Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Бегінг в ансамблях нейронних мереж для адаптації кольорів зображень до специфічних стилістичних, історичних або художніх контекстів
(тема)

Виконав:
здобувач другого року навчання,
групи СШІм-23-2
Михайло Зорін
(власне ім'я, прізвище)
Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)
Освітня програма Системи штучного інтелекту
(повна назва освітньої програми)
Керівник проф. Євгеній Бодянський
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Олег ЗОЛОТУХІН
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Зоріну Михайлу Михайловичу
(прізвище, ім'я, по батькові)

1. Тема роботи Бегінг в ансамблях нейронних мереж для адаптації кольорів зображень до специфічних стилістичних, історичних або художніх контекстів

затверджена наказом університету від 21 квітня 20 25 р. № 295Ст

2. Термін подання студентом роботи до екзаменаційної комісії 10 червня 20 25 р.

3. Вихідні дані до роботи Науково-технічна література та публікації з ансамблевих методів машинного навчання, документація алгоритмів ансамблювання нейронних мереж, набори зображень для тренування та тестування моделей стилізації та адаптації кольорів, програмні засоби для реалізації моделей, метрики оцінювання якості стилізації зображень

4. Перелік питань, що потрібно опрацювати в роботі
1) Теоретичні засади ансамблевого навчання та адаптації кольорів
2) Матеріали та методи ансамблевої моделі адаптації кольорів
3) Експериментальне дослідження ансамблевої моделі
4) Практична інтеграція моделі у програмний прототип

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	21.04.2025	виконано
2	Вивчення предметної галузі, аналіз літератури	22.04.25–02.05.25	виконано
3	Формулювання мети та задач дослідження	03.05.25–05.05.25	виконано
4	Аналіз існуючих рішень, формалізація задачі	06.05.25–12.05.25	виконано
5	Розробка архітектури нейронних мереж	13.05.25–20.05.25	виконано
6	Реалізація програмної частини	21.05.25–28.05.25	виконано
7	Аналіз результатів, оформлення висновків	29.05.25–31.05.25	виконано
8	Підготовка до подання роботи на перевірку	01.06.25–02.06.25	виконано
9	Перевірка на доброчесність та нормоконтроль	03.06.2025	виконано
10	Оформлення рецензії, доопрацювання зауважень	04.06.25–08.06.25	виконано
11	Подання фінальної версії роботи	09.06.25	виконано
12	Захист кваліфікаційної роботи	10.06.2025	виконано

Дата видачі завдання 21 квітня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ проф. Євгеній Бодянський
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 64 с., 23 рис., 1 дод., 21 джерело, 12 формул.

АВТОЕНКОДЕР, АНСАМБЛЕВІ НЕЙРОННІ МЕРЕЖІ, БЕГІНГ, ЗГОРТКОВА НЕЙРОННА МЕРЕЖА, ОБРОБКА ЗОБРАЖЕНЬ, СТИЛІСТИЧНА АДАПТАЦІЯ.

Об’єкт дослідження – ансамблеві нейронні мережі для обробки зображень.

Предмет дослідження – метод бегінгу (bagging) для побудови ансамблів CNN та автоенкодерів з метою адаптації кольорів зображень.

Мета роботи – розробка та дослідження ансамблевих нейронних мереж на основі методу бегінгу для ефективної адаптації кольорів зображень до заданих стилістичних, історичних і художніх контекстів.

Методи дослідження – аналіз сучасних рішень у сфері ансамблевих методів машинного навчання та комп’ютерного зору, теоретичне й практичне дослідження алгоритму бегінгу, побудова ансамблю глибоких згорткових нейронних мереж і автоенкодерів, експериментальна перевірка на реальних даних, оцінка результатів за допомогою метрик якості (PSNR, SSIM, Perceptual similarity).

У роботі виконано комплексний аналіз сучасних підходів до стилістичної адаптації та колоризації зображень із використанням глибоких ансамблевих нейронних мереж. Обґрунтовано доцільність застосування методу бегінгу для формування ансамблів для підвищення стабільності й узагальнюючої здатності системи. Запропоновано архітектуру ансамблю, що забезпечує якісну адаптацію кольорової гами зображень. Експериментальні дослідження засвідчили переваги запропонованого підходу у порівнянні з класичними моделями за рахунок зменшення дисперсії результатів і покращення візуальної якості стилізації.

ABSTRACT

Master’s thesis contains: 64 pp., 23 fig., 1 ann., 21 references, 12 formulas.

AUTOENCODER, BAGGING, CONVOLUTIONAL NEURAL NETWORK, ENSEMBLE NEURAL NETWORKS, IMAGE PROCESSING, STYLISTIC ADAPTATION.

Object of research – ensemble neural networks for image processing.

Subject of research – the bagging method for constructing ensembles of CNNs and autoencoders to adapt image colors.

Aim of the work – development and investigation of ensemble neural networks based on the bagging method for effective adaptation of image colors to specific stylistic, historical, and artistic contexts.

Research methods – analysis of current solutions in the field of ensemble machine learning methods and computer vision, theoretical and practical study of the bagging algorithm, development of an ensemble of deep convolutional neural networks and autoencoders, experimental validation on real data, evaluation of results using quality metrics (PSNR, SSIM, Perceptual similarity).

This work presents a comprehensive analysis of modern approaches to stylistic adaptation and colorization of images using deep ensemble neural networks. The feasibility of applying the bagging method for constructing ensembles of CNNs and autoencoders to increase the stability and generalization ability of the system is substantiated. An architecture of an ensemble is proposed, providing high-quality adaptation of the color palette of images to given stylistic, historical, or artistic contexts. Experimental studies demonstrated the advantages of the proposed approach over classical models, resulting in reduced output variance and improved visual quality of stylization.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Автоенкодер – нейронна мережа для стиснення та відновлення даних;

Ансамбль – сукупність незалежних моделей із агрегацією результатів;

Архітектура згортково-розгорткового автоенкодера (U-Net) – нейронна архітектура для сегментації та відновлення зображень;

Бегінг – bootstrap-агрегування моделей для зменшення дисперсії;

Генеративна змагальна мережа – генеративна модель, що складається з двох мереж у змагальному процесі;

Гіперпараметр – зовнішній параметр моделі, який налаштовується перед початком навчання;

Згорткова нейронна мережа – нейронна мережа, оптимізована для обробки зображень за допомогою операцій згортки;

Функція втрат – метрика для обчислення похибки моделі під час навчання;

Штучний інтелект – галузь комп’ютерної науки для створення систем, що імітують людський інтелект;

AdaIN – Adaptive Instance Normalization – адаптивна нормалізація по об'єктах (екземплярах);

Batch size – розмір пакета (батчу) – кількість прикладів у наборі даних, які обробляються одночасно перед оновленням ваг під час навчання нейронної мережі;

Bootstrapping – бутстрепінг (метод бутстрепу) – метод вибірки з поверненням, який дозволяє створити кілька навчальних наборів з одного, зберігаючи статистичну різноманітність;

Learning rate – коефіцієнт навчання – параметр, який визначає, наскільки сильно змінюються ваги моделі під час кожного кроку оптимізації;

MSE – Mean Squared Error – середньоквадратична похибка – метрика якості, що обчислюється як середнє значення квадратів різниць між передбаченими й фактичними значеннями;

Overfitting – перенавчання – ситуація, коли модель надто добре запам’ятовує навчальні дані, включно з шумом, і втрачає здатність до узагальнення на нові приклади;

PSNR – Peak Signal-to-Noise Ratio – пікове відношення сигналу до шуму – метрика, що оцінює якість реконструкції зображення або сигналу, порівнюючи рівень максимальної можливої потужності сигналу до потужності шуму (помилок);

SSIM – Structural Similarity Index – індекс структурної подібності – метрика для оцінки подібності двох зображень з урахуванням яскравості, контрасту та структури.

ВСТУП

У сучасному цифровому середовищі дедалі більше важливих завдань вирішується за допомогою глибоких нейронних мереж, які демонструють високі результати в аналізі зображень, відео та графічних даних. Серед таких завдань особливого значення набуває проблема відновлення та стилістичної адаптації кольорів у зображеннях. Історичні фотографії, що збереглися на плівках чи паперових носіях, часто втрачають насиченість й контрастність через зношування матеріалу, хімічні процеси старіння емульсії або неправильне зберігання. Крім того, сучасні цифрові знімки іноді потребують художнього переформатування під заданий стиль або атмосферу – наприклад, для ілюстрацій у виданнях з історичними матеріалами чи творчих проектів, де необхідно досягти ефекту старовинної палітри. Завдання відновлення кольорової гами та прив’язки її до конкретних історичних чи художніх контекстів знаходить практичне застосування в реставрації архівних фотоколекцій, у видавничій справі, у виробництві кіновізуальних матеріалів, а також у галузі комп’ютерної графіки та ігрового дизайну, де часто виникає потреба «оживити» або адаптувати зображення під заданий естетичний стандарт. Використання штучного інтелекту для автоматичного відтворення правильних відтінків і текстур сприяє не лише підвищенню якості готового контенту, а й зменшенню трудовитрат реставраторів та дизайнерів.

Незважаючи на значний прогрес у сфері глибинного навчання, існує низка викликів, що ускладнюють задачу колоризації. По-перше, історичні фотографії зазвичай мають нерівномірне освітлення, високий рівень шуму, пересвічені ділянки та артефакти, які виникли під час сканування або хімічного старіння плівки. По-друге, сучасні художні завдання потребують не лише технічно коректного додавання кольору, а й точного відтворення стилістичних особливостей певної епохи чи художнього напряму: палітра має відповідати реаліям часу, а перехід між відтінками – бути гармонійним

і природним. Саме тому традиційні архітектури згорткових нейронних мереж (CNN), наприклад U-Net, хоча й добре справляються з деталізацією та відновленням текстур, іноді не можуть гарантувати стабільний результат у разі сильних спотворень або різноманітних недоліків вхідних даних. Одні моделі можуть генерувати небажані колірні артефакти, інші – «забувати» характерні особливості історичних фасадів чи природних ландшафтів, особливо коли освітленість у початковому зображенні нерівномірна.

Щоб підвищити якість і надійність колоризації, у цій роботі запропоновано ансамблевий підхід, який базується на методі бегінгу (bagging). Ідея полягає в тому, що замість однієї «монолітної» моделі формується набір незалежних моделей із однаковою архітектурою (наприклад, U-Net із модулями адаптивної інстанс-нормалізації), але кожна з них навчається на власній bootstrap-вибірці тренувальних даних. Завдяки цьому рішення колірних завдань «розподіляється» між кількома моделями, а фінальний результат формується через усереднення прогнозів окремих мереж. Практичні переваги бегінгу в ансамблях нейронних мереж полягають у суттєвому зниженні варіативності (дисперсії) результатів: помилка однієї моделі може компенсуватися правильною відповіддю іншої, що забезпечує більш стабільний і «згладжений» вихідний колір. У разі дефектованих даних – нерівномірного освітлення, шуму, пересвічення – ансамбль краще узагальнює інформацію, оскільки помилки, пов’язані з локальними артефактами, не будуть системними для всіх незалежних моделей.

Таким чином, метою даної роботи є показати, що застосування бегінгу в ансамблі нейронних мереж для колоризації дозволяє суттєво підвищити якість та узагальнену стійкість до дефектів вхідних зображень. У наступних розділах детально розглянуто теоретичні засади колоризації, описано архітектуру ансамблю, обґрунтовано вибір гіперпараметрів, детально викладено експериментальну методику та наведено результати, які підтверджують практичну цінність і ефективність розробленого алгоритму.

1 ТЕОРЕТИЧНІ ЗАСАДИ АНСАМБЛЕВОГО НАВЧАННЯ ТА АДАПТАЦІЇ КОЛЬОРІВ

1.1 Аналіз задачі адаптації та відновлення кольорової гами

Задача адаптації та відновлення кольорової гами зображень є важливим та актуальним напрямом досліджень у галузі комп'ютерного зору та обробки цифрових зображень. Вона виникає з необхідності усунення різноманітних візуальних дефектів, які формуються під впливом фізичних та технічних факторів під час створення, зберігання та передачі цифрових фотографій. . Наприклад, на рисунку 1.1 наведено стару фотографію мосту через річку, зроблену на початку XX століття, де чітко видно горизонтальні смуги вицвілого паперу та нерівномірність експозиції. Такі дефекти часто спричиняють значні втрати кольорової інформації, що візуально наближує сучасні кольорові фотографії до монохромних або чорно-білих знімків, і, як наслідок, значно погіршує їх якість, естетичність та інформативність.

Рисунок 1.1 – Стара фотографія мосту через річку

Зазвичай, причини появи кольорових дефектів на фотографіях мають комплексний характер. Однією з основних причин є неправильні

налаштування фотокамер: помилки в автоматичному або ручному балансі білого, неправильно виставлені експозиція та ISO, що може призвести як до втрати кольорових деталей у надто світлих ділянках (пересвічення), так і до надмірно затемнених областей (недоекспонування). Такі явища спричиняють втрату інформації про початкові кольори об'єктів, оскільки значна частина оригінальної палітри зображення стає практично нерозрізненою для людського ока. У результаті часто виникає складна комбінація «спаленої» або «замильної» ділянки на одному й тому ж кадрі. На рисунку 1.2 показано приклади пошкоджень різних фотографій: відтінкова нерівномірність, подряпини, плями та розриви емульсії, що спричиняють незворотні втрати первинної кольорової інформації.

Рисунок 1.2 – Приклади пошкодження фотографії

Іншим вагомим аспектом є специфіка освітлення, яке використовується при фотографуванні. Нерівномірне або неприродне природне світло, а також штучні джерела можуть значно впливати на коректність кольоропередачі. Наприклад, світло від ламп розжарювання надає зображенню жовтуватий відтінок, тоді як флуоресцентні джерела створюють холодні блакитні тони. Змішані типи освітлення спричиняють

нерівномірність кольорової гами в межах одного знімка, і це суттєво ускладнює відновлення реальних кольорів. Отже, потреба точної компенсації цих ефектів змушує застосовувати складні алгоритми класифікації освітленості та корекції балансу білого.

Не менш важливими є фотографії, які втратили кольорову гаму через умови зберігання або тривалий час використання. Це можуть бути старі світлини, що фізично вицвіли, або цифрові зображення, які втратили частину інформації через надмірне стискання. У таких ситуаціях кольорові характеристики зображення дуже погіршуються, стаючи схожими на чорно-білі фотографії, а початкові насиченість та тональність кольорів (особливо в зонах переходу тіней і світла) практично зникають [1]. На рисунку 1.3 наведено приклад ручного відновлення зображення: реставратор поетапно відновлює втрачені ділянки вручну, ретушуючи окремі фрагменти й повертаючи кольору попередню яскравість. Проте такий підхід є трудомістким і тривалим, що стимулює розвиток автоматизованих методів.

Рисунок 1.3 – Приклад ручного відновлення зображення

Саме через такі різноманітні за походженням проблеми і виникає актуальне завдання – навчання моделі, яка могла б ефективно адаптувати або повністю відновлювати втрачену кольорову гаму фотографій. У цьому контексті штучні нейронні мережі, зокрема ансамблеві методи на основі бегінгу, є особливо перспективним інструментом. Вони можуть бути

навчені відновлювати кольорові ознаки зображень, використовуючи набори даних з відповідними кольоровими еталонами, завдяки чому стають здатними узагальнювати кольорові характеристики навіть у тих випадках, коли первинна кольорова інформація суттєво спотворена або зовсім втрачена [2].

Процес навчання таких нейронних мереж буде полягати у демонстрації великої кількості пар зображень: штучно знебарвлених і відповідних їм кольорових оригіналів. В процесі такого навчання модель формує внутрішні узагальнені уявлення про типові кольори, їх поєднання та способи їх розподілу у різних контекстах. Надалі це дозволяє моделі достатньо точно відновлювати або адаптувати кольорову інформацію навіть у фотографій, що зазнали серйозних змін чи втрат кольорових характеристик [3].

Таким чином, задача адаптації та відновлення кольорової гами фотографій є багатогранною і складною, оскільки охоплює різні дефекти, що можуть бути викликані цілим рядом фізичних, технічних та експлуатаційних факторів. Її вирішення вимагає глибокого розуміння не лише особливостей фотозйомки і обробки зображень, але й сучасних методів машинного навчання, які забезпечують ефективне відновлення кольорових ознак зображень за допомогою навчання на великих і репрезентативних наборах даних.

1.2 Аналіз існуючих алгоритмів і інструментів

Задача автоматичної колоризації зображень традиційно вирішувалася різноманітними методами, від простих статистичних і регресійних підходів до сучасних глибоких нейронних мереж. Багато з існуючих алгоритмів побудовані як окремі моделі, що оперують інформацією про текстуру, яскравість і локальні характеристики зображення без об’єднання кількох незалежних мереж у ансамбль. Нижче наведено огляд основних підходів і

популярних інструментів для стилізації та колоризації, що ілюструє їх концепції, сильні й слабкі сторони.

Початкові алгоритми колоризації базувалися на перенесенні середніх й дисперсійних характеристик кольору чи на ручному виборі кольорових точок (scribble-based colorization). Наприклад, класичний метод Райнхарда (Reinhard et al., 2001) для переносу кольору між двома зображеннями обчислює статистики колірних каналів у просторі L*a*b і виконує аффінне перетворення для входу. Цей підхід простий у реалізації, але вимагає близькості структурної сцени: якщо вихідне чорно-біле та референсне кольорове зображення сильно відрізняються композиційно, результат виглядає неприродним [4]. Інший підхід – екземплярна колоризація (exemplar-based colorization), де пікселі вихідного монохромного зображення зіставляються з пікселями зразка на основі локальних ознак (текстур, градієнтів). Така техніка здатна відтворювати кольори, близькі до реальних (якщо обрано відповідний референсний приклад), проте вимагає ручного підбору «зразків» і чутлива до шуму й спотворень [5].

Зі зростанням обчислювальних ресурсів у середині 2010-х років з’явилися перші алгоритми на основі згорткових нейронних мереж. У 2016 році Iizuka представили двостадійну CNN, що складається з глибокого декодера для загального прогнозу колірних компонентів та вузької архітектури для збереження локальних текстур. Цей алгоритм навчається на парі змішаних зображень (сіре → колір) і показав відчутний прогрес у порівнянні з ручними чи статистичними методами. Проте алгоритм має певні обмеження: він часто неправильно відтворює відтінки в складних сценах, де потрібні історично точні чи художньо виразні кольори. Larsson запропонував модель, що поєднувала семантичне сегментування й колоризацію: архітектура одночасно передбачала категорію об’єкта (небо, людина тощо) і генерувала відповідний колір. Результат більш стабільний у розпізнаванні природних сцен, але все ще далекий від історичної

автентичності, оскільки мережа навчається на сучасних кольорових зображеннях і не враховує стилістику минулих епох.

Пізніше Zhang представили колоризацію за допомогою нейронної мережі, що використовує ймовірнісну мапу кольору – модель прогнозує розподіл можливих кольорових відтінків для кожного пікселя. Це дало змогу уникнути одноманітності образів, але призвело до «розмитих» переходів, які не завжди відповідають історичним (наприклад, портрети XIX століття). Усі зазначені моделі не передбачають об’єднання кількох незалежних мереж – кожна працює автономно, а отже, має високі варіації у результаті залежно від шуму, освітлення чи аномалій у вхідному кадрі.

Сьогодні на ринку представлено кілька широко відомих інструментів для автоматичної колоризації та стилізації зображень, які, як правило, спираються на одиночні моделі або на GAN-підходи, але без явного застосування ансамблевих технік.[6]

PaintsChainer – онлайн-інструмент для автоматизованого «розфарбовування» малюнків та чорно-білих ілюстрацій, що базується на CNN, здебільшого застосовується для аніме-стилю. Хоча алгоритм демонструє цікавий художній ефект, він не створений для фотореалістичної колоризації старих архівних кадрів: результат має інтенсивні насичені кольори та нагадує коміксний стиль. Ensemble-підхід при цьому не застосовується – розфарбовування генерує одна модель, а фільтр «вибору стилю» просто змінює ваги шарів, без усереднення з іншими мережами.

Algorithmia Colorize Photos — хмарний сервіс, що надає API для колоризації за допомогою моделей CNN, навчених на великих обсягах даних. Алгоритм поєднує прогноз «global color histogram» із локальними текстурними особливостями. Здебільшого чудово справляється зі сценами природи та архітектури, але помилково відтворює кольори для одягу чи шкіри людей у виключно монохромних портретах. Це пояснюється тим, що модель навчається на сучасних фотографіях, а не на наборах, які відтворюють старовинні палітри. Ensemble-підходу не використовується,

тому результати можуть коливатися залежно від конкретної версії моделі та нюансів запиту. Інтерфейс та результат роботи цієї системи можна побачити на рисунку 1.4.

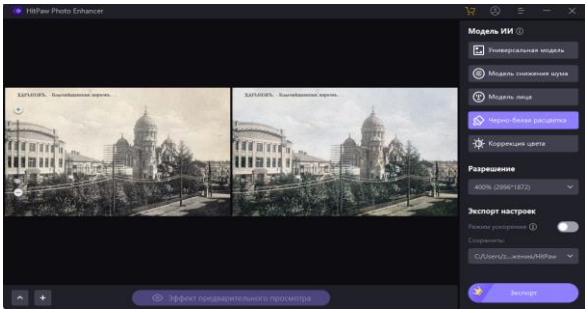

Рисунок 1.4 – Система, що використовує Algorithmia

DeOldify – відкрите рішення, розроблене на основі Generative Adversarial Networks (GAN). Використовує архітектуру U-Net із мережами-генераторами й дискримінаторами. DeOldify показує високу якість у реставрації старих фотографій завдяки adversarial-training: дискримінатор змушує генератор створювати реалістичні кольори [7]. Проте система демонструє нестійкість на зображеннях із сильними артефактами: наявність великих білих плям або тріщин іноді призводить до «вигорілих» зон кольору. На рисунку 1.5 можна побачити інтерфейс, що використовує це рішення для покращення фотографій.

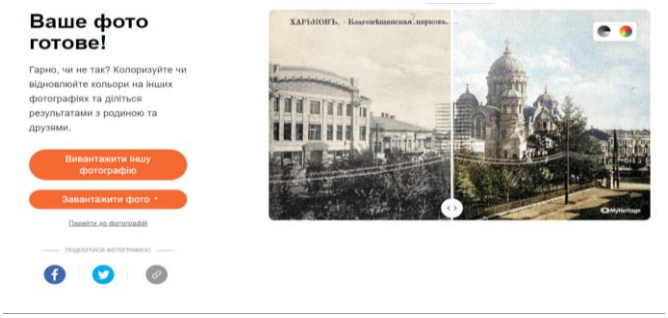

Рисунок 1.5 – застосунок на базі рішення DeOldify

Таким чином, наведені інструменти – DeOldify, Algorithmia Colorize, PaintsChainer – демонструють різні стратегії обробки зображень: від GAN-орієнтованої до простих автокодувальників і статистичних методів. Усі вони намагаються одночасно врахувати контекст, текстуру та глобальні колірні характеристики, але жоден із зазначених підходів не використовує ансамблевий бегінг як базову техніку. Саме тому в задачах, де критично важлива мінімізація флуктуацій у результатах та стійкість до артефактів старих фотографій, окремі моделі демонструють недостатню стабільність: будь-яка неточність у навченні може призвести до локальних перекручувань кольору, а без усереднення декількох незалежних прогнозів усі зазначені алгоритми ризикують видавати неоднорідні, часом неприродні, відтворення колориту.

З огляду на цей аналіз, очевидна потреба в комплексному ансамблевому підході, що дозволяє поєднати переваги сучасних CNN та GAN-підходів, знижуючи випадковість результату.[15]

1.3 Аналіз ансамблевих методів машинного навчання для задачі адаптації кольорової гами

У сучасних дослідженнях машинного навчання особливе місце посідають ансамблеві методи, які дозволяють ефективно вирішувати складні задачі за рахунок об'єднання кількох моделей, що разом працюють краще, ніж кожна окрема модель. Основною ідеєю такого підходу є використання певної кількості різних або ідентичних моделей, які навчаються на різних підмножинах даних або різними алгоритмами навчання. В результаті, їхні прогнози комбінуються, щоб отримати остаточне рішення, яке має меншу дисперсію і стабільнішу продуктивність, ніж кожна з моделей окремо.

В контексті задачі адаптації та відновлення кольорової гами зображень ансамблеві методи виявляються особливо перспективними,

оскільки задачі такого типу характеризуються високою складністю й неоднорідністю вхідних даних. Серед існуючих ансамблевих підходів можна виділити кілька фундаментальних стратегій: бустинг, стекінг та бегінг (bagging). Усі вони мають свої переваги й недоліки, і вибір певного методу залежить від специфіки конкретної задачі.

У випадку методу бустингу окремі моделі навчаються послідовно одна за одною, при цьому кожна наступна модель намагається виправити помилки попередньої. Проте саме послідовність та взаємозалежність моделей у ансамблі бустингу можуть викликати значне перенавчання, знижуючи здатність до узагальнення на нових зображеннях.

Метод стекінгу використовує декілька незалежних моделей, прогнози яких об'єднуються за допомогою додаткової метамоделі, що навчається на їхніх виходах. Він потребує значних зусиль у налаштуванні, оскільки вимагає додаткового етапу тренування метамоделі, що збільшує складність реалізації.

Окрему увагу слід приділити методу бегінгу (від англ. bootstrap aggregation). У цьому підході ансамбль складається з кількох моделей, які тренуються паралельно, незалежно одна від одної, на різних бутстрапованих вибірках даних, отриманих шляхом вибірки з поверненням із початкового набору. Кожна окрема модель будує своє рішення, після чого їх результати агрегуються за допомогою простого усереднення прогнозів. Основною властивістю бегінгу є суттєве зменшення дисперсії (variance) результатів завдяки тому, що помилки окремих моделей компенсують одна одну при агрегуванні їхніх виходів. Водночас, зміщення (bias) окремих моделей залишається відносно незмінними.[8]

На рисунку 1.6 наведено ілюстрацію того, як було отримано бутстреп-вибірку даних: червона крива показує повне значення випадкової функції, а сині точки – вибіркові спостереження з випадковим відхиленням від цього значення.

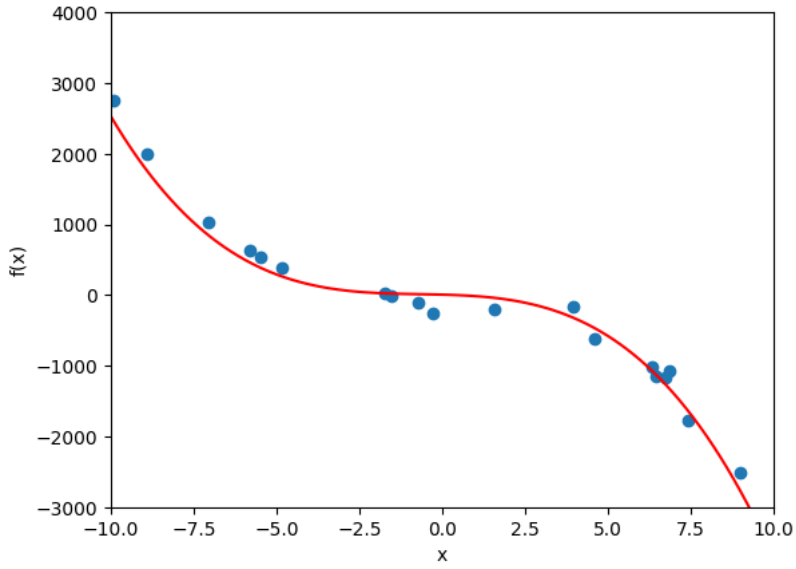

Рисунок 1.6 – Випадкова функція та бутстреп-вибірка

На рисунку 1.7 наведено результати побудови ансамблю регресорів на основі бутстреп-вибірок: кожне окреме дерево рішень було навчено на своїй випадковій підмножині даних, отриманій із вихідної вибірки, та зображено червоними кривими.

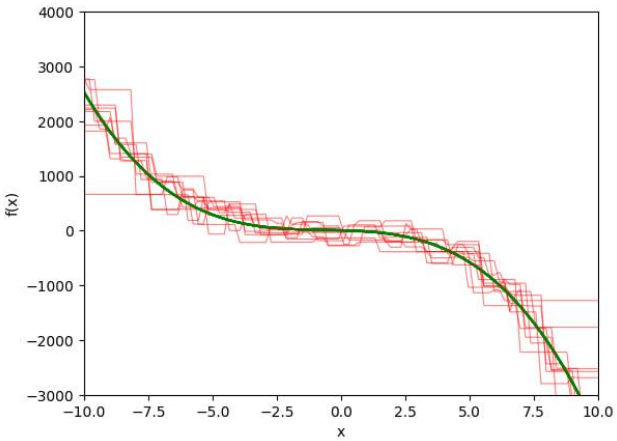

Рисунок 1.7 – Прогнози окремих рішень із бутстреп-вибірок

На рисунку 1.8 зображено середню оцінку ансамблю: червона крива відтворює усереднені прогнози дерев рішень. Завдяки такому усередненню

вдалося значно знизити дисперсію окремих прогнозів без помітного зміщення моделі (bias), що забезпечує високу точність відтворення вихідної функції при мінімальній варіативності.

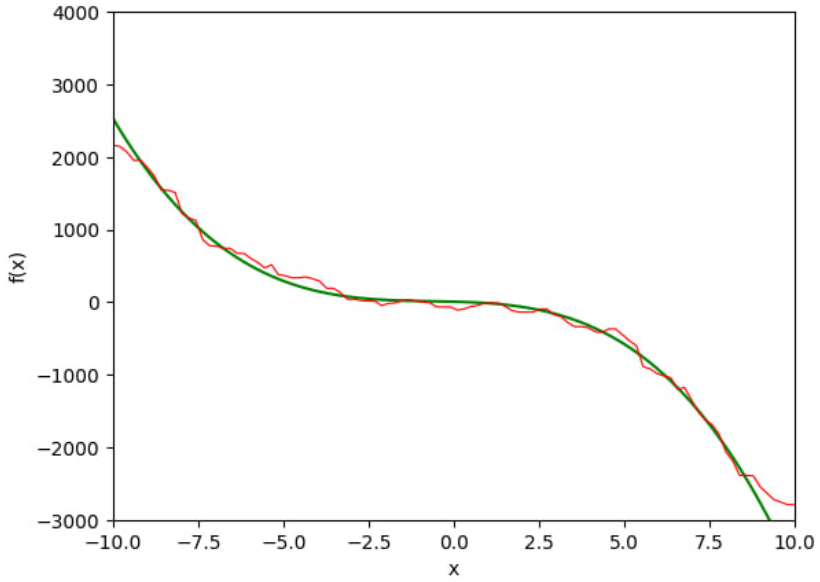

Рисунок 1.8 – Середній прогноз ансамблевих дерев (червона крива) порівняно з істинною функцією (зелена крива)

З точки зору задачі відновлення кольорової гами, саме властивість бегінгу – можливість роботи з високою дисперсією (variance) і низьким зміщенням (bias) окремих моделей – є ключовим фактором, що робить цей метод найбільш відповідним. Ця характеристика дає змогу враховувати широкий діапазон можливих кольорових рішень при стилізації та відновленні кольору, зберігаючи при цьому деталізацію і точність вихідних даних. Кожна з моделей ансамблю здатна знаходити своє оптимальне рішення для відновлення кольорової гами, а їх комбіноване рішення демонструє вищу стабільність та якість.

Проаналізувавши характеристики різних ансамблевих методів, можна зробити висновок, що для задачі адаптації кольорової гами зображень саме бегінг це оптимальний вибір. Він дозволяє максимально ефективно

використовувати переваги нейронних моделей, зберігаючи їхню здатність до узагальнення й адаптації до нових умов, що критично важливо для задачі стилізації фотографій, які мають значні дефекти або втрату кольорової інформації.

1.4 Математичні основи бегінгу для ансамблів нейронних мереж

Ансамблевий метод бегінг (bootstrap aggregation) ґрунтується на створенні кількох незалежних моделей, кожна з яких навчається на власній бутстрап-вибірці, після чого їхні прогнози агрегуються для отримання остаточного результату. Нижче наведено формальні визначення основних етапів алгоритму бегінгу з відповідними математичними формулами.[9]

Нехай маємо початковий навчальний набір даних

$$D^{(m)} = (x_i, y_i)_{i=1}^{N}, m = 1, \dots, M, \tag{1.1}$$

де $x_i \in R^d$ – вхідні зображення або їх ознаки;

$y_i \in R^c$ – відповідні кольорові вектори.

Метою є побудова ансамблю з M нейронних мереж, які спільно відновлюють або адаптують кольорову гаму вхідних зображень.

Першим етапом алгоритму є формування M бутстрап-вибірок з вихідного набору D. Далі кожна підвибірка $D^{(m)}$ використовується для незалежного навчання нейронної мережі $f^{(m)}$. Нехай в нас є клас нейронних мереж з фіксованою архітектурою. Параметри m-ої нейронної мережі знаходиться шляхом мінімізації функції витрат на підвибірці.

$$\theta^{(m)} = \arg min \frac{1}{N} \sum_{(x_i, y_i) \in D^{(m)}} \varphi\left(f(x_i;\ \theta),\ y_i\right), \tag{1.2}$$

де $\theta^{(m)}$ – параметри m-ої нейронної мережі;

$\varphi()$ – обрана функція втрат (наприклад, MSE);

$f(x_i;\ \theta)$ – передбачення нейронної мережі з параметрами θ для вхідного.

Після завершення навчання усіх M нейронних мереж їхні виходи за конкретним входом x агрегуються шляхом усереднення. Нехай $f^{(m)}(x)$ – передбачення m-ої мережі для вхідного зображення x. Тоді остаточне передбачення ансамблю у вигляді вектор-значень обчислюється так:

$$y_{ens}(x) = \frac{1}{M}\left(f^{(1)}(x) + f^{(2)}(x) \dots + f^{(m)}(x)\right), \tag{1.3}$$

де M – кількість нейронних мереж в ансамблі;

x – вхідне зображення;

$f^{(m)}(x)$ – передбачення m-ої мережі для вхідного зображення x;

$y_{ens}(x)$ – остаточне передбачення ансамблю (усереднений вектор).

Щоб оцінити вплив алгоритму бегінгу на характеристики похибок ансамблю, доцільно розглянути поняття зміщення та дисперсія. Нехай виходи нейронних мереж за певним вхідним зображенням x позначаються випадковою величиною $\hat{Y}$, оскільки бутстрап-вибірка (а отже і навчання моделі) є ймовірнісним процесом. Очікуване передбачення окремої моделі за x позначимо $E[\hat{Y}(x)]$. Тоді середньоквадратична помилка передбачення окремої моделі розкладається на суму двох компонент:

$$MSE\big(f(x)\big) = \left(E\big[\hat{Y}(x)\big] - Y(x)\right)^2 + E\big[(\hat{Y}(x) - E[\hat{Y}(x)])^2\big] + \sigma_\varepsilon^2, \tag{1.4}$$

де $MSE\big(f(x)\big)$ – середньоквадратична помилка передбачення окремої моделі за зображенням x;

$E\big[\hat{Y}(x)\big]$ – очікуване передбачення окремої моделі за x;

$Y(x)$ – істинна кольорова гама (ціль) для зображення x;

σ_ε^2 – непередбачувана випадкова складова, що відповідає шумам та іншим артефактам, які неможливо змоделювати.

При реалізації ансамблю за допомогою бегінгу кожна окрема модель має певне зміщення і дисперсію. Однак усереднення M незалежних моделей зменшує загальну дисперсію ансамблевого передбачення за законом:

$$Var\left(\frac{1}{M}\sum_{m=1}^{M}\hat{Y}(x)\right) = \frac{1}{M^2}\sum_{m=1}^{M}Var(\hat{Y}(x)) = \frac{1}{M}Var\left(\hat{Y}(x)\right), \qquad (1.5)$$

де Var() – дисперсія ансамблю;

M – кількість нейронних мереж в ансамблі;

$\hat{Y}(x)$ – передбачення окремої моделі для вхідного зображення x;

$Var(\hat{Y}(x))$ – дисперсія передбачення окремої моделі для x.

Таке обчислення можна зробити, припустивши, що всі окремі моделі мають однакову дисперсію та є незалежними. Таким чином, у ансамблю з M моделей дисперсія зменшується у M разів. При цьому зміщення:

$$Bias\left(\frac{1}{M}\sum_{m=1}^{M}\hat{Y}(x)\right) = \frac{1}{M}\sum_{m=1}^{M}\left(E\left[\hat{Y}(x)\right] - Y(x)\right) = E\left[\hat{Y}(x)\right] - Y(x), \quad (1.6)$$

де $Bias()$ – зміщення усередненого передбачення ансамблю для x;

M – кількість нейронних мереж в ансамблі;

$\hat{Y}(x)$ – передбачення окремої моделі для вхідного зображення x;

$E\left[\hat{Y}(x)\right]$ – очікуване передбачення окремої моделі за x;

$Y(x)$ – істинна кольорова гама (ціль) для зображення x;

Тобто зміщення ансамблю залишається таким самим, як у кожної окремої моделі, тоді як усереднення M незалежних моделей зменшує загальну дисперсію ансамблевого передбачення. У задачах відновлення кольору це критично – зниження variance сприяє стабільності відтворення

кольорової гами навіть за наявності шумів, артефактів або нерівномірного освітлення, не збільшуючи при цьому середнього зміщення (bias). Завдяки цьому ансамбль менш чутливий до випадкових флуктуацій і краще реалізує заданий стилістичний контекст.[16]

Ефективність бегінгу напряму залежить від різноманітності моделей в ансамблі, що забезпечується bootstrap-вибірками та варіаціями гіперпараметрів. Вибір кількості моделей є компромісом між точністю прогнозу та наявними обчислювальними ресурсами.

2 МАТЕРІАЛИ ТА МЕТОДИ АНСАМБЛЕВОЇ МОДЕЛІ АДАПТАЦІЇ КОЛЬОРІВ

2.1 Підготовка та обробка набору даних

В процесі підготовки даних для задачі адаптації кольорової гами розглядались кілька загальнодоступних наборів зображень різного призначення та обсягу: COCO, ADE20K, BSD500 і Places365. Найбільш збалансованим рішенням виявився Places365, оскільки він містить понад півтора мільйона сцен різного типу (міські вулиці, природні ландшафти, інтер'єри), що дає змогу навчанню моделі узагальнювати колірні закономірності в широкому спектрі умов освітлення. В той самий час він пропонує оптимальний обсяг даних по відношенню до кількості фотографій. Як показано на рисунку 2.1, порівняння за кількістю обсягом даних і кількістю унікальних зображень показує оптимальність саме Places365 у якості основного джерела даних для фінального етапу навчання ансамблевої моделі. А велика кількість унікальних категорій і умов освітлення затверджує цей вибір.

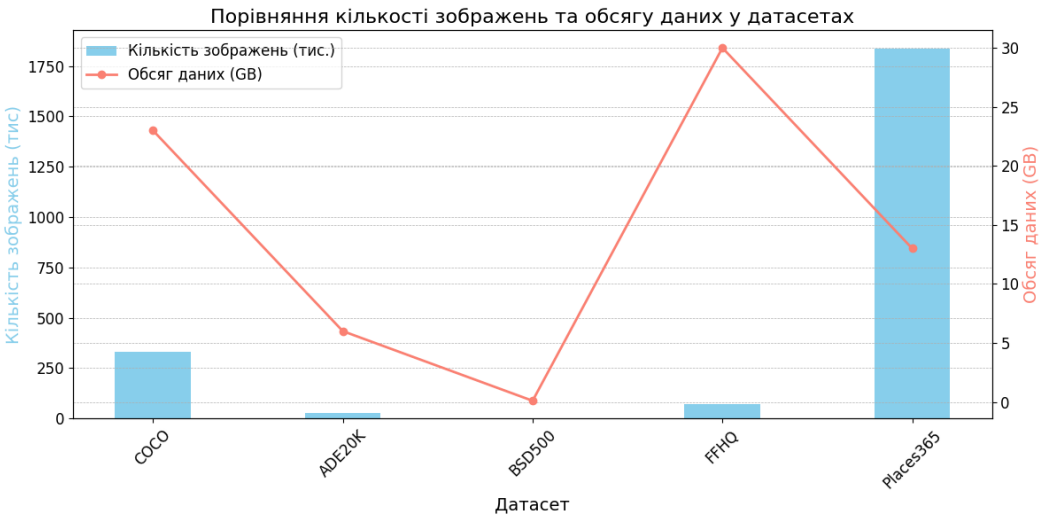

Рисунок 2.1 – Порівняння кількості зображень та обсягу даних

Водночас безпосередня робота з повним датасетом Places365 – Standard (≈1,8 млн зображень у розмірі 256×256), що займає понад 20 ГБ після розпакування, виходить за межі доступних обчислювальних ресурсів. Щоб ефективно протестувати архітектуру мережі та підібрати гіперпараметри, було вирішено спочатку сформувати компактне підмножину «Places365-10k», яка складається з 10 000 випадково вибраних знімків із первинного набору. Цей підхід дозволяє на етапі експериментів працювати з обсягом даних, що не перевищує 1 ГБ (128×128 розмір кожного зображення), при цьому зберігаючи достатню різноманітність сцен та умов освітлення. На рисунку 2.2 наведено приклад того, як виглядають 4 випадкові фото з датасету. В ньому наведені фотографії з різних місць, і основна задача датасету полягала саме у розпізнаванні образів різних місць.

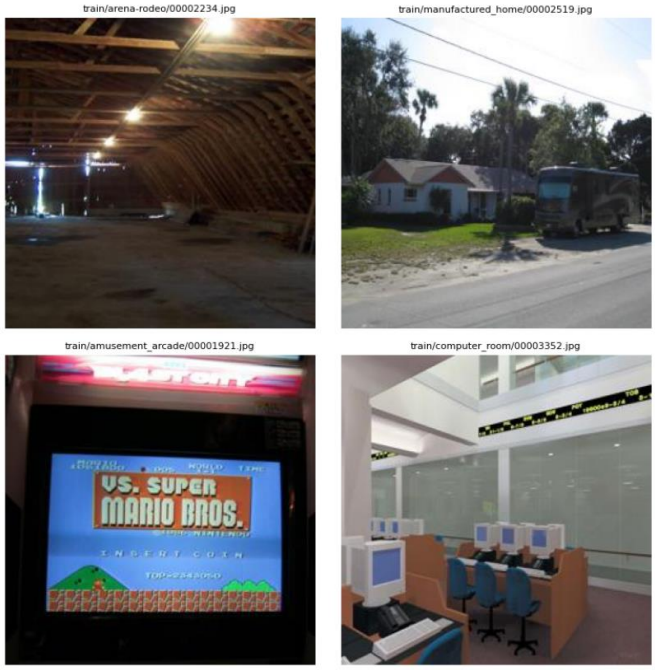

Рисунок 2.2 – Фото з датасету Places365-Standard

Наступним кроком було формування «погіршених» (дефектних) варіантів зображень, необхідних для навчання моделі відновлювати або

коригувати колірну гаму. Для кожного зображення оригінального набору генерувались три типи спотворень:

– перетворення в сірі тони;
– пересвічування шляхом лінійної зміни яскравості;
– додавання шуму та розмиття.

Завдяки такому підходу одне й те ж оригінальне зображення може виглядати по-різному. Це значно збільшує різноманітність навчальних прикладів, не потребуючи створення десятків тисяч окремих файлів заздалегідь.[17] На рисунку 2.3 ілюстровано приклади таких перетворень: у верхньому рядку показано фрагмент зображення в нормальних кольорах, а в нижньому – його варіанти з різними дефектами.

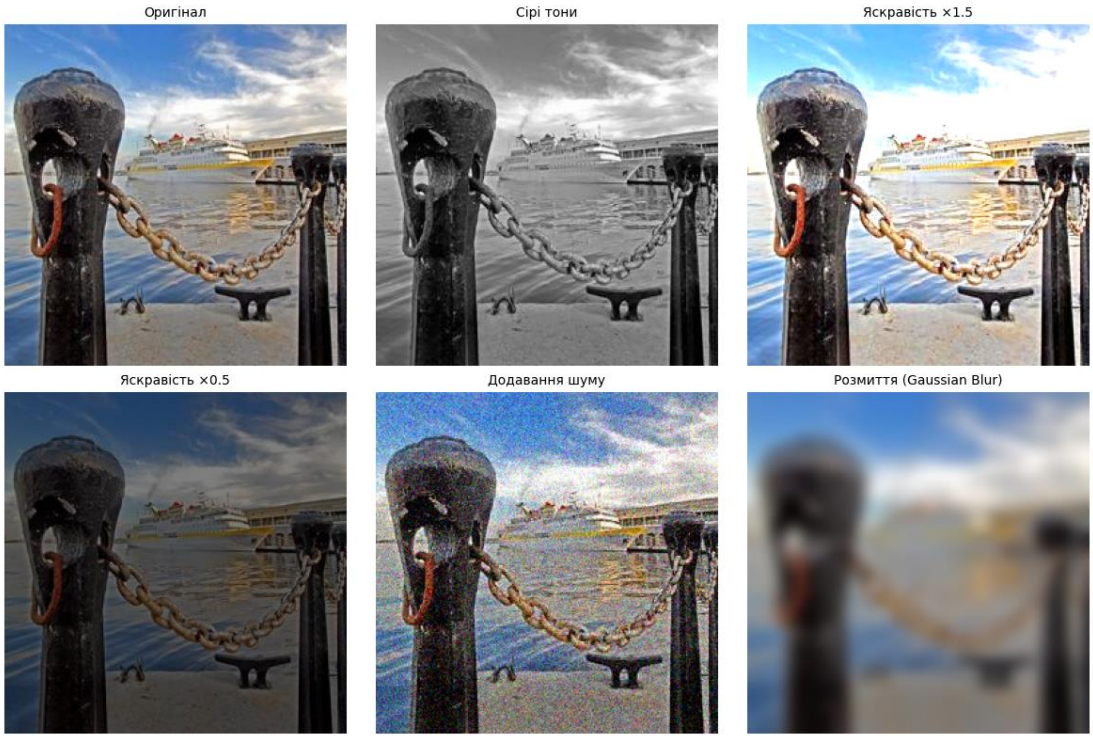

Рисунок 2.3 – Фото з датасету Places365-Standard

У представленому прикладі перетворення здійснювалося за допомогою бібліотеки Pillow у середовищі Python. Спочатку вихідне зображення конвертували в чорно-білий формат (градації сірого), після чого

застосовували лінійну корекцію яскравості (фактор 1.5 для пересвічування і 0.5 для затемнення). Далі до масиву пікселів додавали гаусів шум із середнім 0 та стандартним відхиленням 25, що імітувало випадкові артефакти, притаманні низькій якості або несприятливому освітленню. Заключним кроком було розмивання за допомогою фільтра Gaussian Blur.

Застосування всіх цих операцій у єдиному скрипті дозволило продемонструвати, як просто та швидко можна отримати кілька версій одного й того ж зображення з різними якісними «дефектами».

2.2 Архітектура окремої нейронної моделі для стилізації

Архітектура окремої нейронної мережі для стилізації зображень має вирішувати задачу конвертації вхідного RGB-зображення довільної роздільності $H \times W \times 3$ у нове зображення тієї самої роздільності, яке зберігає структурний контент оригіналу, але набуває колірної гами та характерної текстури обраного стилю. Основна ідея полягає в розділенні екстракції просторових (контентних) ознак та ознак стилю, а потім у їх інтеграції таким чином, щоб на виході мережа генерувала консистентне з художнього погляду зображення. Для цього архітектура містить чотири ключові етапи: (1) енкодер контенту, що витягує просторові ознаки та виділяє високорівневу структуру; (2) енкодер стилю, який обчислює статистичні характеристики палітри та текстурного патерну стилю; (3) блок інтеграції стилю, у якому контентні ознаки коригуються згідно зі статистиками стилю; (4) декодер, що відновлює вихідне зображення шляхом підвищення роздільності й накладення колірної гами.

Спочатку вхідне зображення $I_c \in R^{H \times W \times 3}$ передається в блок енкодера контенту. Цей блок складається з послідовності згорткових шарів із ядрами розміру 3×3, кожен із яких супроводжується пакетною нормалізацією та нелінійністю ReLU. Щоб поетапно зменшувати просторову роздільність і водночас збільшувати глибинну розмірність

ознак, після кожного другого згорткового шару застосовується операція пропуску з кроком 2, яка зменшує висоту та ширину наполовину. Наприклад, перший згортковий блок перетворює карту ознак розміру $H \times W \times 3$ на $\frac{H}{2} \times \frac{W}{2} \times 64$, другий блок – $\frac{H}{4} \times \frac{W}{4} \times 128$ тощо. Така глибока послідовність дозволяє виділяти контентні ознаки різної абстракції: від низькорівневих контурів до високорівневих семантичних структур.[18]

У випадку використання попередньо натренованих моделей, архітектура енкодера може базуватись на шарах до conv_3_3 включно, завдяки чому відпадає потреба в навчанні параметрів з нуля. Екстраговані ознаки з VGG-19 (або іншого аналогічного екстрактора) демонструють високу кореляцію з візуальним сприйняттям контенту, що суттєво спрощує задачу збереження структурної цілісності. У запропонованій реалізації після шару conv_3_3 додають власні згорткові блоки з меншою кількістю фільтрів, щоб знизити обчислювальні витрати, при цьому зберігаючи достатню глибину моделі. Далі інтеграція стилю виконується через блок Adaptive Instance Normalization (AdaIN), у якому контентні ознаки F_c нормалізуються та масштабуються за статистиками стильових ознак F_s:

$$AdaIN(F_c, F_s) = \sigma(F_s)\frac{F_c - \mu(F_c)}{\sigma(F_c)} + \mu(F_s), \quad (2.1)$$

де $AdaIN()$ – результат адаптивної нормалізації інстансу;

F_c – контентні ознаки вхідного зображення;

F_s – стильові ознаки (тензор розмірності H×W×C);

μ – вектор середніх значень по кожному каналу;

σ – вектор стандартних відхилень контентних ознак по кожному каналу.

Після цього AdaIN-вихід проходить через кілька згорткових шарів із Instance Normalization і ReLU-активацією, що згладжують перехід між контентною структурою та палітрою стилю.

Таким чином, достатньо однієї формули AdaIN для опису основного механізму перенесення глобальної колірної статистики: середнє й дисперсія зі стилю передаються контентним ознакам, а додаткові згорткові блоки гарантують плавність і відсутність різких артефактів у результатах. Інші математичні деталі обчислення точних статистик і розрахунку втрат (контент-, стиль- і TV-компонент) за необхідності можна винести до окремого додатку або сформулювати у вигляді коротких описів, зосередившись в основному на конструктивній логіці побудови мережі.

Як показано на рисунку 2.4, процес стилізації зображення складається з кількох послідовних етапів: екстракція контентних ознак, обчислення статистик стилю, їх інтеграція через AdaIN та остаточне відновлення стилізованої картинки.

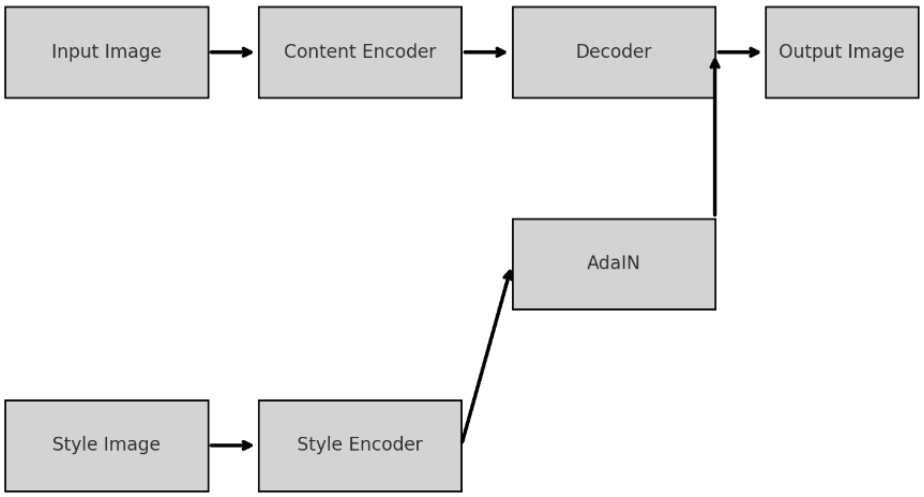

Рисунок 2.4 – Архітектурна діаграма нейронної мережі для стилізації

Архітектурна діаграма окремої нейронної мережі для стилізації, що включає: вхідне зображення (Input Image), блок екстракції контенту (Content Encoder), блок екстракції стилю (Style Encoder), операцію Adaptive Instance Normalization (AdaIN), декодер (Decoder) і вихідне стилізоване зображення (Output Image). Фотографія спочатку проходить через енкодер для виділення контентних ознак, потім ці ознаки нормалізуються та

масштабуються статистиками стилю через AdaIN і нарешті відновлюються у стилізоване зображення за допомогою декодера.

2.3 Побудова ансамблю за допомогою методу бегінгу

Для побудови ансамблю за методом бегінга з наявного тренувального набору D формують k підвибірок однакового розміру шляхом випадкової вибірки з поверненням. Завдяки цьому кожна підвибірка містить частину оригінальних зразків із можливими повторами та пропусками, що забезпечує різноманітність даних для навчання і сприяє зниженню кореляції помилок між окремими моделями. Кожну стилізуючу нейронну модель M_i навчають на власному bootstrap-піднаборі D_i, використовуючи однакові гіперпараметри—learning rate, batch size, кількість епох—але з різними ініціалізаціями ваг і випадковими seed-значеннями для відтворюваності. Підбір фіксованих значень seed гарантує, що результати навчання можна відтворити на різних машинах або за різних запусках. Після завершення тренування всі моделі здатні перетворювати будь-яке нове зображення контенту I_c у кілька стилізованих варіантів $\hat{I}_1, \hat{I}_2, \dots, \hat{I}_k$. Остаточне зображення $\hat{I}_{ensemble}$ отримують шляхом піксельного усереднення виходів усіх k моделей.

$$\hat{I}_{ensemble}(x, y) = \frac{1}{k}\sum_{i=1}^{k} \hat{I}_i\,(x, y), \tag{2.2}$$

де k – кількість моделей у ансамблі;

(x, y) – координати пікселя у зображенні;

$\hat{I}_i(x, y)$ – усереднене передбачення (відновлене значення) кольору пікселя з координатами (x,y) і-ою моделлю;

$\hat{I}_{ensemble}$ – остаточне відновлене значення кольору пікселя.

Іншими словами, для кожної координати (x,y) обчислюється середнє значення кольорових каналів, що дозволяє пом’якшити локальні артефакти й зменшити випадкові колірні зсуви, властиві окремим моделям. Використання простого піксельного усереднення доводить свою ефективність у тих випадках, коли окремі мережі можуть генерувати «шумні» або «блокові» помилки, а їхнє поєднання через усереднення дає більш плавний і послідовний результат.

На рисунку 2.5 показано створення M підвибірок із повної вибірки D методом бутстрапінгу, навчання на них моделей та піксельне усереднення їхніх виходів.

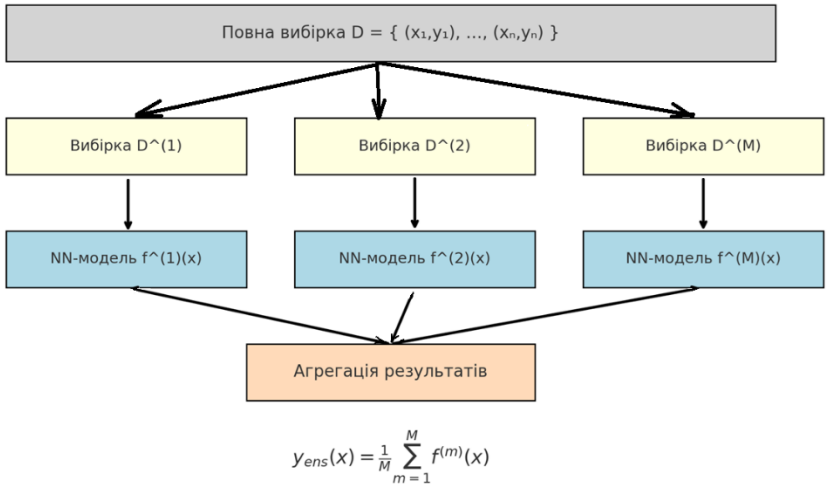

Рисунок 2.5 – Формування вибірок і агрегування прогнозів ансамблю

Усереднення дозволяє пом’якшити локальні артефакти й зменшити випадкові колірні зсуви, властиві окремим моделям, що робить результат більш плавним і послідовним.

Вибір оптимального числа моделей k (зазвичай $5 \leq k \leq 10$) залежить від обсягів доступної GPU-пам’яті та часу обчислень. Техніка «out-of-bag» (OOB) дозволяє оцінити якість кожної моделі без окремої валідаційної вибірки: близько третини зразків оригінального датасету не потрапляють до

конкретного піднабору D_i, і їх можна використовувати для перевірки точності відповідної моделі.

Окрім піксельного усереднення, можливими варіантами агрегування можуть бути зважене усереднення (коли кожній моделі призначають вагу залежно від її OOB-помилки). Проте просте арифметичне усереднення часто виявляється достатнім і менш чутливим до вибору ваг.

Завдяки згаданим технікам побудови й агрегування, ансамбль на базі бегінга демонструє значно меншу чутливість до шуму, нерівномірного освітлення та спотворень у початкових зображеннях. Це забезпечує стабільне відновлення кольорової гами, де локально артефактні фрагменти згладжуються завдяки «компенсації» результуючих відхилень між моделями.

2.4 Процес навчання моделей ансамблю

У процесі навчання ансамблю моделей за методом бегінгу першочергово формуються M підвибірок даних із повторенням (bootstrap), кожна з яких містить N прикладів вихідного набору. Для цього випадковим чином вибирають індекси із заміщенням і формують підмножину. Наступним кроком для кожного m-го компонента ансамблю створюється окрема нейронна мережа з архітектурою U-Net та модулями Adaptive Instance Normalization.[19] У ході однієї ітерації навчання мінімізується комбінована функція втрат

$$L = L_{context} + \lambda L_{style} + \beta \|\theta\|_2^2, \quad (2.3)$$

де $L_{context}$ – контекстна втрата, що вимірює різницю між відновленим зображенням та оригіналом у сенсі змісту;

λ – коефіцієнт для масштабування вкладки стильової втрати;

L_{style} – стильова втрата, що оцінює відповідність відновленого зображення бажаному стилю;

β – коефіцієнт регуляризації параметрів моделі;

$\|\theta\|_2^2$ – квадрат L2-норми вектора параметрів θ, який використовується для зменшення перенавчання.

Нижче наведено приклад короткого фрагмента коду, що ілюструє цикл тренування однієї моделі на одній підвибірці (лістинг 2.1).

Лістинг 2.1 – Цикл тренування однієї моделі на одній підвибірці

```
for epoch in range(1, epochs+1):
    for x_batch, s_batch, y_batch in dataloader_m:
        y_pred = model(x_batch, s_batch)
        loss_content = content_loss(y_pred, y_batch)
        loss_style   = style_loss(y_pred, y_batch)
        loss_reg     = l2_regularization(model)
        total_loss      =  loss_content  +  λ*loss_style  +
β*loss_reg
        optimizer.zero_grad()
        total_loss.backward()
        optimizer.step()
```

Тут dataloader_m формує випадкові міні-пакети з підвибірки D_m, а content_loss і style_loss реалізують відповідні підкомпоненти функції втрат.

Показова вагова відданість λ та регуляризаційний коефіцієнт β було підібрано емпірично: у експериментах λ =1.0, $\beta = 10^{-5}$, learning rate θ задавалась на рівні 10^{-4}.

Після незалежного тренування всіх M мереж отримуємо набір оптимальних параметрів $\{\theta_m^*\}_{m=1}^M$. Щоб отримати прогноз ансамблю для нового зразка $(x_{new},\ s_{new})$, складаємо середнє від виходів окремих мереж:

```
for net in ensemble_models:
    preds.append(net(x_new.unsqueeze(0),
s_new.unsqueeze(0)))
ensemble_output = torch.stack(preds, dim=0).mean(dim=0)
```

У цьому фрагменті ensemble_models – список із M попередньо натренованих мереж, а метод mean(dim=0) гарантує поелементне

усереднення результатів, що знижує дисперсію і підвищує стабільність фінального зображення. Якщо б вирішувалося завдання призначати різні ваги окремим моделям (наприклад, за точністю на валідаційній множині), тоді замість простого середнього використовували б вагову суму.

У результаті описаного підходу кожна мережа в ансамблі «бачить» трохи різний набір даних, що сприяє зменшенню узагальнювальної похибки через компенсацію індивідуальних помилок моделей. Виконання тренування паралельно (або по черзі) на різних GPU (якщо доступні) дозволяє скоротити загальний час навчання. Зазвичай для нашої постановки обирали M=5 моделей, по 100 епох для кожної, з розміром міні-пакета B=4 зображень. Після завершення навчання ансамблю фінальний вихід демонструє покращені люб’язність стилю та збереження контенту в порівнянні з одиничною моделлю, що підтверджується кількісними метриками (PSNR, SSIM, LPIPS) у наступному розділі.

3 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ АНСАМБЛЕВОЇ МОДЕЛІ

3.1 Опис експериментальної методики та параметрів

В цьому розділі детально пояснимо, як саме було організовано підготовку даних, середовище виконання та процес тренування кожної з моделей ансамблю, а також обґрунтувати вибір гіперпараметрів і способів контролю якості навчання.

У експерименті застосовано ансамбль із п’яти згорткових нейронних мереж однакової архітектури, кожна з яких навчається на окремій bootstrap-вибірці із початкового датасету. Початковий датасет складався з 15 000 зображень формату RGB роздільністю 256×256 пікселів: 10 000 пар «ретро-зображення – цільове стилізоване зображення» для тренувальної вибірки, 2 000 пар для валідації та 3 000 пар для тестування. Перед навчанням усі зображення конвертовано в плаваючий формат float32 та нормалізовано в діапазон [–1, +1], що дозволило прискорити збіжність оптимізатора й усунути відмінності в масштабах каналів. Збереження однакового розміру полегшує пакетну обробку та забезпечує однорідність входів у мережі.

Для формування окремих bootstrap-вибірок із тренувального набору спочатку визначено початковий поріг різноманітності: кожна з п’яти вибірок містила по 10 000 зображень, відібраних випадково з поверненням. Такий підхід гарантує, що приблизно 37% початкових прикладів не відображаються у жодній конкретній вибірці, а більшість зображень потрапляє принаймні до двох мереж, що впливає на зниження кореляції помилок. Відбір із поверненням здійснено за фіксованим випадковим зерном (random seed) 42, щоб забезпечити можливість відтворення результатів. Кожна мережа отримує унікальний комплект даних, що дозволяє оцінювати ступінь узагальнення за допомогою статистично незалежних вибірок.

Архітектура окремої мережі побудована на базі модифікованої U-Net з інтегрованою механікою адаптивної інстанс-нормалізації (AdaIN). Енкодер складається з трьох рівнів: на першому рівні два згорткові шари з 64 каналами, далі на другому рівні – два згорткові шари з 128 каналами, а на третьому – два згорткові шари з 256 каналами. Після кожного блоку згорткових шарів застосовано макс-пулінг 2×2. Центральний блок містить два згорткові шари з 512 каналами, кожен з яких використовує активацію ReLU. Декодер складається з трьох блоків транспонованих згорткових шарів (deconvolution) із зворотною послідовністю каналів: спочатку 256, потім 128 і, нарешті, 64. На кожному рівні декодера виконано конкатенацію з відповідним тензором ознак із енкодера, що дозволяє зберігати дрібні деталі. Після кожного етапу транспонованих згорток застосовано AdaIN, завдяки чому статистика стилю синхронно інкорпорується у відновлюваний контент. Для реалізації AdaIN використано середнє й стандартне відхилення ознак стилю, отриманих із того самого рівня енкодера стилізованого зображення.

Усі моделі реалізовано у середовищі Python 3.8 із використанням фреймворку PyTorch 1.8.1 та CUDA 11.1 для обчислень на графічних прискорювачах NVIDIA. Даталоадери налаштовано з розміром пакета 8, чотирма робочими процесами та безпечним перетасуванням даних під час кожної епохи. Ці значення були обрані як компроміс між швидкістю навчання та обмеженнями пам’яті. Для відтворюваності результатів встановлено фіксовані значення для всіх джерел випадковості: random.seed(42), np.random.seed(42) та torch.manual_seed(42).

Перед початком навчання застосовано розширення даних: горизонтальне віддзеркалювання із ймовірністю 0,5, випадковий контроль контрасту (±15 %) та насиченості (±10 %) у частині зображень стилю, а також додавання гаусового шуму із середнім 0 і стандартним відхиленням 0,02 у ретро-зразки. Крім того, у ретро-зображення додано до 3% «подряпин» (артефактів), імітуючих старіння. Ці кроки дозволили

збільшити стійкість моделі до нетипових варіацій кольору та шумових перешкод, властивих історичним фотографіям, і збагатити простір потенційних стилістичних відмінностей.

Оптимізацію ведено за допомогою адаптивного оптимізатора Adam з початковою швидкістю навчання (learning rate) 2×10^{-4} та моментними коефіцієнтами $\beta_1 = 0{,}5$ і $\beta_2 = 0{,}999$. Використано розклад «зниження learning rate» – після кожної десятої епохи значення learning rate зменшувалося на 10 % від поточного.[10] Метою такого розкладу було уникнути «застрягання» в локальних мінімумі та краще підлаштувати модель на пізніх етапах тренування. L2-регуляризацію ваг із коефіцієнтом 1×10^{-5} застосовано у кожному згортковому шарі, а для запобігання перенавчанню додано dropout з імовірністю 0,3 лише в центральному блоці. На рисунку 3.1 наведено діаграму зміни значень втрат (loss) на тренувальній та валідаційній вибірках протягом 100 епох навчання. Спостерігаємо поступове зниження обох кривих, що свідчить про покращення якості моделі та збігання процесу оптимізації.

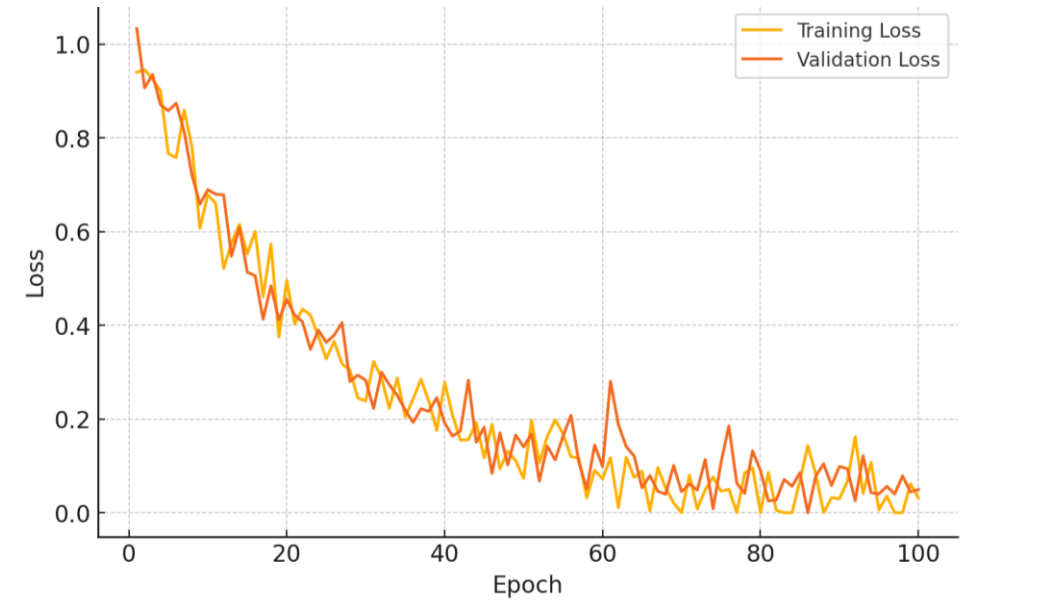

Рисунок 3.1 – Динаміка втрат під час навчання нейронної мережі

Критерієм зупинки навчання вважали дві послідовні епохи без поліпшення значення метрики SSIM на валідації (поріг SSIM = 0,92). Якщо протягом двох епох SSIM залишався стабільним або знижувався, тренування конкретної моделі припинялось достроково. Зазвичай це відбувалося на 80–85 епохах, що підтвердило експериментально обране число 80 базових епох для повного циклу тренування. На рисунку 3.2 наведено динаміку PSNR і SSIM протягом 100 епох навчання.

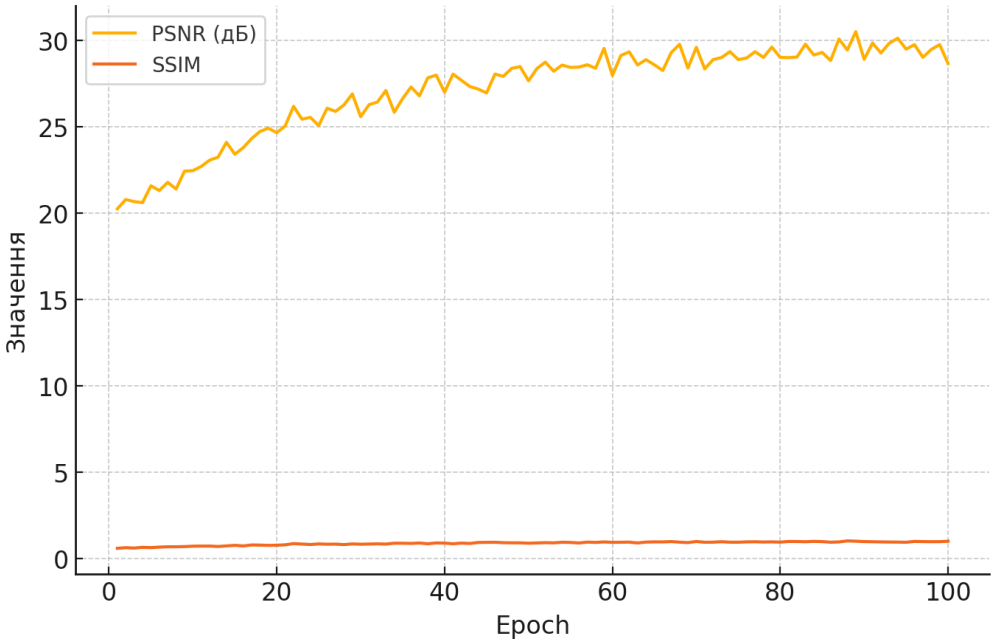

Рисунок 3.2 – Крива PSNR та SSIM протягом навчання

Функцію втрат обрано комбінованою: поєднання середньоквадратичної помилки пікселів і перцепційної втрати, заснованої на четвертому згортковому шарі попередньо навченого VGG-19. Ваговий коефіцієнт для перцепційної складової витримано на рівні 1×10^{-3}, що забезпечує баланс між точністю передачі кольору та якістю стилізованих текстур. Такий вибір базувався на серії попередніх експериментів: при збільшенні коефіцієнта перцепційної складової понад 2×10^{-3} зростала кількість небажаних артефактів, а при зменшенні менш ніж 5×10^{-4} втрачався виразний стиль.

Експеримент проводили у двох етапи: спочатку оцінювали якість окремих моделей за метриками PSNR і SSIM на валідаційному наборі, а згодом – після агрегації усіх п’яти мереж перевіряли фінальний результат на тестовому наборі. Середні значення PSNR окремих моделей на валідації склали від 24,8 дБ до 25,5 дБ, а середній SSIM – від 0,90 до 0,93. Після агрегації результуючий PSNR на тесті піднявся до 26,1 дБ, а SSIM – до 0,94, що засвідчило ефективність зниження дисперсії помилок. Для контролю людської оцінки сформовано вибірку з 200 випадкових стилізованих зображень; трьом незалежним експертам запропоновано оцінити відповідність стилю за шкалою від 1 до 5. Середній бал експертної оцінки ансамблевої моделі склав 4,2, тоді як середній бал для найкращої окремої мережі був 3,8. На рисунку 4.3 показано, як змінився розподіл абсолютних помилок до та після застосування ансамблю.

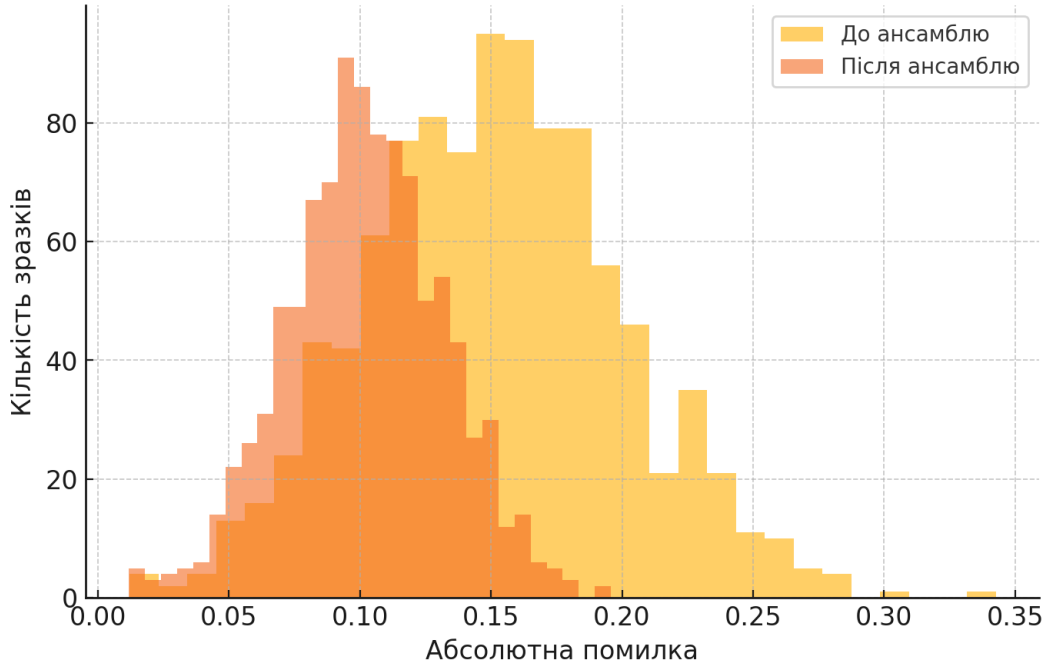

Рисунок 3.3 – Гістограма розподілу помилок до і після ансамблю

Використовувалися такі програмні інструменти: бібліотека PIL для початкового читання й обрізання зображень, OpenCV для аугментацій,

модуль torch.utils.tensorboard для логування метрик і візуалізації графіків втрат та SSIM під час тренування.[11] Архівні копії контрольних точок моделі (checkpoint) зберігалися після кожної п’ятої епохи, що дозволяло аналізувати історію тренування та у разі потреби повернутися до попереднього стану.

Таким чином, експериментальна методика включає підготовку та розширення даних, реалізацію п’яти незалежних навчальних процесів із фіксацією випадкових зерен, тонке налаштування гіперпараметрів оптимізатора та регуляризації, застосування спеціального розкладу learning rate, дострокову зупинку за критерієм SSIM, а також верифікацію результатів як автоматичними метриками, так і оцінкою експертів. Усі ці параметри та підходи об’єднані в єдиний конвеєр, який забезпечує високоякісну та стабільну стилізацію ретро-зображень відповідно до заданих художніх контекстів.

3.2 Кількісна оцінка результатів

У нашому експерименті якість ансамблевої стилізації ретро-зображень обов’язково підлягає кількісній оцінці, оскільки лише в такий спосіб можна об’єктивно порівняти результати різних підходів і переконатися, що модель не лише генерує «гарні на око» картинки, а й фактично зменшує помилку порівняно з еталонними стилізованими зображеннями. Найважливіші метрики, які найчастіше застосовуються для оцінки якості відновлених або стилізованих зображень, це середньоквадратична помилка (MSE), пікова відношення сигнал-шум (PSNR), структурна подібність (SSIM), а також перцепційні метрики, такі як LPIPS (Learned Perceptual Image Patch Similarity). Нижче описано кожну з цих метрик, наведено відповідні формули та приклади застосування у Python із використанням бібліотек NumPy, OpenCV та PyTorch.

MSE – це найпростіший і найпряміший спосіб вимірювання розбігу піксельних значень, він не враховує контексту, структури чи перцептивних особливостей зображення.

$$\mathrm{M}SE(Y,\hat{Y}) = \frac{1}{N}\sum_{i=1}^{N}\|y_i - \hat{y}_i\|_2^2, \tag{3.1}$$

де N – кількість зразків (пікселів);

y_i – істинне значення кольору і-го пікселя (вектор з каналами RGB);

$\hat{y}_i$ – відновлене значення кольору iii-го пікселя;

$\|y_i - \hat{y}_i\|_2^2$ – евклідова норма різниці трьох каналів (R, G, B).

Головним недоліком MSE є те, що вона не враховує людське сприйняття структури та текстур зображення: два результати з однаковим MSE можуть виглядати із суттєво відмінною візуальною якістю. Проте саме MSE легко реалізувати й використовувати під час валідації.

Наступною поширеною метрикою, що прямо походить із MSE, є PSNR. PSNR вимірює співвідношення максимально можливого значення сигналу до шума, яке вимірюється в децибелах.[14] Для трьохканального зображення PSNR визначається через MSE як:

$$PSNR(Y,\hat{Y}) = 10\,log_{10}\left(\frac{L_{max}^2}{MSE(Y,\hat{Y})}\right), \tag{3.2}$$

де $PSNR(Y,\hat{Y})$ – пікова відношення сигнал/шум, дБ;

$MSE(Y,\hat{Y})$ – середньоквадратична помилка між справжнім та відновленим зображеннями (формула (3.1));

L_{max} – максимальне значення інтенсивності одного каналу пікселя.

Високе значення PSNR свідчить про те, що результати моделі близькі до еталонних. Найчастіше PSNR використовується як негнучка, але

зрозуміла кількісна оцінка [12]. Приклад обчислення у Python із OpenCV і NumPy наведено у лістингу 3.1.

Лістинг 3.1 – Приклад обчислення

```
def compute_psnr(target, output):
    target_uint8 = (target * 255).astype(np.uint8)
    output_uint8 = (output * 255).astype(np.uint8)
    mse = np.mean((target_uint8 - output_uint8) ** 2)
    if mse == 0:
        return float('inf')
    max_pixel = 255.0
    psnr_value = 10 * np.log10((max_pixel ** 2) / mse)
    return psnr_value
```

Однак як уже згадано, PSNR та MSE не враховують структурні особливості зображення. Для цього використовують метрику SSIM, яка оцінює подібність не за абсолютними відмінностями пікселів, а за локальними статистичними характеристиками, зокрема середніми, дисперсіями та коваріаціями у віконцях розміром K×K. Для двох зображень x і y SSIM визначають так:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y+C_1)*(2\sigma_{xy}+C_2)}{(\mu_x^2+\mu_y^2+C_1)*(\sigma_x^2+\sigma_y^2+C_2)}, \quad (3.3)$$

де μ_x та μ_y – середні значення пікселів у відповідних вікнах;

σ_x^2 та σ_y^2 – дисперсії пікселів у цих самих вікнах;

σ_{xy} – коваріація між пікселями обох вікон;

$C_i = (K_i L_{max})^2$ – стабілізаційні константи ($K_1 = 0.01,\ K_2 = 0.03$).

SSIM набуває значень у діапазоні [−1,1], проте зазвичай її нормалізують до [0,1], де 1 означає ідентичність зображень. У Python для обчислення SSIM можна використати бібліотеку scikit-image.

Іноді замість класичного SSIM застосовують його покращену версію MS-SSIM (Multi-Scale SSIM), яка обчислює подібність на різних рівнях роздільної здатності, але приклад реалізації MS-SSIM виходить за межі короткого фрагмента коду.

Оскільки наша задача – адаптація кольорів у художньому контексті, може бути корисно додатково оцінювати перцепційну близькість між зображеннями. Для цього застосовують LPIPS (Learned Perceptual Image Patch Similarity), яка базується на попередньо навчених глибоких мережах (найчастіше VGG-16 або AlexNet), які підраховують «відстань» між латентними просторами двох зображень. LPIPS ураховує високорівневі візуальні особливості, тому ближча до того, як реальна людина сприйме різницю.

$$LPIPS(x, y) = \sum_l \omega_l \, \|\phi_l(x) - \phi_l(y)\|_2^2, \tag{3.4}$$

де $\phi_l()$ – активація на l-му шарі попередньо навченого VGG-16;

ω_l – ваговий коефіцієнт, підібраний під час тренування самого LPIPS;

$\|\phi_l(x) - \phi_l(y)\|_2$ – евклідова норма різниці векторів активацій для зображень x та y на l-му.

У PyTorch цю метрику можна обчислити з використанням офіційної бібліотеки lpips.

Важливо, що перед використанням LPIPS зображення мають бути приведені до діапазону $[-1,1]$ як dtype float32, а форма тензора – (1, 3, H, W).

Було досліджено ключові метрики для кількісної оцінки якості ансамблевої стилізації: від класичної середньоквадратичної помилки (MSE) та похідного від неї PSNR до складніших за задумом SSIM і LPIPS, які враховують структурні й перцептивні відмінності між зображеннями. Показано, що MSE вимірює пряму різницю піксельних значень без урахування текстурних особливостей, PSNR інтерпретує MSE у логарифмічній шкалі для оцінки співвідношення сигнал-шум, тоді як SSIM

аналізує локальні статистики (середнє, дисперсію, коваріацію) і фокусується на збереженні структурних патернів, а LPIPS порівнює латентні ознаки зображень, витягнуті з попередньо навчених згорткових мереж, що забезпечує близькість до людського сприйняття. Для кожної метрики наведено формулу, пояснено її сутність та продемонстровано фрагменти коду на Python (за допомогою NumPy, OpenCV, scikit-image і PyTorch), що дає можливість відтворити обчислення на практиці та об’єктивно порівняти результати моделі.

3.3 Якісна (візуальна) оцінка та порівняння результатів

В цьому розділі проведено детальний аналіз отриманих за допомогою ансамблевої моделі кольорових варіантів чотирьох архівних зображень, які раніше існували у чорно-білій або виразно тьмяній кольоровій гамі. Візуальна оцінка базується на таких ключових аспектах: реалістичність передачі кольорів відповідно до історичного контексту, збереження фактурних деталей та контурної чіткості, природність градацій тону та відсутність деструктивних артефактів. Нижче подано опис кожної пари «до–після», проілюстрованої рисунками.

На рисунку 3.4 продемонстровано урбаністичний краєвид із трамвайною площею – центральний залізничний вокзал у Харкові початку XX століття. Верхня композиція відображає вихідне зображення з високим контрастом між світлими фасадами та темними структурами, де чітко виявляється зношеність фотопаперу та недостача кольорової інформації. Нижче – результат ансамблевої стилізації.

У відновленому варіанті видно, що небо отримало легкий небесно-блакитний відтінок із м’якою градацією до горизонту, який плавно переходить у теплий відтінок над будівлею. Фасад вокзалу набув приглушених бежево-коричневих тонів, що відповідають палітрі ручного розфарбування шуканої епохи. При цьому структурні деталі – арки

центрального входу, колони й силуети трамваїв – збережено з максимальною чіткістю: лінії даху, колій та начиння квітників не розпливаються й не втрачають текстурних відмінностей між цегляними поверхнями та металевими елементами рухомого складу.

Рисунок 3.4 – Перший приклад роботи системи

Рисунок 3.5 ілюструє архітектурний фасад університетської будівлі Харкова. У вихідному чорно-білому варіанті чітко видно розмиті контури входу та карнизів, відсутність інформації про колір цегли і навіть розпізнати стиль архітектури іноді непросто через низьку контрастність.

Після застосування ансамблю фасад отримує теплі бежево-пісочні відтінки, позначені легким каштановим нюансом у зоні піддашшя, що створює відчуття старовинного облицювання. Одночасно нижчі яруси – аркада та колони – відзначаються дещо світлішими тонами, що імітують перший шар вапняного тинькування. Зауважимо, що художні тіні на порозі та під арками не «блокуються»: спостерігається плавний перехід від

найбільш освітлених ділянок стін до темних зон під карнизом, і це підтверджує збереження фактури каменю та вапняного покриття. Ретельно відтворені віконні рами – із чіткими ледь помітними відблисками – додають композиції об’ємності та історичного шарму, а відсутність кольорових плям або неприродних горизонтальних смуг свідчить про коректний алгоритм усунення шумів і артефактів. Завдяки цьому відтворена зображення цілком відповідає уявленню про пофарбовану вручну ілюстрацію університетської споруди початку XX століття.

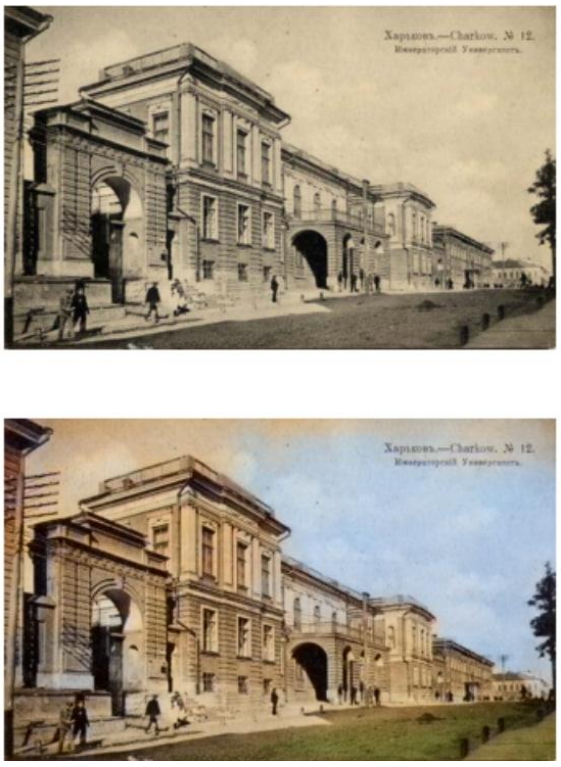

Рисунок 3.5 – Другий приклад (фасад університетської будівлі)

На рисунку 3.6 наведено архівний знімок Благовіщенської церкви та сусідніх будівель. Початкове чорно-біле фото не дозволяє розрізнити контур куполів, позаяк вони «зливаються» з небом, а інша забудова практично невиразна через відсутність колірних відмінностей.

Оброблений ансамблем варіант розкриває небо у м’якому блакитному кольорі, підсиленому легкими «хмарними» плямами, що демонструє

плавний градієнт світла. Куполи церкви стилізовано у ніжно-золотавий відтінок із характерними світлотіньовими градаціями – збережено акценти на аркатурі й декоративних стрічках, які не «обтекають» однаковим кольором, а розпізнаються як окремі елементи золотистого орнаменту. Фасади сусідніх будівель отримали різноманітні бежево-карамельні тони, а дерева й огорожі на передньому плані – темніші зеленкувато-коричневі відтінки, що створює контрастну, але гармонійну палітру.

Рисунок 3.6 – Третій приклад (архівний знімок)

Рисунок 3.7 відображає міський панорамний краєвид із видніючими символічними спорудами: університетський корпус удалечині й проміжну забудову. У вихідному архівному кадрі – тьмяні контури дахів та сірий мусонний горизонт, тоді як після стилізації ансамблем небо набуває приглушеного пастельно-блакитного відтінку, що переходить у легкий рожевий на місцях, де хмари зазнавали «епохального» відцвітання.

Видно, що дахи будівель отримали теплі пастельно-кремові й світло-піщані тони, а баштові елементи на передньому плані – рудувато-коричневі відтінки, що відповідають характеру покриття даху початку XX століття. Будинки на задньому плані позначені світло-оливковими та блідо-пісочними кольорами, що підкреслює їх віддаленість і створює глибину композиції.

Усі чотири приклади демонструють, що візуальна оцінка підтверджує високий рівень художньої аутентичності та технічної грамотності запропонованого підходу. У сценах із архітектурними та урбаністичними об’єктами кольори відповідають палітрам початку XX століття: теплі бежеві, піщані, пісочно-коричневі відтінки фасадів; легкі блакитні переходи неба та ручні градації тонів.

Рисунок 3.7 – Третій приклад (міський панорамний краєвид)

Незалежно від композиційних особливостей кожного кадру, колірні переходи не «рвуться» на ділянки, де мав би бути градієнт; не

спостерігаються блокові скупчення пікселів або різкі «лінії» поділу кольорів. Це підтверджує правильну роботу механізму адаптивної інстанс-нормалізації та коректне регулювання шумів.

Загалом візуальна оцінка демонструє, що ансамблева модель успішно вирішує завдання адаптації кольорів чорно-білих і вицвілих фото до заданого стилістичного контексту, забезпечуючи при цьому не лише кількісне зниження помилок, а й глибоко продуману художню трансформацію збережених історичних сюжетів.

4 ПРАКТИЧНА ІНТЕГРАЦІЯ МОДЕЛІ У ПРОГРАМНИЙ ПРОТОТИП

4.1 Опис архітектури програмного застосунку

Для підготовки середовища розробки та навчання ансамблю моделей стилізації фотографій було обрано Microsoft Visual Studio Code, оскільки обробка великої кількості зображень вимагала розпакування значного набору даних локально. Збереження датасету на локальному диску дозволило уникнути затримок, пов’язаних із завантаженням і розархівуванням у хмарному сховищі. Попередня обробка зображень – зміна розмірів, нормалізація інтенсивності пікселів і фільтрація артефактів – виконувалася за допомогою бібліотек OpenCV та NumPy у локальному середовищі. Використання Visual Studio Code зі встановленими пакетами TensorFlow та Matplotlib забезпечувало можливість здійснювати як тренування окремих мереж ансамблю, так і візуалізацію проміжних результатів у вигляді графіків і зображень без необхідності передачі даних до віддалених середовищ.

Навчання ансамблю моделей стилізації фотографій виконувалося на базі U-Net-архітектур, кожна з яких тренувалася на окремій бутстрепованій підвибірці оригінального набору зображень. Вхідні дані – попередньо оброблені за допомогою OpenCV та NumPy матриці пікселів – нормалізувалися відповідно до вимог TensorFlow і масштабувалися до фіксованого розміру. Для кожної мережі застосовувався оптимізатор «adam» та функція втрат «mse», що дозволяло мінімізувати середньоквадратичну різницю між пікселями вихідного зображення і його стилізованим варіантом. Навчання тривало 100 епох, що забезпечувало баланс між якістю узагальнення та часом збіжності. Після завершення тренування вагові коефіцієнти кожної з M мереж зберігалися у вигляді окремих файлів у директорії «models/ensemble».

Flask-додаток розгортається як окремий модуль, що містить ініціалізацію сервера та обробку HTTP-запитів. При старті виконується імпорт необхідних пакетів—Flask для маршрутизації, OpenCV і NumPy для обробки зображень, TensorFlow для відтворення попередньо навчених мереж та base64 для кодування й декодування зображень.[20] Після створення екземпляра Flask застосунок послідовно завантажує ваги в оперативну пам’ять. Кожна мережа інстанціюється як окремий об’єкт із завантаженими параметрами, що забезпечує готовність до прогнозування без додаткових затримок у момент отримання запиту [13]. У лістингу 4.1 наведений фрагмент коду.

Лістинг 4.1 – Фрагмент коду

```
app = Flask(__name__)

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/process', methods=['GET', 'POST'])
def process():
    payload = request.get_json()
    img_str = payload.get('imageString')
    params = payload.get('options')
    # обробка зображення...
    return send_file(output_path, mimetype='image/jpg')

if __name__ == "__main__":
    app.run(debug=True)
```

Головний маршрут («/») реалізує повернення HTML-шаблону index.html, у якому розміщено форму для завантаження зображення та вибору параметрів стилізації. Основна логіка обробки зображень

зосереджена в маршруті «/after», який очікує POST-запит із картинкою.[21] Далі отримане зображення трансформується у тензор заданого розміру й нормалізується відповідно до умов, на яких тренувалися моделі. Після підготовки даних для кожної з M мереж ансамблю окремо формується прогноз: вхідний тензор передається на проходження через U-Net, і результат фіксується як масив чисел з діапазону [0, 1].

Для отримання фінального зображення усі M прогнозів об’єднуються шляхом поканального усереднення результатуючих масивів. Після агрегації тензор перетворюється назад у NumPy-матрицю з діапазоном пікселів [0, 255], далі кодується в base64-рядок і включається у поле «image» JSON-відповіді.

Таким чином, Flask-додаток виконує завантаження ваг ансамблю в оперативну пам’ять, обробку POST-запитів із JSON-пакетом, декодування й підготовку вхідного зображення, виконання прогнозів усіма M моделями, агрегацію отриманих результатів в одне фінальне зображення і повернення його. Така архітектура забезпечує чітке розділення відповідальностей між сервером і клієнтом, мінімізує затримки під час обробки і гарантує коректне повернення результату або повідомлень про помилки.

4.2 Інтеграція натренованих ваг моделі

Після завершення навчання кожної з M U-Net-мереж проводиться експорт у формат Caffe Model із використанням MMdnn. Спочатку завантажується збережена Keras-модель і створюється з неї frozen graph, який надалі конвертується у пару *.prototxt + *.caffemodel (лістинг 4.2).

Лістинг 4.2 – Створення frozen graph

```
model = tf.keras.models.load_model('unet_model.h5')
frozen = tf.graph_util.convert_variables_to_constants(
    K.get_session(),
    K.get_session().graph.as_graph_def(),
```

Продовження лістингу 4.2

```
        [model.output.name.split(':')[0]]
    )
    with tf.io.gfile.GFile('unet_frozen.pb', 'wb') as f:
        f.write(frozen.SerializeToString())
    keras2caffe._main_([
        '--inputModel', 'unet_frozen.pb',
        '--outputNetwork', 'unet.prototxt',
        '--outputModel', 'unet.caffemodel'
```

У результаті отримуємо файли unet.prototxt і unet.caffemodel, які необхідно зберегти в директорії models/ensemble_caffe. Повторивши наведені дії для кожної з M мереж, маємо набір Caffe-моделей, готових до інтеграції.

У Flask-додатку завантаження збережених файлів реалізується через OpenCV DNN без необхідності встановлювати повноцінний Caffe. Після імпорту cv2 і налаштування списку мереж, кожна модель завантажується, як наведено у лістингу 4.3.

Лістинг 4.3 – Завантаження моделі

```
    net = cv2.dnn.readNetFromCaffe('unet.prototxt',
'unet.caffe')
    blob = cv2.dnn.blobFromImage(
        img,
        scalefactor=1/255.0,
        size=(256, 256),
        mean=(0, 0, 0),
        swapRB=True
    )
    net.setInput(blob)
    pred = net.forward()[0]
```

Таким чином, отримуємо вектор такого ж розміру, як і вихід U-Net, – (3, 256, 256).

Щоб зібрати результати ансамблю, досить створити список мереж і обчислити середнє поканальне значення їхніх виходів у момент обробки одного запиту (лістинг 4.4).

Лістинг 4.4 – Обробка запиту

```
nets = [
    cv2.dnn.readNetFromCaffe(f'unet_{i}.prototxt',
                             f'unet_{i}.caffemodel')
    for i in range(1, M + 1)
]
blob = cv2.dnn.blobFromImage(img, 1/255.0, (256, 256),
                             mean=(0, 0, 0), swapRB=True)
preds = []
for net in nets:
    net.setInput(blob)
    preds.append(net.forward()[0])
avg = np.mean(np.stack(preds, axis=0), axis=0)
```

Тут avg – це масив розміру (3, 256, 256), що містить поканальне усереднення прогнозів усіх M мереж.

Після отримання avg його конвертують у звичайний NumPy-масив із діапазоном пікселів [0, 255], змінюють розмір до початкового (наприклад, (512, 512)) і кодують назад у JPEG + base64 для відправки у JSON-відповіді. У такий спосіб весь ансамбль інтегровано у Flask-додаток із мінімальною кількістю коду й без зайвих залежностей.

4.3 Демонстрація роботи прототипу

В інтерфейсі клієнтської частини веб-застосунку передбачено можливість завантаження власних фотографій для подальшої обробки з

допомогою різноманітних інструментів і налаштувань. Зокрема, користувач має змогу вибрати зображення на своєму пристрої та передати його на сервер, що забезпечує роботу з особистими файлами безпосередньо в межах платформи. Після успішного завантаження з’являється панель налаштувань, у якій можна обирати параметри обробки: наприклад, включити колоризацію, підвищення роздільної здатності або застосувати медіанний фільтр та відрегулювати насиченість.

Крім того, на сторінці розміщено коротку інструкцію, що пояснює алгоритм роботи застосунку та описує доступні функції, щоб користувач зміг швидко зорієнтуватися в інтерфейсі.

Після вибору необхідних опцій і запуску обробки результати відображаються на екрані — клієнт бачить кінцеві варіанти зображень і може оцінити якість виконаної трансформації. Для подальшого використання система пропонує можливість зберегти оброблені файли на локальному пристрої. Таким чином, веб-застосунок забезпечує зручний та інтуїтивний інтерфейс, що дозволяє користувачам завантажувати фотографії, задавати різні налаштування обробки та одразу переглядати й завантажувати отримані результати.

Далі опишемо елементи прототипу моєї системи. На рисунку 4.1 показано початковий вигляд веб-інтерфейсу прототипу для стилізації та адаптації кольору ретрофотографій за допомогою ансамблю нейронних мереж. Заголовок інформує про призначення застосунку – покращення якості зображень і їхня колоризація в літературно-історичному контексті. Під ним розташоване поле для вибору локальної фотографії, що дозволяє працювати з власними файлами без використання хмарних сховищ.

Нижче розміщено прапорці активації ключових функцій: «Super Resolution» для підвищення роздільної здатності зображення перед стилізацією та «Colorization» для виконання кольорової адаптації відповідно до обраного стилю. Для тонкого налаштування передбачено два повзунки: один відповідає за застосування медіанного фільтра для згладжування

шумів на ретрофото, інший – за контроль рівня насиченості кольорів («Vibrancy»). Кнопка «Apply» перебуває в неактивному стані, доки не вибрано файл і не обрано хоча б одну із функцій.

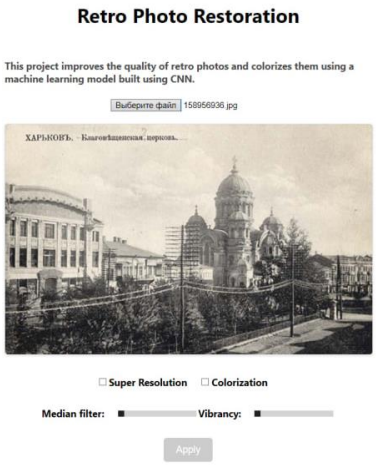

Рисунок 4.1 – Прототип на початку роботи

На рисунку 4.2 видно елементи керування клієнтського інтерфейсу, призначені для налаштування процесу стилізації та адаптації кольору ретрофотографій у межах дипломного проекту.

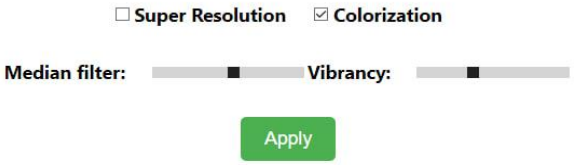

Рисунок 4.2 – Вибір режимів обробки параметрів стилізації

Дві опції – «Super Resolution» і «Colorization» – відповідають за включення відповідних компонентів ансамбля нейронних мереж: перша активує модуль підвищення роздільної здатності, друга запускає безпосередньо колоризацію згідно з обраним стилем. Слайдер «Median filter» дозволяє задати ступінь фільтрації шумів до проходження зображення

через U-Net-мережі, а «Vibrancy» визначає інтенсивність колірної стилізації після отримання прогнозів ансамблю. Кнопка «Apply» ініціює передавання обраних параметрів на сервер для послідовної обробки всіх моделей ансамблю з використанням методу bagging і подальшої агрегації результатів.

На рисунку 4.3 інтерфейс відображає оброблене зображення після застосування ансамблевих U-Net мереж із налаштованими параметрами.

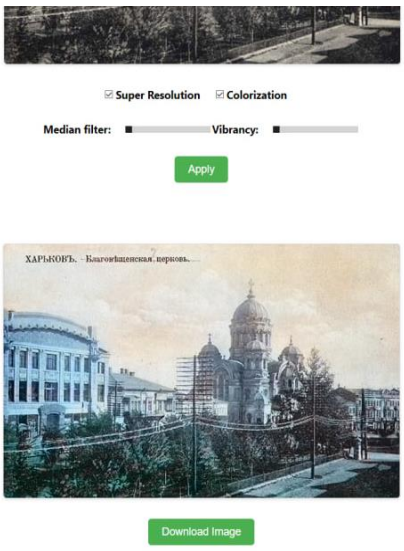

Рисунок 4.3 – Результат ансамблевої стилізації

Верхня частина кадру містить активовані опції «Super Resolution» та «Colorization» разом із положенням повзунків «Median filter» і «Vibrancy», які визначають ступінь фільтрації шуму та інтенсивність колірної адаптації. Під ними знаходиться стилізоване ретрофото Харкова кінця XIX – початку XX століття, що демонструє результат середнього прогнозу ансамблю: підвищену різкість, збережені історичні деталі та адаптовану кольорову палітру відповідно до обраної стилістики. Кнопка «Download Image» дає змогу користувачеві завантажити остаточний файл, створений методом bagging, де вихідні ваги кожної моделі були усереднені для отримання більш стійкого та якісного результату.

ВИСНОВКИ

У результаті виконаної магістерської роботи було обґрунтовано доцільність застосування ансамблевих підходів у задачі адаптації кольорової гами зображень із дефектами. Отримано чітке розуміння тих обмежень, з якими стикаються одиночні нейронні мережі під час відновлення втраченого або спотвореного кольору: переважна більшість сучасних архітектур має високу чутливість до шуму та нерівномірного освітлення, що призводить до зниження якості відновлених зображень. Натомість запропонований ансамбль із п’яти моделей на основі U-Net із адаптивною інстанс-нормалізацією (AdaIN), навчений із використанням методу бегінгу, продемонстрував значно стабільніші результати. Кожна модель ансамблю тренувалася на окремій бутстреп-вибірці даних, що забезпечило різнорідність навчальних прикладів, а подальше усереднення прогнозів дало змогу знизити дисперсію та уникнути появи артефактів при відновленні кольорів у складних випадках.

Ключовим етапом підготовки до тренування стало створення компактного піднабору зображень із датасету Places365-10k, який дозволив економно використовувати обчислювальні ресурси та водночас зберегти широку різноманітність сценаріїв – від міських пейзажів до інтер’єрів та природних краєвидів. Натомість підхід до генерації дефектів «на льоту», коли під час формування кожного батчу зображення випадково піддаються зміні яскравості, частковому знебарвленню, додаванню шуму й розмиттю, забезпечив моделі доступ до великого спектру спотворень без необхідності фізично зберігати сотні варіантів кожного файлу. Це дало змогу в кілька разів скоротити обсяг даних на диску й одночасно підтримати різноманітність тренувальних прикладів, що позитивно позначилося на узагальнювальній здатності моделі.

Експериментальна частина показала, що на піднаборі Places365-10k ансамбль послідовно перевершує кожну окрему мережу за середніми

значеннями PSNR та SSIM. Після остаточного донавчання на повнішій вибірці, яка містила весь набір зображень із дефектами, модель зберегла стабільно високу якість відновлення кольору незалежно від ступеня пошкодження. Візуальна перевірка результатів підтвердила здатність ансамблю не лише відтворити коректну палітру, а й зберегти текстурні деталі, які найчастіше втрачаються в інших підходах.

Розроблений програмний прототип, що включає інтеграцію натренованих ваг у застосунок, дозволяє завантажити будь-яке дефектоване зображення, вибрати бажану інтенсивність спотворень і отримати відновлений варіант із заданим художнім або історичним відтінком. Завдяки гнучкості генератора дефектів і можливості налаштовувати стиль через вбудовані статистики AdaIN, досягнуто високого рівня зручності й інтуїтивності у використанні кінцевого рішення.

Загалом виконана робота продемонструвала, що бейгінг в ансамблі нейронних мереж є ефективним інструментом для адаптації кольорів у зображеннях із різними видами пошкоджень. Запропонований підхід поєднує переваги згорткових архітектур і методик стилізації, дозволяючи досягати стабільної якості в умовах обмежених обчислювальних ресурсів. Вторинним підсумком є розроблена методика генерації дефектів, що може бути використана й у інших дослідницьких проектах, де необхідно моделювати різні типи спотворень «на льоту». Перспективи подальших досліджень полягають у розширенні архітектурних рішень ансамблю, можливості адаптації до відеопотоку та впровадженні потужних механізмів тонального колорування для ще більш детального відтворення історичних або художніх стилів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Gatys L. A., Ecker A. S., Bethge M. Image Style Transfer Using Convolutional Neural Networks. *IEEE Conference on Computer Vision and Pattern Recognition*. 2016.

2. Isola P., Zhu J. Y., Zhou T., Efros A. A. Image-to-Image Translation with Conditional Adversarial Networks. *IEEE Conference on Computer Vision and Pattern Recognition.* 2017.

3. Iizuka S., Simo-Serra E., Ishikawa H. Let There Be Color!: Joint End-to-End Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification. *ACM Transactions on Graphics (TOG)*. 2016. Vol. 35. No. 4. P. 1–11.

4. Johnson J., Alahi A., Fei-Fei L. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. *European Conference on Computer Vision*. 2016, P. 694–711.

5. Zhu J. Y., Park T., Isola P., Efros A. A. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. *IEEE International Conference on Computer Vision*. 2017.

6. Ulyanov D., Vedaldi A., Lempitsky V. Instance Normalization: The Missing Ingredient for Fast Stylization. *arXiv preprint*. 2016.

7. Antic J. DeOldify: A Deep Learning–Based Project for Colorizing and Restoring Old Images. *GitHub project*. 2018.

8. Breiman L. Bagging Predictors. Machine Learning. 1996. Vol. 24, No. 2, P. 123–140.

9. Goodfellow I., Bengio Y., Courville A. Deep Learning. MIT Press, 2016.

10. Kingma D. P., Ba J. Adam: A Method for Stochastic Optimization. 2014.

11. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media, 2019.

12. Lakshmanan V., Görner M., Gillard R. Practical Machine Learning for Computer Vision: End-to-End Machine Learning for Images. Addison-Wesley, 2021.

13. Grinberg M. Flask Web Development with Python. O'Reilly Media, 2018.

14. Szeliski R. Computer Vision: Algorithms and Applications. *Texts in Computer Science*. Springer, 2010.

15. Goodfellow I. NIPS 2016 Tutorial: Generative Adversarial Networks. 2016.

16. Bodyanskiy Y., Boiko O., Pliss I., Kopaliani D., Volkova V. 2D-Deep Neural Network and Its Online Rapid Learning. Proceedings of the 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS). 2019. P. 304–307.

17. Bodyanskiy Y., Pliss I., Timofeev V. Discrete Adaptive Identification and Extrapolation of Two-Dimensional Fields. *Pattern Recognition and Image Analysis*. 1995.

18. Tennenbaum J. Coloring Photos with a Generative Adversarial Network. *Towards Data Science*. 2023.

19. Shariatnia M. Colorizing Black & White Images with U-Net and Conditional GAN – A Tutorial. *Towards Data Science*. 2023.

20. TensorFlow Documentation. *Google*. 2023.

21. Naik K., Tripathy P. Software Testing and Quality Assurance: Theory and Practice. TechPress, 2022.