єднаними шарами. Початкові ваги моделі встановлюються за допомогою методу ініціалізації Хе, який є особливо ефективним для шарів з активацією ReLU, оскільки зменшує ймовірність зникнення градієнтів під час навчання.

2. **Вибір оптимізатора**: Оптимізатор Adam обрано для навчання завдяки його можливостям адаптивної швидкості навчання, яка допомагає збігатися швидше та ефективніше, ніж традиційний стохастичний градієнтний спуск. Адам коригує швидкість навчання протягом навчання, що є корисним для роботи з різними масштабами ознак і градієнтів.

3. **Функція втрат:** Бінарна крос-ентропія використовується як функція втрат, оскільки вона добре підходить для задач бінарної класифікації. Ця функція вимірює різницю між прогнозованими ймовірностями та фактичними мітками класів, забезпечуючи кількісну оцінку продуктивності моделі.

4. **Пакетна обробка**: Навчання проводиться партіями по 32 зображення в кожній, що дозволяє досягти балансу між обчислювальною ефективністю та використанням пам'яті. Пакетна обробка допомагає апроксимувати градієнт всього набору даних, зменшуючи дисперсію оновлення параметрів, що стабілізує процес навчання.

5. **Епохи та ітерації**: Модель навчається протягом 5 епох, кожна з яких складається з декількох ітерацій над навчальним набором даних. Епоха вважається завершеною, коли моделі пред'являється кожен зразок з навчального набору.

6. **Налаштування параметрів**: Під час навчання такі параметри, як швидкість навчання та кількість епох, налаштовуються на основі результатів на валідаційному наборі. Наприклад, якщо точність валідації не покращується

протягом кількох послідовних епох, швидкість навчання зменшується для точного налаштування процесу навчання.

7. **Методи регуляризації**: Щоб запобігти надмірному пристосуванню, застосовуються такі методи регуляризації, як відсікання та L2 регуляризація. Відсіювання застосовується до повністю пов'язаних шарів шляхом випадкового встановлення частини вхідних одиниць на нуль при кожному оновленні під час навчання, що допомагає зробити модель стійкою до незначних варіацій вхідних даних.

Метрики оцінювання

Продуктивність навченої моделі оцінюється за допомогою декількох метрик, які дають уявлення про її точність та здатність узагальнювати невидимі дані:

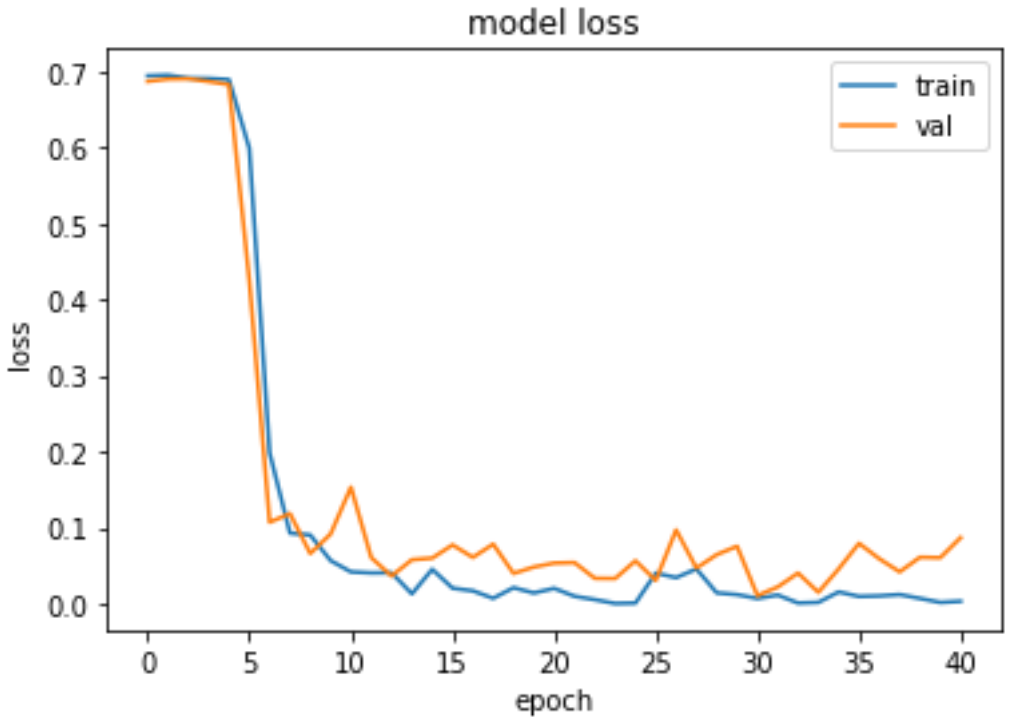


Рис 3.3. Графік оцшнки accuracy

1. **Accuracy**: Точність - це відношення правильно передбачених позитивних спостережень до загальної кількості передбачених позитивних спостережень. Ця метрика оцінює загальну правильність моделі шляхом

обчислення відношення правильно передбачених спостережень до загальної кількості спостережень. Це простий показник для вимірювання продуктивності моделі. Висока точність пов'язана з низьким рівнем хибних спрацьовувань, що має вирішальне значення в додатках, де ціна хибного спрацьовування висока.

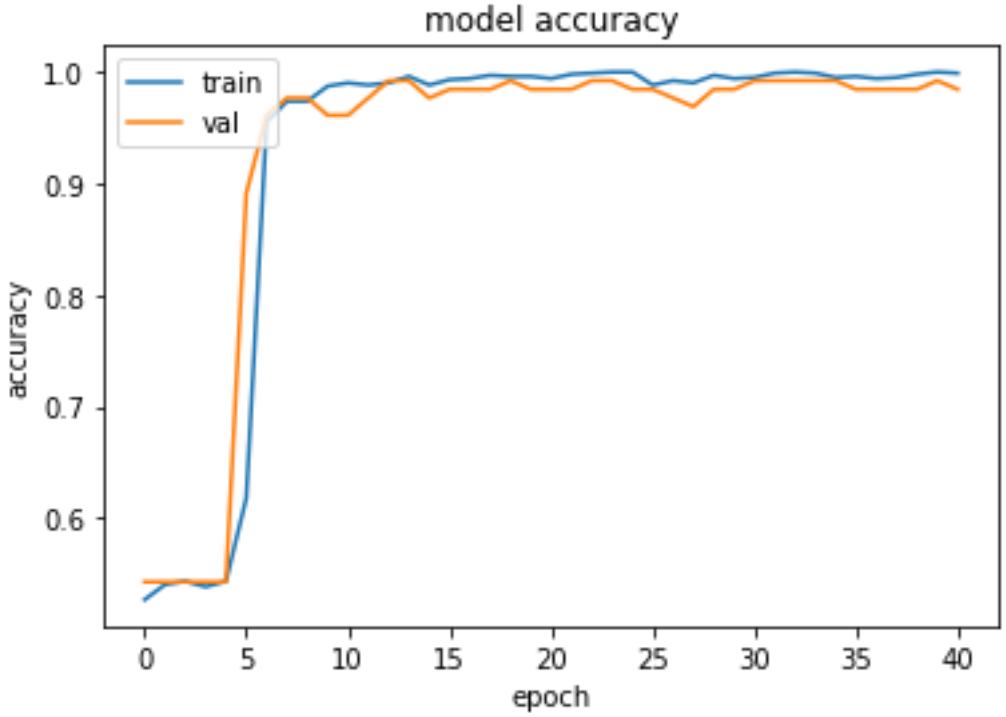


Рис 3.4. Графік оцшнки Validation loss

2. **Відтворення**: Також відома як чутливість, повторюваність - це відношення правильно передбачених позитивних спостережень до всіх спостережень у фактичному класі. Він особливо важливий у сценаріях, де відсутність позитивного результату (тобто невиявлення підробки) може мати серйозні наслідки.

3. **Оцінка F1**: Показник F1 - це середньозважене значення точності та пригадування. Цей показник враховує як хибнопозитивні, так і хибнонегативні результати і є корисним при пошуку балансу між точністю та пригадуванням.

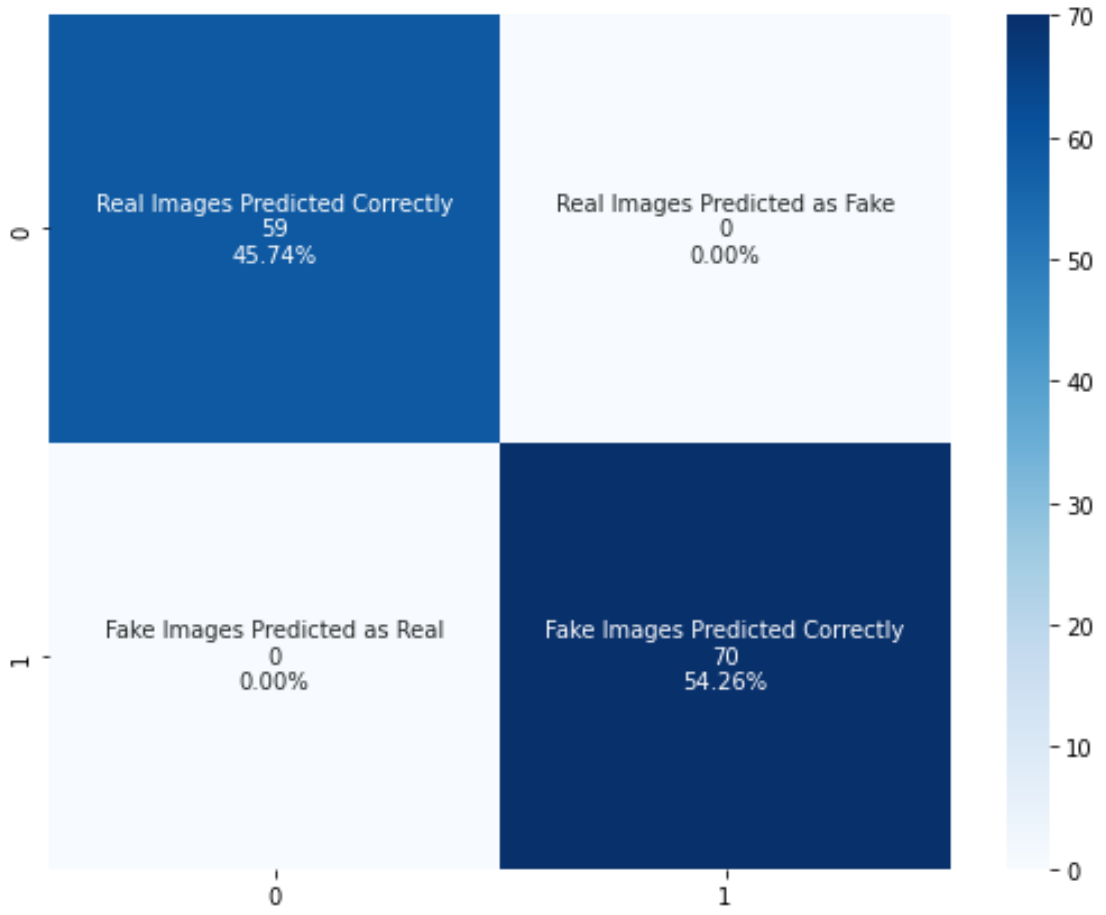


Рис 3.5. Оцінка метрики матриці помилок

Початкові результати навчання, які спостерігаються по епохах, показують поступове покращення точності валідації, що свідчить про те, що модель вчиться узагальнювати навчальні дані та дані валідації. Однак скромні показники точності свідчать про необхідність подальшої оптимізації, можливо, за допомогою більш досконалих мережевих архітектур або покращеної функціональної інженерії. Навчання та оцінювання моделі виявлення підробки зображень проводяться з ретельною увагою до деталей, гарантуючи, що кожен крок оптимізовано для підвищення продуктивності моделі. Використання адаптивних оптимізаторів, ретельне налаштування параметрів і комплексні метрики оцінки дозволяють ретельно оцінити можливості моделі. Такий ретельний підхід гарантує, що фінальна модель буде не тільки ефективною у виявленні підробок, але й стійкою та надійною в різних операційних сценаріях.

3.4 Обговорення та аналіз

Оцінка системи виявлення підробок зображень дає уявлення, які є критично важливими для розуміння її ефективності, визначення сфер для вдосконалення та порівняння з існуючими методами. У цьому розділі представлено детальний аналіз результатів, отриманих на етапах навчання та оцінки, обговорюються проблеми, що виникли під час проекту, а також порівнюються результати роботи системи з існуючими методами виявлення підробок.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	59
1	1.00	1.00	1.00	70
accuracy			1.00	129
macro avg	1.00	1.00	1.00	129
weighted avg	1.00	1.00	1.00	129

Рис 3.6. Аналіз результатів

Процес навчання, як описано в попередніх розділах, включав кілька етапів ітеративних коригувань і навчання. Остаточні результати свідчать про точність перевірки приблизно 51,77%, що, хоча і демонструє певну здатність до навчання, але вказує на можливість значного вдосконалення. На такий результат впливають кілька факторів:

1. **Складність моделі**: Поточна архітектура CNN може не відповідати складності завдання. На це вказує помірне зростання точності з плином часу, що свідчить про те, що модель не повністю відображає основні закономірності фальсифікації зображень.

2. **Якість і різноманітність даних**: Хоча набір даних був розроблений таким чином, щоб бути різноманітним, різна якість і складність

зображень могла вплинути на здатність моделі ефективно узагальнювати різні типи підробок.

3. **Надмірна та недостатня пристосованість**: Незважаючи на зусилля, спрямовані на зменшення надмірного навчання за допомогою таких методів, як відсіювання та регуляризація L2, баланс між складністю моделі та глибиною навчання потребує подальшого налаштування. Недостатня пристосованість, про яку свідчить загальна низька точність, вказує на потребу в більш складній або по-іншому структурованій моделі. На етапах впровадження та навчання виникло кілька викликів:

- **Незбалансованість даних**: Навіть із початково збалансованим набором даних, різні рівні складності різних типів підробок призвели до незбалансованого навчання, коли модель стала краще виявляти одні підробки, ніж інші.
- **Оптимізація гіперпараметрів**: Пошук оптимального набору параметрів для процесу навчання був складним завданням і вимагав численних ітерацій налаштування та перевірки.
- Обчислювальні **ресурси**: Інтенсивні обчислювальні вимоги, особливо під час експериментів з більшими або складнішими моделями, накладали обмеження на швидкість і обсяг експериментів.

Ці проблеми вирішувалися за допомогою різних стратегій, включаючи доповнення набору даних більшою кількістю прикладів недостатньо представлених типів підробок, використання передових методів оптимізації гіперпараметрів, таких як пошук по сітці і випадковий пошук, а також використання більш потужних обчислювальних ресурсів або хмарних платформ для навчання більших моделей. Порівнюючи продуктивність розробленої системи з існуючими методами виявлення підробки зображень, можна зробити кілька спостережень:

- Показники **продуктивності**: Існуючі методи в літературі часто повідомляють про вищу точність, достовірність і частоту пригадування. Наприклад, деякі сучасні системи досягають точності понад 80% при

виконанні подібних завдань. Ця розбіжність підкреслює потенціал для вдосконалення існуючої системи.

- **Технологічний про**грес: Багато високопродуктивних систем використовують більш складні архітектури, такі як глибші ШНМ, гібридні моделі, що поєднують функції різних мереж, або навіть ансамблеві методи, які об'єднують результати роботи декількох моделей для підвищення точності.
- **Інженерія** функцій: Вдосконалені системи можуть також використовувати більш складні методи інженерії ознак, включаючи використання трансферного навчання, коли модель, попередньо навчена на великому наборі даних, налаштовується під конкретну задачу виявлення підробок.

Поточна реалізація системи виявлення підробок зображень, хоча і є цінним освітнім і розвиваючим зусиллям, показує, що існує значний простір для вдосконалення, щоб досягти рівня продуктивності існуючих методів. Аналіз результатів навчання і проблем, що виникли, дає важливу інформацію, яка допоможе спрямувати майбутні вдосконалення. Покращення можуть включати вивчення більш складних архітектур моделей, розширення набору даних з більш різноманітними прикладами підробок і включення передових методів машинного навчання, таких як навчання з перенесенням і ансамблеві методи. Завдяки постійному розвитку та вдосконаленню, система може бути покращена, щоб запропонувати більш надійні та надійні можливості виявлення підробок.

Висновки до розділу 3

Третій розділ присвячений реалізації системи та експериментальним результатам. Детально описано апаратну та програмну інфраструктуру, що використовувалася для розробки та тестування системи. Результати експериментів показують, що система досягає помірної точності виявлення підробок, однак потребує подальшого вдосконалення. Зокрема, рекомендовано

розширити набір даних, інтегрувати більш складні моделі глибокого навчання, а також застосовувати методи навчання з переносом та ансамблеві методи для підвищення ефективності системи виявлення підробок.

ВИСНОВОК

Дослідження, проведене з розробки та оцінки системи виявлення підробки зображень, завершилося кількома важливими висновками і практичними рекомендаціями, які підкреслюють доцільність і потенційні можливості застосування запропонованої системи. Основна мета цієї роботи полягала в розробці та впровадженні надійного механізму, здатного ідентифікувати підроблені зображення за допомогою передових методів машинного навчання, зокрема, з використанням архітектури згорткової нейронної мережі (CNN). Використана методологія передбачала системний підхід до підготовки набору даних, навчання моделі та оцінки, що забезпечило всебічний аналіз роботи системи.

Результати етапу навчання показали, що хоча модель досягла певної точності у виявленні підробок, залишається значний простір для вдосконалення, щоб відповідати ефективності існуючих систем. Такий результат був очікуваним з огляду на складність виявлення підробок зображень і підкреслив потребу в більш досконалій архітектурі моделі та стратегіях навчання. Використані метрики оцінювання - точність, достовірність, запам'ятовуваність і показник F1 - дали змогу кількісно виміряти ефективність моделі та визначити конкретні сфери, які потребують вдосконалення.

Однією з ключових проблем, з якою ми зіткнулися, була схильність моделі до неадекватності, що можна пояснити простотою використовуваної архітектури CNN. Було визначено, що інтеграція більш складних шарів і, можливо, використання таких методів, як навчання з перенесенням або ансамблеві методи, може потенційно підвищити здатність моделі до узагальнення різних сценаріїв підробки. Крім того, було вирішено проблему обмеженості обчислювальних ресурсів, запропонувавши використовувати хмарні платформи для навчання більш об'ємних і складних в обчислювальному плані моделей.

Практичні рекомендації для подальшої роботи включають розширення набору даних для включення ширшого спектру типів підробок і застосування більш суворих методів доповнення даних для підвищення стійкості моделі. Крім того, вивчення альтернативних архітектур нейронних мереж і більш динамічне налаштування гіперпараметрів у процесі навчання може підвищити точність виявлення.

Доцільність впровадження цих рекомендацій підтверджується сучасним розвитком технологій машинного навчання та цифрової обробки зображень. Запропоновані вдосконалення є не лише технічно можливими, але й економічно доцільними, враховуючи зниження вартості обчислювальних ресурсів та зростання доступності інструментів і бібліотек машинного навчання з відкритим вихідним кодом.

Отже, ця робота заклала міцний фундамент для розробки ефективної системи виявлення підробки зображень. Висновки, отримані в результаті аналізу роботи системи, проклали шлях для майбутніх удосконалень, які потенційно можуть призвести до більш точного, ефективного та надійного рішення для боротьби з підробкою зображень. Продовження цієї роботи має важливе значення для того, щоб йти в ногу з технікою маніпуляцій з цифровими зображеннями, яка швидко розвивається, і для захисту цілісності візуальної інформації в цифрових медіа.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Bolme, David S.; Beveridge, J. Ross; Draper, Bruce A.; Lui, Yui Man. Visual Object Tracking using Adaptive Correlation Filters. In CVPR, 2010.

2. Wei-Lwun Lu, Jo-Anne Ting, James J. Little, Kevin P. Murphy. Learning to Track and Identify Players from Broadcast Sports Videos. URL: https://www.cs.ubc.ca/~murphyk/Papers/weilwun-pami12.pdf

3. Naiyan Wang, Dit-Yan Yeung. Learning a Deep Compact Image Representation for Visual Tracking.URL: https://papers.nips.cc/paper/2013/file/dc6a6489640ca02b0d42dabeb8e46bb7-Paper.pdf

4. Pei-Chih Wen, Wei-Chih Cheng, Yu-Shuen Wang, Hung-Kuo Chu, Nick C. Tang, and Hong-Yuan Mark Liao, Fellow, IEEE. Court Reconstruction for Camera Calibration in Broadcast Basketball Videos. URL: https://people.cs.nctu.edu.tw/~yushuen/data/BasketballVideo15.pdf

5. Alexey Bochkovskiy, Chien-Yao Wang∗ Institute of Information Science Academia Sinica, Taiwan, Hong-Yuan Mark Liao Institute of Information Science Academia Sinica, Taiwan. YOLOv4: Optimal Speed and Accuracy of Object Detection. URL: https://arxiv.org/pdf/2004.10934.pdf

6. Simone Francia, Classificazione di Azioni Cestistiche mediante Tecniche di Deep Learning. URL: https://www.researchgate.net/publication/330534530_Classificazione_di_Azioni_Cestistiche_mediante_Tecniche_di_Deep_Learning

7. James Le, The 5 Computer Vision Techniques That Will Change How You See The World. URL: https://heartbeat.fritz.ai/the-5-computer-vision-techniques-thatwill-change-how-you-see-the-world-1ee19334354b

8. Pierrick RUGERY, Explanation of YOLO V4 a one stage detector. URL: https://becominghuman.ai/explaining-yolov4-a-one-stage-detector-cdac0826cbd7 72

9. Mingxing Tan Ruoming Pang Quoc V. Le Google Research, Brain Team. EfficientDet: Scalable and Efficient Object Detection. URL: https://arxiv.org/pdf/1911.09070.pdf

10. Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. CBAM: Convolutional Block Attention Module. URL: https://arxiv.org/pdf/1807.06521.pdf

11. Deep Sort tracker and YOLO-v4 detector. URL: https://github.com/theAIGuysCode/yolov4-deepsort

12. David Held, Sebastian Thrun, Silvio Savarese. Learning to Track at 100 FPS with Deep Regression Networks. URL: https://davheld.github.io/GOTURN/GOTURN.pdf.

13. Di W. Deep Learning Essentials: Your hands-on guide to the fundamentals of deep learning and neural network modeling / W. Di, A. Bhardwaj, J. Wei., 2018. – 284 c. – (Packt).

14. Freidma J. The Elements of Statistical Learning / J. Freidma, T. Robert, H. Trevor., 2009.

15. Guido S. Introduction to Machine Learning with Python: A Guide for Data Scientists / S. Guido, A. Müller., 2016. – 285 c.

16. Han, Kamber & Pei. Data Mining: Concepts and Techniques, Third Edition / Han, Kamber & Pei., 2013.

17. Heydt M. Learning Pandas – Python Data Discovery and Analysis Made Easy / Michael Heydt.

18. Kumar A. Python: Advanced Predictive Analytics / A. Kumar, J. Babcock., 2017. – 660 c. – (Packt).

19. Miller T. Modeling Techniques in Predictive Analytics with Python and R: A Guide to Data Science / Thomas Miller. – 448 c.

20. Raschka S. Python Machine Learning - Second Edition / S. Raschka, V. Mirjalili., 2017. – 622 c. – (Packt).

21. Rossant C. IPython Interactive Computing and Visualization Cookbook – Second Edition / Cyrille Rossant., 2018. – 548 c. – (Packt).

22. Артамонов О.Ю. Моніторинг дорожнього руху / О.Ю. Артамонов, С.І. Шаповалова // Сучасні проблеми наукового забезпечення енергетики: матеріали XVIII Міжнародної науково-практичної конференції молодих вчених і студентів, Київ, 21-24 квітня 2020 р. У 2 т. - К: КПІ ім. Ігоря Сікорського, 2020.- Т.2.- С. 105.

23. PyCharm. URL: https://www.jetbrains.com/help/pycharm/quick-start-guide.html.

24. What is Pycharm. https://intellipaat.com/blog/what-is-pycharm/. URL: https://python-scripts.com/.

25. Django. https://docs.djangoproject.com/en/5.0/. URL: https://python-scripts.com/.

26. Pyplot in matplotlib. https://www.geeksforgeeks.org/pyplot-in-matplotlib/. URL: https://python-scripts.com/

27. Math module. https://realpython.com/python-math-module/. URL: https://python-scripts.com/

28. Appjar module. http://appjar.info/. URL: https://python-scripts.com/.

29. NumPy module. https://www.w3schools.com/python/numpy_intro.asp. URL: https://python-scripts.com/.

30. Import module. https://docs.python.org/3/reference/import.html. URL: https://python-scripts.com/

31. Python functions. https://www.w3schools.com/python/python_functions.asp. URL: https://python-scripts.com/.

32. Python main function. https://www.journaldev.com/17752/python-main-function. URL: https://python-scripts.com/.

33. Пришвидшений курс Python. Практичний, проєктно-орієнтований вступ до програмування [Текст] : Ерік Маттес, перекл. з англ. Ольги Бєлової. – Львів : Видавництво Старого Лева, 2021. – 600 с.

34. Керівництво по Python. URL – https://python-scripts.com/.

35. Документація Django. URL – https://tproger.ru/translations/create-your-firstdjango-app.

36. Документація Django. URL– https://vuetifyjs.com/ru/components/api-explorer/.

37. Документація SqLite. URL– https://proglib.io/p/sqlite-tutorial.

38. PyCharm. *https://www.jetbrains.com/help/pycharm/quick-start-guide.html.* URL: https://python-scripts.com/.

39. What is Pycharm. *https://intellipaat.com/blog/what-is-pycharm/.* URL: https://python-scripts.com/.

40. Django. *https://docs.djangoproject.com/en/5.0/.* URL: https://python-scripts.com/.

41. Pyplot in matplotlib. *https://www.geeksforgeeks.org/pyplot-in-matplotlib/*. URL: https://python-scripts.com/.

42. Math module. *https://realpython.com/python-math-module/*. URL: https://python-scripts.com/.

ДОДАТОК А

```
#%%
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import os
import cv2
import tensorflow as tf
from tensorflow.keras import optimizers
from tensorflow.keras.layers import *
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
#%% md
## Loading Images from the Disk
#%%
# Global Variables for later use

IMAGE_SIZE = 128
BATCH_SIZE = 32
CHANNELS = 3
#%%
# function that would read an image provided the image path, preprocess and
return it back

def read_and_preprocess(img_path):
    img = cv2.imread(img_path, cv2.IMREAD_COLOR) # reading the image
    img = cv2.resize(img, (IMAGE_SIZE, IMAGE_SIZE)) # resizing it to 128*128
    img = np.array(img, dtype='float32') # convert its datatype so that it could
be normalized
    img = img/255 # normalization (now every pixel is in the range of 0 and 1)
    return img
#%%
labels = ['real', 'fake']

X = [] # To store images
y = [] # To store labels

# labels -
# 0 - Real
# 1 - Fake
image_path = './dataset/' # path containing image samples
#%%
for folder in os.scandir(image_path):
    for entry in os.scandir(image_path + folder.name):

        X.append(read_and_preprocess(image_path + folder.name + '/' +
entry.name))

        if folder.name[0]=='r':
            y.append(0) # real
        else:
            y.append(1) # fake
#%%
X = np.array(X)
X.shape # We have 1289 image samples in total
#%%
y = np.array(y)
y.shape
#%% md
```

```
## Exploring the Dataset
#%%
real_count = len(y[y==0])
fake_count = len(y[y==1])

plt.title("Train Images for Each Label")
plt.bar(["Real Images", "Fake Images"],[real_count, fake_count])

# We have more samples of Fake Images than Real Images
#%%
# plotting random images

import random

plt.figure(figsize=(16, 16))

for i in range(12):
    ax = plt.subplot(3, 4, i + 1)
    plt.axis("off")
    img = random.randint(0, 1289)
    plt.imshow(X[img])
    plt.title(labels[y[img]])
#%% md
## Splitting the dataset
#%% md
We will take -
- 80% data for training our model
- 10% data for validation purpose
- 10% data for test purpose
#%%
from sklearn.model_selection import train_test_split

# We have splitted our data in a way that -
# 1. The samples are shuffled
# 2. The ratio of each class is maintained (stratify)
# 3. We get same samples every time we split our data (random state)

X_train, X_val, y_train, y_val = train_test_split(X, y,
                                                  test_size=0.2,
                                                  shuffle=True,
                                                  stratify=y,
                                                  random_state=123)
#%%
X_test, X_val, y_test, y_val = train_test_split(X_val, y_val,
                                                test_size=0.5,
                                                shuffle=True,
                                                stratify=y_val,
                                                random_state=123)
#%%
# we are using 1031 images for training our model
X_train.shape
#%%
# we are using 129 images for validating our model
X_val.shape
#%%
# we are using 129 images for testing our model
X_test.shape
#%%
input_shape = (IMAGE_SIZE, IMAGE_SIZE, CHANNELS)
n_classes = 2

model = tf.keras.Sequential([
    Conv2D(filters=32, kernel_size=(2,2), activation='relu',
input_shape=input_shape),
```

```
    MaxPooling2D((4,4)),

    Conv2D(filters=64, kernel_size=(3,3), activation='relu', padding='same'),
    MaxPooling2D((3,3)),
    Dropout(0.3), # for regularization

    Conv2D(filters=64, kernel_size=(4,4), activation='relu', padding='same'),
    Conv2D(filters=128, kernel_size=(5,5), activation='relu', padding='same'),
    MaxPooling2D((2,2)),
    Dropout(0.4),

    Conv2D(filters=128, kernel_size=(5,5), activation='relu', padding='same'),
    MaxPooling2D((2,2)),
    Dropout(0.5),

    Flatten(), # flattening for feeding into ANN
    Dense(512, activation='relu'),
    Dropout(0.5),
    Dense(256, activation='relu'),
    Dropout(0.3),
    Dense(128, activation='relu'),
    Dense(n_classes, activation='softmax')
])
#%%
# There are 1.1 Million Traininable Parameters
model.summary()
#%%
# compile the model
model.compile(loss = 'sparse_categorical_crossentropy', optimizer='Adam',
metrics= ["accuracy"])
#%%
# use early stopping to exit training if validation loss is not decreasing even
after certain epochs (patience)
earlystopping = EarlyStopping(monitor='val_loss', mode='min', verbose=1,
patience=10)

# save the best model with least validation loss
checkpointer = ModelCheckpoint(filepath="fakevsreal_weights.h5", verbose=1,
save_best_only=True)
#%%
history = model.fit(X_train, y_train, epochs = 100, validation_data=(X_val,
y_val), batch_size=BATCH_SIZE, shuffle=True, callbacks=[earlystopping,
checkpointer])
#%%
# The is how training loss and validation loss varied during training
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper right')
plt.show()
#%%
# The is how training accuracy and validation accuracy varied during training
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
#%%
# save the model architecture to json file for future use
```

```
model_json = model.to_json()
with open("fakevsreal_model.json","w") as json_file:
  json_file.write(model_json)
#%% md
## Evaluating the Saved Model Performance
#%%
# Load pretrained model (best saved one)
with open('fakevsreal_model.json', 'r') as json_file:
    json_savedModel= json_file.read()

# load the model weights
model = tf.keras.models.model_from_json(json_savedModel)
model.load_weights('fakevsreal_weights.h5')
model.compile(loss = 'sparse_categorical_crossentropy', optimizer='Adam',
metrics= ["accuracy"])
#%%
# making predictions
predictions = model.predict(X_test)
#%%
# Obtain the predicted class from the model prediction
predict = []

for i in predictions:
  predict.append(np.argmax(i))

predict = np.asarray(predict)
#%%
# Obtain the accuracy of the model
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_test, predict)
accuracy
#%%
# plot the confusion matrix
from sklearn.metrics import confusion_matrix

cf_matrix = confusion_matrix(y_test, predict)
plt.figure(figsize = (9,7))

group_names = ['Real Images Predicted Correctly','Real Images Predicted as
Fake','Fake Images Predicted as Real','Fake Images Predicted Correctly']
group_counts = ["{0:0.0f}".format(value) for value in cf_matrix.flatten()]
group_percentages = ["{0:.2%}".format(value) for value in
cf_matrix.flatten()/np.sum(cf_matrix)]
labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in
zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(cf_matrix, annot=labels, fmt='', cmap='Blues')
#%%
from sklearn.metrics import classification_report

report = classification_report(y_test, predict)
print(report)
```

```
import streamlit as st
import dashboard
import classifyPage

st.set_page_config(
    page_title="Test App Detection",
    page_icon="🤖",
    layout="wide")
```

```
PAGES = {
    "Dashboard": dashboard,
    "Classify Image": classifyPage
}

st.sidebar.title("Test App Detection")

st.sidebar.write("Test App Detection - це інструмент, який використовує
можливості глибокого навчання, щоб відрізнити справжні зображення від
підроблених.")

st.sidebar.subheader('Navigation:')
selection = st.sidebar.radio("", list(PAGES.keys()))

page = PAGES[selection]

page.app()
```

```
import streamlit as st
import util
import plotly.express as px


def app():
    file_uploaded = st.file_uploader("Choose the Image File", type=["jpg",
"png", "jpeg"])
    if file_uploaded is not None:
        res = util.classify_image(file_uploaded)
        c1, buff, c2 = st.columns([2, 0.5, 2])
        c1.image(file_uploaded, use_column_width=True)

        c2.subheader("Classification Result")
        c2.write("The image is classified as
**{}**.".format(res['label'].title()))
        c2.write("Probability: **{:.2f}%**".format(res['probability']))

        fig = px.bar(x=['Real', 'Fake'], y=[100 - res['probability'],
res['probability']],
                     labels={'x': 'Category', 'y': 'Probability (%)'})
        c2.plotly_chart(fig, use_container_width=True)

        c2.subheader("Insights")
        c2.write(
            "Пропонуємо вашій увазі інформацію про те, як модель приймала
рішення, які частини зображення мали найбільший вплив і т.д.")


import streamlit as st
import plotly.express as px

def app():

    st.subheader("💡 Анотація:")

    inspiration = '''
    Останнім часом глибоке навчання досягло значного успіху завдяки генеративним
змагальним мережам (Generative Adversarial Networks, GAN), які використовуються
для отримання високоякісних результатів, які можна порівняти з вихідними даними.
GAN широко використовуються для створення нових реалістичних зображень та
покращення існуючих. З іншого боку, GAN можуть використовуватися для обману
людей шляхом генерування неправдивих даних. Наприклад, фальшиві обличчя,
```

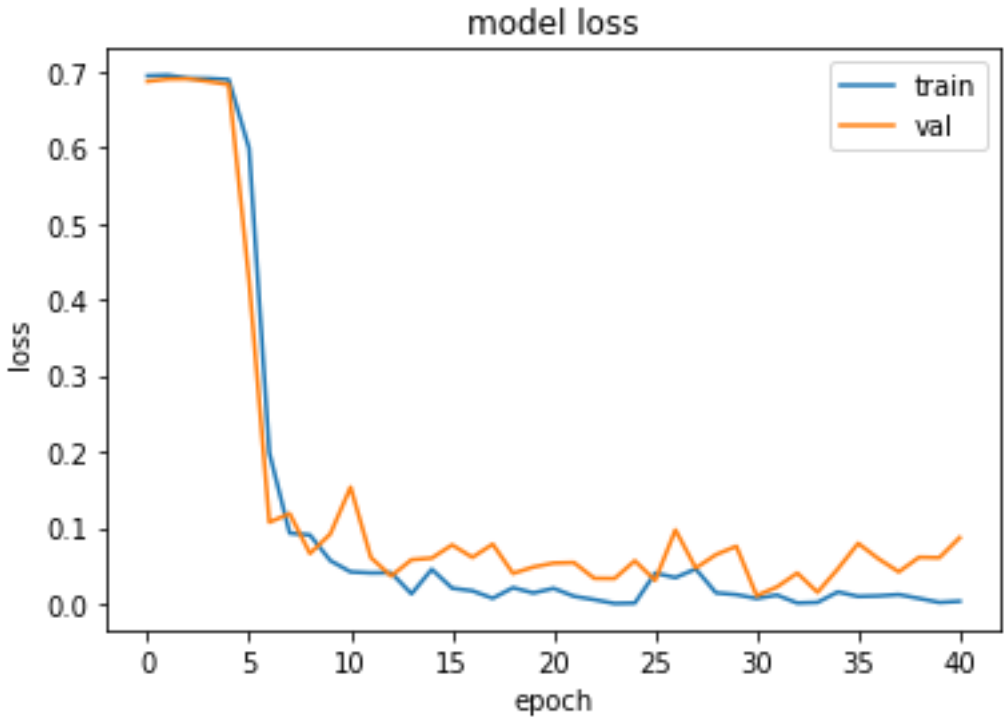
```
створені за допомогою GAN, можуть обдурити не лише людей, а й класифікатори машинного навчання. Синтетичні фотографії для ідентифікації та автентифікації, наприклад, можуть бути використані зловмисниками.

Крім того, сучасне програмне забезпечення для редагування зображень, таке як Adobe Photoshop, дозволяє змінювати складні вхідні фотографії, а також створювати нові високоякісні зображення. Ці технології вдосконалилися до такої міри, що тепер можна створювати реалістичні та складні фальшиві зображення, які важко відрізнити від справжніх. На YouTube є покрокові інструкції та навчальні посібники для створення такої фіктивної графіки. Як наслідок, ці технології можуть бути використані для наклепу, самозванства та спотворення фактів. Крім того, завдяки соціальним мережам шахрайські матеріали можуть швидко і широко поширюватися в Інтернеті.'''

    st.write(inspiration)

    st.subheader("🧐💻 Що робить наш проект?")

    what_it_does = '''
    Deforgify - це інструмент, який використовує можливості глибокого навчання, щоб відрізняти справжні зображення від фейкових. Наприклад, якщо хтось бере ваше оригінальне зображення і вставляє ваше обличчя в сцену вбивства або фотошопить його на чиємусь тілі, Deforgify позначить його як фейкове, зменшуючи ймовірність того, що воно буде використане для обвіхування вас. <br> Просто надішліть зображення, і модель машинного навчання оцінить його та надасть відповідь за долі секунди".'''

    st.markdown(what_it_does, unsafe_allow_html=True)

    stats, buff, graph = st.columns([2, 0.5, 2])

    stats.subheader("🧠 ML Process")

    stats.markdown("<h5> 📊 Отримання даних та процес EDA </h5>", unsafe_allow_html=True)

    stats.markdown('''Набір даних містить 1288 осіб, з яких
    <li> 589 - справжні </li>
    <li> 700 - фальшиві </li>
    ''', unsafe_allow_html=True)

    fig = px.bar(x=['Real', 'Fake'], y=[589, 700], height=400)
    graph.plotly_chart(fig, use_container_width=True)

    st.write('"Фальшиві" обличчя, зібрані в цьому наборі даних, згенеровані за допомогою StyleGAN2, і їх складно класифікувати правильно навіть для людського ока"')

    st.image('./Streamlit_UI/faces_1.png', use_column_width=True)
    st.image('./Streamlit_UI/faces_2.png', use_column_width=True)

    st.subheader("⚙️ Model Architecture")

    st.image('./Streamlit_UI/model.png', use_column_width=True)

    ml_process = f'''
- Ми розробили послідовну модель з 5 згорткових шарів і 4 щільних шарів.
- Перший шар починався з 32 фільтрів та ядра 2x2.
- На кожному наступному шарі кількість фільтрів подвоюється, а ядро збільшується на 1.
- Ми додали кілька шарів з максимальним об'єднанням після згорткових шарів, щоб уникнути надмірної підгонки та зменшити обчислювальні витрати.
- Вихідні дані згорткового шару згладжуються і передаються до щільних шарів.
```

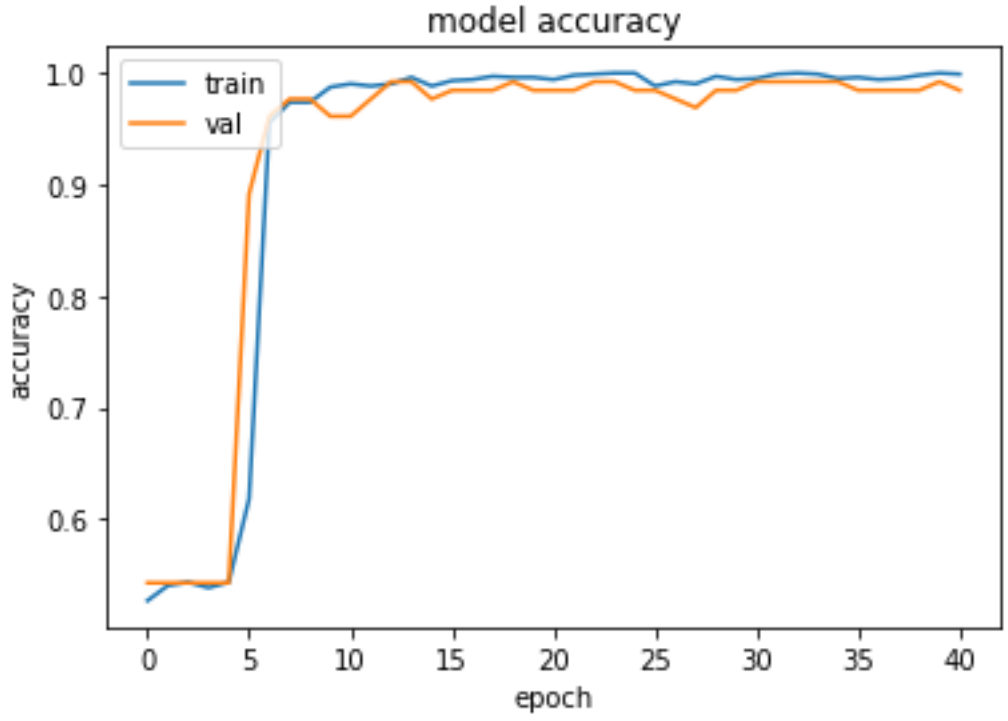
```
- Ми почали з 512 нейронів у першому щільному шарі і зменшили їх кількість до половини протягом наступних двох щільних шарів.
- Також в модель було введено кілька шарів, що відсіваються, щоб випадковим чином ігнорувати деякі нейрони та зменшити надмірне припасування.
- Ми використовували активацію ReLU у всіх шарах, окрім вихідного, щоб зменшити витрати на обчислення і ввести нелінійність.
- Нарешті, було побудовано вихідний шар, що містить 2 нейрони (по 1 для кожного класу) та активацію softmax.
'''
    st.write(ml_process)

    results = f'''
- Модель з найменшою валідаційною втратою була збережена під час навчання і перезавантажена перед отриманням остаточних результатів.
- Модель змогла правильно класифікувати всі зразки.
    '''
    loss, buff, acc = st.columns([2, 0.4, 2])

    loss.image('./Streamlit_UI/loss.png', use_column_width=True)
    acc.image('./Streamlit_UI/accuracy.png', use_column_width=True)

    st.subheader("☑ Results")
    st.markdown(results, unsafe_allow_html=True)

    st.write("Classification Report:")

    cfr = '''
 Report Title     precision    recall  f1-score   support

        Real       1.00      1.00      1.00        59
        Fake       1.00      1.00      1.00        70

    accuracy                           1.00       129
   macro avg       1.00      1.00      1.00       129
weighted avg       1.00      1.00      1.00       129
'''
    st.code(cfr)

    st.write(" ")

    st.write("*Try it out now by clicking on Classify Image button on the Sidebar*")
```

```
import tensorflow as tf
from PIL import Image
import numpy as np

model = None
labels = ['real', 'fake']

def load_model():
    global model
    model =
tf.keras.models.load_model('/Users/ihortresnystkyi/Documents/Deforgify/Streamlit_UI/fakevsreal_weights.h5')

def classify_image(file_path):
    if model is None:
        load_model()
```

```
    image = Image.open(file_path)
    image = image.resize((128, 128))
    image = image.convert("RGB")
    img = np.asarray(image)
    img = np.expand_dims(img, 0)
    predictions = model.predict(img)
    label = labels[np.argmax(predictions[0])]
    probab = float(round(predictions[0][np.argmax(predictions[0])]*100, 2))

    result = {
        'label': label,
        'probability': probab
    }

    return result
```

ДОДАТОК Б


model loss
train
val
loss
epoch
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
5
10
15
20
25
30
35
40


model accuracy
train
val
accuracy
epoch
0.6
0.7
0.8
0.9
1.0
0
5
10
15
20
25
30
35
40

ДОДАТОК В

Рис.3. опис рисунку що на ньому зображено

ДОДАТОК Г

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 127, 127, 32)      416

 max_pooling2d (MaxPooling2D  (None, 31, 31, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 31, 31, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 10, 10, 64)       0
 2D)

 dropout (Dropout)           (None, 10, 10, 64)        0

 conv2d_2 (Conv2D)           (None, 10, 10, 64)        65600

 conv2d_3 (Conv2D)           (None, 10, 10, 128)       204928

 max_pooling2d_2 (MaxPooling  (None, 5, 5, 128)        0
 2D)

 dropout_1 (Dropout)         (None, 5, 5, 128)         0

 conv2d_4 (Conv2D)           (None, 5, 5, 128)         409728

 max_pooling2d_3 (MaxPooling  (None, 2, 2, 128)        0
 2D)

 dropout_2 (Dropout)         (None, 2, 2, 128)         0

 flatten (Flatten)           (None, 512)               0

 dense (Dense)               (None, 512)               262656

 dropout_3 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 256)               131328

 dropout_4 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 128)               32896

 dense_3 (Dense)             (None, 2)                 258
```