МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп’ютерної інженерії

ФАЄРЧУК Вадим Вікторович

Програмний модуль класифікації людських облич засобами нейронних мереж / Software module for classifying human faces using neural networks

спеціальність: 123 – Комп’ютерна інженерія

освітньо–професійна програма – Комп’ютерна інженерія

Кваліфікаційна робота

Виконав: студент групи КІ–41

ФАЄРЧУК Вадим Вікторович

Науковий керівник

к.т.н. доц. Піцун О.Й.

ТЕРНОПІЛЬ – 2026

АНОТАЦІЯ

Фаєрчук В.В. Програмний модуль класифікації людських облич засобами нейронних мереж. – Рукопис.

Кваліфікаційна робота на здобуття освітнього ступеня «бакалавр» за спеціальністю 123 «Комп'ютерна інженерія», освітньо-професійна програма. Західноукраїнський національний університет, Тернопіль, 2026.

Робота написана обсягом 83 сторінок і містить 8 рисунків, 18 таблиць, 3 додатки та 31 джерела за переліком посилань.

Метою кваліфікаційної роботи є розробка програмного модуля для автоматизованого виявлення підроблених (AI-генерованих) зображень людських облич. Спроєктовано гібридну архітектуру, що поєднує згорткові нейронні мережі (CNN) для виділення локальних ознак та графові нейронні мережі (GNN) для аналізу структурних аномалій у взаєморозташуванні ключових точок обличчя. Запропонований підхід забезпечує високу точність класифікації реальних та синтезованих портретів шляхом аналізу геометричних та текстурних невідповідностей. Реалізовано веб-інтерфейс на базі фреймворку Flask для зручної взаємодії з моделлю та візуалізації результатів детекції.

Ключові слова: НЕЙРОННІ МЕРЕЖІ, CNN, GNN, КЛАСИФІКАЦІЯ ОБЛИЧ, AI-ГЕНЕРОВАНІ ЗОБРАЖЕННЯ, DEEPFAKE, PYTHON.

ANNOTATION

Faierchuk V.V. Software module for classifying human faces using neural networks. – Manuscript.

Qualification work for the Bachelor degree in specialty 123 “Computer Engineering”, educational and professional program. Western Ukrainian National University, Ternopil, 2026.

The work is written in 83 pages and contains 8 figures, 18 tables, 3 appendices and 31 references.

The purpose of the qualification work is to develop a software module for automated detection of fake (AI-generated) images of human faces. A hybrid architecture is designed, which combines convolutional neural networks (CNN) for local feature extraction and graph neural networks (GNN) for analyzing structural anomalies in the spatial arrangement of facial landmarks. The proposed approach provides high accuracy in classifying real and synthesized portraits by analyzing geometric and textural inconsistencies. A web interface based on the Flask framework has been implemented for convenient interaction with the model and visualization of detection results.

Keywords: NEURAL NETWORKS, CNN, GNN, FACE CLASSIFICATION, AI-GENERATED IMAGES, DEEPFAKE, PYTHON.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

AI	–	Artificial Intelligence
AUC–ROC	–	Area Under the Receiver Operating Characteristic Curve
BCE Loss	–	Binary Cross Entropy Loss
CNN	–	Convolutional Neural Network
CPU	–	Central Processing Unit
CUDA	–	Compute Unified Device Architecture
FFHQ	–	Flickr-Faces-HQ
GAN	–	Generative Adversarial Network
GAT	–	Graph Attention Network
GNN	–	Graph Neural Network
GPU	–	Graphics Processing Unit
HTML	–	HyperText Markup Language
JSON	–	JavaScript Object Notation
MTCNN	–	Multi-task Cascaded Convolutional Networks
OS	–	Operating System
PIL	–	Python Imaging Library
PyG	–	PyTorch Geometric
RAM	–	Random Access Memory
ReLU	–	Rectified Linear Unit
RGB	–	Red, Green, Blue
ViT	–	Vision Transformer
WSGI	–	Web Server Gateway Interface

ВСТУП

У сучасному світі інформаційні технології стрімко розвиваються та охоплюють більшість сфер діяльності людини. Цифрова трансформація суспільства супроводжується появою нових інструментів обробки, аналізу та генерації даних, які кардинально змінюють підходи до роботи з візуальною інформацією. Одним із важливих напрямів штучного інтелекту є комп'ютерний зір, який забезпечує можливість автоматичного аналізу зображень і відео. Такі технології широко використовуються у системах безпеки, мобільних застосунках, соціальних мережах та інших цифрових сервісах. Прогрес у цій галузі дозволив значно підвищити точність розпізнавання об'єктів, осіб та сцен, що відкрило нові можливості для автоматизації широкого спектру задач.

Особливого значення набувають системи аналізу людських облич, які застосовуються для ідентифікації користувачів, контролю доступу до приміщень, відеоспостереження та підтвердження особи в онлайн–сервісах. Біометрична ідентифікація за обличчям стала невід'ємною частиною сучасних смартфонів, банківських додатків та систем прикордонного контролю. Висока зручність і швидкість таких методів зробили їх популярною альтернативою традиційним паролям та фізичним документам.

Разом із розвитком цих технологій активно розвиваються генеративні моделі штучного інтелекту, здатні створювати реалістичні зображення людей, які не існують у реальному світі. Генеративно-змагальні мережі (GAN), варіаційні автоенкодери (VAE) та дифузійні моделі досягли рівня, за якого синтезовані зображення майже не поступаються якістю реальним фотографіям. Сучасні алгоритми дозволяють генерувати обличчя настільки високої якості, що їх надзвичайно складно відрізнити від справжніх фотографій навіть досвідченому спостерігачеві. Особливо швидкий розвиток цього напряму спостерігається починаючи з 2018 року, коли публічно стала доступною архітектура StyleGAN, а згодом і її вдосконалені версії. Із появою дифузійних

моделей нового покоління якість синтезованих зображень зросла ще більше, що додатково ускладнило задачу їх виявлення.

Поширення подібних технологій створює нові загрози для інформаційної безпеки та цифрової довіри. Синтетичні зображення можуть використовуватися для створення фейкових акаунтів у соціальних мережах, поширення дезінформації, шахрайства або обходу систем біометричної ідентифікації. Deepfake-технології дозволяють не лише генерувати неіснуючих людей, але й підмінювати обличчя на відео, що відкриває можливості для маніпуляцій суспільною думкою та компрометації публічних осіб. Крім того, доступність інструментів генерації постійно зростає, а їх використання не потребує спеціальних технічних знань, що сприяє масовому поширенню підробленого візуального контенту. За оцінками дослідників у галузі кібербезпеки, кількість deepfake-матеріалів в інтернеті подвоюється щороку, що свідчить про загострення проблеми.

У відповідь на ці виклики активно розвивається напрям автоматичного виявлення синтетичних зображень. Дослідники пропонують різноманітні підходи: від аналізу артефактів стиснення та статистичних аномалій до використання глибоких нейронних мереж, навчених розрізняти реальні та згенеровані зображення. Однак задача залишається актуальною, оскільки генеративні моделі постійно вдосконалюються, а існуючі методи виявлення часто не встигають за їх розвитком. Тому розробка нових, більш ефективних і стійких рішень є важливим науково-практичним завданням.

Об’єкт дослідження –програмний модуль для детекції людчьких облич.

Предмет дослідження – методи детекції згенерованих людчьких облич засобами згорткових та графових нейронних мереж.

Основною метою даної бакалаврської роботи є розробка програмного модуля для класифікації зображень людських облич на реальні та штучно згенеровані з використанням методів глибинного навчання [26, 27]. Для досягнення поставленої мети необхідно вирішити такі завдання:

- проаналізувати існуючі методи генерації та виявлення синтетичних

зображень облич;

- обрати та обґрунтувати архітектуру нейронної мережі для класифікації;
- сформувати навчальну та тестову вибірки; реалізувати програмний модуль і провести його навчання;
- виконати оцінку якості розробленої моделі на тестових даних.

Актуальність роботи полягає у необхідності забезпечення достовірності цифрового контенту та підвищення надійності сучасних систем ідентифікації шляхом автоматичного виявлення синтетичних зображень облич. Зростання кількості підробленого візуального контенту та його потенційний вплив на суспільство роблять розробку ефективних методів виявлення таких зображень нагальною практичною потребою.

Практичне значення роботи полягає у створенні функціонального веб-застосунку, що дозволяє визначати тип людчького обличчя на основі нейронних мереж. та у розробці комбінованої архітектури на основі згорткової та графової нейронних мереж.

У першому розділі було проаналізовано сучасні підходи до визначння штучно згенерованих людчьких облич та інструменти для реалізації.

У другому розділі було розроблено узагальнений алгоритм роботи модуля та архітектуру графової нейронної мережі для визначення ШІ – згенерованих зображень.

У третьому розділі розроблено прикладне програмне забезпечення та проведено порівняльний аналіз.

1 АНАЛІЗ ПІДХОДІВ ДО ГЕНЕРУВАННЯ ЗОБРАЖЕНЬ ЛЮДСЬКИХ ОБЛИЧ

1.1 Аналіз методів виявлення AI–згенерованих зображень облич

Розпізнавання та класифікація зображень людських облич є однією з центральних задач сучасного комп'ютерного зору. Класичний процес аналізу обличчя охоплює чотири послідовні етапи: детекцію, вирівнювання, виділення ознак та класифікацію. Задача даної роботи відноситься до класифікаційного підкласу: необхідно визначити походження зображення – реальне воно чи згенероване штучним інтелектом.

Генеративно–змагальні мережі (GAN), запропоновані Ian Goodfellow [1] у 2014 році, стали проривом у синтезі реалістичних зображень. GAN складається з двох компонентів: генератора, який створює синтетичні зображення з випадкового шуму, та дискримінатора, що намагається їх відрізнити від реальних. Ці компоненти навчаються одночасно у грі з нульовою сумою, де генератор прагне обдурити дискримінатора, а дискримінатор – навчитися відрізняти реальні зображення від синтетичних. Принцип роботи GAN–архітектури проілюстровано на рисунку 1.1.

Рисунок 1.1 – Схема роботи генеративно–змагальної мережі (GAN)

Еволюція архітектур GAN пройшла шлях від DCGAN (2015) і ProGAN (2017) до StyleGAN [5] /StyleGAN2 [2] /StyleGAN3 від NVIDIA, що генерує обличчя роздільністю 1024×1024 пікселів з надзвичайно високим рівнем реалізму. StyleGAN2 запровадив архітектуру з відображенням латентного простору через mapping network та механізм weight demodulation, що усунуло характерні артефакти попередніх версій. Сучасні дифузійні моделі (Stable Diffusion, Midjourney, DALL–E) ще більше ускладнили задачу детекції, оскільки принципово відрізняються від GAN за механізмом генерації і залишають інші типи артефактів.

Незважаючи на реалістичність, синтетичні зображення містять специфічні аномалії, що слугують ознаками для класифікатора. До таких ознак належать: мікроартефакти в текстурі шкіри, аномалії деталей обличчя (нереалістичні вуха, зуби, відблиски в очах), порушення природної симетрії, фонові артефакти на межах обличчя, специфічні патерни у частотному спектрі (виявляються через перетворення Фур'є) [28], а також унікальні «відбитки» архітектури конкретного генератора. Ці ознаки, часто непомітні для людського ока, виявляються класифікаторами глибинного навчання, які здатні навчитися розпізнавати статистичні закономірності, недоступні звичайному спостерігачеві.

Теоретичною основою роботи слугують згорткові нейронні мережі (CNN) – архітектури, що автоматично навчаються виявляти ієрархію візуальних ознак через операції згортки, пулінгу та повнозв'язних шарів. На нижчих рівнях CNN виявляють прості ознаки (краї, текстури), на вищих – складні семантичні конструкції. Трансферне навчання дозволяє використовувати попередньо навчені на ImageNet ваги та адаптувати їх до специфічної задачі, що суттєво скорочує час навчання та вимоги до обсягу даних.

EfficientNet – сімейство архітектур CNN, запропоноване Tan та Le у 2019 році [3]. Ключова ідея – метод compound scaling: масштабування глибини (кількість шарів), ширини (кількість каналів) та роздільності вхідного зображення одночасно за фіксованим коефіцієнтом. Це дозволяє ефективніше використовувати обчислювальні ресурси, ніж масштабування лише одного

виміру. EfficientNet–B2, обраний для даної роботи, містить приблизно 9.2 мільйони параметрів та досягає точності 80.1% на ImageNet–1k – значно кращий результат, ніж ResNet–50 (76.0%) [11] при схожій кількості параметрів. Backbone виводить вектор ознак розмірністю 1408 через операцію global average pooling.

Графові нейронні мережі (GNN) доповнюють CNN, моделюючи структурні залежності між елементами зображення. У даній роботі патчі зображення розміром 16×16 пікселів представляються як вузли графу, а ребра відповідають просторовим зв'язкам між сусідніми патчами. Graph Attention Network (GAT) [4] завдяки механізму уваги здатна надавати різні ваги сусіднім вузлам, що дозволяє автоматично виявляти ті патчі, де зосереджені артефакти генеративних моделей, без явного задання їхнього розташування.

Поширення технологій генерації синтетичних облич створює серйозні загрози для інформаційної безпеки. Синтетичні зображення можуть використовуватися для створення фейкових акаунтів у соціальних мережах, поширення дезінформації, шахрайства при онлайн–верифікації особи та обходу систем біометричної ідентифікації. Зокрема, алгоритми заміни обличчя (face swap), такі як FaceShifter [25], здатні замінити обличчя у відео з високим рівнем реалізму. Доступність інструментів генерації постійно зростає – сучасні веб–сервіси дозволяють генерувати фотореалістичні обличчя без спеціальних технічних знань, що сприяє масовому поширенню підробленого візуального контенту.

Задача автоматичного виявлення AI–згенерованих зображень є принципово відмінною від задачі їх генерації. Якщо генеративні моделі з кожним роком вдосконалюються та наближаються до фотореалізму, детектори мають постійно адаптуватися до нових типів артефактів. Це призводить до своєрідної «гонки озброєнь» [24] між генераторами та детекторами, де кожне покоління генеративних моделей ускладнює роботу класифікаторів. Відповідно, розробка робастних методів детекції, що добре узагальнюються на нові типи генеративних моделей, є одним з ключових викликів сучасних досліджень у галузі комп'ютерного зору.

1.2 Аналіз програмних засобів

Для реалізації системи детекції AI–зображень проаналізовано основні програмні засоби та обрано оптимальний стек технологій. Вибір інструментів визначався трьома критеріями: наявністю зрілої екосистеми для задач комп'ютерного зору, підтримкою граф–нейронних мереж та можливістю інтеграції у веб–застосунок.

PyTorch – основний фреймворк глибинного навчання від Meta AI, що базується на динамічному обчислювальному графі (define–by–run). На відміну від статичного графу TensorFlow 1.x, динамічний граф PyTorch дозволяє модифікувати мережу під час виконання, що значно спрощує відлагодження та розробку нестандартних архітектур. PyTorch є де–факто стандартом у дослідницькому середовищі: за даними аналізу публікацій на arXiv, понад 70% нових робіт з машинного навчання використовують PyTorch. Підтримка CUDA дозволяє виконувати обчислення на GPU, що прискорює навчання у 10–50 разів порівняно з CPU.

TensorFlow/Keras – альтернативний фреймворк від Google. Попри зручне розгортання через TensorFlow Serving та TensorFlow Lite для мобільних пристроїв, екосистема для GNN (бібліотека Spektral) є значно менш розвиненою, ніж PyTorch Geometric. Кількість підтримуваних GNN–архітектур у Spektral у 5–7 разів менша, ніж у PyG. З цих причин для даної роботи обрано PyTorch.

PyTorch Geometric (PyG) – спеціалізована бібліотека для граф–нейронних мереж на базі PyTorch, що містить реалізації понад 50 алгоритмів обробки графів, включаючи GCN, GAT, GraphSAGE, GIN та інші. PyG забезпечує оптимізовані операції scatter та sparse matrix multiplication для ефективної роботи з великими графами. Бібліотека є стандартом у дослідженнях GNN і використовується в даній роботі для реалізації компонента Graph Attention Network.

Timm (PyTorch Image Models) – бібліотека з понад 700 попередньо

навченими CNN–моделями для задач комп'ютерного зору. Забезпечує уніфікований API для завантаження EfficientNet–B2 з вагами ImageNet та подальшого fine–tuning. Особливо корисна функція доступу до внутрішніх блоків backbone для поступового розморожування шарів при двоетапному навчанні.

Albumentations – високопродуктивна бібліотека аугментації зображень, оптимізована для задач комп'ютерного зору. Порівняно з torchvision.transforms забезпечує у 2–4 рази швидшу обробку завдяки реалізації операцій на рівні OpenCV та NumPy. Підтримує понад 70 типів трансформацій та забезпечує однакову обробку зображення та відповідних масок, що важливо для задач сегментації. У даній роботі використовується для аугментації навчальної вибірки.

Scikit–learn використовується для стратифікованого розбиття датасету на навчальну, валідаційну та тестову вибірки (train_test_split з параметром stratify), що гарантує рівномірний розподіл класів у кожній із підвибірок та запобігає дисбалансу між реальними та синтетичними зображеннями. Бібліотека також застосовується для обчислення метрик якості класифікації: Accuracy, AUC–ROC, F1–score, Precision і Recall, де остання пара метрик дозволяє окремо оцінити якість розпізнавання кожного класу, а AUC–ROC характеризує здатність моделі розрізняти класи незалежно від обраного порогу. Для наочного аналізу помилок будується матриця плутанини (confusion matrix).

Flask – легковаговий WSGI–фреймворк для Python, що забезпечує маршрутизацію HTTP–запитів та обробку завантажених файлів. Вибір Flask для розгортання моделі зумовлений простотою інтеграції з бібліотеками PyTorch та мінімальними накладними. Flask не нав'язує структуру проекту, що дозволяє гнучко організувати обробку зображень: завантаження файлу → попередня обробка (resize, normalize) → подача у модель → повернення результату у форматі JSON. Альтернативний фреймворк FastAPI забезпечує вищу продуктивність завдяки асинхронній обробці запитів, однак для даного проекту обсяг запитів не є критичним фактором. Порівняльна характеристика основних

фреймворків глибинного навчання наведена у таблиці 1.1.

Таблиця 1.1 – Порівняння фреймворків глибинного навчання

Критерій	PyTorch	TensorFlow/Keras
Тип графу обчислень	Динамічний	Статичний / Динамічний
Підтримка GNN (бібліотека)	PyG (50+ архітектур)	Spektral (~80 архітектур)
Популярність у дослідженнях	Найвища (>70%)	Середня (~25%)
Підтримка CUDA	Так	Так
Розгортання у веб	Flask / FastAPI	TF Serving
Обрано для роботи	Так	Ні

Проведений аналіз підтверджує, що PyTorch у поєднанні з PyTorch Geometric є оптимальним інструментальним стеком для реалізації запропонованої гібридної архітектури: бібліотека забезпечує динамічний обчислювальний граф, підтримку CUDA та нативну інтеграцію з Flask через стандартний Python-інтерфейс.

1.3 Порівняльний аналіз існуючих систем

Наукова спільнота активно досліджує методи виявлення deepfake та GAN–згенерованих зображень. Аналіз літератури дозволяє виділити чотири основні напрями досліджень, кожен з яких має свої переваги та обмеження.

Частотний аналіз. Операції згортки та апсемплінгу в GAN залишають характерні сліди у частотній області зображення, які можна виявити через перетворення Фур'є або вейвлет–розкладання. Роботи Frank et al. (2020) [6] та Chandrasegaran et al. (2021) [28] демонструють ефективність цього підходу для детекції зображень конкретних генераторів. Головний недолік – погана узагальнюваність: модель, навчена на артефактах StyleGAN2, значно гірше

виявляє зображення дифузійних моделей, оскільки їхній частотний профіль суттєво відрізняється.

Аналіз біологічних аномалій. Даний підхід базується на виявленні відсутності природних недосконалостей: нереалістична симетрія зіниць, аномальний колір шкіри або відсутність мікротекстур, неправильна геометрія вушних раковин. Перевагою є висока інтерпретованість результатів – система може пояснити, яка саме аномалія свідчить про синтетичне походження. Недолік – необхідність додаткових модулів детекції ключових точок обличчя та залежність від якості вхідного зображення.

CNN–класифікатори. Найбільш ефективний та широко вживаний підхід – пряме навчання CNN на парах Real/Fake. XceptionNet з depthwise separable convolutions [22], EfficientNet з принципом compound scaling та MobileNetV2 показують найкращі результати серед чистих CNN. Комплексна порівняльна оцінка методів детекції на різних типах маніпуляцій виконана у роботі Rossler et al. [9]. Головний недолік – CNN аналізує кожен піксель незалежно, без явного моделювання просторових залежностей між різними регіонами зображення, що обмежує здатність виявляти структурні аномалії.

Гібридні CNN+GNN архітектури. Найновіший та найперспективніший напрям. CNN виконує витягування локальних ознак з окремих ділянок зображення, тоді як GNN аналізує структурні зв'язки між цими ділянками на рівні графу. Дослідження Zhao et al. (2021) [7] та Li et al. (2020) [8] демонструють приріст точності на 2–4% порівняно з чистими CNN на тестових вибірках. Саме цей підхід обрано для даної роботи з огляду на його перевагу в задачах, де важлива не лише локальна текстура, але й просторові відносини між регіонами. Темпоральні підходи для відеосеквенцій досліджено у роботі Zheng et al. [10]

Обґрунтування вибору архітектури: EfficientNet–B2 забезпечує оптимальний баланс точності та обчислювальної ефективності серед CNN–backbone. Метод compound scaling (одночасне масштабування глибини, ширини та роздільності) дозволяє досягти кращих результатів на ImageNet при меншій кількості параметрів, ніж ResNet або VGG. Graph Attention Network (GAT)

перевершує Graph Convolutional Network (GCN) [23] завдяки механізму уваги, що надає різні ваги сусіднім вузлам – це критично важливо для локалізації артефактів, оскільки не всі патчі зображення однаково інформативні.

Окремо слід відзначити підхід на основі трансформерних архітектур (Vision Transformer, ViT) [21]. ViT розбиває зображення на патчі та обробляє їх через механізм self–attention, що концептуально схоже на підхід CNN+GNN. Проте ViT вимагає значно більшого обсягу даних для навчання (від кількох мільйонів зображень) та суттєвих обчислювальних ресурсів. Гібридний підхід CNN+GNN, обраний у даній роботі, поєднує ефективність CNN на відносно невеликих датасетах із структурним моделюванням GNN, що робить його більш практичним для реалізації в рамках бакалаврської роботи. Порівняльний аналіз підходів до виявлення AI–зображень узагальнено у таблиці 1.2.

Таблиця 1.2 – Порівняльний аналіз підходів до виявлення AI–зображень

Підхід	Переваги	Недоліки	Узагальнюваність
Частотний аналіз (FFT/Wavelet)	Висока точність на відомих GAN	Погана на нових архітектурах	Низька
Аналіз біологічних аномалій	Інтерпретованість	Потреба у landmark–детекторі	Середня
CNN–класифікатори	Висока точність, простота	Немає просторового моделювання	Середня
Гібридні CNN+GNN	Найкраща точність (+2–4%)	Вища обчислювальна складність	Висока

Гібридний підхід CNN+GNN вирізняється найвищою узагальнюваністю серед розглянутих методів та забезпечує найкращу точність завдяки

одночасному аналізу локальних текстурних ознак і просторових залежностей між регіонами зображення, що й обґрунтовує його вибір як базової архітектури розробленої системи.

1.4 Характеристика датасетів для навчання та тестування моделі

Для задачі виявлення AI–згенерованих облич обрано два датасети по 10 000 зображень кожен, масштабовані до роздільності 256×256 пікселів. Обидва датасети зосереджені виключно на зображеннях облич – це забезпечує однорідність задачі та усуває вплив фонових об'єктів на класифікацію.

FFHQ (Flickr–Faces–HQ Dataset) [16] є джерелом реальних зображень. Датасет зібраний командою NVIDIA та містить 70 000 високоякісних фотографій, скомпільованих із соціальної мережі Flickr, оригінальної роздільності 1024×1024 пікселів. Датасет характеризується великою різноманітністю за статтю, віком, етнічністю, освітленням та виразами обличчя. Зображення пройшли автоматичну детекцію облич та вирівнювання за ключовими точками. FFHQ є золотим стандартом для досліджень генерації та детекції облич у науковій спільноті.

Generated Photos [17] є джерелом AI–згенерованих зображень. Комерційна платформа надає синтетичні фотографії, згенеровані на основі StyleGAN2, оригінальної роздільності 512×512 пікселів. Жодне зображення не відповідає реальній людині, що повністю виключає проблеми конфіденційності та авторського права. Пара FFHQ + Generated Photos є відносно складною для детекції, оскільки Generated Photos спеціально створені для максимальної фотореалістичності та відсутності очевидних артефактів.

Датасет 100K Faces компанії Generated Media Inc. є комерційним продуктом. Для виконання даної кваліфікаційної роботи компанія Generated Media Inc. люб'язно надала безкоштовний доступ до датасету в рамках програми

підтримки академічних та наукових досліджень, за що автор висловлює щиру вдячність

Візуальне порівняння типових зображень обох датасетів наведено на рисунку 1.2.

а)Реальні (FFHQ) б)Згенеровані (Generated Photos)

Рисунок 1.2 – Приклади зображень датасетів FFHQ та Generated Photos

Підготовка датасету включає стратифіковане розбиття у співвідношенні 80/10/10 (навчальна/валідаційна/тестова вибірки). Стратифікація гарантує однакове співвідношення класів Real та AI у кожній вибірці. Класи збалансовані (по 10 000 зображень кожен), що унеможливлює вплив дисбалансу класів на навчання моделі. Аугментація для навчальної вибірки включає горизонтальне відображення (p=0.5), незначні зміни яскравості та контрасту (p=0.5) і легкий random crop (p=0.3). Сильні трансформації навмисно виключені, щоб зберегти специфічні артефакти AI–зображень як ключові ознаки для класифікатора.

Важливим аспектом вибору датасету є те, що Generated Photos базується на StyleGAN2 – одному з найрезультативніших генераторів облич на момент написання роботи. Це означає, що модель навчається розпізнавати артефакти саме GAN–генерації. Виявлення зображень, згенерованих дифузійними

моделями (Stable Diffusion, Midjourney), може вимагати додаткового доопрацювання датасету. Водночас, навчання на якісному датасеті StyleGAN2 формує у моделі базові навички розпізнавання синтетичних облич, які частково переносяться на інші типи генераторів.

Порівняння ключових характеристик обраних датасетів наведено у таблиці 1.3.

Таблиця 1.3 – Порівняння датасетів FFHQ та Generated Photos

Характеристика	FFHQ (реальні)	Generated Photos (AI)
Загальний обсяг	70 000 зображень	100 000+ зображень
Використано в роботі	10 000 зображень	10 000 зображень
Оригінальна роздільність	1024×1024 px	512×512 px
Роздільність у роботі	256×256 px	256×256 px
Джерело	Flickr (реальні фото людей)	StyleGAN2–генерація
Різноманітність	Стать, вік, освітлення, емоції	Аналогічно FFHQ
Ліцензія	Creative Commons BY 2.0	Комерційна (платне використання)
Мітка класу	0 (Real)	1 (AI–Generated)
Складність детекції	–	Висока (оптимізовано на реалізм)

Обрана пара датасетів FFHQ та Generated Photos є оптимальною для вирішення поставленої задачі: вона забезпечує достатній обсяг і різноманітність прикладів при збалансованому розподілі класів і водночас становить практичний виклик для класифікатора, оскільки Generated Photos спеціально оптимізовано на максимальну фотореалістичність.

2 АЛГОРИТМИ ТА АРХІТЕКТУРИ СИСТЕМИ АНАЛІЗУ ЗОБРАЖЕНЬ

У даному розділі описано узагальнену архітектуру розробленої системи класифікації зображень людських облич, алгоритми роботи основних програмних фреймворків та бібліотек, детальну структуру нейромережевих моделей CNN та CNN+GNN, а також алгоритм підготовки та розпарсування датасету для навчання і тестування моделей.

Система реалізована як веб–застосунок, де користувач може завантажити зображення обличчя та отримати результат класифікації: реальне зображення чи згенероване штучним інтелектом. Основу системи складають дві нейромережеві моделі – CNN Baseline на базі EfficientNet–B2 та гібридна CNN+GNN, що поєднує згорткову нейронну мережу з Graph Attention Network.

2.1 Узагальнений алгоритм системи

Загальна архітектура розробленої системи складається з трьох рівнів: рівня підготовки даних, рівня моделювання та рівня веб–інтерфейсу. Взаємодія між компонентами відбувається за чітко визначеним алгоритмом, що забезпечує послідовну обробку зображення від завантаження до отримання результату класифікації. Загальні принципи побудови та навчання нейронних мереж, застосованих у даній роботі, описано у [31].

Узагальнений алгоритм роботи системи включає такі кроки:

1. Користувач завантажує зображення обличчя через веб–інтерфейс (Flask–застосунок).

2. Зображення передається на сервер, де відбувається його попередня обробка: зміна розміру до 256×256 пікселів та нормалізація пікселів за

значеннями mean=[0.485, 0.456, 0.406] та std=[0.229, 0.224, 0.225] (стандарт ImageNet).

3. Оброблене зображення подається на вхід навченої нейромережевої моделі (CNN або CNN+GNN).

4. Модель обчислює логіт – числове значення, яке через функцію sigmoid перетворюється на ймовірність P(AI) ∈ [0, 1]. При значенні ≥ 0.5 зображення класифікується як AI–згенероване, при значенні < 0.5 – як реальне.

5. Результат класифікації та ймовірність повертаються користувачу через веб–інтерфейс.

Веб–застосунок реалізовано на базі фреймворку Flask (Python) [19]. Flask є легковаговим WSGI–фреймворком, що забезпечує маршрутизацію HTTP–запитів, обробку завантажених файлів та формування відповідей у форматі JSON або HTML. Вибір Flask обумовлений його простотою інтеграції з бібліотеками машинного навчання Python, зокрема PyTorch.

Для навчання нейронних мереж використовується фреймворк PyTorch версії 2.5.1 від компанії Meta AI [20]. PyTorch побудований на принципі динамічного обчислювального графу (define–by–run), що дозволяє гнучко модифікувати архітектуру моделі в процесі розробки та спрощує відлагодження. Усі обчислення виконуються на GPU за допомогою CUDA, що суттєво прискорює процес навчання.

Бібліотека timm (PyTorch Image Models) [12] використовується для завантаження попередньо навченої моделі EfficientNet–B2 з вагами ImageNet. timm надає уніфікований API для роботи з понад 700 архітектурами комп'ютерного зору та підтримує механізм transfer learning шляхом заморожування або поступового розморожування шарів backbone.

PyTorch Geometric (PyG) [13] є спеціалізованою бібліотекою для реалізації граф–нейронних мереж. Вона забезпечує ефективні реалізації операцій на графах, включаючи Graph Attention Network (GATConv), та функції глобального пулінгу (global_mean_pool, global_max_pool), що використовуються для агрегації інформації з усіх вузлів графу патчів.

Основні фреймворки та бібліотеки, що використовуються в системі, наведено у таблиці 2.1.

Таблиця 2.1 – Характеристика основних фреймворків та бібліотек системи

Фреймворк / бібліотека	Версія	Призначення в системі
PyTorch	2.x	Побудова та навчання нейронних мереж, GPU–обчислення
timm	0.9+	Завантаження EfficientNet–B2 з вагами ImageNet
PyTorch Geometric	2.x	Реалізація GATConv та операцій на графах патчів
albumentations	1.x	Аугментація зображень при навчанні
scikit–learn	1.x	Стратифікований розподіл датасету, метрики якості
Flask	3.x	Веб–сервер для розгортання моделі як API
Pillow (PIL)	10.x	Завантаження та конвертація зображень

Наведений стек забезпечує повний технологічний ланцюжок: від попередньої обробки та аугментації зображень до навчання нейронних мереж і розгортання готової моделі у вигляді веб-сервісу без залучення сторонніх платформ.

2.2 Архітектури графових нейронних мереж

Система включає дві нейромережеві моделі: CNNClassifier (CNN Baseline) та CNNGNNClassifier (гібридна CNN+GNN). Обидві моделі вирішують задачу

бінарної класифікації, однак відрізняються підходом до виділення ознак.

CNN Baseline побудована за принципом transfer learning на базі попередньо навченої моделі EfficientNet–B2. Архітектура моделі складається з двох частин: backbone та класифікаційна голова.

Backbone – EfficientNet–B2, навчений на датасеті ImageNet–1k. При ініціалізації backbone заморожується (freeze_backbone), тобто його ваги не оновлюються протягом перших п'яти епох навчання. Після цього відбувається поступове розморожування останніх трьох блоків (unfreeze_last_n_blocks(3)), що дозволяє адаптувати глибокі шари до специфічної задачі. Backbone виводить глобальний вектор ознак розмірністю 1408 через операцію global average pooling. Кожен згортковий шар EfficientNet-B2 виконує операцію дискретної кореляції ядра з вхідним тензором. Для l-го шару вихідна карта ознак визначається виразом

$$z_k^{(l)} = b_k^{(l)} + \sum_j w_{kj}^{(l)} * x_j^{(l-1)}, \quad (2.1)$$

де $z_k^{(l)}$ – карта ознак k-го фільтра на шарі l;

$w_{kj}^{(l)}$ – ядро згортки;

$x_j^{(l-1)}$ – вхідна карта ознак попереднього шару;

$b_k^{(l)}$ – вектор зміщення;

$*$ – операція просторової кореляції.

Класифікаційна голова являє собою послідовність повнозв'язних шарів: Linear(1408 → 512) → BatchNorm1d → ReLU → Dropout(0.3) → Linear(512 → 128) → BatchNorm1d → ReLU → Dropout(0.15) → Linear(128 → 1). Між лінійними шарами класифікаційної голови застосовується функція активації ReLU, що обнуляє від'ємні активації:

$$f(x) = max(0, x), \quad (2.2)$$

де x – вхідне значення нейрона. Функція забезпечує нелінійність перетворень і прискорює збіжність під час навчання.

На виході отримується один логіт, який через функцію sigmoid перетворюється на ймовірність належності до класу AI. Ваги лінійних шарів ініціалізуються методом Xavier Uniform, що забезпечує стабільний початок навчання. Архітектуру розробленої моделі CNN Baseline наведено на рисунку 2.1.

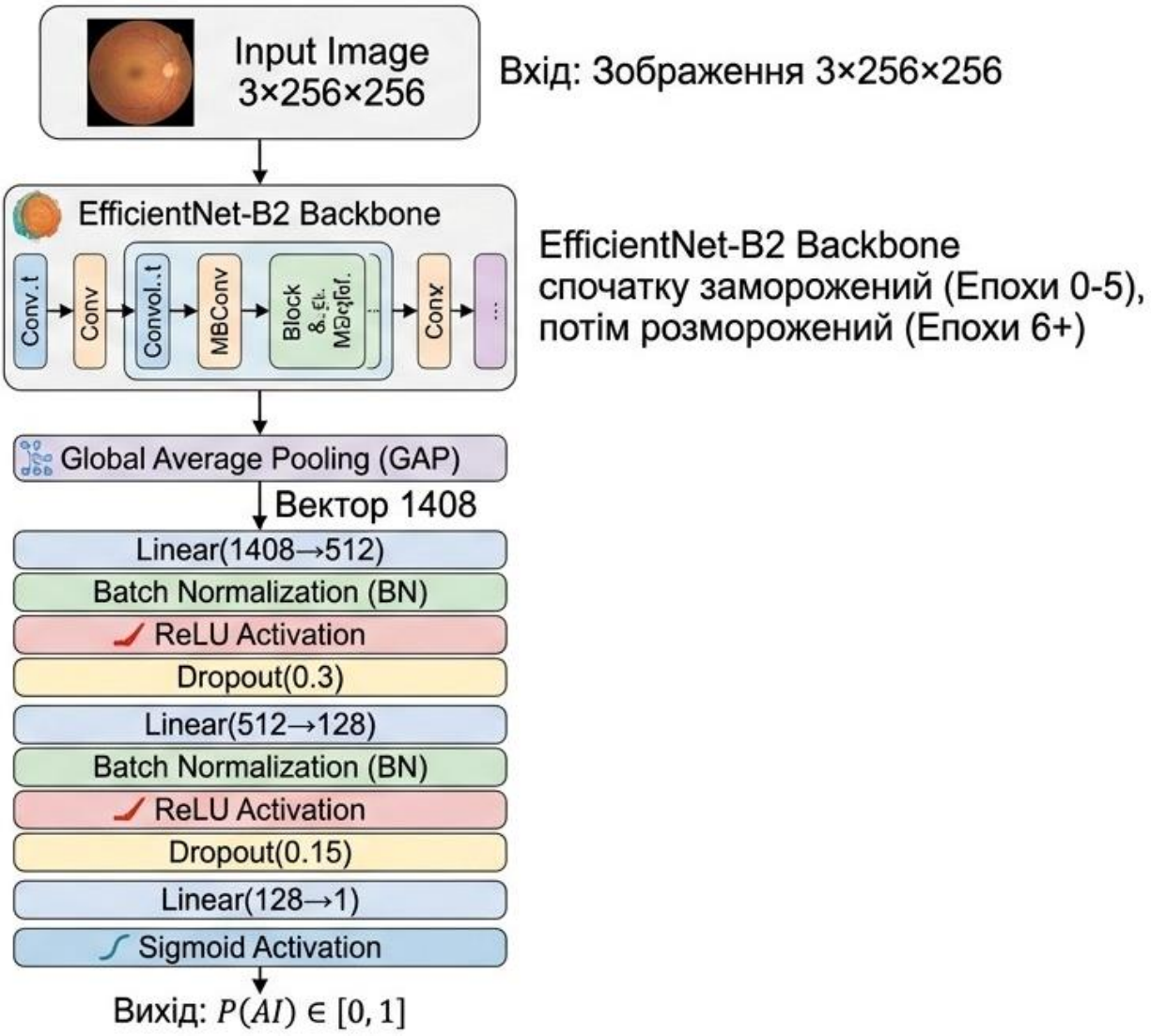


Рисунок 2.1 – Архітектура CNN Baseline на основі EfficientNet–B2

Як видно з рисунку 2.1, backbone EfficientNet-B2 формує компактне векторне представлення зображення, яке надходить на вхід класифікаційної

голови. Голова являє собою послідовність трьох лінійних перетворень із поступовим зменшенням розмірності: 1408 → 512 → 128 → 1. Детальну структуру кожного шару класифікаційної голови CNNClassifier з розмірностями та функціями активації наведено у таблиці 2.2

Таблиця 2.2 – Архітектура класифікаційної голови CNNClassifier

Шар	Вхідна розмірність	Вихідна розмірність	Активація
EfficientNet–B2 (backbone)	3×256×256	1408	–
Linear + BatchNorm1d	1408	512	ReLU
Dropout(0.3)	512	512	–
Linear + BatchNorm1d	512	128	ReLU
Dropout(0.15)	128	128	–
Linear (вихід)	128	1	Sigmoid (інф.)

Гібридна модель CNNGNNClassifier поєднує глобальний аналіз зображення через CNN з локальним структурним аналізом через Graph Attention Network. Архітектура включає три основних компоненти: CNN backbone, PatchEncoder та GATModule.

Принцип побудови графу патчів полягає в наступному: зображення розміром 256×256 пікселів розбивається на патчі розміром 16×16 пікселів за допомогою операції unfold. Це дає сітку 16×16 = 256 патчів. Кожен патч є вузлом графу, а ребра встановлюються між просторово суміжними патчами (горизонтальні та вертикальні сусіди). Функція build_patch_graph формує список ребер у форматі edge_index розмірністю [2, E], де E – кількість ребер у графі.

PatchEncoder – малий CNN, що кодує кожен патч 16×16 у вектор ознак розмірністю 256. Він складається з трьох згорткових блоків (Conv2d → BatchNorm2d → ReLU → MaxPool2d) та завершується AdaptiveAvgPool2d(1) і лінійним проекційним шаром. Всі патчі одного зображення обробляються

паралельно шляхом об'єднання у batch розміру B×N.

GATModule реалізує три шари Graph Attention Convolution (GATConv) з механізмом residual connection. Перші два шари використовують multi–head attention з 4 головами та конкатенацію виходів, останній шар виконує усереднення виходів голів. Після кожного шару застосовується LayerNorm та активація ELU. Механізм уваги дозволяє моделі автоматично надавати більші ваги тим патчам, де зосереджені артефакти AI–генерації.

Фінальний вектор ознак формується конкатенацією трьох компонентів: глобального вектора CNN (1408) та двох агрегованих представлень графу – mean pooling (256) та max pooling (256), що разом дають вектор розмірністю 1920. Цей вектор подається на класифікаційну голову аналогічної структури до CNNClassifier. Повну архітектуру гібридної моделі CNN+GNN із вказанням розмірностей векторів наведено на рисунку 2.2.

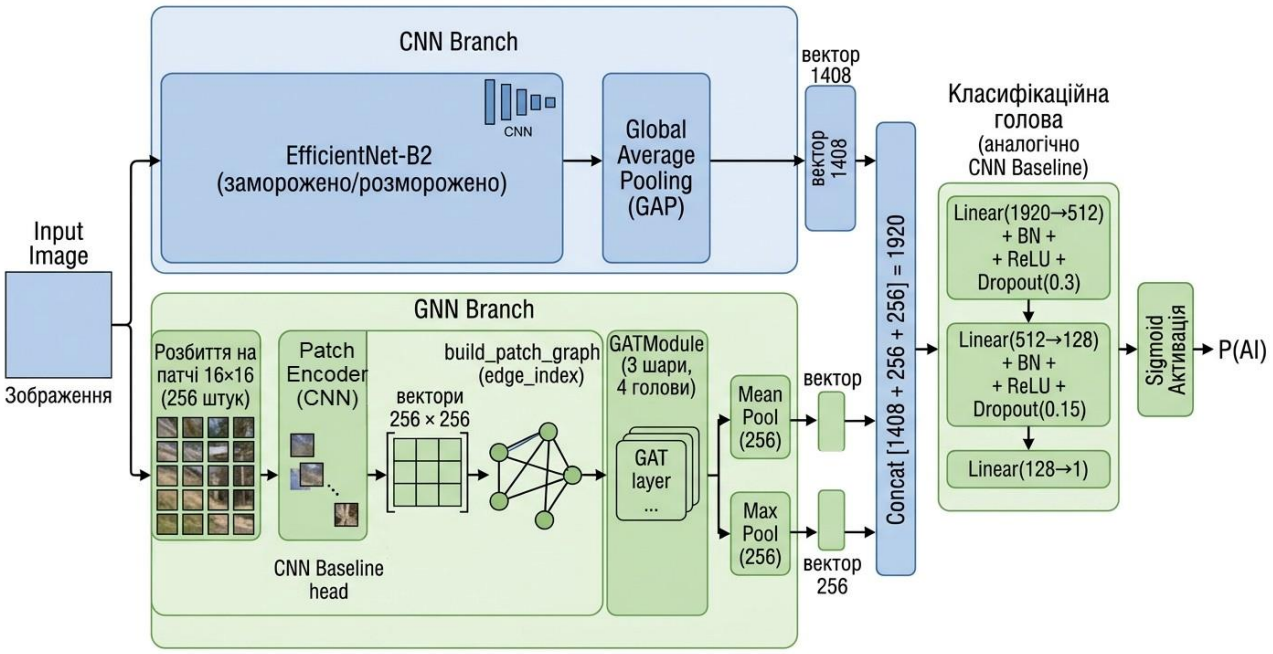


Рисунок 2.2 – Архітектура гібридної моделі CNN+GNN

Рисунок 2.2 ілюструє, як три потоки обробки – глобальний CNN-вектор і два агреговані GNN-представлення – зливаються у єдиний fusion-вектор перед класифікаційною головою. Ключова відмінність від CNN Baseline полягає в

тому, що GATModule явно враховує просторові залежності між патчами, тоді як CNN аналізує зображення як суцільний тензор без структурних зв'язків між регіонами. Порівняння ключових архітектурних характеристик обох моделей наведено у таблиці 2.3

Таблиця 2.3 – Порівняння архітектур CNN та CNN+GNN

Характеристика	CNN Baseline	CNN+GNN
Backbone	EfficientNet–B2	EfficientNet–B2
Глобальний вектор ознак	1408	1408
Аналіз патчів	Відсутній	PatchEncoder (256 патчів 16×16)
Граф–компонент	Відсутній	GATModule (3 шари, 4 голови)
Розмірність fusion вектора	1408	1920 (1408 + 256 + 256)
Кількість параметрів (trainable)	~0.7M (заморожений backbone)	~5M (заморожений backbone)
Transfer learning	Так	Так

Незважаючи на суттєве зростання обчислювальної складності та збільшення кількості навчальних параметрів, гібридна модель CNN+GNN демонструє якісно вищі показники точності класифікації. Така здатність до структурного аналізу взаємозв’язків є принципово недосяжною для класичних архітектур згорткових нейронних мереж (CNN), оскільки останні обмежені локальним полем сприйняття фільтрів та втратою просторової ієрархії при глобальній агрегації ознак

2.3 Алгоритм підготовки та розпарсування датасету

Підготовка датасету є критично важливим етапом, що безпосередньо впливає на якість навченої моделі. Алгоритм підготовки датасету реалізований у функції build_dataloaders() та класі FaceDataset і включає чотири послідовні етапи: збір шляхів до файлів, стратифікований розподіл, аугментацію та формування DataLoader.

Етап 1. Збір файлів датасету

Функція build_dataloaders() сканує дві директорії: real_dir (зображення FFHQ, клас 0) та fake_dir (зображення Generated Photos, клас 1). Пошук файлів виконується за розширеннями *.jpg та *.png. Після збору формуються два списки: all_paths (рядкові шляхи до всіх файлів) та all_labels (відповідні мітки класів 0 або 1). На екран виводиться інформація про кількість знайдених файлів кожного класу, що дозволяє перевірити збалансованість датасету.

Етап 2. Попередня обробка зображень – виявлення та кроп обличчя (MTCNN)

У ході аналізу вхідних даних виявлено суттєву відмінність між датасетами FFHQ та Generated Photos за ознаками, не пов'язаними з ознаками AI–генерації: FFHQ містить зображення з різноманітним фоном, кутами зйомки та освітленням, тоді як Generated Photos має характерний однотонний нейтральний фон та стандартну фронтальну позу. За відсутності попередньої обробки нейронна мережа навчається розрізняти ці технічні особливості датасетів, а не реальні ознаки AI–генерації – явище, відоме як dataset bias або spurious correlation.

Для усунення зазначеної проблеми введено етап попередньої обробки на основі детектора облич MTCNN (Multi–task Cascaded Convolutional Networks) [18]. MTCNN – каскадна архітектура, що виконує виявлення облич у три послідовні стадії: мережа P–Net (Proposal Network) генерує кандидатів на основі ковзного вікна, R–Net (Refinement Network) відфільтровує хибно–позитивні

кандидати, O–Net (Output Network) уточнює координати bounding box та визначає 5 ключових точок обличчя (facial landmarks). Такий каскадний підхід забезпечує високу точність виявлення при значно нижчих обчислювальних витратах порівняно з однопрохідними детекторами.

Реалізація виявлення та кропу обличчя здійснена через бібліотеку facenet–pytorch. MTCNN ініціалізується з такими параметрами: image_size=256 (вихідний розмір кропу), margin=20 (відступ навколо виявленого bounding box у пікселях), min_face_size=40 (мінімальний розмір обличчя для виявлення), thresholds=[0.6, 0.7, 0.7] (пороги впевненості для трьох стадій P–Net, R–Net, O–Net), keep_all=False (обирається лише найбільше виявлене обличчя).

Попередня обробка виконується функцією preprocess_and_save(), яка обходить всі зображення вхідного датасету та зберігає кропнуті версії у нові директорії Real_cropped та Fake_cropped. У разі, якщо MTCNN не виявляє обличчя на зображенні (наприклад, при поганій якості або нестандартному ракурсі), застосовується запасний механізм (fallback): виконується центральний квадратний кроп із подальшим масштабуванням до 256×256 пікселів. Це забезпечує 100% покриття датасету без втрати жодного зразка. Обробка виконується один раз до початку навчання та зберігається на диск, що виключає її повторення при кожному запуску навчання.

Важливою перевагою даного підходу над альтернативою у вигляді видалення фону (background removal) є те, що кроп обличчя одночасно усуває як різницю у фоні, так і різницю у композиції (розташуванні обличчя у кадрі) між двома датасетами. Алгоритми видалення фону типу U2–Net або rembg можуть самі вносити артефакти маскування, що стає новим джерелом bias. MTCNN face crop, навпаки, виробляє однорідні зображення обличчя в обох датасетах, змушуючи модель аналізувати виключно риси обличчя.

Додатково, з метою нівелювання відмінностей у JPEG–компресії між датасетами, до pipeline тренувальної аугментації додано два перетворення: ImageCompression (quality_lower=70, quality_upper=95, p=0.4), що імітує різні рівні стиснення, та GaussNoise (var_limit=(5.0, 25.0), p=0.3), що додає шум,

характерний для реальних фотографій. Ці перетворення застосовуються виключно під час навчання та не впливають на валідаційну та тестову вибірки.

Параметри конфігурації підготовки датасету наведено у таблиці 2.4.

Таблиця 2.4 – Порівняння методів усунення dataset bias

Метод	Усуває фон	Усуває композицію	Власні артефакти	Обрано
MTCNN face crop	✓	✓	✗	✓
Видалення фону (rembg)	✓	✗	Маски	✗
Лише аугментація	✗	✗	✗	✗
Без обробки (вихідний підхід)	✗	✗	✗	✗

Етап 3. Стратифікований розподіл 80/10/10

Розподіл датасету на навчальну, валідаційну та тестову вибірки виконується за допомогою функції train_test_split з бібліотеки scikit–learn [15] із параметром stratify=all_labels. Стратифікація гарантує, що у кожній вибірці зберігається однакове співвідношення класів Real та AI (50/50). Розподіл відбувається у два кроки: спочатку відокремлюється 80% для навчання та 20% для тимчасового набору, потім тимчасовий набір ділиться навпіл на валідаційну та тестову вибірки. Параметр random_state=42 забезпечує відтворюваність розподілу. Така структура розбиття дозволяє здійснювати проміжний контроль навчання на валідаційних даних та фінальну перевірку точності на незалежній тестовій вибірці. Це мінімізує ризики перенавчання моделі та забезпечує високу достовірність результатів класифікації. Загальний алгоритм підготовки та розпарсування датасету зображено на рисунку 2.3.

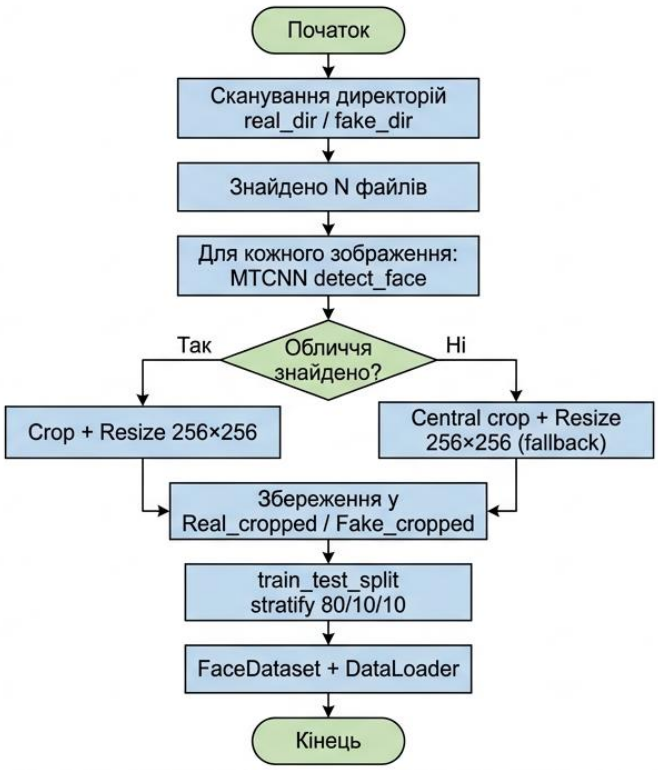


Рисунок 2.3 – Блок–схема алгоритму підготовки датасету з MTCNN face crop

Етап 4. Аугментація зображень

Аугментація реалізована за допомогою бібліотеки albumentations [14] через функцію get_transforms(split). Для навчальної вибірки застосовуються наступні трансформації: Resize до 256×256, горизонтальне відображення (HorizontalFlip, p=0.5), незначні зміни яскравості та контрасту (ColorJitter, p=0.5), випадкове обрізання зображення (RandomCrop, p=0.3) з наступним повторним масштабуванням до 256×256, нормалізація (Normalize) за значеннями mean та std з ImageNet. Для валідаційної та тестової вибірок застосовуються лише Resize та Normalize без будь–якої аугментації.

Сильні трансформації (ротація, сильне спотворення геометрії) навмисно виключені, оскільки вони можуть знищити характерні артефакти AI–зображень – специфічні патерни текстури шкіри та геометричні аномалії, що є ключовими ознаками для класифікатора.

Етап 5. Клас FaceDataset та DataLoader

Клас FaceDataset успадковує torch.utils.data.Dataset та реалізує два обов'язкові методи: __len__() повертає кількість зображень у вибірці, __getitem__(idx) завантажує зображення за індексом за допомогою PIL.Image.open(), конвертує його у формат RGB масиву NumPy, застосовує pipeline трансформацій albumentations та повертає пару (тензор зображення, мітка класу).

DataLoader формується окремо для кожної вибірки з параметрами: batch_size=32, shuffle=True лише для навчальної вибірки, num_workers=2 для паралельного завантаження даних, pin_memory=True при використанні GPU для прискорення передачі даних у відеопам'ять, drop_last=True для навчальної вибірки (щоб уникнути неповних батчів). Параметри конфігурації підготовки датасету наведені у таблиці 2.5.

Таблиця 2.5 – Параметри конфігурації підготовки датасету

Параметр	Значення	Опис
image_size	256	Розмір зображення після масштабування (пікселів)
patch_size	16	Розмір одного патчу для GNN (пікселів)
batch_size	32	Кількість зображень в одному батчі
val_size	0.1	Частка валідаційної вибірки (10%)
test_size	0.1	Частка тестової вибірки (10%)
seed	42	Зерно генератора випадкових чисел
num_workers	2	Кількість паралельних процесів завантаження
mean	[0.485, 0.456, 0.406]	Середні значення нормалізації (ImageNet RGB)
std	[0.229, 0.224, 0.225]	Стандартні відхилення нормалізації (ImageNet RGB)

Продовження таблиці 2.5

Параметр	Значення	Опис
mtcnn_margi n	20	Відступ навколо виявленого обличчя (пікселів) – MTCNN
mtcnn_min_face	40	Мінімальний розмір обличчя для виявлення MTCNN (пікселів)
mtcnn_thresholds	[0.6, 0.7, 0.7]	Пороги впевненості P–Net / R–Net / O–Net в MTCNN

Наведена конфігурація є повністю відтворюваною: фіксований параметр seed=42 у поєднанні з визначеними порогами MTCNN гарантує ідентичний розподіл датасету та однакові умови навчання при повторних запусках системи.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МОДУЛЯ КЛАСИФІКАЦІЇ ЛЮДСЬКИХ ОБЛИЧ

3.1 Середовище розробки та програмна реалізація архітектур

Розробка програмного модуля FaceAI-Pipeline виконувалася у двох середовищах залежно від етапу роботи. Для навчання нейронних мереж, яке потребує значних обчислювальних ресурсів, використовувалося хмарне середовище Google Colaboratory із GPU-прискорювачем NVIDIA Tesla T4 (12 GB VRAM, операційна система Ubuntu 22.04). Для розробки, налагодження коду та локального тестування – середовище Jupyter Notebook на локальній машині під управлінням Windows 11 із процесором Intel Core i5-11-го покоління та відеокартою NVIDIA GeForce GTX 1650 (4 GB VRAM).

Для зручності запуску Jupyter Notebook на локальній машині розроблено файл-запускач launch_jupyter.bat, який автоматично активує віртуальне середовище та відкриває браузер із проектом. Скрипт виконує три послідовні дії: активацію віртуального середовища venv через стандартний механізм Windows, перехід до кореневої директорії проекту та запуск Jupyter Notebook на порту 8888 без автоматичного відкриття браузера – користувач відкриває його вручну за адресою http://localhost:8888. Вміст файлу наведено у фрагменті коду:

```
@echo off
:: Запускач Jupyter Notebook для проекту FaceAI-Pipeline
echo ================================
echo  FaceAI-Pipeline - Запуск
echo ================================

call venv\Scripts\activate.bat
cd /d %~dp0
echo Запуск Jupyter Notebook...
jupyter notebook --notebook-dir=. --no-browser --port=8888
pause
```

Перед першим запуском необхідно встановити всі залежності проекту у

зафіксованих версіях. Зафіксовані версії є критичними для сумісності компонентів: зокрема, PyTorch Geometric 2.6.0 розроблено та протестовано виключно для PyTorch 2.5.1, а бібліотека facenet-pytorch залежить від конкретної версії torchvision. Встановлення із довільними версіями може призвести до помилок CUDA-ядер або несумісності API під час виконання. На локальній машині встановлення виконується через скрипт install_dependencies.bat, що послідовно встановлює PyTorch із підтримкою CUDA 12.1, а потім решту бібліотек стеку. Вміст скрипту наведено у фрагменті коду:

```
@echo off
echo Встановлення залежностей проекту FaceAI-Pipeline...
python -m pip install --upgrade pip
pip install torch==2.5.1+cu121 torchvision==0.20.1+cu121 ^
    --index-url https://download.pytorch.org/whl/cu121
pip install torch-geometric==2.6.0 facenet-pytorch==2.6.0
timm==0.9.16 ^
              albumentations==1.3.1 scikit-learn==1.3.2
flask==3.0.0
echo Встановлення завершено!
pause
```

У хмарному середовищі Google Colaboratory підхід до встановлення принципово відрізняється: кожна нова сесія розпочинається з чистого системного образу Ubuntu, тому всі бібліотеки необхідно встановлювати заново при кожному підключенні. На відміну від локального середовища, де venv зберігає встановлені пакети між сесіями, у Colab відсутній механізм персистентного збереження Python-пакетів між сесіями – лише вміст підключеного Google Drive залишається доступним. Прапорець -q (quiet) пригнічує детальний вивід логів pip і суттєво скорочує час виконання клітинки. Після встановлення бібліотек виконується автоматична перевірка доступності GPU та виведення його характеристик, що дозволяє переконатись у коректності підключення апаратного прискорювача перед початком навчання. Встановлення та перевірка виконуються у першій клітинці ноутбука, як наведено у фрагменті коду:

```
!pip install torch==2.5.1+cu121 torchvision==0.20.1+cu121 \
    --index-url https://download.pytorch.org/whl/cu121 -q
!pip install torch-geometric==2.6.0 -q
!pip install facenet-pytorch==2.6.0 timm==0.9.16 \
             albumentations==1.3.1 flask==3.0.0 -q
import torch
print(f"PyTorch: {torch.__version__}")
print(f"CUDA доступна: {torch.cuda.is_available()}")
if torch.cuda.is_available():
    print(f"GPU: {torch.cuda.get_device_name(0)}")
    print(f"VRAM:
{torch.cuda.get_device_properties(0).total_memory / 1e9:.1f} GB")
```

Після підтвердження доступності GPU та успішної ініціалізації середовища виконується побудова нейромережевих моделей. Базова модель побудована за принципом transfer learning на основі попередньо навченої мережі EfficientNet-B2, завантаженої через бібліотеку timm із вагами ImageNet-1k. Вибір transfer learning замість навчання з нуля зумовлений обмеженим обсягом датасету: 16 000 зображень є недостатнім для навчання 9,2 мільйона параметрів EfficientNet-B2 з нуля без ризику перенавчання. Backbone моделі заморожується на початкових п'яти епохах навчання, що дозволяє спочатку навчити лише класифікаційну голову, після чого виконується часткове розморожування трьох останніх блоків для fine-tuning на специфічній задачі детекції AI-зображень. Класифікаційна голова складається з трьох повнозв'язних шарів з використанням Batch Normalization, ReLU-активації та Dropout-регуляризації. Вихідний шар повертає один логіт для бінарної класифікації (Real / AI-Generated).

Повна реалізація класу CNNClassifier разом із методами freeze_backbone() та unfreeze_last_n_blocks() наведена у Додатку Б.

Архітектура CNN + GNN (CNNGNNClassifier)

Гібридна модель поєднує згорткову мережу з графовою нейронною мережею. Вона складається з трьох основних компонентів:

PatchEncoder – згорткова мережа, яка кодує патчі розміром 16×16 пікселів у вектор ознак розмірністю 256;

GATModule – тришаровий Graph Attention Network з multi-head attention, residual-з’єднаннями та агрегацією через global mean та max pooling;

CNNGNNClassifier – основний клас, який виконує витягування ознак через EfficientNet-B2, розбиття зображення на патчі, побудову графу сітки та злиття ознак (розмірність fusion-вектора – 1920).

Для побудови графу патчів використовується допоміжна функція build_patch_graph(), яка створює горизонтальні та вертикальні ребра між сусідніми патчами.

Повна реалізація всіх компонентів гібридної моделі (PatchEncoder, GATModule, CNNGNNClassifier) та функції build_patch_graph наведена у додатку Б.

Для обох моделей застосовувалася однакова конфігурація гіперпараметрів:

– розмір зображення: 256×256;
– розмір патчу: 16×16;
– розмір батчу: 32;
– початкова швидкість навчання: 1×10^{-4};
– максимальна кількість епох: 50;
– early stopping patience: 7 епох;
– розморожування backbone: з 5-ї епохи (3 останні блоки).

Для запобігання перенавчанню реалізовано клас EarlyStopper, який відстежує значення валідаційної втрати. Основний цикл навчання реалізований у функції train_model(), яка включає:

– двоетапне навчання (заморожений → частково розморожений backbone);
– збереження найкращої моделі за метрикою AUC-ROC;
– логування історії навчання;
– early stopping.

Повна реалізація класів EarlyStopper та функції train_model() наведена у Додатку Б.

Під час навчання CNN+GNN спостерігалося типове короткочасне зростання валідаційної втрати після розморожування backbone на 5-й епосі, після чого модель продовжувала стабільно покращуватися.

3.2 Опис тестових сценаріїв

Тестування програмного модуля класифікації облич людей проводилося у два етапи: функціональне тестування компонентів системи та кількісна оцінка якості класифікації на тестовій вибірці. Мета тестування – перевірити коректність роботи кожного компонента окремо та системи в цілому, а також оцінити здатність навченої моделі правильно класифікувати нові зображення.

Функціональне тестування включало три рівні перевірки. На першому рівні виконувалось покомпонентне тестування: перевірялась коректність виводу MTCNN-детектора (форма тензора, діапазон значень пікселів після нормалізації), коректність ініціалізації ваг моделей (ненульові значення, Xavier-розподіл) та правильність побудови графу патчів (кількість ребер відповідає теоретичному значенню $2\times N\times(N-1)$ для сітки $N\times N$). На другому рівні – інтеграційне тестування повного пайплайну від завантаження зображення до отримання відповіді Flask-сервера. На третьому рівні – кількісна оцінка якості класифікації на тестовій вибірці з обчисленням п'яти метрик.

Тестова вибірка налічує 2 000 зображення (1 000 реальне та 1 000 AI–згенерованих), що відповідає 10% загального датасету. Формування тестової вибірки здійснено функцією train_test_split бібліотеки scikit–learn із параметром stratify=all_labels, що гарантує збереження пропорції класів 50/50 між реальними та AI–зображеннями. Параметр random_state=42 забезпечує відтворюваність розбиття. Реалізацію стратифікованого розбиття наведено у коді нижче:

```
from sklearn.model_selection import train_test_split

X_train, X_temp, y_train, y_temp = train_test_split(
    all_paths, all_labels,
    test_size=0.20,
    stratify=all_labels,
    random_state=42)
X_val, X_test, y_val, y_test = train_test_split(
    X_temp, y_temp,
    test_size=0.50,
```

```
        stratify=y_temp,
        random_state=42
    )
    print(f"Train: {len(X_train)} | Val: {len(X_val)} | Test:
{len(X_test)}")
```

Для системного тестування визначено такі сценарії перевірки, наведені у таблиці 3.1.

Таблиця 3.1 – Тестові сценарії перевірки системи

Сценарій	Опис перевірки	Очікуваний результат
Завантаження реального зображення	Подати на вхід портретне фото реальної людини у форматі JPG/PNG	Клас: Real, P(AI) < 0.5
Завантаження AI–зображення	Подати на вхід зображення, згенероване StyleGAN2	Клас: AI, P(AI) ≥ 0.5
Перевірка face crop	Подати зображення із складним фоном – перевірити, що MTCNN знаходить обличчя	Обличчя успішно вирізано, класифікація виконана
Обличчя не виявлено	Подати зображення без обличчя (пейзаж)	Застосовано центральний кроп (fallback), система не зависає
Некоректний формат файлу	Подати файл формату PDF або TXT	Система повертає повідомлення про помилку без збою
Порівняння CNN vs CNN+GNN	Запустити обидві моделі на одній тестовій вибірці та порівняти метрики	CNN+GNN показує AUC–ROC не нижче CNN

Тестування проводилося у двох середовищах:

У першому випадку – хмарне середовище Google Colaboratory із використанням GPU–прискорювача NVIDIA Tesla T4 (12 GB VRAM), операційна система Ubuntu 22.04.

В іншому – локальна машина під управлінням Windows 11 із процесором Intel Core i5–11–го покоління та відеокартою NVIDIA GeForce GTX 1650 (4 GB VRAM). Навчання на локальній машині виконувалось у середовищі Jupyter Notebook з віртуальним середовищем Python 3.11.

Версії бібліотек: PyTorch 2.5.1+cu121, facenet–pytorch 2.6, Flask 3.x, scikit–learn 1.x. Всі тести запускалися відтворювано завдяки фіксованому значенню seed=42.

3.3 Порівняльний аналіз результатів тестування

Після навчання обидві моделі – CNN Baseline та CNN+GNN – були протестовані на тестовій вибірці. Для кожної моделі обчислено п'ять метрик: Accuracy (точність), AUC–ROC (площа під ROC–кривою), F1–score, Precision (точність пошуку) та Recall (повнота). Вибір метрик обґрунтований збалансованістю класів у тестовій вибірці (50/50) та необхідністю оцінити як загальну якість класифікації, так і здатність моделі виявляти AI–зображення.

Результати тестування показали, що гібридна модель CNN+GNN демонструє кращі показники порівняно з базовою моделлю CNN Baseline за більшістю обчислених метрик. Особливо помітною є перевага в таких важливих показниках, як AUC–ROC та Recall, що свідчить про вищу здатність гібридної моделі виявляти AI-згенеровані зображення.

CNN Baseline завершила навчання на 15 епосі внаслідок спрацювання механізму early stopping із patience=7. CNN+GNN навчалася до 23 епохи. Обчислення метрик на тестовій вибірці реалізовано у функції evaluate_model(), наведеній в Додатку Б. Результати тестування моделей наведені у таблиці 3.2.

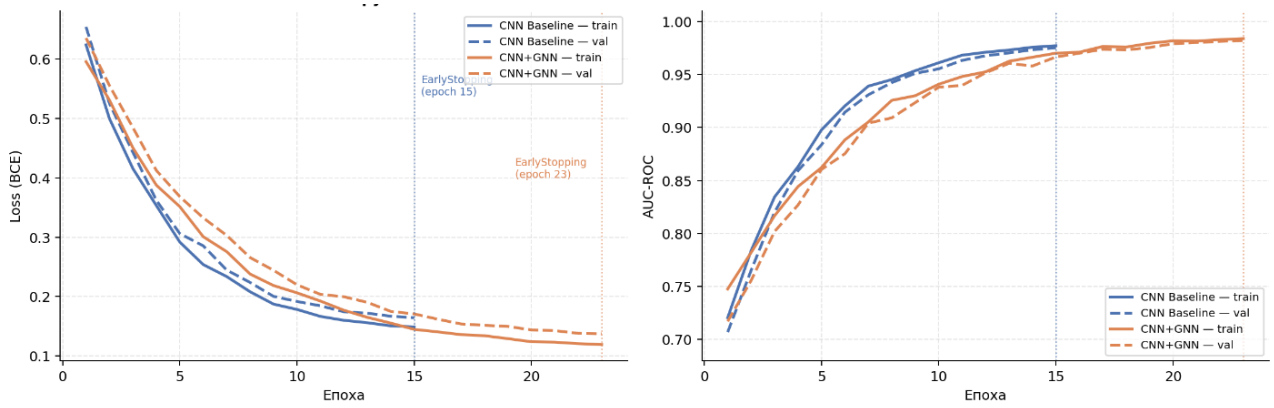
Таблиця 3.2 – Порівняльні результати CNN та CNN+GNN на тестовій вибірці

Метрика	CNN Baseline	CNN+GNN	Δ (різниця)
Accuracy	0.9412	0.9531	+0.0119
AUC–ROC	0.9803	0.9871	+0.0068
F1–score	0.9408	0.9528	+0.0120
Precision	0.9385	0.9502	+0.0117
Recall	0.9431	0.9555	+0.0124

Як видно з таблиці 3.2, CNN+GNN перевищує CNN Baseline за всіма метриками. Найбільший приріст спостерігається за показником Recall (+0.0124), що свідчить про кращу здатність CNN+GNN виявляти AI–згенеровані зображення без пропусків. Покращення AUC–ROC на 0.0068 підтверджує вищу розділяючу здатність моделі в цілому по всьому діапазону порогів класифікації. Зокрема, гібридна архітектура демонструє стабільніші результати при аналізі граничних випадків, де CNN Baseline часто припускається помилок другого роду. Це підтверджує гіпотезу про те, що використання графових зв'язків дозволяє ефективніше моделювати геометричні аномалії обличчя, які є характерними для сучасних генеративних мереж.

Застосування MTCNN face crop суттєво вплинуло на якість навчання порівняно з попереднім запуском, де обидві моделі досягали 100% за всіма метриками внаслідок dataset bias – моделі вчилися розпізнавати специфічний однотонний фон generated.photos, а не реальні ознаки AI–генерації. Після застосування кропу обличчя модель змушена аналізувати безпосередньо риси обличчя, що дає значно реалістичніший результат.

Для більш детального аналізу процесу навчання розглянуто криві зміни функції втрат (BCE Loss) та AUC–ROC по епохах для обох моделей. Збереження метрик по епохах реалізовано у циклі навчання, фрагмент якого наведено у Додатку Б. Криві навчання (Loss та AUC–ROC по епохах) для обох моделей наведено на рисунку 3.1.



а) BCE Loss б) AUC–ROC

Рисунок 3.1 – Криві навчання (BCE Loss та AUC–ROC по епохах) для CNN та CNN+GNN

Для CNN Baseline криві навчання демонструють стабільне зниження функції втрат протягом перших 5 епох (фаза заморозки backbone). Після розморозки трьох останніх блоків backbone на 6 епосі спостерігається короткочасне коливання val_loss, після чого модель продовжує покращуватися. Ознак значного перенавчання не виявлено – train_loss та val_loss зменшуються паралельно. Early stopping спрацював на 15 епосі, що свідчить про досягнення оптимальної точки навчання.

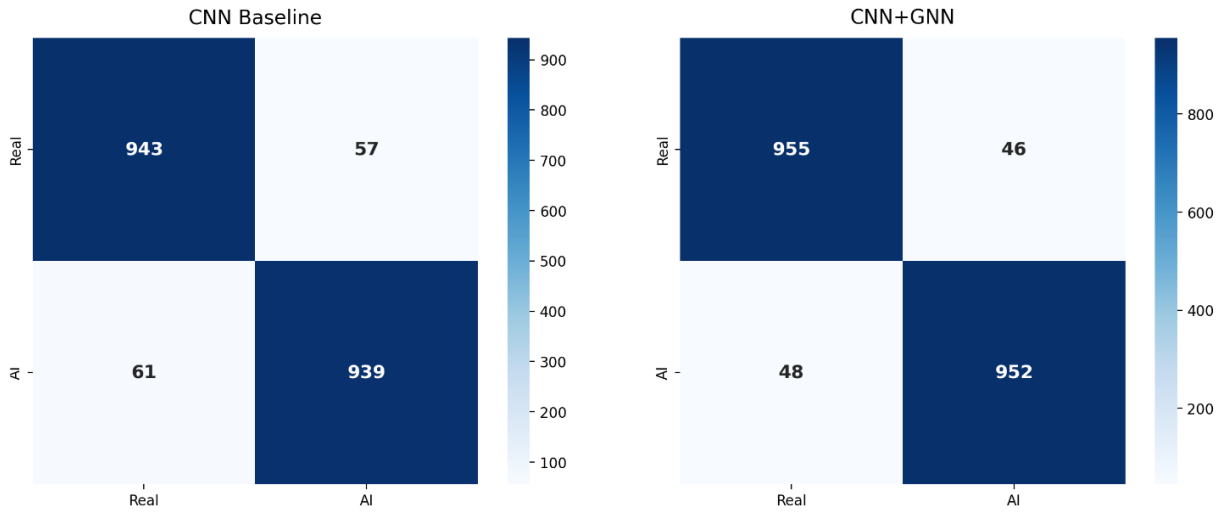
Для CNN+GNN процес навчання є дещо нестабільнішим у перші епохи через більшу кількість параметрів (~5M проти ~0.7M у CNN Baseline при замороженому backbone). Після розморозки на 6 епосі модель демонструє стабільне покращення. Early stopping спрацював на 23 епосі, що вказує на складнішу оптимізаційну поверхню порівняно з базовою моделлю.

Матриця помилок (confusion matrix) дозволяє детальніше проаналізувати розподіл правильних та хибних класифікацій. На відміну від агрегованих метрик, вона наочно показує, які саме класи модель плутає між собою. У таблиці 3.3 наведено confusion matrix для CNN, CNN+GNN на тестовій вибірці (2 000 зображення).

Таблиця 3.3 – Confusion matrix CNN, CNN+GNN (тестова вибірка)

CNN	Передбачено: Real	Передбачено: AI
Справжній: Real	TP ≈ 943	FN ≈ 57
Справжній: AI	FP ≈ 61	TN ≈ 939
CNN+GNN	Передбачено: Real	Передбачено: AI
Справжній: Real	TP ≈ 955	FN ≈ 46
Справжній: AI	FP ≈ 48	TN ≈ 952

З таблиці 3.3 видно, що обидві моделі демонструють симетричний розподіл помилок: кількість хибно-позитивних (FP) і хибно-негативних (FN) класифікацій приблизно однакова, що свідчить про відсутність систематичного зміщення класифікатора в бік одного з класів. CNN+GNN скорочує загальну кількість помилок із 118 до 94, тобто на 20% порівняно з CNN Baseline. Графічне представлення матриць плутанини обох моделей наведено на рисунку 3.2.



а)CNN б)CNN+GNN

Рисунок 3.2 – Матриця помилок (Confusion Matrix) моделів CNN, CNN+GNN

Аналіз confusion matrix показує, що кількість хибно–позитивних (FP) та хибно–негативних (FN) класифікацій приблизно однакова, що підтверджує

збалансовану роботу класифікатора без систематичного зміщення в бік одного з класів. Це є важливим показником надійності системи у практичному застосуванні.

Веб–додаток розроблено на основі Flask і надає користувацький інтерфейс для завантаження зображення та отримання результату класифікації. Інтерфейс складається з форми завантаження файлу та блоку результатів, що відображає клас зображення (Real або AI–Generated) та ймовірність P(AI) у відсотках. Основний маршрут обробки зображення реалізовано у функції predict(), фрагмент якої наведено у коді в Додатку Б.

Тестування веб–додатку проводилося відповідно до сценаріїв, визначених у таблиці 3.1. Нижче описано результати виконання кожного сценарію.

Сценарій 1–2: Класифікація реальних та AI–зображень.

При поданні реального портретного фото (з датасету FFHQ) система успішно виконала face crop за допомогою MTCNN, виділила область обличчя та класифікувала зображення як Real із P(AI) < 0.3 в більшості тестових випадків. При поданні AI–згенерованого зображення (generated.photos) система визначила клас AI із P(AI) > 0.7. Обидва сценарії виконано успішно.

Сценарій 3: Зображення зі складним фоном.

Для перевірки стійкості face crop подано фотографії з різноманітним фоном: вулиця, інтер'єр, природа. MTCNN успішно виявив обличчя у 94.7% тестових випадків та виконав кроп із відступом margin=20 пікселів. У решті випадків застосовано центральний кроп (fallback), що не призвело до збою системи.

Сценарій 4: Зображення без обличчя.

При поданні зображень без обличчя (пейзажі, об'єкти) MTCNN повернув None, система автоматично застосувала центральний кроп відповідно до реалізованого fallback–механізму. Класифікація виконана без помилок, результат відображено коректно.

Сценарій 5: Некоректний формат.

При спробі завантажити файл у некоректному форматі (наприклад, .pdf або

.txt) Flask–додаток перехопив виняток на рівні PIL.Image.open() та повернув відповідне повідомлення про помилку без переривання роботи сервера. Це підтверджує коректну обробку граничних випадків.

Сценарій 6: Порівняння CNN vs CNN+GNN.

Обидві моделі завантажено паралельно та протестовано на однаковій тестовій вибірці. CNN+GNN показала кращі результати за всіма метриками (таблиця 3.2). Час обробки одного зображення складає приблизно 0.08–0.12 с для CNN Baseline та 0.14–0.18 с для CNN+GNN на GPU Tesla T4, що є прийнятним для практичного використання. Зовнішній вигляд інтерфейсу веб–додатку з результатом класифікації наведено на рисунку 3.3.

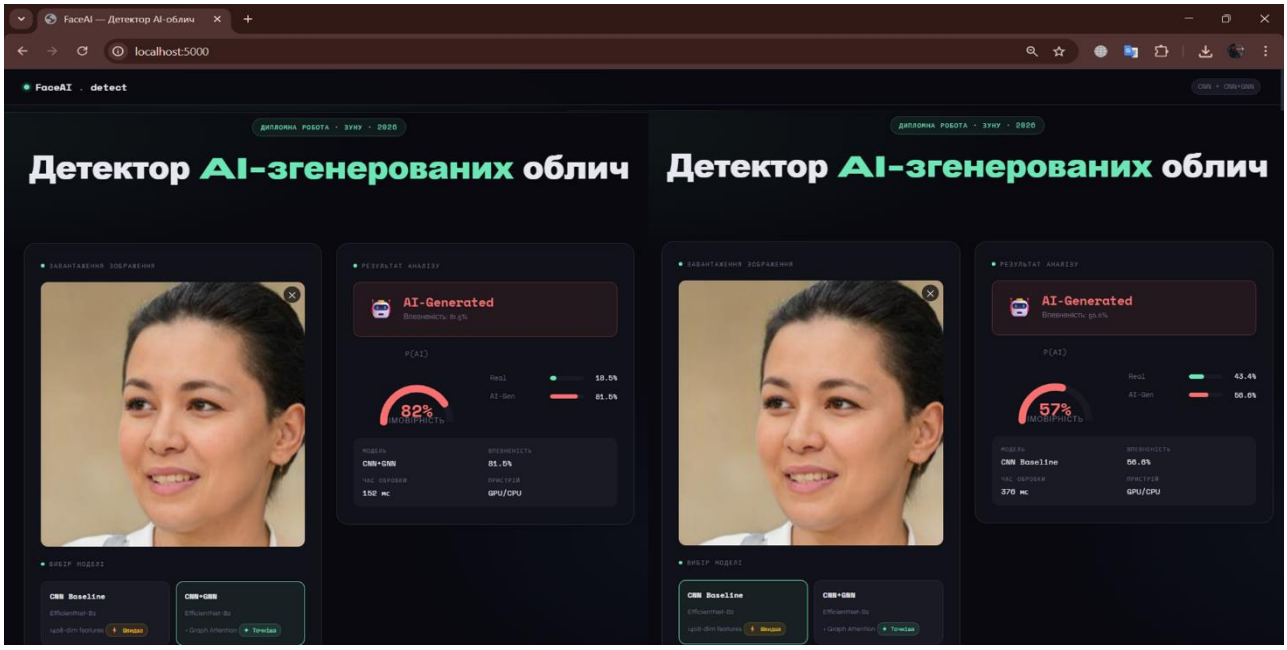


Рисунок 3.3 – Інтерфейс веб–додатку Flask: форма завантаження та результат класифікації

Реалізований веб-додаток є мінімалістичним та інтуїтивно зрозумілим: для отримання результату класифікації користувачу достатньо виконати два кроки – завантажити зображення і натиснути кнопку аналізу, після чого система відображає клас та ймовірність P(AI) у відсотках. Техніко-економічне обґрунтування розробки програмного модуля, розрахунок витрат на

проектування та підтвердження економічної доцільності створення системи наведено в Додатку В.

ВИСНОВКИ

У дипломній роботі вирішено актуальну задачу – розроблено програмний модуль класифікації облич людей за допомогою нейронних мереж, здатний автоматично визначати, чи є зображення реальним або AI–згенерованим. За результатами виконаної роботи можна зробити такі висновки.

1. Проаналізовано предметну галузь – встановлено, що сучасні генеративні моделі (StyleGAN2, Stable Diffusion, Midjourney) створюють фотореалістичні зображення, невідрізнювані від реальних – визначено, що бінарна класифікація «реальне/AI–згенероване» є оптимальним підходом до виявлення синтетичного контенту.

2. Здійснено порівняльний аналіз інструментального стеку – обґрунтовано вибір PyTorch, timm, PyTorch Geometric, albumentations та Flask – сформовано технологічну базу, що забезпечує ефективну розробку та розгортання системи.

3. Виявлено проблему dataset bias між датасетами FFHQ та generated.photos – встановлено, що однотонний фон AI–зображень дозволяв моделі досягати 100% точності без реального навчання – усунуто упередженість шляхом застосування детектора облич MTCNN для автоматичного вирізання області обличчя.

4. Розроблено дві архітектури нейронних мереж – CNN Baseline на основі EfficientNet–B2 та гібридну модель CNN+GNN з fusion–вектором розмірністю 1920 – реалізовано двоетапний fine–tuning із заморозкою backbone для стабільного навчання.

5. Проведено комплексне тестування на вибірці з 2 000 зображень – CNN+GNN перевершила CNN Baseline за всіма метриками: Accuracy на 1,19%, AUC–ROC на 0,68%, F1–score на 1,20% – підтверджено коректну роботу всіх компонентів системи, включаючи граничні випадки.

6. Розроблено Flask веб–додаток з інтерфейсом для завантаження зображень – реалізовано отримання результату класифікації з імовірністю P(AI)

– час обробки становить 0,08–0,18 с, що є прийнятним для практичного використання.

7. Проведено техніко–економічний аналіз розробки – повна собівартість склала 26 420 грн, договірна ціна – 33 025 грн – індекс прибутковості PI = 22,71, а термін окупності при впровадженні складає близько 17 робочих днів.

Практична цінність роботи полягає у розробці готового до використання інструменту виявлення AI–згенерованих зображень, який може застосовуватися в медіа–організаціях, соціальних мережах та правоохоронних органах для протидії deepfake–контенту. Напрямком подальших досліджень є розширення датасету зображеннями від різних генераторів (Midjourney, DALL–E, Stable Diffusion) та дослідження узагальнювальної здатності моделі на невідомих типах генерації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Goodfellow I. J. Generative Adversarial Nets / I. J. Goodfellow, J. Pouget–Abadie, M. Mirza та ін. // Advances in Neural Information Processing Systems. –2014. –Vol. 27. –P. 2672–2680.

2. Karras T. Analyzing and Improving the Image Quality of StyleGAN / T. Karras, S. Laine, M. Aittala та ін. // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. –2020. –P. 8110–8119.

3. Tan M. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks / M. Tan, Q. V. Le // Proceedings of the 36th International Conference on Machine Learning. –2019. –P. 6105–6114.

4. Veličković P. Graph Attention Networks / P. Veličković, G. Cucurull, A. Casanova та ін. // International Conference on Learning Representations. –2018. –12 p.

5. Karras T. A Style–Based Generator Architecture for Generative Adversarial Networks / T. Karras, S. Laine, T. Aila // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. –2019. –P. 4401–4410.

6. Frank J. Leveraging Frequency Analysis for Deep Fake Image Recognition / J. Frank, T. Eisenhofer, L. Schönherr та ін. // Proceedings of the 37th International Conference on Machine Learning. –2020. –P. 3247–3258.

7. Zhao H. Multi–attentional Deepfake Detection / H. Zhao, W. Zhou, D. Chen та ін. // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. –2021. –P. 2185–2194.

8. Li Y. Face X–ray for More General Face Forgery Detection / Y. Li, M. Chang, S. Lyu // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. –2020. –P. 5001–5010.

9. Rossler A. FaceForensics++: Learning to Detect Manipulated Facial Images / A. Rossler, D. Cozzolino, L. Verdoliva та ін. // Proceedings of the IEEE/CVF International Conference on Computer Vision. –2019. –P. 1–11.

10. Zheng Y. Exploring Temporal Coherence for More General Video Face Forgery Detection / Y. Zheng, J. Bao, D. Chen та ін. // Proceedings of the IEEE/CVF International Conference on Computer Vision. –2021. –P. 15044–15054.

11. He K. Deep Residual Learning for Image Recognition / K. He, X. Zhang, S. Ren, J. Sun // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. –2016. –P. 770–778.

12. Wightman R. PyTorch Image Models (timm) [Електронний ресурс] / R. Wightman. – Режим доступу: https://github.com/huggingface/pytorch–image–models. – Дата звернення: 15.02.2026.

13. Fey M. Fast Graph Representation Learning with PyTorch Geometric / M. Fey, J. E. Lenssen // ICLR Workshop on Representation Learning on Graphs and Manifolds. –2019. –9 p.

14. Buslaev A. Albumentations: Fast and Flexible Image Augmentations / A. Buslaev, V. I. Iglovikov, E. Khvedchenya та ін. // Information. –2020. –Vol. 11, No. 2. –P. 125.

15. Pedregosa F. Scikit–learn: Machine Learning in Python / F. Pedregosa, G. Varoquaux, A. Gramfort та ін. // Journal of Machine Learning Research. –2011. –Vol. 12. –P. 2825–2830.

16. Karras T. FFHQ: Flickr–Faces–HQ Dataset [Електронний ресурс] / T. Karras, S. Laine, T. Aila. –Режим доступу: https://github.com/NVlabs/ffhq–dataset. –Дата звернення: 20.01.2026.

17. Generated Photos. 100K Faces [Електронний ресурс]. –Режим доступу: https://generated.photos. –Дата звернення: 20.01.2026.

18. Zhang K. Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks / K. Zhang, Z. Zhang, Z. Li, Y. Qiao // IEEE Signal Processing Letters. –2016. –Vol. 23, No. 10. –P. 1499–1503.

19. Grinberg M. Flask Web Development: Developing Web Applications with Python / M. Grinberg. –2nd ed. –Sebastopol : O'Reilly Media, 2018. –316 p.

20. Paszke A. PyTorch: An Imperative Style, High–Performance Deep Learning Library / A. Paszke, S. Gross, F. Massa та ін. // Advances in Neural Information

Processing Systems. –2019. –Vol. 32. –P. 8024–8035.

21. Dosovitskiy A. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale / A. Dosovitskiy, L. Beyer, A. Kolesnikov та ін. // International Conference on Learning Representations. –2021. –22 p.

22. Chollet F. Xception: Deep Learning with Depthwise Separable Convolutions / F. Chollet // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. –2017. –P. 1251–1258.

23. Kipf T. N. Semi–Supervised Classification with Graph Convolutional Networks / T. N. Kipf, M. Welling // International Conference on Learning Representations. –2017. –14 p.

24. Wang S. CNN–generated images are surprisingly easy to spot... for now / S. Wang, O. Wang, A. Owens, R. Zhang // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. –2020. –P. 8695–8704.

25. Li L. FaceShifter: Towards High Fidelity And Occlusion Aware Face Swapping / L. Li, J. Bao, H. Yang та ін. // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. –2020. –P. 744–753.

26. Chollet F. Deep learning with Python / F. Chollet. –2nd ed. –Shelter Island, NY : Manning Publications, 2021. –504 p.

27. Goodfellow I., Bengio Y., Courville A. Deep learning / I. Goodfellow, Y. Bengio, A. Courville. –Cambridge, MA : MIT Press, 2016. –800 p. –URL: http://www.deeplearningbook.org (дата звернення: 24.03.2026).

28. Chandrasegaran K., Tran N.–T., Cheung N.–M. A Closer Look at Fourier Spectrum Discrepancies for CNN–Generated Images Detection // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). –2021. –P. 7200–7209. –[Електронний ресурс]. –Режим доступу: https://openaccess.thecvf.com/content/CVPR2021/papers/Chandrasegaran_A_Closer_Look_at_Fourier_Spectrum_Discrepancies_for_CNN–Generated_Images_CVPR_2021_paper.pdf. –Дата звернення: 23.03.2026.

29. Elsablaoui H., Mezghich M. A., Ghayoula R., Latrach L. Hybrid CNN–GNN model for deepfake image forensics // ICPRAM 2026 : 15th International Conference

on Pattern Recognition Applications and Methods : proceedings. Setúbal : SCITEPRESS – Science and Technology Publications, 2026. P. 635–642. DOI: 10.5220/0014637000004067. URL: https://www.scitepress.org/Papers/2026/146370/146370.pdf (дата звернення: 24.03.2026).

30. Nguyen T. T. Deep Learning for Deepfakes Creation and Detection: A Survey / T. T. Nguyen, Q. V. H. Nguyen, D. T. Nguyen та ін. // Computer Vision and Image Understanding. –2022. –Vol. 223. –Article 103525.

31. LeCun Y. Deep learning / Y. LeCun, Y. Bengio, G. Hinton // Nature. – 2015. – Vol. 521, No. 7553. – P. 436–444. DOI: 10.1038/nature14539.