

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
Навчально-науковий фізико-технічний інститут
Кафедра математичного моделювання та аналізу даних

«На правах рукопису»

УДК 004.93:004.8

«До захисту допущено»

В.о. зав. кафедри

_____ Іван ТЕРЕЩЕНКО

«___» _____ 2026 р.

Дипломна робота
на здобуття ступеня бакалавра
за освітньо-професійною програмою
«Математичні методи моделювання, розпізнавання образів та комп'ютерного
зору»

зі спеціальності: 113 Прикладна математика
на тему: **«Інтерактивна багатоваріантна колоризація зображень на**
базі гібридної архітектури U-Net та V-Mamba»

Виконав:

студент 4 курсу, групи ФІ-21
Климентьєв Максим Андрійович

Керівник:

асистент кафедри ММАД д-р філософії
Железняков Дмитро Валентинович

Рецензент:

асистент кафедри СП д-р філософії
Яковчук Олег Костянтинович

Засвідчую, що у цій дипломній
роботі немає запозичень з праць
інших авторів без відповідних
посилань.

Студент _____

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»**

**Навчально-науковий фізико-технічний інститут
Кафедра математичного моделювання та аналізу даних**

Рівень вищої освіти — перший (бакалаврський)
Спеціальність — 113 Прикладна математика,
ОПП «Математичні методи моделювання, розпізнавання образів та
комп'ютерного зору»

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

_____ Іван ТЕРЕЩЕНКО

«___» _____ 2026 р.

**ЗАВДАННЯ
на дипломну роботу**

Студент: Климентьєв Максим Андрійович

1. Тема роботи: *«Інтерактивна багатоваріантна колоризація зображень на базі гібридної архітектури U-Net та V-Mamba»*, науковий керівник: асистент кафедри ММАД д-р філософії Железняков Дмитро Валентинович, затверджені наказом по університету №__ від «___» _____ 2026 р.

2. Термін подання студентом роботи: «___» _____ 2026 р.

3. Об'єкт дослідження: процес інтерактивної багатоваріантної колоризації зображень.

4. Предмет дослідження: моделі глибокого навчання для колоризації зображень, метрики для їх порівняльного аналізу, а також методи суб'єктивного оцінювання якості інтерактивної колоризації.

5. Перелік завдань:

- проаналізувати сучасний стан досліджень у задачі колоризації зображень та виявити обмеження існуючих методів глибокого навчання, зокрема архітектур на базі CNN, GAN, Transformer та Diffusion;

- сформувати набори даних для навчання і тестування, а також розробити алгоритм генерації кольорових підказок для імітації взаємодії з користувачем;

- спроектувати та реалізувати власну модель колоризації на базі гібридної архітектури U-Net та V-Mamba, а також інтегрувати базові моделі інших архітектур для порівняння;

- здійснити навчання розробленої моделі;

- розробити програмне забезпечення із графічним інтерфейсом для виконання інтерактивної багатоваріантної колоризації;

- провести комплексне оцінювання її ефективності шляхом порівняльного тестування за об'єктивними метриками і проведення користувацького дослідження.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу:
Презентація доповіді.

7. Дата видачі завдання: 3 вересня 2025 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання	Примітка
1	Узгодження теми роботи із науковим керівником	01-15 вересня 2025 р.	Виконано
2	Огляд опублікованих джерел за тематикою дослідження	Вересень-грудень 2025 р.	Виконано
3	Планування кроків проведення дослідження та збирання датасетів	Квітень 2026 р.	Виконано
4	Написання програмного забезпечення та проведення досліджень	Квітень-травень 2026 р.	Виконано
5	Аналіз отриманих результатів	Травень 2026 р.	Виконано
6	Оформлення дипломної роботи	Травень 2026 р.	Виконано
7	Попередній захист бакалаврської роботи	2 червня 2026 р.	Виконано
8	Подача закінченої бакалаврської роботи на кафедру	8 червня 2026 р.	Виконано
9	Захист бакалаврської роботи	18 червня 2026 р.	Виконано

Студент

_____ Максим КЛИМЕНТЬЄВ

Керівник

_____ Дмитро ЖЕЛЕЗНЯКОВ

РЕФЕРАТ

Кваліфікаційна робота містить: 74 стор., 15 рисунків, 5 таблиць, 36 джерел.

Мета роботи – розробка обчислювально ефективної системи інтерактивної багатоваріантної колоризації. Об’єкт дослідження – процес керованого відновлення кольору на монохромних зображеннях. Предмет дослідження – гібридні генеративні нейромережеві моделі на базі згорток та селективних моделей простору станів.

Спроектовано та реалізовано гібридну архітектуру B-Mamba, що поєднує точність локального вилучення ознак U-Net із глобальним рецептивним полем двонаправленої Mamba. Для багатоваріантності розроблено алгоритм стохастичної генерації підказок на основі градієнтної значущості. Впровадження спільних ваг дозволило мінімізувати апаратні вимоги. Створено вебзастосунок для керування колоризацією у реальному часі.

Експериментальний аналіз показав, що модель перевершує базові моделі, досягнувши FID 3.078 та LPIPS 0.143. Забезпечено низьке значення похибки дотримання підказок ($MSE_{HINT} = 6.1 \times 10^{-5}$) при інференсі 15.08 мс та споживанні 187 МБ відеопам’яті. Користувацьке дослідження підтвердило високу реалістичність генерації та точність просторового керування кольором.

КОЛОРИЗАЦІЯ ЗОБРАЖЕНЬ, ГЛИБОКЕ НАВЧАННЯ, U-NET, МАМБА, СЕЛЕКТИВНІ МОДЕЛІ ПРОСТОРУ СТАНІВ, ІНТЕРАКТИВНА КОЛОРИЗАЦІЯ, БАГАТОВАРІАНТНА КОЛОРИЗАЦІЯ

ABSTRACT

The qualification work contains: 74 pages, 15 figures, 5 tables, 36 references.

The purpose of the work is to develop a computationally efficient system for interactive multi-variant image colorization. The object of research is the process of controlled color restoration in monochrome images. The subject of research is hybrid generative neural network models based on convolutions and selective state space models.

A hybrid architecture, B-Mamba, was designed and implemented, combining the precision of U-Net local feature extraction with the global receptive field of bidirectional Mamba. For multi-variant colorization, an algorithm for stochastic hint generation based on gradient saliency was developed. The introduction of shared weights minimized hardware requirements. A web application was created for real-time colorization control.

Experimental analysis showed that the model outperforms baseline models, achieving an FID of 3.078 and LPIPS of 0.143. A low value of hint-following error was achieved ($\text{MSE}_{\text{HINT}} = 6.1 \times 10^{-5}$) with a 15.08 ms inference time and 187 MB of video memory consumption. A user study confirmed the high visual realism of the generation and the accuracy of spatial color control.

IMAGE COLORIZATION, DEEP LEARNING, U-NET, MAMBA, SELECTIVE STATE SPACE MODELS, INTERACTIVE COLORIZATION, MULTI-VARIANT COLORIZATION

ЗМІСТ

Перелік умовних позначень, скорочень і термінів	8
Вступ	9
1 Теоретичні основи колоризації зображень	13
1.1 Поняття та історія колоризації	13
1.2 Підходи до колоризації	15
1.2.1 Методи на основі втручання користувача	15
1.2.2 Методи перенесення кольору зі зразка	15
1.2.3 Згорткові нейронні мережі	16
1.2.4 Генеративно-змагальні мережі	17
1.2.5 Трансформери та механізми уваги	17
1.2.6 Дифузійні моделі	19
1.2.7 Селективні моделі простору станів	20
1.3 Інтерактивна колоризація	22
1.4 Метрики якості колоризації	22
1.5 Порівняння різних методів колоризації	25
Висновки до розділу 1	27
2 Проєктування та реалізація гібридної архітектури і системи інтерактивної багатоваріантної колоризації	28
2.1 Загальна концепція та конвеєр обробки даних	28
2.1.1 Стратегія симуляції підказок користувача під час навчання	29
2.1.2 Автоматизована симуляція підказок на етапі валідації	31
2.1.3 Алгоритм стохастичної генерації підказок для багатоваріантності	31
2.2 Компонентна деталізація та алгоритми навчання гібридної моделі ...	33
2.2.1 Згорткові блоки енкодера та декодера	33
2.2.2 Двонаправлене просторове сканування	34
2.2.3 Механізм спільного використання ваг	34
2.2.4 Загальна архітектура	35
2.2.5 Формулювання функцій втрат	36

2.3	Програмна реалізація та графічний інтерфейс користувача	38
	Висновки до розділу 2	39
3	Експериментальний аналіз результатів та оцінка ефективності системи...	40
3.1	Апаратне забезпечення та параметри навчання.....	40
3.2	Методологія тестування та базові моделі.....	41
3.3	Об’єктивний аналіз якості генерації та швидкодії.....	43
3.4	Оцінка перцептивної реалістичності та точності керування	44
3.5	Візуальні результати	45
3.6	Користувацьке дослідження якості колоризації.....	45
3.6.1	Методологія проведення опитування	46
3.6.2	Результати першого етапу: оцінка реалістичності	48
3.6.3	Результати другого етапу: точність слідування підказкам	49
3.6.4	Результати третього етапу: оцінка багатоваріантності	52
3.6.5	Кореляція між об’єктивними метриками та суб’єктивним сприйняттям	53
3.6.6	Висновки за результатами користувацького дослідження	55
	Висновки до розділу 3	55
	Висновки	56
	Перелік посилань.....	58
	Додаток А Фрагменти вихідного коду програми.....	64
A.1	Реалізація двонаправленого блоку Shared Mamba	64
A.1.1	Допоміжні модулі	64
A.1.2	Основний модуль.....	65
A.2	Генерація Гауссівських підказок.....	66
A.3	Багатоваріантність	68
A.4	Комбінована функція втрат	71
	Додаток Б Протокол узгодження.....	73

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

cGAN	— умовна генеративно-змагальна мережа (англ. conditional Generative Adversarial Network)
CLIP	— контрастивне попереднє навчання мови та зображень (англ. Contrastive Language-Image Pre-training)
CNN	— згорткова нейронна мережа (англ. Convolutional Neural Network)
FID	— відстань Фреше (англ. Fréchet Inception Distance)
GAN	— генеративно-змагальна мережа (англ. Generative Adversarial Network)
KID	— ядерна відстань (англ. Kernel Inception Distance)
LPIPS	— перцептивна метрика на базі глибоких ознак (англ. Learned Perceptual Image Patch Similarity)
MAE	— середньоабсолютна похибка (англ. Mean Absolute Error)
MSE	— середньоквадратична похибка (англ. Mean Squared Error)
PSNR	— пікове відношення сигналу до шуму (англ. Peak Signal-to-Noise Ratio)
SSIM	— індекс структурної схожості (англ. Structural Similarity Index)
SSM	— селективна модель простору станів (англ. Selective State Space Model)
VRAM	— відеопам'ять (англ. Video Random Access Memory)
ViT	— візуальний трансформер (англ. Vision Transformer)

ВСТУП

Актуальність дослідження. Стрімкий розвиток методів глибокого навчання відкрив нові можливості для розв’язання складних задач генерації та обробки візуальних даних, однією з яких є інтерактивна колоризація монохромних зображень. Актуальність розробки нових підходів у цій сфері зумовлена практичним попитом на інтелектуальні інструменти керування кольором для складного фотографічного контенту. Класичні методи інтерактивної колоризації на основі математичної оптимізації дозволяють поширювати колір від заданих користувачем підказок, проте вони суттєво обмежені відсутністю розуміння високорівневої семантики сцени. Оскільки такі алгоритми спираються переважно на низькорівневі градієнти яскравості, вони вимагають надмірно щільного нанесення маркерів для уникнення «розтікання» кольору за межі об’єктів зі слабо вираженими контурами. Натомість впровадження сучасних семантично обізнаних моделей глибокого навчання дозволяє досягти високоякісних результатів за мінімальної кількості підказок. Такі інструменти є затребуваними для швидкої генерації багатоваріантних колірних рішень у реальних прикладних задачах, наприклад, в електронній комерції, рекламній фотографії чи дизайні інтер’єрів, де виникає потреба миттєво перефарбовувати окремі об’єкти зі збереженням їхньої природної текстури та тіней.

Незважаючи на значний прогрес у галузі комп’ютерного зору, існуючі архітектурні рішення мають ряд суттєвих обмежень. Традиційні архітектури прямого відображення на базі згорткової нейронної мережі (англ. Convolutional Neural Network, CNN), умовної генеративно-змагальної мережі (англ. Conditional Generative Adversarial Network, cGAN) та стандартного візуального трансформера (англ. Vision Transformer, ViT) найчастіше є детермінованими. Такі моделі зводять задачу до однозначного математичного відображення, що ускладнює природну генерацію альтернативних кольорових рішень для одного й того ж вхідного зображення.

З іншого боку, стохастичні генеративні архітектури здатні забезпечувати багатоваріантність. Зокрема, генеративно-змагальні мережі (англ. Generative Adversarial Network, GAN) формують різні результати завдяки перетворенню випадкових векторів із латентного простору шуму. Своєю чергою, генеративні трансформери, наприклад авторегресивні, забезпечують варіативність за рахунок імовірнісного семплювання токенів із передбаченого розподілу на кожному кроці генерації, а дифузійні моделі (англ. Diffusion Models) досягають цього завдяки ітеративному процесу зняття шуму з різних випадкових початкових станів. Однак складність керування та значна ресурсоемність таких систем, що зумовлені важкістю інтеграції точних просторових умов у глобальний латентний простір (у випадку GAN), послідовним характером декодування токенів у поєднанні з квадратичною складністю уваги (у генеративних трансформерах) та потребою у багаторазовому проходженні мережі під час інференсу (у дифузійних моделях), роблять їх важкозастосовними для миттєвої інтерактивної взаємодії у реальному часі, де від системи вимагається швидкий відгук на колірні підказки.

У цьому контексті перспективним напрямком є дослідження новітніх архітектур, зокрема селективних моделей простору станів (англ. Selective State Space Models, SSM). На сьогодні рішення на основі SSM для задач інтерактивної багатоваріантної колоризації зображень загального вигляду відсутні, а існуючі поодинокі дослідження обмежені вузькоспеціалізованими доменами даних. Оскільки чисті архітектури такого класу мають обмежені можливості щодо збереження чітких локальних просторових ознак та текстур зображення, виникає обґрунтована необхідність проектування саме гібридних моделей. Це зумовлює доцільність поєднання глобального контексту SSM із локальними властивостями згорткових підходів для створення обчислювально ефективних архітектур, здатних забезпечити високу якість генерації та миттєвий відгук на дії користувача.

Мета дослідження. Розробка моделі колоризації на базі гібридної архітектури U-Net + V-Mamba, створення програмного забезпечення інтерактивної колоризації з можливістю вибору різних моделей та проведення

порівняльного аналізу із сучасними базовими моделями (англ. baseline models).

Задача дослідження. Для досягнення поставленої мети необхідно вирішити такі завдання:

1) проаналізувати сучасний стан досліджень у задачі колоризації зображень та виявити обмеження існуючих методів глибокого навчання, зокрема архітектур на базі CNN, GAN, Transformer та Diffusion;

2) сформулювати набори даних для навчання і тестування, а також розробити алгоритм генерації кольорових підказок для імітації взаємодії з користувачем;

3) спроектувати та реалізувати власну модель колоризації на базі гібридної архітектури U-Net та B-Mamba, а також інтегрувати базові моделі інших архітектур для порівняння;

4) здійснити навчання розробленої моделі;

5) розробити програмне забезпечення із графічним інтерфейсом для виконання інтерактивної багатоваріантної колоризації;

6) провести комплексне оцінювання її ефективності шляхом порівняльного тестування за об'єктивними метриками і проведення користувацького дослідження.

Об'єктом дослідження є процес інтерактивної багатоваріантної колоризації зображень.

Предметом дослідження є моделі глибокого навчання для колоризації зображень, метрики для їх порівняльного аналізу, а також методи суб'єктивного оцінювання якості інтерактивної колоризації.

При розв'язанні поставлених завдань використовувались такі *методи дослідження*: методи глибокого навчання та методи комп'ютерного зору, методи об'єктивного та суб'єктивного оцінювання якості результатів.

Наукова новизна. Наукова новизна отриманих результатів полягає у наступному:

- вперше застосовано SSM до задачі інтерактивної багатоваріантної колоризації монохромних зображень загального вигляду, що дозволило розширити сферу використання цього класу моделей за межі аналізу вузькоспеціалізованих даних, а саме інфрачервоних та супутникових знімків;

- запропоновано нову модифіковану гібридну архітектуру, на базі поєднання U-Net та B-Mamba, для задач інтерактивної колоризації. За рахунок інтеграції механізму двонаправленого сканування зі спільними вагами вперше вдалося ефективно об'єднати здатність SSM до захоплення глобального контексту з локальною просторовою точністю згорткових мереж, суттєво оптимізувавши споживання відеопам'яті.

Практичне значення. Практичне значення отриманих результатів полягає у створенні швидкого та зручного програмного інструментарію для задач інтерактивної колоризації. Зокрема:

- розроблена гібридна модель пропонує ефективний компроміс між обчислювальною легкістю та точністю керування. Вона перевершує класичні інтерактивні моделі у точності локалізації підказок і водночас може слугувати швидкою альтернативою важким дифузійним моделям та трансформерам у сценаріях, де пріоритетом є миттєвий відгук у режимі реального часу;

- створено програмний застосунок із графічним інтерфейсом, який спрощує користувацьку взаємодію з моделями. Інструментарій дозволяє зручно наносити просторові підказки, генерувати зображення та легко перемикатися між розробленою і базовими архітектурами для прямого візуального порівняння їхніх результатів.

1 ТЕОРЕТИЧНІ ОСНОВИ КОЛОРИЗАЦІЇ ЗОБРАЖЕНЬ

У цьому розділі розглядаються основні теоретичні аспекти колоризації зображень, включно з історією розвитку методів, класичними алгоритмічними підходами та сучасними нейромережевими моделями. Особлива увага приділяється різноманітним підходам: від архітектур на основі CNN до дифузійної моделі, а також методам оцінки якості отриманих результатів.

Основна мета цього розділу – проаналізувати еволюцію методів колоризації від класичних CNN до сучасних архітектур типу трансформер, виявити їхні обмеження щодо обчислювальної складності та збереження просторових контурів об'єктів, і на основі цього обґрунтувати необхідність переходу до гібридних ієрархічних архітектур на базі SSM.

1.1 Поняття та історія колоризації

Колоризація – це процес додавання кольорової інформації до монохромних зображень. З математичної точки зору, цю задачу можна сформулювати як відображення одноканального сірого зображення у триканальне кольорове зображення, наприклад, у просторі RGB, CIELAB або YUV [1]. Оскільки одному значенню яскравості може відповідати безліч кольорових відтінків, задача є невизначеною та має характер «один до багатьох», що вимагає використання апріорних знань про семантику сцени.

Історично методи колоризації можна поділити на три основні етапи:

1) **ручна та напіваавтоматична колоризація:** вимагала значних зусиль художників або застосування алгоритмів на основі локальних кольорових підказок, сегментації зображень та методів перенесення кольору між зображеннями [2, 3, 4];

2) **методи на основі глибокого навчання:** поява великих наборів даних, таких як ImageNet та COCO, дозволила тренувати CNN для передбачення

кольору [5]. Знаковим став підхід *Colorful Image Colorization* [6], де задачу було переформульовано з класичної регресії на класифікацію. Замість того, щоб мінімізувати усереднену похибку, яка призводить до тьмяних кольорів, модель передбачає розподіл ймовірностей для кожного пікселя по палітрі відтінків. Подальший розвиток цього етапу ознаменувався впровадженням GAN, що дозволило подолати проблему розмитості та суттєво підвищити фотореалістичність результатів завдяки змагальному процесу навчання;

3) **генеративні та послідовні моделі:** сучасний етап, що характеризується використанням механізмів глобальної уваги трансформерів та дифузійних моделей для генерації високоякісних текстур, а також впровадженням SSM для лінійного моделювання контексту зображень та забезпечення високої семантичної узгодженості [7, 8].

Розвиток методів глибокого навчання, зокрема поява архітектур U-Net та згаданих раніше GAN, дозволив перейти від простого розфарбовування до генерації фотореалістичних зображень [9]. Останні дослідження все частіше фокусуються на дифузійних моделях, які демонструють передові результати у задачах синтезу зображень [10].

Поряд із цим, активного розвитку набули архітектури на базі механізмів глобальної уваги. Проте висока обчислювальна складність класичних трансформерів, а також обмеження лінійних та інших ефективних апроксимацій у точному моделюванні глобального контексту [11, 12], роблять їх менш придатними для інтерактивних задач.

Водночас новітні архітектури на базі SSM здатні забезпечити глобальне рецептивне поле з лінійною складністю. Хоча вони й почали з'являтися у вузькоспеціалізованих задачах відновлення інфрачервоних [13, 14] та супутникових знімків [15], вони залишаються недослідженими в контексті інтерактивної колоризації фотографій. Відсутність рішень, які б поєднували лінійну обчислювальну ефективність SSM із точним просторовим контролем за допомогою користувацьких підказок, створює простір для наукового пошуку та підкреслює актуальність запропонованого в цій роботі гібридного підходу.

1.2 Підходи до колоризації

Підходи до колоризації базуються на різних архітектурах нейронних мереж. Опустимо гібридні архітектури та розглянемо деякі з базових. До епохи глибокого навчання колоризація розглядалася переважно як задача оптимізації. Класичні алгоритмічні методи можна розділити на дві великі групи: методи на основі втручання користувача та методи перенесення кольору зі зразка.

1.2.1 Методи на основі втручання користувача

Цей підхід базується на припущенні, що сусідні пікселі з подібною інтенсивністю повинні мати подібний колір. Користувач наносить на чорно-біле зображення кольорові штрихи, які слугують граничними умовами. Задача зводиться до мінімізації цільової функції енергії J [2]:

$$J(U) = \sum_r \left(U(r) - \sum_{s \in N(r)} w_{rs} U(s) \right)^2,$$

де $U(r)$ – колір у пікселі r , $N(r)$ – окіл пікселя, а w_{rs} – ваговий коефіцієнт, що залежить від подібності яскравості пікселів r та s .

Перевагою методу є повний контроль користувача, проте він вимагає значних ручних зусиль і часто дає артефакти на межах об'єктів зі слабким контрастом.

1.2.2 Методи перенесення кольору зі зразка

У цьому підході колір переноситься з референсного кольорового зображення на цільове чорно-біле шляхом зіставлення статистик яскравості та текстур.

Ранні роботи [3, 4] використовували зіставлення гістограм у

декорельованому колірному просторі $l\alpha\beta$ (простір Рудермана), що дозволяло маніпулювати колірними каналами незалежно один від одного без виникнення візуальних артефактів.

Хоча цей метод є більш автоматизованим, він критично залежить від вдалого вибору референсного зображення та часто призводить до помилкового забарвлення семантично різних об'єктів, що мають схожу текстуру.

1.2.3 Згорткові нейронні мережі

Поява CNN дозволила моделювати складні залежності між формою об'єкта та його ймовірним кольором, використовуючи великі набори даних.

Перші нейромережеві методи намагалися прямо передбачити значення каналів ab у просторі Lab, мінімізуючи MSE:

$$\mathcal{L}_{L2} = \frac{1}{2} \sum_{h,w} \|Y_{gt} - \hat{Y}\|_2^2,$$

де Y_{gt} – справжнє зображення, а $\hat{Y}$ – згенероване. Проте, як зазначають Zhang et al. [6], використання MSE призводить до усереднення всіх можливих кольорів, що дає ненасичені колірні результати.

Для вирішення цієї проблеми задачу було переформульовано як мультимодальну класифікацію, де мережа передбачає розподіл ймовірностей для квантованих відтінків кольору, що значно підвищило яскравість результатів. В роботах [16, 6] запропонували використовувати ознаки, вивчені для класифікації об'єктів, для задачі колоризації, демонструючи тісний зв'язок між семантикою об'єкта та його кольором. Інший популярний метод, *Deep Koalarization* [17], використовує Inception-ResNet-v2 для вилучення ознак та подвійний декодер для відновлення кольору.

1.2.4 Генеративно-змагальні мережі

З метою усунення ефекту недостатньої насиченості кольорів були запропоновані різні GAN. Мережі GAN, зокрема архітектура Pix2Pix [9], здійснили прорив у чіткості зображень. Генератор G намагається створити реалістичне зображення, щоб обдурити дискримінатор D , який вчиться відрізняти справжні кольорові фото від згенерованих.

Функція втрат cGAN має вигляд:

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z)))],$$

де x – вхідне чорно-біле зображення, y – цільове кольорове зображення, а z – вектор випадкового шуму.

Зокрема, модель *Pix2Pix* [9] використовує cGAN для перетворення зображень, що дозволяє генерувати більш чіткі та реалістичні результати. Своєю чергою, метод *ChromaGAN* [18] поєднує геометричну інформацію з семантичним розподілом класів для покращення правдоподібності розфарбовування, що дозволяє уникнути грубих помилок, наприклад, зеленого неба. Також існують підходи для непарного навчання, такі як CycleGAN [19], що дозволяють тренувати моделі без наявності пар «чорно-біле – кольорове».

1.2.5 Трансформери та механізми уваги

Останні дослідження [20, 21, 22] показують, що класичні CNN мають локальне рецептивне поле, через що вони погано справляються з моделюванням глобальних зв'язків. Це часто призводить до семантичних помилок: наприклад, різні частини одного об'єкта, перекриті іншими елементами сцени, можуть бути розфарбовані в різні кольори.

Перехід до архітектур на базі трансформерів дозволив вирішити цю проблему. Вони використовують механізм глобальної уваги, який дозволяє кожному патчу зображення взаємодіяти з усіма іншими патчами одночасно,

враховуючи контекст усієї сцени. Це є критично важливим для просторово узгодженого забарвлення великих областей та підтримки єдиної логіки освітлення.

Зокрема, модель *Colorization Transformer* [20] вперше успішно застосувала цей підхід, розбивши зображення на послідовність патчів і розглядаючи задачу колоризації як задачу прогнозування послідовності токенів, подібно до генерації тексту. Модель генерує кольори авторегресійно, що забезпечує високу якість, проте робить процес інференсу надзвичайно повільним.

Значним кроком у напрямку інтерактивності стала модель *UniColor* [21]. Вона запропонувала уніфікований фреймворк, який здатний обробляти мультимодальні умови – від автоматичної колоризації до генерації на основі текстових описів, референсних зображень або точкових підказок. Використання трансформера дозволило ефективно інтегрувати ці різноманітні підказки у єдиний простір репрезентацій.

Для вирішення проблеми розмиття контурів, яка часто виникає у чистих трансформерів, архітектура *DDColor* [22] впровадила систему подвійних декодерів та запити кольору (англ. Color Queries) для досягнення фотореалістичності. Один декодер відповідає виключно за відновлення просторової роздільної здатності, тобто за піксельний рівень, а інший – за формування кольорових семантичних ознак, тобто за глобальний контекст, після чого їхні результати зливаються.

Проте, незважаючи на беззаперечні переваги в якості генерації, використання класичного механізму уваги накладає жорсткі апаратні обмеження. Обчислювальна складність глобальної уваги зростає квадратично $\mathcal{O}(N^2)$ зі збільшенням роздільної здатності зображення. Це робить такі моделі вкрай вимогливими до обсягу відеопам'яті (англ. Video Random Access Memory, VRAM) і ускладнює їх використання для інтерактивної колоризації зображень у високій якості в реальному часі.

Спроби подолати цю проблему за рахунок так званих «ефективних трансформерів» призводять до нових компромісів. Наприклад, обчислення уваги в локальних вікнах, як-от, в Swin Transformer [23], або використання

математичних лінійних та інших апроксимацій можуть обмежувати здатність моделі до повного глобального моделювання контексту [11, 12]. У задачі інтерактивної колоризації це стає критичним недоліком, оскільки нейромережа втрачає здатність швидко та безшовно поширювати колір від точкової підказки користувача на віддалені ділянки об'єкта.

1.2.6 Дифузійні моделі

Найбільш перспективним напрямком на сьогодні є дифузійні моделі. На відміну від GAN, які генерують результат за один прохід, дифузійні моделі формують зображення ітеративно. Математична основа класичних дифузійних моделей базується на двох марковських процесах: прямому, який поступово руйнує дані шляхом додавання гаусового шуму, та зворотному, де нейронна мережа вчиться крок за кроком відновлювати оригінальний сигнал, керуючись вхідним чорно-білим зображенням як просторовою умовою. Хоча у класичних підходах під деградацією розуміють поступове додавання випадкового гаусового шуму, сучасні методи розширюють це поняття на будь-які типи спотворень.

Цей клас моделей здійснив революцію в синтезі зображень завдяки високій стабільності навчання, тобто відсутності проблеми колапсу мод, який характерний для GAN, та неперевершеній якості генерації дрібних текстур.

Зокрема, *Palette* [10] є універсальним фреймворком для перетворення зображень, який перевершує GAN та регресійні моделі без необхідності налаштування під конкретну задачу. Поряд із цим, метод *Cold Diffusion* [24] пропонує підхід до інверсії довільних детермінованих перетворень зображень без використання гаусового шуму, що відкриває нові можливості для відновлення зображень та розширює теоретичні межі застосування дифузійних алгоритмів.

Проте фундаментальним недоліком дифузійних моделей залишається їхня надзвичайно висока обчислювальна вартість на етапі інференсу. Оскільки для створення фінального зображення модель повинна виконати десятки або сотні послідовних ітерацій денойзингу, час генерації зростає на порядки порівняно з моделями прямого проходу. У контексті інтерактивної колоризації,

де фундаментальною вимогою є миттєвий візуальний зворотний зв'язок на кожну нову підказку користувача, використання чистих дифузійних моделей є недоцільним через руйнування користувацького досвіду.

1.2.7 Селективні моделі простору станів

Новітнім трендом у глибокому навчанні є впровадження SSM, зокрема архітектури Mamba [8]. Вони розроблені як обчислювально ефективна альтернатива архітектурам типу трансформер. Новітнім класом нейромережових архітектур, що дозволяє подолати квадратичну складність трансформерів та повільність дифузійних моделей, є SSM. Головною перевагою SSM є здатність моделювати наскрізні глобальні контекстні зв'язки з суворою лінійною обчислювальною складністю $\mathcal{O}(N)$ відносно довжини послідовності, що є критичним фактором для забезпечення інтерактивності та роботи систем у реальному часі.

Математично базова система SSM відображає вхідний одновимірний сигнал $x(t) \in \mathbb{R}$ у прихований стан $h(t) \in \mathbb{R}^N$ через лінійні диференціальні рівняння:

$$\begin{aligned} h'(t) &= A \cdot h(t) + B \cdot x(t), \\ y(t) &= C \cdot h(t) + D \cdot x(t), \end{aligned}$$

де A, B, C, D – матриці параметрів системи. При цьому матриця A характеризує внутрішню динаміку системи та керує еволюцією прихованого стану, тобто відповідає за довготривалу пам'ять, матриця D описує прямий зв'язок між входом і виходом, виконуючи роль класичного пропускового з'єднання, а матриці B та C визначають коефіцієнти взаємодії прихованого стану із вхідним та вихідним сигналами відповідно.

Для обробки дискретних послідовностей безперервні параметри трансформують у дискретні $(\bar{A}, \bar{B})$ із використанням кроку дискретизації Δ . Зазвичай для цього застосовують метод фіксації нульового порядку (англ.

Zero-Order Hold):

$$\begin{aligned}\bar{A} &= \exp(\Delta A), \\ \bar{B} &= (\Delta A)^{-1}(\exp(\Delta A) - I) \cdot \Delta B,\end{aligned}$$

що дозволяє переписати систему у дискретному вигляді: $h_k = \bar{A} \cdot h_{k-1} + \bar{B} \cdot x_k$,
 $y_k = C \cdot h_k + D \cdot x_k$.

Головна інновація архітектури Mamba полягає у селективності її внутрішніх механізмів: параметри B , C та крок дискретизації Δ стають динамічними функціями від вхідного сигналу, на відміну від статичних матриць A та D . Завдяки такій дискретизації параметрів та використанню апаратно-оптимізованого алгоритму паралельного сканування в пам'яті графічного процесора, модель динамічно фільтрує нерелевантну інформацію та ефективно обробляє довгі послідовності без обчислювальних обмежень механізму уваги.

Однак пряме застосування оригінальної одновимірної Mamba до двовимірних зображень призводить до проблеми втрати локального контексту, оскільки розгортання матриці пікселів у вектор розриває їхню просторову суміжність по вертикалі. Для вирішення цієї проблеми візуальні адаптації на кшталт Vision Mamba [25] та новітні моделі міжспектрального перенесення зображень [13] використовують механізми двонаправленого або перехресного сканування. Це дозволяє моделі агрегувати ознаки з різних напрямків, повністю відновлюючи просторове розуміння сцени.

У контексті інтерактивної колоризації використання Mamba відкриває унікальну можливість: здатність моделі миттєво та узгоджено поширювати колірну інформацію від точкової підказки користувача на весь об'єкт завдяки глобальному рецептивному полю, уникаючи при цьому експоненційного зростання споживання відеопам'яті. Саме цей синергетичний баланс обчислювальної ефективності та просторової експресивності робить SSM найбільш перспективним фундаментом для розробки сучасних інтерактивних систем розфарбовування.

1.3 Інтерактивна колоризація

Оскільки колоризація є відносно суб'єктивною задачею, важливу роль відіграють методи, що дозволяють користувачеві впливати на результат:

1) **колоризація на основі прикладів:** методи, такі як *Deep Exemplar-based Colorization* [26], використовують референсне кольорове зображення для перенесення колірної палітри на цільове чорно-біле;

2) **колоризація на основі тексту:** з розвитком мультимодальних моделей, таких як CLIP, з'явилася можливість керувати кольором за допомогою текстових описів, зокрема *L-CAD* [27] використовує дифузійні пріори для колоризації на основі описів будь-якого рівня деталізації, а *Diffusing Colors* [28] пропонує фреймворк, де текстові підказки керують процесом дифузії, забезпечуючи баланс між автоматизацією та контролем;

3) **колоризація на основі підказок користувача:** методи *Real-Time User-Guided Image Colorization* [29] та *iColoriT* [30] дозволяють користувачеві ставити кольорові точки або штрихи, які мережа поширює на відповідні семантичні регіони.

Сучасна тенденція, як показано в *Control Color* [31] та *UniColor* [21], полягає у створенні уніфікованих фреймворків, що підтримують різні типи умов, такі як текст, зразок, точки, штрихи, в одній моделі.

1.4 Метрики якості колоризації

Оцінка якості колоризації є нетривіальним завданням через фундаментальну неоднозначність розв'язку: одному чорно-білому вхідному сигналу відповідає ціла множина семантично та візуально коректних колірних варіацій. Оскільки відображення має характер $1 \rightarrow N$, використання лише одного критерію не дозволяє повноцінно оцінити роботу моделі. Для комплексного аналізу розроблених архітектур у роботі застосовується комбінований набір метрик, що охоплює такі аспекти.

Метрики локальної та глобальної похибки. У задачах інтерактивної колоризації похибку доцільно розділяти за областю обчислення. Зокрема, метрика середньоквадратичної похибки (англ. Mean Squared Error, MSE) та середньоабсолютної похибки (англ. Mean Absolute Error, MAE) оцінюють загальне відхилення кольору на всьому зображенні. Вони обчислюються за формулами:

$$\text{MSE}(x, y) = \frac{1}{N} \sum_{i=1}^N (x_i - y_i)^2, \quad \text{MAE}(x, y) = \frac{1}{N} \sum_{i=1}^N |x_i - y_i|,$$

де x_i та y_i – значення i -го пікселя згенерованого та еталонного зображень відповідно; N – загальна кількість пікселів.

Спеціалізовані метрики MSE_{HINT} та MAE_{HINT} розраховуються за тими ж математичними формулами, але виключно в просторових координатах наданих користувачем підказок. Наближення MSE_{HINT} та MAE_{HINT} до нуля є індикатором того, що мережа не ігнорує користувацький ввід, а ідеально інтегрує його у фінальний результат.

Піксельні та структурні метрики. Пікове відношення сигналу до шуму (англ. Peak Signal-to-Noise Ratio, PSNR) [32] та індекс структурної схожості (англ. Structural Similarity Index, SSIM) [33] – класичні методи порівняння, які вимірюють пряме відхилення згенерованого зображення від еталонного кольорового зображення. PSNR обчислюється як:

$$\text{PSNR}(I, J) = 10 \log_{10} \left(\frac{\text{MAX}^2}{\text{MSE}(I, J)} \right),$$

де MAX – максимальне можливе значення пікселя зображення. Метрика SSIM розраховується за формулою:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)},$$

де μ_x, μ_y – локальні середні значення яскравості зображень; σ_x^2, σ_y^2 – дисперсії; σ_{xy} – коваріація; c_1, c_2 – малі константи для забезпечення чисельної стабільності.

Хоча ці метрики зручні для математичного аналізу, у задачах колоризації

їхня репрезентативність є обмеженою: якщо модель згенерує семантично правильну червону машину замість оригінальної синьої, вони зафіксують значну помилку, незважаючи на високу візуальну правдоподібність результату.

Перцептивні генеративні метрики. Перцептивна метрика на базі глибоких ознак (англ. Learned Perceptual Image Patch Similarity, LPIPS) [34] використовується для оцінки попарної схожості зображень на рівні глибоких ознак нейронної мережі VGG:

$$\text{LPIPS}(x, x_0) = \sum_l \frac{1}{H_l W_l} \sum_{h,w} \|w_l \odot (\hat{y}_{hw}^l - \hat{y}_{0hw}^l)\|_2^2,$$

де $\hat{y}^l, \hat{y}_0^l$ – карти ознак l -го шару для згенерованого та еталонного зображень; H_l, W_l – просторові розміри цих карт; w_l – вектор ваг; $\odot$ – поелементне множення.

Натомість відстань Фреше (англ. Fréchet Inception Distance, FID) [35] та ядерна відстань (англ. Kernel Inception Distance, KID) [36] оцінюють відстань між розподілами ознак усього набору даних:

$$\text{FID}(x, y) = \|\mu_x - \mu_y\|^2 + \text{Tr} \left(\Sigma_x + \Sigma_y - 2(\Sigma_x \Sigma_y)^{\frac{1}{2}} \right),$$

де μ_x, μ_y – вектори середніх значень ознак; Σ_x, Σ_y – матриці коваріації; Tr – слід матриці.

$$\text{KID}(x, y) = \text{MMD}(x, y)^2,$$

де MMD – максимальна середня розбіжність (Maximum Mean Discrepancy) між представленнями ознак. Зниження цих показників свідчить про те, що статистика згенерованих зображень наближається до реальних фотографій. Оскільки KID обчислюється ітеративно на випадкових підвбірках, результат подається у форматі середнього значення KID_{mean} та стандартного відхилення KID_{std} , що характеризує стабільність моделі.

Обчислювальна та апаратна ефективність. Оскільки розроблювана архітектура орієнтована на інтерактивність, критичними параметрами є час інференсу, Time, та пікове споживання VRAM.

1.5 Порівняння різних методів колоризації

Комплексне порівняння розглянутих методів (табл. 1.1) базується на чотирьох ключових критеріях (Рис. 1.1).

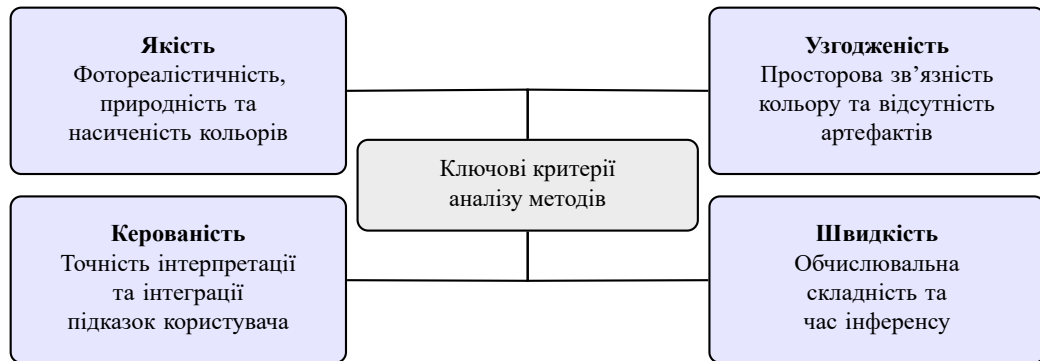


Рисунок 1.1 – Ключові критерії порівняння методів колоризації

Для візуального представлення розглянутих методів на рис. 1.2 та рис. 1.3 наведено приклади результатів їхньої роботи.

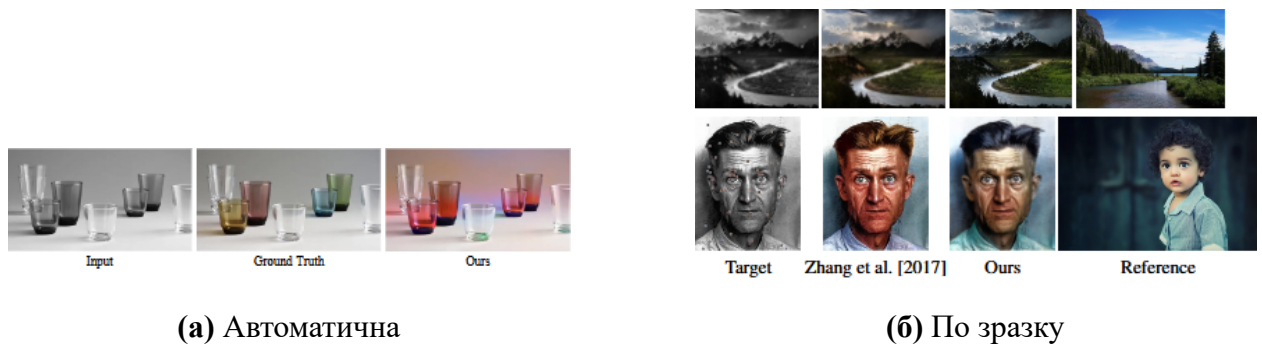
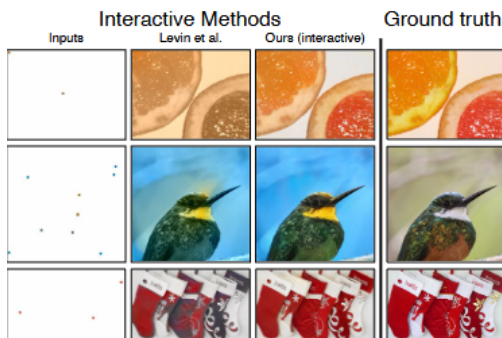


Рисунок 1.2 – Приклади результатів роботи автоматичного підходу та колоризації за зразком (ілюстрації взято з робіт [22, 26])

Проведений аналіз демонструє наявність компромісу між обчислювальною лінійністю та збереженням локальних просторових контурів у сучасних послідовних моделях. Чисті трансформери забезпечують високу якість зв'язності, але стають апаратно непридатними для обробки великих зображень у реальному часі, тоді як SSM пропонують бажану лінійну ефективність $\mathcal{O}(N)$,

Таблиця 1.1 – Порівняльний аналіз методів колоризації зображень

Метод / Архітектура	Представники	Переваги	Недоліки
Класичні оптимізаційні	Levin et al. [2], Welsh [4]	Точний локальний контроль користувача, відсутність семантичних «галюцинацій»	Обчислювально повільні, вимагають значної ручної розмітки, пасують перед складними текстурами
CNN	Zhang et al. [6], Larsson [16]	Висока швидкість інференсу, стабільність процесу навчання	Обмежене рецептивне поле; схильність до генерації усереднених, тьмяних відтінків при мінімізації MSE
GAN	Pix2Pix [9], ChromaGAN [18]	Висока чіткість контурів, насичені та візуально привабливі кольори	Нестабільність змагального навчання, схильність до появи кольірних артефактів та плям
Transformers	ColTran [20], UniColor [21], DDColor [22]	Моделювання глобального контексту всієї сцени, гнучка мультимодальна керованість	Квадратична обчислювальна складність $\mathcal{O}(N^2)$, критично високі вимоги до обсягу VRAM
Diffusion Models	Palette [10], Cold Diffusion [24]	Найвища якість генерації (SOTA), бездоганний фотореалізм текстур	Екстремально низька швидкість інференсу через сотні послідовних ітерацій денойзингу
Mamba SSM	Vision Mamba [25], ColorMamba [13]	Суворо лінійна складність $\mathcal{O}(N)$, стабільне споживання VRAM, глобальне рецептивне поле	Чутливість до одновимірного розгортання 2D-даних, ризик втрати або розмиття локальних контурів об'єктів

**(а) Інтерактивна****(б) По тексту****Рисунок 1.3 – Приклади результатів роботи інтерактивного та текстового підходів до колоризації (ілюстрації взято з робіт [29, 27])**

але потребують додаткових механізмів локального посилення контурів.

Таким чином, гібридні підходи, що поєднують генеративну силу та швидкість селективних лінійних моделей простору станів із точністю локального вилучення ознак за допомогою ієрархічних згорткових декодерів,

виглядають найбільш обґрунтованими та перспективними для вирішення задачі інтерактивної колоризації. Саме розробці та дослідженню такої гібридної архітектури присвячено наступні розділи цієї роботи.

Висновки до розділу 1

У цьому розділі було проведено огляд теоретичних основ та сучасних методів колоризації зображень. Було показано, що еволюція методів пройшла шлях від простих евристик до складних генеративних моделей, таких як GAN, Transformers, дифузійні фреймворки та селективні моделі простору станів.

Попри значні успіхи безумовної колоризації у досягненні реалістичності результатів, залишаються відкритими питання обчислювальної складності моделей під час роботи із зображеннями високої роздільної здатності, керованості процесом колоризації та забезпечення багатоваріантності результатів. Зокрема, гібридні підходи, мультимодальні підходи та уніфіковані дифузійні й рекурентні структури є найбільш актуальним напрямком досліджень.

Порівняння нейромережових підходів показало, що хоча архітектури GAN забезпечують швидкий інференс, вони часто страждають від нестабільності навчання та появи кольірних артефактів. Дифузійні моделі дозволяють досягти найвищої якості та фотореалізму текстур, проте їхнє практичне застосування жорстко обмежене тривалим часом інференсу через ітеративний характер процесу генерації. У свою чергу, підходи на базі трансформерів ефективно моделюють глобальний контекст усієї сцени, однак квадратичне зростання обчислювальних витрат $\mathcal{O}(N^2)$ суттєво ускладнює їх масштабування для зображень високої роздільної здатності.

У наступному розділі буде запропоновано власний підхід до вирішення задачі колоризації зображень, що базується на гібридизації архітектури U-Net та Mamba SSM з використанням спільних ваг. Це дозволить ефективно поєднати вилучення низькорівневих просторових ознак, для боротьби з розтіканням кольору, із лінійним моделюванням глобального контексту та суттєвим збереженням пам'яті графічного процесора.

2 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ГІБРИДНОЇ АРХІТЕКТУРИ І СИСТЕМИ ІНТЕРАКТИВНОЇ БАГАТОВАРІАНТНОЇ КОЛОРИЗАЦІЇ

У цьому розділі наведено опис практичної реалізації системи інтерактивної багатоваріантної колоризації зображень та її основної моделі на базі гібридної архітектури U-Net та V-Mamba.

Розглянуто процеси попередньої обробки вхідних даних та алгоритми симуляції користувацьких підказок.

Деталізовано внутрішню будову розробленої нейромережі, наведено обґрунтування двонаправленого сканування послідовностей ознак та описано обрану функцію втрат для забезпечення стабільного навчання моделі.

2.1 Загальна концепція та конвеєр обробки даних

Розроблена система реалізує метод інтерактивної колоризації, за якого нейромережа розфарбовує монохромне зображення, спираючись як на власне розуміння семантики об'єктів, так і на точкові колірні підказки користувача.

Математично вхідний потік даних формується шляхом поєднання двох складових:

- канал яскравості $I_L \in \mathbb{R}^{1 \times H \times W}$, який представляє компонент L кольорного простору CIELAB, нормалізований до діапазону $[-1, 1]$;
- триканальний тензор підказок $I_{\text{hints}} \in \mathbb{R}^{3 \times H \times W}$, який містить хроматичні значення та інформацію про місцезнаходження кольорових підказок.

Структурно тензор I_{hints} складається з двох каналів кольоровості a та b розмірністю $2 \times H \times W$ (нормалізованих у $[-1, 1]$) та маски інтенсивності підказок $M \in [0, 1]^{1 \times H \times W}$.

Маска M необхідна для того, щоб мережа могла відрізнити зони з явними підказками від порожнього простору. Без цього каналу модель сприймала б нульові значення у вільних зонах як вимогу розфарбувати ту частину зображення

у нейтральні ахроматичні кольори.

Перед подачею на вхід енкодера канал яскравості конкатенується з тензором підказок за виміром каналів, утворюючи загальний чотириканальний вхідний масив:

$$X_{\text{in}} = \text{Concat}(I_L, I_{\text{hints}}) \in \mathbb{R}^{4 \times H \times W}.$$

2.1.1 Стратегія симуляції підказок користувача під час навчання

Оскільки під час реального використання кількість, розмір та розташування точок є випадковими, процес навчання моделі вимагає динамічної симуляції дій користувача на кожній ітерації. Навчання на фіксованих шаблонах призвело б до перенавчання моделі та ігнорування підказок на інференсі.

Конвеєр підготовки даних під час тренування реалізує таку стохастичну процедуру для кожного батчу:

1) з оригінального кольорового зображення вилучаються істинні канали кольоровості $Y \in \mathbb{R}^{2 \times H \times W}$;

2) випадковим чином визначається цільова кількість підказок K з дискретного геометричного розподілу $K \sim \text{Geom}(p = 0.2)$, в якому математичне сподівання становить $\mathbb{E}[K] = \frac{1-p}{p}$, тобто в середньому генерується 4 підказки. Ймовірність появи конкретної кількості підказок k обчислюється як $P(K = k) = p(1 - p)^k$. Зокрема, ймовірність генерації зображення взагалі без підказок, тобто $K = 0$, становить 20%, що змушує модель також вчитися безумовній автоматичній колоризації;

3) для імітації фокусу людської уваги координати підказок генеруються за допомогою алгоритму виділення візуальної значущості. Алгоритм обчислює двовимірну функцію щільності ймовірності (PDF) на основі каналу яскравості I_L .

Спочатку за допомогою згортки з операторами Собеля обчислюється магнітуда градієнтів:

$$G = \sqrt{(I_L * S_x)^2 + (I_L * S_y)^2} + \epsilon,$$

де I_L – матриця каналу яскравості зображення; S_x, S_y – ядра оператора Собеля для виділення горизонтальних та вертикальних перепадів яскравості; $*$ – операція двовимірної згортки; ϵ – мала константа, наприклад, 10^{-6} , для уникнення обчислення кореня з нуля та забезпечення чисельної стабільності. Для поширення значущості від контурів усередину об’єктів до матриці G застосовується просторове усереднення. Отримана карта нормалізується та змішується з рівномірним розподілом U з ваговим коефіцієнтом $\alpha = 0.15$, щоб забезпечити ненульову ймовірність вибору точок навіть в абсолютно однорідних областях зображення:

$$\text{PDF} = (1 - \alpha) \frac{G_{\text{blur}}}{\sum G_{\text{blur}}} + \alpha U,$$

де G_{blur} – матриця градієнтів G після застосування операції просторового згладжування; $\sum G_{\text{blur}}$ – сума всіх елементів згладженої матриці, використовується як нормалізуючий дільник; U – матриця рівномірного розподілу ймовірностей, де кожен елемент дорівнює $\frac{1}{H \cdot W}$; α – ваговий коефіцієнт змішування, який наразі $\alpha = 0.15$.

Координати семплюються з цього розподілу без повернення. Додатково застосовується евристика мінімальної просторової відстані, яка відкидає нові координати, якщо вони знаходяться занадто близько до вже існуючих, уникаючи неприродного скупчення кліків;

4) формується маска інтенсивності M : у вибраних координатах розміщуються двовимірні радіальні гаусові ядра випадкового радіуса та випадкової амплітуди інтенсивності (від 0.2 до 1.0). Це імітує різні розміри пензля користувача та ступінь «впевненості» підказки. Якщо гаусові плями перекриваються, маска оновлюється за принципом максимуму $M_{\text{new}} = \max(M_{\text{old}}, G_{\text{patch}})$;

5) формуються канали підказок a та b шляхом поелементного множення істинних значень Y на згенеровану маску M .

2.1.2 Автоматизована симуляція підказок на етапі валідації

Для об'єктивної оцінки якості моделі під час валідації та тестування, алгоритм перемикається з випадкової генерації у детермінований режим. Алгоритм автоматично знаходить найбільш інформативні зони для колоризації, діючи за такою логікою:

- 1) обчислюється матриця спектральної магнітуди, насиченості кольору, для всього зображення як сума квадратів значень каналів a та b ;
- 2) визначаються координати пікселя з максимальною магнітудою, і в цій точці розміщується гаусова підказка фіксованого розміру;
- 3) щоб запобігти вибору точок в одному й тому самому об'єкті, застосовується механізм просторового пригнічення: значення магнітуди в заданому радіусі навколо обраної точки примусово встановлюються у від'ємні значення, а саме -1.0 ;
- 4) процес ітеративно повторюється для заданої кількості підказок, наприклад, $K = 3$.

Такий конвеєр валідації гарантує, що модель отримує підказки саме в найскладніших, найбільш яскравих ділянках зображення, що дозволяє коректно оцінити її здатність до поширення кольору на великі просторові відстані.

2.1.3 Алгоритм стохастичної генерації підказок для багатоваріантності

З огляду на детерміновану природу базової архітектури, механізм багатоваріантної колоризації винесено на етап попередньої обробки даних. Отримання множини альтернативних результатів для одного зображення забезпечується алгоритмом автоматичного доповнення підказок, який складається з таких етапів:

- 1) на основі користувацького тензора підказок формується маска зайнятості. Усі пікселі, де маска інтенсивності $M > 0.05$, позначаються

як пріоритетні та захищаються від автоматичної модифікації, що гарантує збереження намірів користувача.

2) для вибору координат нових підказок алгоритм стохастично, з рівною ймовірністю, обирає одну з двох стратегій семплювання:

- *семплювання на основі градієнтної значущості*, що діє аналогічно до процесу навчання, описаного у п. 2.1.1, та спрямоване на дрібні високочастотні деталі;

- *семплювання семантичних центрів*, де до каналу I_L застосовується детектор границь Canny, контури інвертуються, після чого обчислюється карта відстаней $\mathcal{D}$. Області, які вже містять підказки, пригнічуються за допомогою алгоритму зв'язної заливки із порогом толерантності $\tau = 20$. Координати обираються як глобальні максимуми карти $\mathcal{D}$, що гарантує точне потрапляння підказок у геометричні центри порожніх об'єктів.

3) для кожної знайденої координати обчислюється випадковий колір. Для забезпечення візуальної природності та запобігання надмірній кислотності результатів, хроматичні значення беруться з нормального розподілу із нульовим математичним сподіванням та обмеженим стандартним відхиленням $\sigma = 0.25$, після чого обрізаються до робочого діапазону $[-1, 1]$:

$$C_{\text{rand}} = \text{Clamp}(\mathcal{N}(0, \sigma^2), -1, 1).$$

4) нові підказки формуються у вигляді двовимірних гаусових ядер випадкового радіуса від 2 до 8 пікселів та випадкової амплітуди. Згенеровані маски та кольорові значення поєднуються з оригінальним тензором I_{hints} за допомогою операції попіксельного максимуму для масок та попіксельного додавання для колірних каналів, утворюючи модифікований вхідний масив для мережі.

Отже, інтеграція стохастичного семплювання до конвеєра даних уможливорює генерацію широкого спектра альтернативних рішень, гарантуючи при цьому абсолютне збереження тих кольорових умов, які були явно задані користувачем.

2.2 Компонентна деталізація та алгоритми навчання гібридної моделі

Розроблена гібридна модель базується на ієрархічній архітектурі типу U-Net, у якій класичний центральний згортковий вузол замінено на Mamba.

Такий підхід дозволяє делегувати вилучення локальних просторових ознак та збереження контурів перевіреним згортковим шарам, тоді як завдання глобального поширення кольору від точкових підказок виконується SSM з лінійною обчислювальною складністю.

2.2.1 Згорткові блоки енкодера та декодера

Для вилучення просторових ознак та поступового зниження роздільної здатності використовується згортковий енкодер. Кожен його рівень побудований за допомогою *DoubleConv*, структурну схему якої зображено на рис. 2.1.

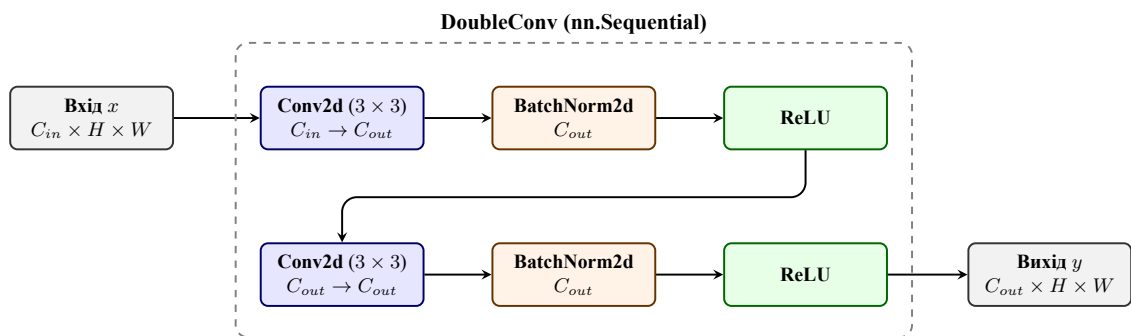


Рисунок 2.1 – Архітектура блоку DoubleConv для енкодера та декодера

Блок складається з двох послідовних згорткових шарів (3×3), за кожним з яких слідує нормалізація батчів та нелінійна функція активації ReLU. Після кожного блоку в енкодері застосовується операція просторового стиснення – Max Pooling 2×2 , яка зменшує розмірність карт ознак удвічі.

Декодер працює симетрично, використовуючи білінійну інтерполяцію для масштабування.

Критичним елементом архітектури є наскрізні з'єднання, які прокидають карти ознак з енкодера безпосередньо у відповідні шари декодера, що запобігає втраті дрібних контурів об'єктів, тобто розтіканню кольору.

2.2.2 Двонаправлене просторове сканування

Оригінальна архітектура Mamba розроблена для обробки одновимірних послідовностей, наприклад, тексту, і сканує дані лише в одному напрямку. Для двовимірних зображень це створює проблему просторової асиметрії: пікселі не отримують контекстної інформації від тих областей, що знаходяться нижче або правіше від них по сітці сканування.

Для вирішення цієї проблеми у центральному згортковому вузлі реалізовано алгоритм двонаправленого сканування. Розгорнута у вектор послідовність просторових ознак $X \in \mathbb{R}^{B \times L \times D}$, де L – це H помножена на W та поділена на певний коефіцієнт, який залежить від U-Net, а C – кількість початкових каналів, у нашому випадку це 4, обробляється паралельно: у природному напрямку та в інвертованому.

Інверсія досягається за допомогою безвитратної операції перегортання тензора у пам'яті (`torch.flip`). Фінальний вихід Y обчислюється як середнє арифметичне двох проходів:

$$Y = \frac{1}{2} (\text{Mamba}_{\text{fwd}}(X) + \text{Flip}(\text{Mamba}_{\text{rev}}(\text{Flip}(X)))) .$$

Це забезпечує кожному просторовому токеноу доступ до повного глобального контексту сцени з усіх напрямків.

2.2.3 Механізм спільного використання ваг

Використання глибоких послідовностей Mamba-блоків вимагає значного обсягу відеопам'яті графічного процесора. Для оптимізації апаратного споживання без втрати генеративної здатності мережі імплементовано механізм

спільного використання ваг, структурну логіку якого зображено на рис. 2.2.

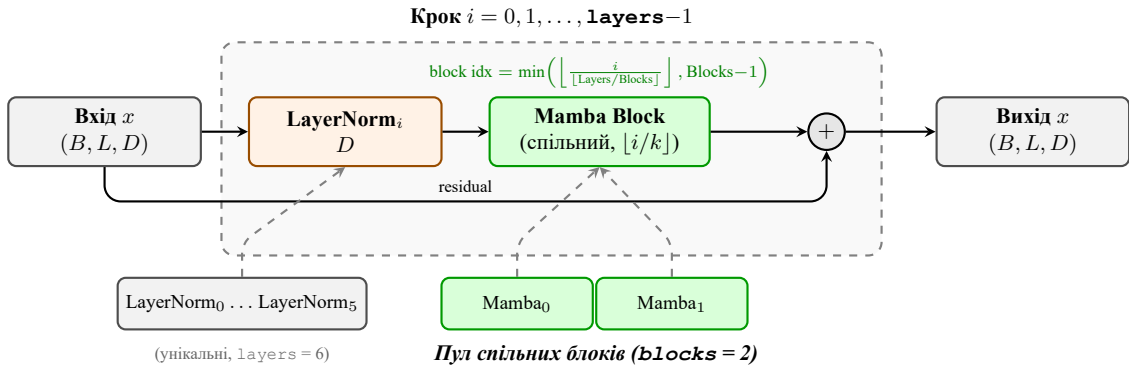


Рисунок 2.2 – Внутрішня архітектура та механізм розділення ваг у блоці Shared Mamba

Замість ініціалізації L унікальних блоків Mamba, модель створює лише обмежений фізичний пул з K блоків (де $K < L$). Під час прямого проходу шари послідовно перевикористовують ці фізичні блоки згідно з математичним індексом:

$$\text{idx} = \min\left(\left\lfloor \frac{i}{\lfloor L/K \rfloor} \right\rfloor, K - 1\right), \quad \text{де } i \in [0, L - 1].$$

Водночас, шари пакетної нормалізації (LayerNorm), які передують кожному блоку, залишаються унікальними для кожного кроку i . Такий підхід дозволяє моделі проєктувати ознаки в різні підпростори та успішно адаптуватися до різних рівнів абстракції попри спільні вагові матриці A, B, C, D всередині самих блоків простору станів.

2.2.4 Загальна архітектура

Загальна архітектура розробленої мережі представлена на рис. 2.3.

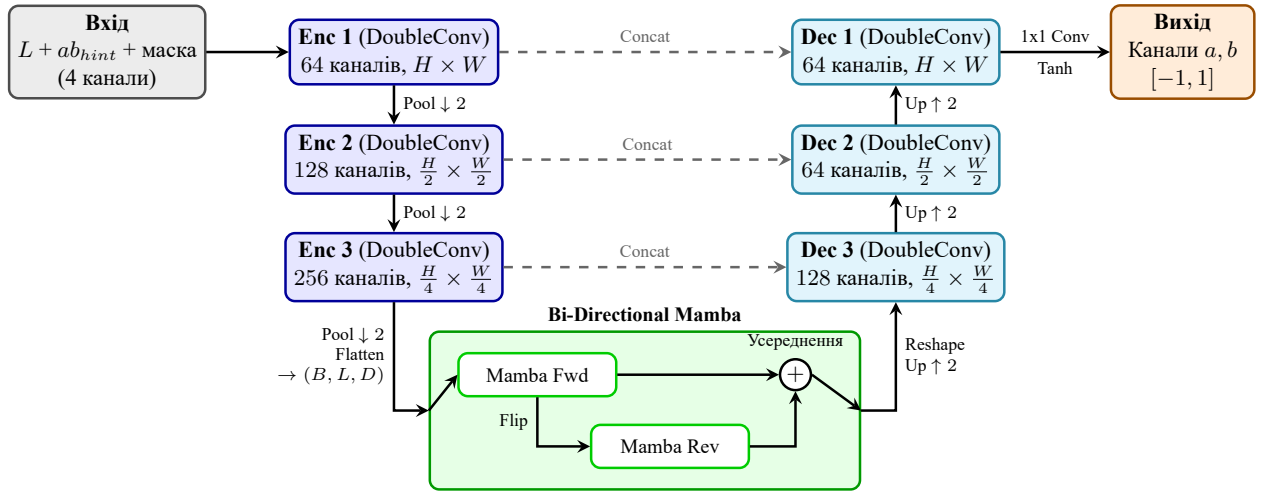


Рисунок 2.3 – Архітектура гібридної нейронної мережі Bidirectional Shared Mamba

2.2.5 Формулювання функцій втрат

Процес оптимізації гібридної мережі керується складеною цільовою функцією втрат, яка балансує піксельну точність відтворення кольору, суворе дотримання підказок користувача та візуальний реалізм згенерованого зображення.

Загальна функція втрат $\mathcal{L}_{\text{total}}$ формується як зважена сума трьох незалежних компонентів:

$$\mathcal{L}_{\text{total}} = \lambda_{L1} \mathcal{L}_{L1} + \lambda_{\text{hint}} \mathcal{L}_{\text{hint}} + \lambda_{\text{LPIPS}} \mathcal{L}_{\text{LPIPS}},$$

де λ_{L1} , λ_{hint} , λ_{LPIPS} – відповідні вагові коефіцієнти, що задають пріоритет кожної метрики.

Глобальна похибка кольоровості ($\mathcal{L}_{L1}$). Базовою метрикою для оцінки правильності прогнозування каналів a та b обрано середню абсолютну похибку. На відміну від MSE, яка схильна до генерації тьмяних та усереднених кольорів через надмірне штрафування значних відхилень, $\mathcal{L}_{L1}$ стимулює модель до створення більш насичених результатів. Похибка обчислюється по всьому простору зображення між згенерованими $\hat{Y} = (\hat{a}, \hat{b})$ та істинними $Y = (a, b)$

каналами:

$$\mathcal{L}_{L1} = \frac{1}{N} \sum_{i=1}^N |Y_i - \hat{Y}_i|.$$

Штраф за ігнорування підказок ($\mathcal{L}_{\text{hint}}$). Для забезпечення жорсткої реакції моделі на взаємодію з користувачем вводиться спеціалізований штраф, який обчислюється виключно в тих пікселях, де маска підказок M є активною. Щоб запобігти дисбалансу градієнтів між зображеннями з різною кількістю підказок, похибка нормалізується на фактичну кількість пікселів підказки у конкретному зображенні з додаванням константи $\epsilon = 10^{-8}$ для уникнення ділення на нуль:

$$\mathcal{L}_{\text{hint}} = \frac{1}{\max(\epsilon, \sum M_i)} \sum_{i=1}^N M_i \cdot |Y_i - \hat{Y}_i|.$$

Цей компонент змушує нейромережу ставити найвищий пріоритет на бездоганне поширення цільового кольору саме з тих точок, які визначив користувач.

Перцептивна похибка ($\mathcal{L}_{\text{LPIPS}}$). Оскільки піксельні метрики, такі як L1, часто не корелюють із суб'єктивним сприйняттям якості зображення людиною, до конвеєра навчання інтегровано перцептивну похибку LPIPS. Вона обчислює відстань між зображеннями у просторі високорівневих ознак попередньо навченої згорткової мережі, у нашій системі – архітектури VGG.

Оскільки VGG-екстрактор натренований на зображеннях у колірному просторі RGB, математичний граф обчислення похибки включає диференційовне перетворення колірних просторів «на льоту». Спочатку нормалізовані канали денормалізуються до своїх фізичних масштабів – до каналу L додається 1 і разом це масштабується коефіцієнтом 50.0, канали a, b – коефіцієнтом 110.0. Далі відновлений тензор CIELAB конвертується у простір RGB, обмежується у діапазоні $[0, 1]$ та нормалізується до $[-1, 1]$:

$$\mathcal{L}_{\text{LPIPS}} = \text{VGG} \left(\text{Norm}(\text{LABtoRGB}(L, \hat{a}, \hat{b})), \text{Norm}(\text{LABtoRGB}(L, a, b)) \right).$$

Завдяки використанню цієї функції втрат гібридна модель вчиться генерувати не лише математично близькі, але й візуально реалістичні та структуровані

текстури без ефекту неприродних градієнтних переходів.

2.3 Програмна реалізація та графічний інтерфейс користувача

Програмна реалізація розробленої системи виконана мовою програмування Python із використанням фреймворку глибокого навчання PyTorch. Це забезпечує ефективне паралельне виконання тензорних обчислень на графічних процесорах та мінімізує час генерації результату. Для побудови інтерактивного вебзастосунку було використано спеціалізований фреймворк Gradio.

Особливістю програмного бекенду є наявність розробленого диспетчера ресурсів ColorizationEngineAPI, який автоматично завантажує ваги нейромережі, відстежує зміни у конфігураційних файлах та здійснює примусове вивільнення відеопам'яті за допомогою збирача сміття під час перемикання між різними архітектурами. Це гарантує стабільну роботу системи навіть на комп'ютерах з обмеженими апаратними можливостями.

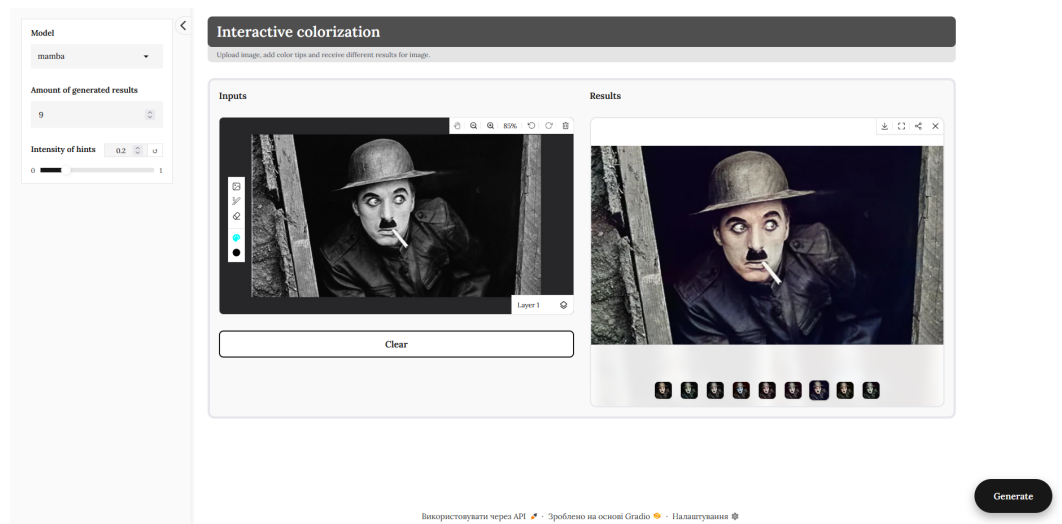


Рисунок 2.4 – Інтерфейс розробленого застосунку для інтерактивної колоризації

Інтерфейс програми (рис. 2.4) реалізовано у вигляді інтерактивного вебзастосунку. Робочий процес алгоритмізовано за такими етапами:

1) **ініціалізація та налаштування:** користувач завантажує зображення, яке програмно перетворюється у простір LAB із виділенням каналу яскравості L як вхідної умови. Через бічну панель задаються гіперпараметри генерації: архітектура моделі, кількість вихідних варіацій та коефіцієнт глобальної інтенсивності підказок;

2) **введення просторових підказок:** за допомогою графічного інтерфейсу користувач наносить кольорові штрихи поверх чорно-білого зображення. Система фіксує ці мазки, вилучає їхні кольорові значення та формує розріджений тензор підказок разом із маскою інтенсивності підказок M ;

3) **інференс та візуалізація:** застосунок виконує прямий прохід обраної моделі в оптимізованому режимі `torch.inference_mode()`. Згенеровані канали a та b конкатенуються з вхідним каналом L , конвертуються у колірний простір RGB формату `uint8` та виводяться у блоці результатів як галерея багатоваріантних рішень.

Завдяки оптимізованій архітектурі моделі процес інференсу займає лічені мілісекунди.

Висновки до розділу 2

У цьому розділі було детально описано процес інженерного проектування та практичної реалізації системи інтерактивної багатоваріантної колоризації. Розроблено оригінальну архітектуру B-Mamba, яка поєднує U-Net для контролю локальних меж та двонаправлену Mamba зі спільними вагами. Окрім того, створено зручний графічний застосунок.

3 ЕКСПЕРИМЕНТАЛЬНИЙ АНАЛІЗ РЕЗУЛЬТАТІВ ТА ОЦІНКА ЕФЕКТИВНОСТІ СИСТЕМИ

У цьому розділі наводяться результати комплексного тестування розробленої моделі B-Mamba та її порівняльний аналіз із передовими архітектурами в галузі колоризації зображень. Дослідження ефективності проведено як за допомогою об'єктивних математичних метрик PSNR, SSIM та MSE, так і за допомогою перцептивних генеративних критеріїв FID, KID та LPIPS.

Окрім аналізу якості генерованого зображення, особлива увага приділяється апаратній ефективності запропонованого методу – споживанню VRAM та кількості Time, що є критично важливими параметрами для систем інтерактивного розфарбовування реального часу.

3.1 Апаратне забезпечення та параметри навчання

Процес навчання моделі здійснювався з використанням відеокарти NVIDIA GeForce RTX 5060 Ti та процесора AMD Ryzen 5 5600. Для забезпечення стабільності обчислень усі вхідні зображення масштабувалися до розміру 256×256 пікселів, а розмір пакету (англ. batch size) становив 16 екземплярів.

Оптимізація вагових коефіцієнтів виконувалася за допомогою алгоритму AdamW із базовою швидкістю навчання 1×10^{-4} та коефіцієнтом згасання ваг (англ. weight decay) 1×10^{-4} . Для динамічного адаптивного керування кроком оптимізації застосовувався планувальник CosineAnnealingLR із мінімальною швидкістю навчання $\eta_{\min} = 1 \times 10^{-6}$ та тривалістю циклу, що відповідає загальній кількості епох навчання.

Модель навчалася протягом 43 епох на тренувальній вибірці COCO 2017 Train, а валідація проводилася на COCO 2017 Val. Для розрахунку загальної цільової функції втрат було емпірично підібрано такі вагові коефіцієнти:

$\lambda_{L1} = 1.0$, $\lambda_{\text{hint}} = 5.0$ та $\lambda_{\text{LPIPS}} = 1.0$.

Для прискорення навчання та зменшення споживання відеопам'яті застосовувалася змішана точність обчислень у форматі BFloat16 (англ. Brain Floating Point 16). На відміну від стандартного Float32, цей формат майже вдвічі скорочує обсяг пам'яті, необхідний для зберігання проміжних активацій, тоді як основні вагові коефіцієнти зберігаються у форматі Float32 для чисельної стабільності градієнтного спуску. Завдяки цьому тривалість однієї епохи скоротилася приблизно з 90 до 60 хвилин.

За підсумками процесу навчання значення функції втрат на тренувальній вибірці склало 0.1826. Під час контрольної перевірки значення функції втрат, усереднене для повністю автоматичної валідації та валідації з інтерактивними підказками, було на рівні 0.2236.

Така незначна розбіжність між тренувальними та валідаційними показниками підтверджує повну відсутність перенавчання та високу узагальнювальну здатність архітектури. Водночас динаміка похибки свідчить про те, що градієнтний спуск ще не досягнув абсолютного плато, вказуючи на наявність резерву для подальшого зниження похибки.

3.2 Методологія тестування та базові моделі

Для забезпечення об'єктивності результатів порівняння проводилося на комплексному наборі даних, в який увійшли такі датасети: COCO 2017 Test, Natural Color Dataset та Oxford-102 Flowers. Цей набір даних містить як зображення високої складності, наприклад заклад, який гармонійно поєднує в собі невелике кафе та крамницю свіжих овочів, так і зображення з усього одним об'єктом, наприклад яблуком. Усі моделі тестувалися в однакових апаратних умовах з ідентичною роздільною здатністю вхідних тензорів.

Для проведення об'єктивного порівняння застосовувався комбінований підхід до умов навчання моделей, деталі якого зведено у таблиці 3.1. Розроблена гібридна архітектура B-Mamba та модель Pix2Pix навчалися з нуля на ідентичному тренувальному наборі даних COCO 2017 Train з використанням

однакової симуляції користувацьких підказок. Самостійне навчання Pix2Pix було зумовлене відсутністю відкритих попередньо навчених ваг для задачі інтерактивної колоризації. Модель навчалася протягом 60 епох. Натомість для інших базових архітектур під час тестування використовувалися їхні оригінальні офіційні попередньо навчені ваги, завантажені з репозиторіїв авторів.

Хоча в ідеальних умовах еталонне порівняння вимагає навчання всіх моделей на ідентичних даних, такий компромісний підхід застосовано з метою уникнення заниження результатів базових методів. Використання офіційних ваг гарантує, що ці архітектури демонструють максимальну можливу якість генерації, заявлену в їхніх оригінальних публікаціях, і не страждають від потенційно неоптимальних гіперпараметрів під час стороннього процесу навчання.

Таблиця 3.1 – Умови навчання та параметри порівнюваних моделей

Модель	Джерело ваг	Тренувальний датасет	Епохи	Додаткова інформація
ECCV16 [6]	Офіційні	ImageNet	-	-
SIGGRAPH17 [29]	Офіційні	ImageNet	-	-
DDColor [22]	Офіційні	ImageNet	-	Використано найменшу версію (ddcolor_paper_tiny)
Pix2Pix [9] (Our Impl.)	Навчено нами	COCO 2017 Train	60	Навчання зупинено до досягнення плато (відсутні ознаки перенавчання)
Ours (B-Mamba)	Навчено нами	COCO 2017 Train	43	Навчання зупинено до досягнення плато (відсутні ознаки перенавчання)

Загалом для порівняння було обрано чотири базові моделі, які представляють різні етапи еволюції методів колоризації:

1) **ECCV16 [6]** – автоматична CNN, яка розв’язує задачу як мультимодальну класифікацію пікселів (тестувалася з офіційними вагами).

2) **SIGGRAPH17 [29]** – інтерактивна CNN, розроблена спеціально для підтримки користувацьких підказок (тестувалася з офіційними вагами). Вона слугує прямим конкурентом розробленій системі.

3) **Pix2Pix [9]** – інтерактивна cGAN, цілеспрямовано адаптована та

навчена нами на наборі COCO приймати просторові підказки як додаткову умову. Вона дозволяє оцінити ефективність класичного змагального підходу в задачах керованої генерації.

4) **DDColor [22]** – сучасна надважка автоматична модель на базі подвійних декодерів та просторової уваги, яка демонструє найвищу якість генерації серед автоматичного режиму обраних моделей (тестувалася з офіційними вагами).

3.3 Об’єктивний аналіз якості генерації та швидкодії

Основні результати кількісного оцінювання моделей показано в таблиці 3.2. Усі моделі отримали на вхід 3 підказки з найбільшою хроматичною магнітудою.

Таблиця 3.2 – Основні результати моделей

Модель	PSNR ↑	SSIM ↑	LPIPS ↓	VRAM ↓	Time ↓
ECCV16 [6] (Auto)	20.11	0.656	0.231	165.51 MB	6.05 ms
DDColor [22] (Auto)	19.94	0.653	0.199	1092.30 MB	35.37 ms
Pix2Pix [9] (Hinted, Our Impl.)	19.51	0.610	0.246	241.28 MB	3.11 ms
SIGGRAPH17 [29] (Hinted)	20.77	0.651	0.185	353.57 MB	15.53 ms
Ours (B-Mamba)	22.53	0.664	0.143	187.29 MB	15.08 ms

Аналіз даних таблиці 3.2 свідчить про перевагу розробленої за більшістю критеріїв системи окрім VRAM та Time. Показник PSNR для B-Mamba склав 22.53 дБ, що на +2.42 дБ перевершує найближчого автоматичного конкурента ECCV16 та на +1.76 дБ перевершує інтерактивну модель SIGGRAPH17. Високий SSIM, який дорівнює 0.664, підтверджує, що архітектура U-Net бездоганно зберігає просторову геометрію та контури об’єктів. LPIPS, яка найкраще корелює зі сприйняттям людського ока, показала найкращий результат – 0.143.

З точки зору обчислювальної ефективності розроблена гібридна модель продемонструвала видатні результати. Завдяки лінійній складності блоку Mamba та використанню спільних ваг, споживання VRAM під час інференсу склало

лише 187.29 МБ. Для порівняння, потужна модель DDColor вимагає майже в 6 разів більше пам'яті – 1092.30 МБ – та працює у 2.3 рази повільніше – 35.37 мс проти 15.08 мс у B-Mamba. Це доводить високу придатність розробленої системи для розгортання на побутових пристроях.

3.4 Оцінка перцептивної реалістичності та точності керування

Додатково було проведено оцінку за допомогою FID, KID та MSE, зведених у таблиці 3.3.

Таблиця 3.3 – Додаткові метрики якості генерації моделей

Модель	FID ↓	KID mean ↓	KID std ↓	MSE LAB ↓	MSE HINT ↓
ECCV16 [6] (Auto)	23.000	0.01770	0.00720	0.02300	0.03807
DDColor [22] (Auto)	5.031	0.00485	0.00848	0.02290	0.03773
Pix2Pix [9] (Hinted, Our Impl.)	8.437	0.00562	0.00842	0.02633	0.00039
SIGGRAPH17 [29] (Hinted)	7.937	0.00452	0.00891	0.01889	0.00486
Ours (B-Mamba)	3.078	0.00313	0.00662	0.01332	6.1×10^{-5}

Модель B-Mamba досягла досить низького значення FID – 3.078, порівняно з 5.031 у DDColor та 23.0 у ECCV16. Це, разом із найнижчим значенням KID_{mean} , яке дорівнює 0.00313, доводить, що кольорова палітра розробленої моделі є надзвичайно природною.

Критичним показником для розробленої інтерактивної системи є MSE_{HINT} – SIGGRAPH17 має 0.00486, що означає періодичне ігнорування або розмиття мережею вимог користувача. Натомість архітектура B-Mamba досягла 6.1×10^{-5} . Це беззаперечно доводить, що розроблена нейромережа ідеально слухається вводу оператора, жорстко фіксує колір у заданій точці та розумно поширює його на сусідні області без конфліктів у латентному просторі.

3.5 Візуальні результати

Окрім об'єктивного математичного оцінювання, для наочного візуального порівняння було відібрано по одному характерному прикладу з кожного тестового набору даних для всіх досліджуваних моделей (рис. 3.1). Для глибшого аналізу під результатами кожної моделі додатково наведено теплові мапи просторового розподілу похибки. На цих мапах інтенсивність кольору прямо пропорційна величині відхилення згенерованих каналів кольоровості від еталонного зображення: що яскравіша ділянка, то більшою є локальна помилка генерації.

Візуальний аналіз показує, що повністю автоматична модель DDColor продемонструвала високу якість колоризації квітки (Oxford-102 Flowers). Проте на інших тестових зображеннях її результати виявилися менш стабільними: через відсутність механізму інтерактивного керування модель покладається виключно на внутрішні апріорні знання. У випадках візуальної невизначеності це призводить до вибору «безпечної», тьмяної палітри – зокрема, яблуко набуло блілого жовто-зеленого відтінку, а поле стало однорідно піщано-коричневим, і користувач не має інструментів для коригування цього результату. Натомість запропонована гібридна модель розв'язує цю проблему: завдяки підтримці просторових підказок вона забезпечує не лише збереження природної насиченості кольорів, але й дозволяє точно керувати процесом колоризації на різних типах сцен.

3.6 Користувацьке дослідження якості колоризації

Для всебічної оцінки візуальної якості результатів роботи запропонованої гібридної моделі та її порівняння з базовим методом було проведено користувацьке дослідження. Об'єктивні метрики не завжди корелюють із тим, як людське око сприймає реалістичність кольорів, тому суб'єктивна оцінка є критично важливою для задач генерації зображень.

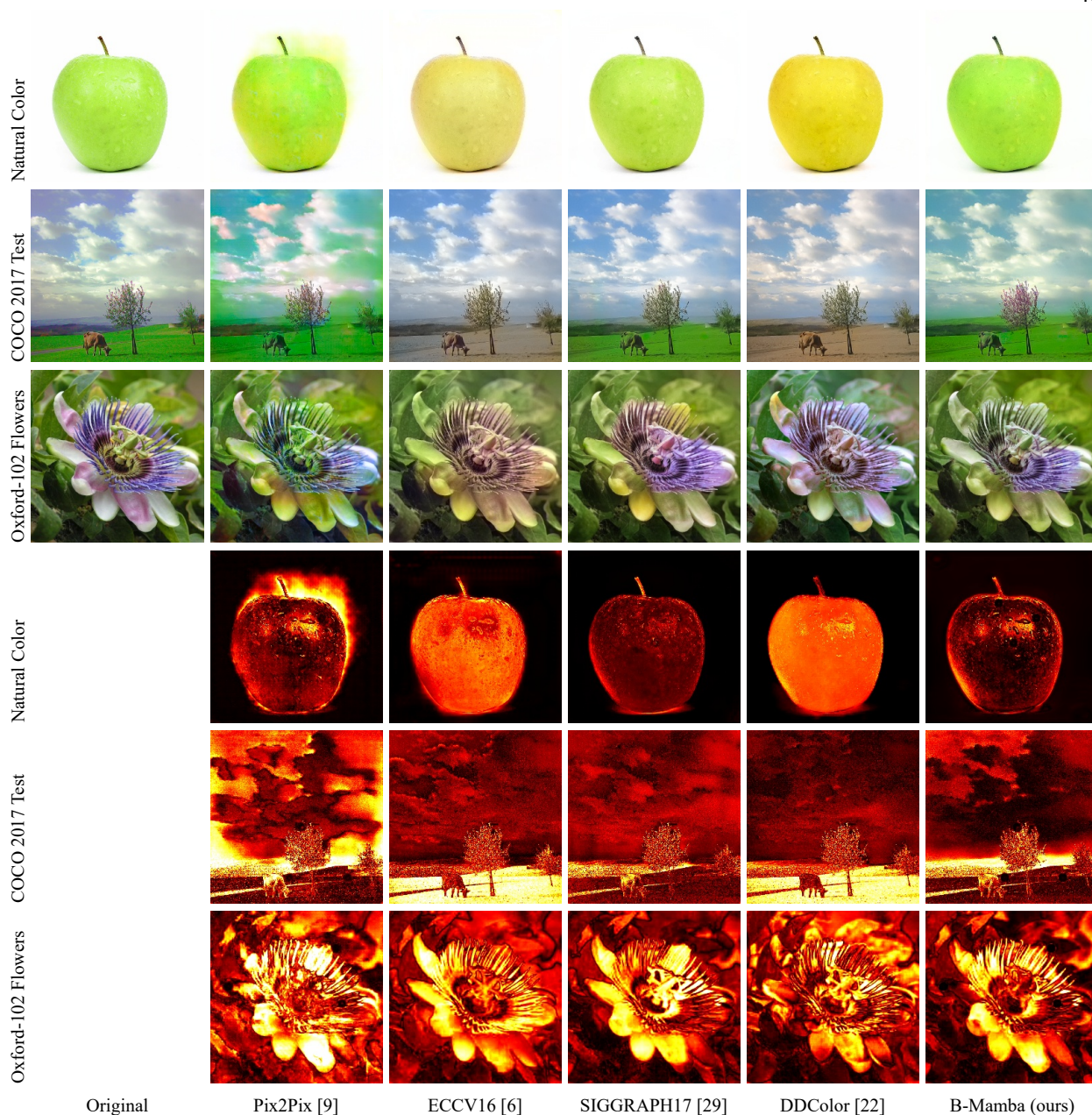


Рисунок 3.1 – Візуальне порівняння результатів колоризації на різних датасетах

3.6.1 Методологія проведення опитування

Приклад інтерфейсу етапів опитування наведено на рис. 3.2.

В опитуванні взяли участь 18 респондентів. Участь була повністю добровільною та анонімною, результати збиралися через платформу Google Forms. Для тестування було відібрано 6 категорій зображень різної складності:

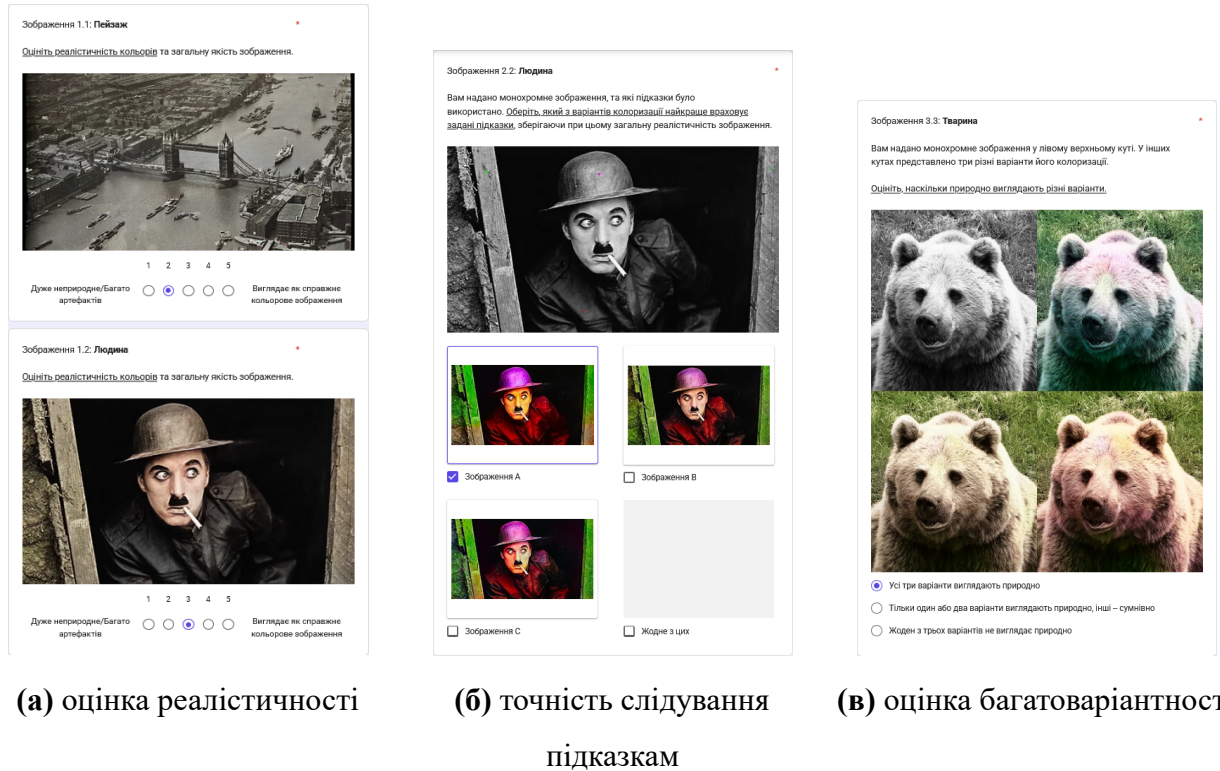


Рисунок 3.2 – Інтерфейс користувацького дослідження

пейзаж, людина, тварина, кімната/інтер'єр, простий одиничний об'єкт та вулиця.

Опитування складалося з ознайомлення з протоколом погодження та трьох послідовних етапів, кожен з яких був спрямований на оцінку конкретного аспекту роботи моделі:

1) протокол погодження (див. додаток Б): містить інформацію про мету дослідження, політику збору та обробки даних, а також умови використання результатів;

2) оцінка реалістичності: респондентам демонстрували колоризоване зображення та пропонували оцінити його природність за 5-бальною шкалою (від 1 – «Дуже неприродне / Багато артефактів» до 5 – «Виглядає як справжнє кольорове зображення»);

3) точність слідування підказкам: респондентам демонстрували монохромне зображення із нанесеними кольорними підказками та три варіанти колоризації від різних моделей, а саме SIGGRAPH17, Pix2Pix та нашої запропонованої моделі. Завданням було обрати варіант, який найкраще враховує задані підказки, зберігаючи реалістичність;

4) оцінка багатоваріантності: респондентам демонстрували 3 варіанти колоризації одного й того ж монохромного зображення, згенерованих нашою моделлю для однакового набору підказок користувача. Завданням було для кожного варіанту окремо визначити, чи виглядає він природно, та підрахувати кількість природних результатів.

3.6.2 Результати першого етапу: оцінка реалістичності

На першому етапі середня оцінка реалістичності згенерованих зображень склала 3.509 балів з 5 можливих. Усереднені за зображеннями оцінки можна побачити на рис. 3.3



Рисунок 3.3 – Розподіл відповідей респондентів щодо реалістичності автоматично згенерованих зображень

Найвищі оцінки отримали зображення в категоріях «простий одиничний об'єкт» та «людина», 4.056 та 3.944 бали відповідно, що свідчить про високу здатність моделі коректно відновлювати природні відтінки шкіри та локальні кольори ізольованих об'єктів.

Зображення з великою кількістю дрібних деталей або зняті з висоти (категорії «пейзаж» і «вулиця») отримали дещо нижчі бали, 2.333 та 3.556 відповідно. Це пояснюється складністю просторового розподілу кольору в

дрібних текстурах та тяжінням моделі до безпечних сірих або сепійних відтінків у зонах із високою візуальною невизначеністю.

3.6.3 Результати другого етапу: точність слідування підказкам

Цей етап мав на меті порівняти ефективність механізму інтерактивних підказок запропонованої моделі з існуючими альтернативними рішеннями (Pix2Pix та SIGGRAPH17). Головним критерієм оцінки була здатність неймережі локалізувати заданий колір у межах цільового об'єкта, не допускаючи його неприродного розтікання на сусідні області, та водночас зберігати загальну візуальну цілісність сцени. Оскільки результати роботи моделей можуть бути однаково вдалимими або невдалимими, респондентам дозволялося обрати кілька кращих варіантів одночасно або ж вказати, що жодна з моделей не впоралася із завданням.

Розподіл відповідей, наведений на діаграмі (рис. 3.4), демонструє, у яких саме категоріях зображень розроблена гібридна архітектура найкраще впоралася із завданням за оцінками респондентів.



Рисунок 3.4 – Розподіл відповідей респондентів щодо точності слідування підказкам (наша модель)

Найвищі показники точності для нашої моделі зафіксовано в категоріях «Людина» (16 відповідей з 18 можливих) та «Простий одиничний об'єкт» (15 відповідей). Це підтверджує ефективність інтеграції блоків V-Mamba у структуру

U-Net при роботі з об'єктами, що мають чіткі семантичні межі. Мережа коректно реєструє просторові координати підказок і поширює заданий колір виключно в межах однорідних зон, не допускаючи ефекту «розтікання» на фон.

Натомість найменшу кількість голосів гібридна модель отримала в категорії «Пейзаж» (4 відповіді). Це пояснюється високою структурною складністю таких сцен та великою кількістю дрібних деталей. У складних ландшафтах кількох розріджених точкових маркерів виявляється недостатньо для подолання семантичної неоднозначності. Саме в таких складних сценаріях респонденти найчастіше користувалися опцією «Жодне з цих» (загалом по всьому опитуванню цей варіант було обрано у 14.8% випадків). Це вказує на те, що для подібних зображень користувачеві необхідно надавати більш щільні та продумані колірні орієнтири.

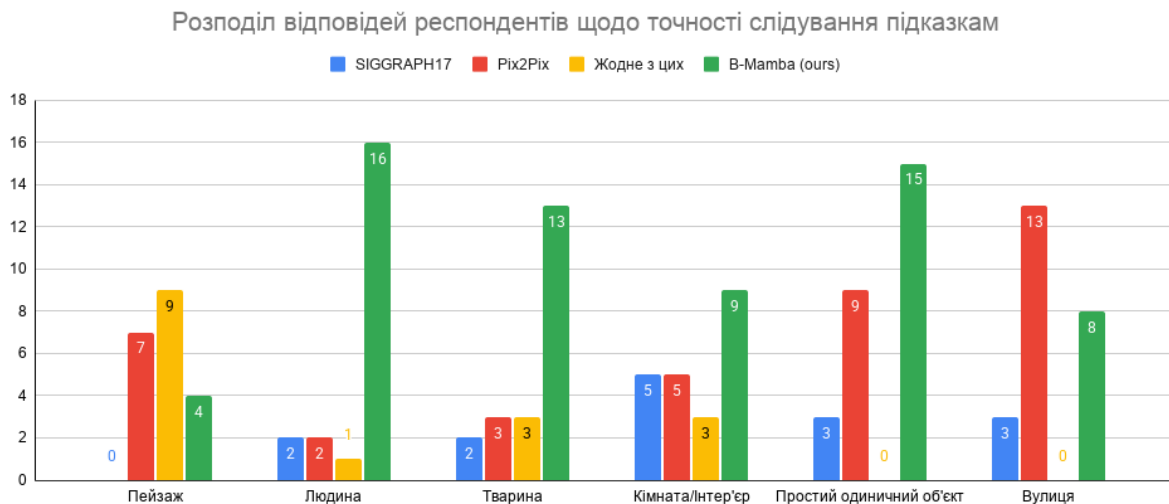
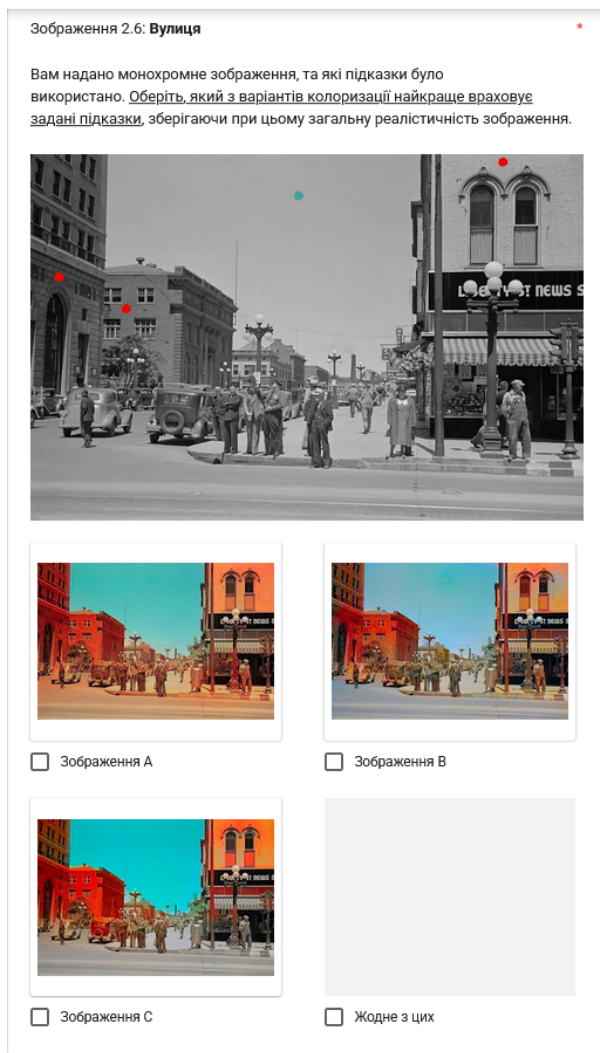
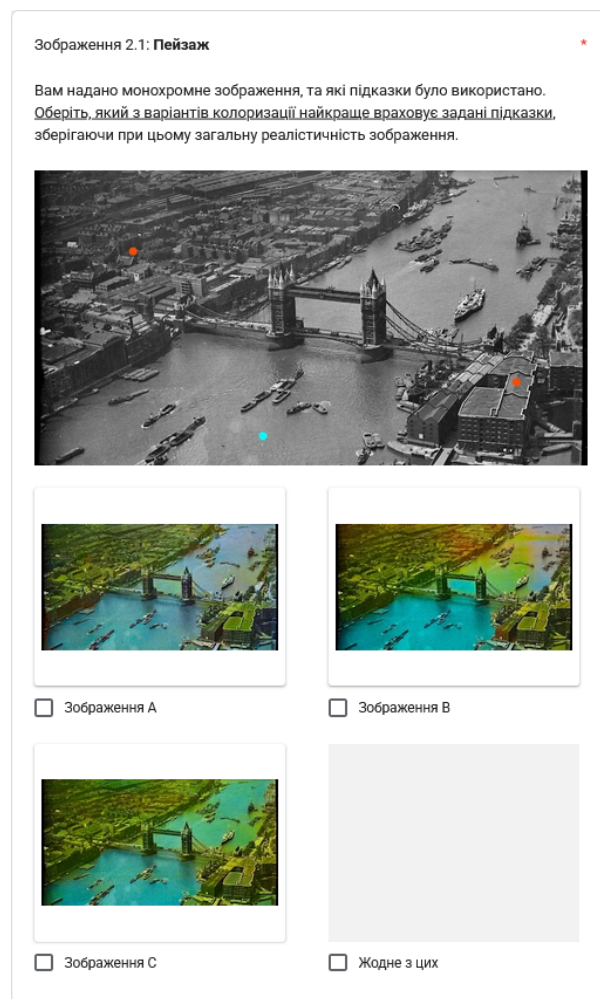


Рисунок 3.5 – Розподіл відповідей респондентів щодо точності слідування підказкам (порівняльний аналіз)

Більш детальний порівняльний аналіз усіх досліджуваних методів (рис. 3.5) виявляє специфіку роботи альтернативних архітектур. Зокрема, аналіз результатів у категорії «Вулиця», де модель Pix2Pix отримала найбільшу кількість голосів (13 відповідей), виявив специфічний артефакт роботи розробленої архітектури B-Mamba (рис. 3.6a). Під час розфарбовування міських



(а) Міський пейзаж: артефакт розтікання кольору. А – B-Mamba, В – Pix2Pix, С – SIGGRAPH17.



(б) Природний ландшафт: ефект глобального змішування. А – Pix2Pix, В – SIGGRAPH17, С – B-Mamba.

Рисунок 3.6 – Приклади візуальних артефактів під час колоризації складних сцен різними методами

пейзажів наша модель інколи демонструвала надмірне розтікання кольору від цільових об'єктів, зокрема фасадів будівель, на сусідні великі площини, такі як асфальтове покриття доріг. Водночас метод Pix2Pix виявився більш консервативним у поширенні кольору, що дозволило йому зберегти нейтральні та природні відтінки дорожнього покриття. Це обґрунтовує вищу оцінку Pix2Pix респондентами в цій категорії порівняно із запропонованим гібридним методом (13 голосів проти 8). З іншого боку, метод SIGGRAPH17 продемонстрував

стабільно низькі результати у більшості категорій через схильність до надмірної сатурації та неприродного змішування кольорів, змігши скласти конкуренцію лише у замкнених просторах категорії «Кімната/Інтер'єр» (5 відповідей).

Як і очікувалося, категорія «Пейзаж» виявилася найскладнішою для всіх без винятку алгоритмів, про що свідчить домінування варіанта «Жодне з цих» (9 відповідей). Візуальний аналіз результатів (рис. 3.6б) показує, що через відсутність чітких структурних бар'єрів та високу щільність дрібних деталей моделі не здатні ефективно ізолювати вплив просторових підказок. Це призводить до глобального змішування цільових кольорів, що утворює неприродний загальний тон сцени.

Загалом за результатами цього етапу у 60.2% усіх оцінюваних сцен респонденти обрали саме зображення, згенероване розробленою гібридною моделлю, як таке, що найкраще враховує задані підказки без втрати загальної візуальної якості.

3.6.4 Результати третього етапу: оцінка багатоваріантності

На фінальному етапі оцінювалася здатність моделі генерувати правдоподібні альтернативні варіанти, результати можна побачити на рис. 3.7.

Розподіл відповідей респондентів щодо правдоподібності альтернативних варіантів

- Жоден з трьох варіантів не виглядає природно
- Тільки один або два варіанти виглядають природно, інші – сумнівно
- Усі три варіанти виглядають природно

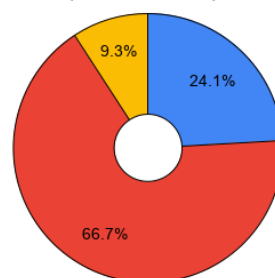


Рисунок 3.7 – Розподіл відповідей респондентів щодо правдоподібності альтернативних варіантів

У 9.3% відповідей респонденти зазначили, що «Усі три варіанти

виглядають природно». У 66.7% випадків користувачі обрали варіант «Тільки один або два варіанти виглядають природно, інші – сумнівно».

Цей результат підтверджує, що додавання генерації випадкових підказок дозволяє створювати різні версії розфарбовування, але зберігати їх максимальну візуальну правдоподібність в очах людини-спостерігача буде можливо за умови, якщо колір для підказки генеруватиметься додатковою моделлю на основі семантики об'єкта, а не просто вибиратиметься випадковим чином із нормального розподілу.

3.6.5 Кореляція між об'єктивними метриками та суб'єктивним сприйняттям

Аналіз отриманих результатів дозволяє простежити чітку кореляцію між формальними математичними метриками (табл. 3.2, 3.3) та оцінками респондентів (рис. 3.5), що підтверджує валідність обраної методології тестування.

Абсолютний збіг спостерігається в оцінці точності керування (табл. 3.4).

Таблиця 3.4 – Кореляція між відсотком голосів та похибкою MSE_{HINT}

Модель	Відсоток голосів $\uparrow$	MSE_{HINT} $\downarrow$
Жодне з цих	14.8%	-
SIGGRAPH17 [29] (Hinted)	13.9%	0.00486
Pix2Pix [9] (Hinted, Our Impl.)	36.1%	0.00039
Ours (B-Mamba)	60.2%	6.1×10^{-5}

Наявність строгої оберненої залежності між об'єктивною метрикою MSE_{HINT} та суб'єктивним вибором респондентів підтверджується коефіцієнтом рангової кореляції Спірмена ($\rho = -1.0$). Цю залежність наочно візуалізовано на рис. 3.8, де графіки обох величин мають яскраво виражений зустрічний характер.

Це доводить, що мінімізація похибки MSE_{HINT} під час навчання моделі безпосередньо веде до підвищення реалістичності слідування просторовим

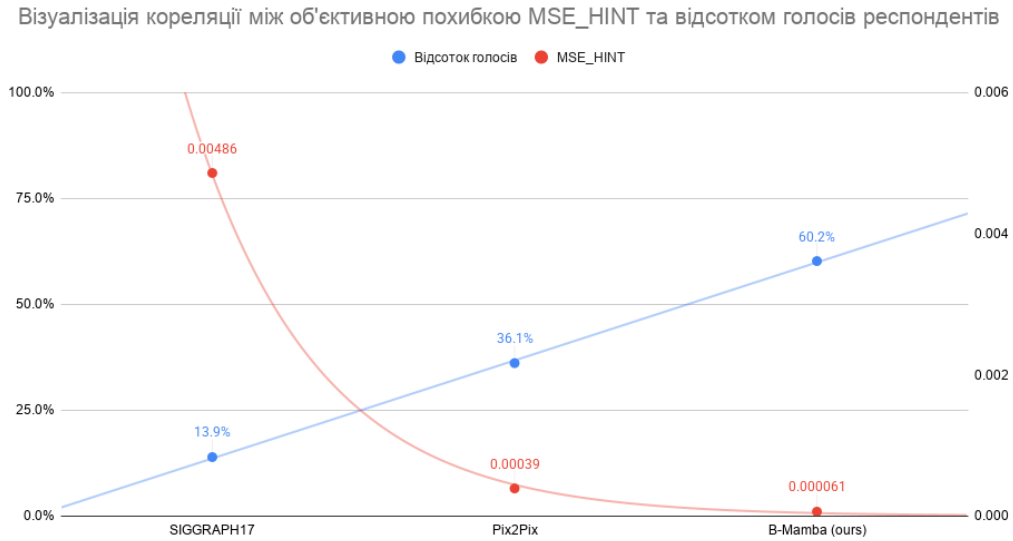


Рисунок 3.8 – Візуалізація зворотної залежності між об'єктивною похибкою MSE_{HINT} та результатами користувацького оцінювання

підказкам з точки зору кінцевого користувача.

Оскільки респонденти мали змогу обирати декілька прийнятних варіантів одночасно, опитування дозволило об'єктивно оцінити загальний рівень довіри до кожної моделі. Перемога розробленої архітектури B-Mamba (60.2% голосів на другому етапі дослідження) високо корелює з її низькою похибкою MSE_{HINT} , яка дорівнює 6.1×10^{-5} .

Модель Pix2Pix, яка посіла друге місце за метрикою MSE_{HINT} (0.00039), так само стала другою за частотою вибору в користувацькому опитуванні (36.1%). Відповідно, архітектура SIGGRAPH17, що продемонструвала третій результат за метрикою MSE_{HINT} (0.00486), очікувано посіла третє місце за результатами опитування (13.9%).

Варто також зазначити, що у 14.8% випадків респонденти не обрали жодну з моделей. Це додатково підкреслює загальну складність задачі просторово-керованої генерації, особливо для сцен із високою структурною невизначеністю, де навіть найкращі алгоритми можуть генерувати артефакти або ігнорувати вимоги користувача.

Таким чином, формальні результати повністю підтверджуються суб'єктивним зоровим сприйняттям: розроблена архітектура найефективніше

засвоює просторові підказки та генерує візуально достовірні результати.

3.6.6 Висновки за результатами користувацького дослідження

Проведене опитування підтвердило гіпотезу про те, що запропонована гібридна архітектура ефективно справляється із задачею інтерактивної багатоваріантної колоризації. Модель отримала високі суб'єктивні оцінки щодо загальної реалістичності, продемонструвала значну перевагу над базовим підходом у точності обробки локальних кольорних підказок користувача, а також довела здатність генерувати серію альтернативних кольорових рішень для одного зображення.

Висновки до розділу 3

В ході експериментального дослідження розроблена неймережева модель B-Mamba продемонструвала абсолютну перевагу над існуючими класичними аналогами ECCV16, SIGGRAPH17, Pix2Pix та перевершила сучасну важку архітектуру DDColor за перцептивними показниками якості генерації: FID 3.078, LPIPS 0.143. Водночас оптимізація через Shared Mamba дозволила скоротити використання VRAM до 187 МБ.

Рекордно низький $MSE_{HINT} = 6.1 \times 10^{-5}$ підтверджує створення бездоганно керованої інтерактивної системи, яка здатна в реальному часі генерувати багатоваріантні результати за бажанням користувача.

Окрім об'єктивних метрик, результати користувацького дослідження довели високу візуальну реалістичність згенерованих зображень (середня оцінка 3.509 з 5 балів) та значну перевагу моделі в точності обробки локальних підказок (перевага у 60.2% випадків порівняно з базовими методами).

ВИСНОВКИ

У кваліфікаційній роботі розв'язано актуальну науково-прикладну задачу розробки обчислювально ефективною системи інтерактивної колоризації зображень із можливістю багатоваріантної генерації. За результатами проведених досліджень сформовано такі висновки:

1) У ході роботи було **повністю виконано завдання щодо аналізу** науково-технічної літератури. Дослідження існуючих методів комп'ютерного зору на базі CNN, GAN, Transformer та Diffusion показало, що традиційні моделі зводять задачу до детермінованого відображення, а багатоваріантні аналоги є занадто ресурсоемними для роботи в реальному часі. Це обґрунтувало необхідність проєктування гібридної архітектури з використанням SSM.

2) Завдання з **підготовки даних та розробки алгоритму генерації підказок виконано в повному обсязі**. Сформовано комплексний набір даних для навчання та тестування на базі COCO 2017, Natural Color Dataset та Oxford-102 Flowers. Розроблено та програмно реалізовано алгоритм стохастичної генерації кольірних підказок на основі градієнтної значущості та семантичних центрів, що дозволило імітувати дії користувача та забезпечити багатоваріантність генерації без ускладнення архітектури мережі.

3) Успішно **спроєктовано та реалізовано гібридну нейромережеву архітектуру B-Mamba**, що поєднує локальну точність U-Net із глобальним рецептивним полем двонаправленої Mamba. Впровадження спільних ваг дозволило суттєво оптимізувати обчислювальні витрати. Для проведення порівняльного аналізу було успішно інтегровано базові моделі інших архітектурних класів, а саме ECCV16, SIGGRAPH17, Pix2Pix, DDCOLOR.

4) Практичне завдання з **розробки програмного забезпечення виконано успішно**. Створено інтерактивний вебзастосунок із графічним інтерфейсом, який дозволяє користувачу в режимі реального часу керувати процесом колоризації за допомогою підказок.

5) Завдання щодо **навчання та комплексного оцінювання розробленої**

моделі виконано. За результатами об'єктивного тестування модель B-Mamba продемонструвала абсолютну перевагу над аналогами. Показник якості PSNR склав 22.53 дБ, що на 2.42 дБ перевищує результат автоматичної моделі ECCV16 та на 1.76 дБ – інтерактивної SIGGRAPH17. Архітектура забезпечила найкращу перцептивну якість із показниками LPIPS на рівні 0.143 та FID 3.078 (порівняно з 5.031 у важкої моделі DDColor). Система довела бездоганну точність дотримання користувацьких умов із рекордно низькою похибкою підказок MSE_{HINT} на рівні 6.1×10^{-5} . Підтверджено високу апаратну ефективність: час інференсу склав 15.08 мс при споживанні 187.29 МБ відеопам'яті, що в 2.3 раза швидше та майже в 6 разів економніше порівняно з еталонною моделлю DDColor.

б) Проведено **користувацьке дослідження для суб'єктивної оцінки** якості генерації. Підтверджено, що запропонована гібридна архітектура ефективно компенсує недоліки суто автоматичної колоризації. Модель отримала високі оцінки загальної реалістичності (3.509 з 5 балів), перевершила базові підходи у здатності точно слідувати просторовим підказкам (перевага у 60.2% випадків) та довела здатність генерувати серію правдоподібних багатоваріантних рішень для складних сцен.

Перспективи подальших досліджень у даній предметній області включають:

- теоретичне дослідження перспективних архітектур, зокрема аналіз можливостей інтеграції дифузійних процесів із SSM (Diffusion-Mamba) для підвищення якості генерації;
- удосконалення конвеєра багатоваріантності шляхом розробки допоміжної моделі для автоматичного передбачення природного кольору об'єктів;
- розширення набору базових моделей для проведення більш вичерпного та репрезентативного бенчмаркінгу;
- адаптацію розробленої архітектури для колоризації відеопотоку із забезпеченням часової узгодженості кадрів;
- розширення функціоналу програмного комплексу через впровадження додаткових інструментів для підвищення зручності користувацької взаємодії.

ПЕРЕЛІК ПОСИЛАНЬ

- [1] Saeed Anwar et al. “Image Colorization: A Survey and Dataset”. English. In: *Information Fusion* 114 (Feb. 2025), p. 102720. ISSN: 1566-2535. DOI: 10.1016/j.inffus.2024.102720. URL: <https://doi.org/10.1016/j.inffus.2024.102720>.
- [2] Anat Levin, Dani Lischinski, and Yair Weiss. “Colorization using optimization”. English. In: *ACM Transactions on Graphics* 23.3 (2004), pp. 689–694. ISSN: 0730-0301. DOI: 10.1145/1015706.1015780. URL: <https://doi.org/10.1145/1015706.1015780>.
- [3] Erik Reinhard et al. “Color transfer between images”. English. In: *IEEE Computer graphics and applications* 21.5 (2001), pp. 34–41. DOI: 10.1109/38.946629. URL: <https://github.com/chia56028/Color-Transfer-between-Images>.
- [4] Tomihisa Welsh, Michael Ashikhmin, and Klaus Mueller. “Transferring color to greyscale images”. English. In: *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*. New York, NY, USA: Association for Computing Machinery, 2002, pp. 277–280. ISBN: 978-1-58113-521-3. DOI: 10.1145/566570.566576. URL: <https://doi.org/10.1145/566570.566576>.
- [5] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. English. Cambridge, MA, USA: MIT Press, 2016. ISBN: 978-0262035613. URL: <http://www.deeplearningbook.org>.
- [6] Richard Zhang, Phillip Isola, and Alexei A. Efros. “Colorful Image Colorization”. English. In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe et al. Cham: Springer International Publishing, 2016, pp. 649–666. ISBN: 978-3-319-46486-2. DOI: 10.1007/978-3-319-46487-9_40. URL: http://dx.doi.org/10.1007/978-3-319-46487-9_40.

- [7] Ling Yang et al. “Diffusion Models: A Comprehensive Survey of Methods and Applications”. English. In: *ACM Computing Surveys* 56.4 (Nov. 2024), pp. 1–39. ISSN: 0360-0300. DOI: 10.1145/3626235. URL: <https://doi.org/10.1145/3626235>.
- [8] Albert Gu and Tri Dao. “Mamba: Linear-Time Sequence Modeling with Selective State Spaces”. English. In: *arXiv preprint arXiv:2312.00752* (2023). arXiv: 2312.00752. URL: <https://arxiv.org/abs/2312.00752>.
- [9] Phillip Isola et al. “Image-To-Image Translation With Conditional Adversarial Networks”. English. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, July 2017, pp. 5967–5976. DOI: 10.1109/CVPR.2017.632. URL: <https://doi.org/10.48550/arXiv.1611.07004>.
- [10] Chitwan Saharia et al. “Palette: Image-to-Image Diffusion Models”. English. In: *ACM SIGGRAPH 2022 Conference Proceedings*. SIGGRAPH ’22. Vancouver, BC, Canada: Association for Computing Machinery, 2022. ISBN: 9781450393379. DOI: 10.1145/3528233.3530757. URL: <https://doi.org/10.1145/3528233.3530757>.
- [11] Yi Tay et al. “Long Range Arena: A Benchmark for Efficient Transformers”. English. In: *International Conference on Learning Representations (ICLR)*. 2021. URL: <https://arxiv.org/abs/2011.04006>.
- [12] Yi Tay et al. “Efficient Transformers: A Survey”. English. In: *ACM Computing Surveys (CSUR)* 55.6 (2022), pp. 1–28. DOI: 10.1145/3530811. URL: <https://arxiv.org/abs/2009.06732>.
- [13] Huiyu Zhai et al. “ColorMamba: Towards High-Quality NIR-to-RGB Spectral Translation with Mamba”. English. In: *arXiv preprint arXiv:2408.08087* (2024). arXiv: 2408.08087. URL: <https://arxiv.org/abs/2408.08087>.
- [14] Abhinav Atti et al. “Histogram Assisted Quality Aware Generative Model for Resolution Invariant NIR Image Colorization”. English. In: *arXiv preprint arXiv:2601.01103* (2026). arXiv: 2601.01103. URL: <https://arxiv.org/abs/2601.01103>.

- [15] Qian Jiang et al. “MFmamba: A Multi-function Network for Panchromatic Image Resolution Restoration Based on State-Space Model”. English. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 40. 7. 2026, pp. 5406–5414. arXiv: 2511.18888. URL: <https://arxiv.org/abs/2511.18888>.
- [16] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. “Learning representations for automatic colorization”. English. In: *European conference on computer vision*. Vol. 9908. Lecture Notes in Computer Science. Springer. Cham, 2016, pp. 577–593. ISBN: 978-3-319-46492-3. DOI: 10.1007/978-3-319-46493-0_35. URL: https://doi.org/10.1007/978-3-319-46493-0_35.
- [17] Federico Baldassarre, Diego González Morín, and Lucas Rodés-Guirao. “Deep koalarization: Image colorization using cnns and inception-resnet-v2”. English. In: *arXiv preprint arXiv:1712.03400* (Dec. 2017). Project: <https://github.com/baldassarreFe/deep-koalarization>. DOI: 10.48550/arXiv.1712.03400. URL: <https://doi.org/10.48550/arXiv.1712.03400>.
- [18] Patricia Vitoria, Lara Raad, and Coloma Ballester. “ChromaGAN: Adversarial Picture Colorization with Semantic Class Distribution”. English. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Mar. 2020, pp. 2434–2443. DOI: 10.1109/WACV45572.2020.9093389. URL: https://openaccess.thecvf.com/content_WACV_2020/html/Vitoria_ChromaGAN_Adversarial_Picture_Colorization_with_Semantic_Class_Distribution_WACV_2020_paper.html.
- [19] Jun-Yan Zhu et al. “Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks”. English. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2017, pp. 2242–2251. DOI: 10.1109/ICCV.2017.244. URL: <https://doi.org/10.48550/arXiv.1703.10593>.
- [20] Manoj Kumar, Dirk Weissenborn, and Nal Kalchbrenner. “Colorization transformer”. English. In: *arXiv preprint arXiv:2102.04432* (2021). DOI: 10.48550/arXiv.2102.04432. URL: <https://doi.org/10.48550/arXiv.2102.04432>.

- [21] Zhitong Huang, Nanxuan Zhao, and Jing Liao. “UniColor: A Unified Framework for Multi-Modal Colorization with Transformer”. English. In: *ACM Transactions on Graphics* 41.6 (Nov. 2022). ISSN: 0730-0301. DOI: 10.1145/3550454.3555471. URL: <https://doi.org/10.1145/3550454.3555471>.
- [22] Xiaoyang Kang et al. “DDColor: Towards Photo-Realistic Image Colorization via Dual Decoders”. English. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2023, pp. 328–338. DOI: 10.1109/ICCV51070.2023.00037. URL: https://openaccess.thecvf.com/content/ICCV2023/html/Kang_DDColor_Towards_Photo-Realistic_Image_Colorization_via_Dual_Decoders_ICCV_2023_paper.html.
- [23] Ze Liu et al. “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows”. English. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 10012–10022. arXiv: 2103.14030. URL: <https://arxiv.org/abs/2103.14030>.
- [24] Arpit Bansal et al. “Cold Diffusion: Inverting Arbitrary Image Transforms Without Noise”. English. In: *Advances in Neural Information Processing Systems*. Ed. by A. Oh et al. Vol. 36. Curran Associates, Inc., 2023, pp. 41259–41282. DOI: 10.48550/arXiv.2208.09392. URL: <https://doi.org/10.48550/arXiv.2208.09392>.
- [25] Lianghui Zhu et al. “Vision Mamba: Efficient Visual Representation Learning with Bidirectional State Space Model”. English. In: *arXiv preprint arXiv:2401.09417* (2024). arXiv: 2401.09417. URL: <https://arxiv.org/abs/2401.09417>.
- [26] Mingming He et al. “Deep exemplar-based colorization”. English. In: *ACM Transactions on Graphics* 37.4 (July 2018). ISSN: 0730-0301. DOI: 10.1145/3197517.3201365. URL: <https://doi.org/10.1145/3197517.3201365>.
- [27] Zheng Chang et al. “L-CAD: Language-based Colorization with Any-level Descriptions using Diffusion Priors”. English. In: *Advances in Neural Information Processing Systems*. Ed. by A. Oh et al. Vol. 36. Curran Associates, Inc., 2023, pp. 77174–77186. DOI: 10.48550/arXiv.2305.15217. URL: <https://pro>

ceedings.neurips.cc/paper_files/paper/2023/file/f3bfbd65743e60c685a3845bd61ce15f-Paper-Conference.pdf.

- [28] Nir Zabari et al. “Diffusing Colors: Image Colorization with Text Guided Diffusion”. English. In: *SIGGRAPH Asia 2023 Conference Papers*. New York, NY, USA: Association for Computing Machinery, 2023, 61:1–61:11. ISBN: 9798400703157. DOI: 10.1145/3610548.3618180. URL: <https://doi.org/10.1145/3610548.3618180>.
- [29] Richard Zhang et al. “Real-time user-guided image colorization with learned deep priors”. English. In: *arXiv preprint arXiv:1705.02999* (2017). DOI: 10.48550/arXiv.1705.02999. URL: <https://doi.org/10.48550/arXiv.1705.02999>.
- [30] Jooyeol Yun et al. “iColoriT: Towards Propagating Local Hints to the Right Region in Interactive Colorization by Leveraging Vision Transformer”. English. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. IEEE, Jan. 2023, pp. 1787–1796. DOI: 10.1109/WACV56688.2023.00183. URL: https://openaccess.thecvf.com/content/WACV2023/html/Yun_iColoriT_Towards_Propagating_Local_Hints_to_the_Right_Region_in_WACV_2023_paper.html.
- [31] Zhexin Liang et al. “Control Color: Multimodal Diffusion-Based Interactive Image Colorization: Z. Liang et al.” English. In: *International Journal of Computer Vision* 133.11 (2025), pp. 7897–7923. DOI: 10.1007/s11263-025-02549-6. URL: <https://doi.org/10.1007/s11263-025-02549-6>.
- [32] Alain Hore and Djemel Ziou. “Image Quality Metrics: PSNR vs. SSIM”. English. In: *2010 20th International Conference on Pattern Recognition*. IEEE, 2010, pp. 2366–2369. DOI: 10.1109/ICPR.2010.579. URL: https://www.researchgate.net/publication/220931731_Image_quality_metrics_PSNR_vs_SSIM.
- [33] Zhou Wang et al. “Image quality assessment: from error visibility to structural similarity”. English. In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612. DOI: 10.1109/TIP.2003.819861. URL: <https://github.com/zhenglab>

/IQA/blob/master/Papers/Algorithms/Image%20Quality%20Assessment%20From%20Error%20Visibility%20to%20structural%20similarity(SSIM).pdf.

- [34] Richard Zhang et al. “The Unreasonable Effectiveness of Deep Features as a Perceptual Metric”. English. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, 2018, pp. 586–595. DOI: 10.1109/CVPR.2018.00068. URL: <https://doi.org/10.48550/arXiv.1801.03924>.
- [35] Martin Heusel et al. “GANs trained by a two time-scale update rule converge to a local Nash equilibrium”. English. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. DOI: 10.48550/arXiv.1706.08500. URL: <https://doi.org/10.48550/arXiv.1706.08500>.
- [36] Mikołaj Bińkowski et al. “Demystifying MMD GANs”. English. In: *International Conference on Learning Representations (ICLR)*. 2018. DOI: 10.48550/arXiv.1801.01401. URL: <https://doi.org/10.48550/arXiv.1801.01401>.

ДОДАТОК А ФРАГМЕНТИ ВИХІДНОГО КОДУ ПРОГРАМИ

У цьому додатку наведено фрагменти вихідного коду найбільш критичних модулів розробленої системи інтерактивної колоризації. Повний вихідний код проєкту, включно зі скриптами для навчання, валідації та графічним інтерфейсом користувача, доступний у публічному репозиторії за посиланням: <https://github.com/KlymentievMaksym/interactive-colorization-engine>.

A.1 Реалізація двонаправленого блоку Shared Mamba

Нижче наведено реалізацію основного архітектурного блоку B-Mamba, що використовує механізм розділення ваг для прямого та зворотного сканування послідовності ознак.

A.1.1 Допоміжні модулі

Лістинг 1: Клас модуля Shared Mamba (PyTorch)

```

1 import torch.nn as nn
2 from mamba_ssm import Mamba
3
4 class MambaShared(nn.Module):
5     def __init__(
6         self,
7         d_model: int,
8         d_state: int = 64,
9         d_conv: int = 4,
10        expand: int = 2,
11        layers: int = 6,
12        blocks: int = 2
13    ):
14        super().__init__()
15        self.layers = layers
16
17        self.blocks = nn.ModuleList([
18            Mamba(d_model=d_model, d_state=d_state, d_conv=d_conv, expand=expand) for _ in range(blocks)
19        ])
20
21        self.norms = nn.ModuleList([nn.LayerNorm(d_model) for _ in range(layers)])
22
23    def forward(self, x):
24        expected_val_range = self.layers // len(self.blocks)
25
26        for i in range(self.layers):
27            idx_block = min(i // expected_val_range, len(self.blocks) - 1)
28            x = self.blocks[idx_block](self.norms[i](x)) + x
29
30        return x

```

Лістинг 2: Клас модуля DoubleConv (PyTorch)

```

1 import torch
2 import torch.nn as nn

```

```

3
4 class DoubleConv(nn.Module):
5     def __init__(self, in_channels: int, out_channels: int):
6         super().__init__()
7         self.conv = nn.Sequential(
8             nn.Conv2d(in_channels, out_channels, kernel_size=3, padding=1, bias=False),
9             nn.BatchNorm2d(out_channels),
10            nn.ReLU(inplace=True),
11            nn.Conv2d(out_channels, out_channels, kernel_size=3, padding=1, bias=False),
12            nn.BatchNorm2d(out_channels),
13            nn.ReLU(inplace=True)
14        )
15
16    def forward(self, x: torch.Tensor) -> torch.Tensor:
17        return self.conv(x)

```

A.1.2 Основний модуль

Лістинг 3: Клас модуля MambaWrapper (PyTorch)

```

1 import torch
2 import torch.nn as nn
3
4 from colorization_engine.models.util_models import MambaShared, BaseColorizer
5 from colorization_engine.factory.registry import register_model
6
7 @register_model("mamba")
8 class MambaWrapper(BaseColorizer):
9     def __init__(self, d_model: int = 256, layers: int = 6, blocks: int = 2):
10         super().__init__()
11
12         # --- ENCODER ---
13         # Вхід: 1 канал L + 2 канали ab підказки() + 1 канал маски = 4 канали
14         self.enc1 = DoubleConv(4, 64)
15         self.pool1 = nn.Conv2d(64, 64, kernel_size=4, stride=2, padding=1) # H/2
16
17         self.enc2 = DoubleConv(64, 128)
18         self.pool2 = nn.Conv2d(128, 128, kernel_size=4, stride=2, padding=1) # H/4
19
20         self.enc3 = DoubleConv(128, d_model)
21         self.pool3 = nn.Conv2d(d_model, d_model, kernel_size=4, stride=2, padding=1) # H/8
22
23         # --- Mamba ---
24         self.mamba = MambaShared(d_model=d_model, layers=layers, blocks=blocks)
25
26         # --- DECODER ---
27         self.up3 = nn.Upsample(scale_factor=2, mode='bilinear', align_corners=False)
28         self.dec3 = DoubleConv(d_model + d_model, 128) # Concat enc3
29
30         self.up2 = nn.Upsample(scale_factor=2, mode='bilinear', align_corners=False)
31         self.dec2 = DoubleConv(128 + 128, 64) # Concat enc2
32
33         self.up1 = nn.Upsample(scale_factor=2, mode='bilinear', align_corners=False)
34         self.dec1 = DoubleConv(64 + 64, 64) # Concat enc1
35
36         # --- FINAL LAYER ---
37         self.final_conv = nn.Sequential(
38             nn.Conv2d(64, 2, kernel_size=1),
39             nn.Tanh() # [-1, 1]
40         )
41
42     def forward(self, l_norm: torch.Tensor, hints: torch.Tensor | None = None) -> torch.Tensor:
43         """
44         l_norm: (B, 1, H, W)
45         hints: (B, 3, H, W) - канали [a, b, mask], згенеровані _receive_hints
46         """
47         batch_size, _, height, width = l_norm.shape
48         device = l_norm.device
49
50         if hints is None:
51             hints = torch.zeros((batch_size, 3, height, width), device=device)
52
53         x_in = torch.cat([l_norm, hints], dim=1)
54
55         # --- ENCODER FORWARD ---
56         feat1 = self.enc1(x_in) # (B, 64, H, W)
57         p1 = self.pool1(feat1) # (B, 64, H/2, W/2)

```

```

59     feat2 = self.enc2(p1)          # (B, 128, H/2, W/2)
60     p2 = self.pool2(feat2)        # (B, 128, H/4, W/4)
61
62     feat3 = self.enc3(p2)          # (B, 256, H/4, W/4)
63     p3 = self.pool3(feat3)        # (B, 256, H/8, W/8)
64
65     # --- MAMBA FORWARD ---
66     b, c, h, w = p3.shape
67     x_flat = p3.view(b, c, h * w).permute(0, 2, 1).contiguous() # [B, L, D]
68
69     x_fwd = self.mamba(x_flat)
70     x_flat_rev = torch.flip(x_flat, dims=[1]) # Перевертаємо послідовність справа наліво, знизу вверху
71     x_rev = self.mamba(x_flat_rev)
72     x_rev = torch.flip(x_rev, dims=[1])
73
74     x_mamba = (x_fwd + x_rev) / 2.0
75
76     x_res = x_mamba.permute(0, 2, 1).contiguous().view(b, c, h, w)
77
78     # --- DECODER FORWARD ---
79     d3 = self.up3(x_res)
80     d3 = self.dec3(torch.cat([d3, feat3], dim=1)) # Злиття Mamba + локальний контекст H/4
81
82     d2 = self.up2(d3)
83     d2 = self.dec2(torch.cat([d2, feat2], dim=1)) # Злиття H/2
84
85     d1 = self.up1(d2)
86     d1 = self.dec1(torch.cat([d1, feat1], dim=1)) # Злиття H
87
88     ab_channels = self.final_conv(d1)
89
90     return ab_channels

```

A.2 Генерація Гауссівських підказок

Наступні два лістинг демонструють механізм перетворення точкових кліків користувача на м'які розподіли кольору під час формування вхідного тензора для тренування, валідації та тестування.

Лістинг 4: Функція генерації маски інтерактивності

```

1  import torch
2
3  def get_gaussian_patch_box(
4      size: int,
5      sigma: float | None = None,
6      device: str | torch.device = 'cpu'
7  ) -> torch.Tensor:
8      """Generate 2D Gauss kernel with max 1.0"""
9      if sigma is None:
10         sigma = size / 4.0
11         coords = torch.arange(size, dtype=torch.float32, device=device) - size // 2
12         g = torch.exp(-(coords**2) / (2 * sigma**2))
13         g = g / g.max()
14         return g.view(-1, 1) * g.view(1, -1)
15
16  def get_gaussian_patch_circle(
17      size: int,
18      sigma: float | None = None,
19      device: str | torch.device = 'cpu'
20  ) -> torch.Tensor:
21      """Generate 2D Radial Gauss kernel with max 1.0 and strict circular bounds"""
22      if sigma is None:
23         sigma = size / 4.0
24
25      pad = size // 2
26      coords = torch.arange(size, dtype=torch.float32, device=device) - pad
27
28      Y, X = torch.meshgrid(coords, coords, indexing="ij")
29      dist_sq = X**2 + Y**2
30
31      patch = torch.exp(-dist_sq / (2 * sigma**2))
32      edge_val = torch.exp(torch.tensor(-(pad**2) / (2 * sigma**2), device=device))
33      patch = patch - edge_val

```

```

34
35 patch[dist_sq > pad*2] = 0.0
36 patch = patch.clamp(min=0.0)
37
38 patch = patch / patch.max()
39
40 return patch

```

Лістинг 5: Функція генерації Гауссівських підказок

```

1 import torch
2 from albumentations import Compose
3 import cv2
4 import numpy as np
5
6 from colorization_engine.utils.saliency import FastSaliencySampler
7 from colorization_engine.utils.patches import get_gaussian_patch_circle as get_gaussian_patch
8
9 saliency_sampler = FastSaliencySampler(blur_kernel_size=15, uniform_prior=0.15)
10
11 def _receive_hints(
12     ab_tensor: torch.Tensor,
13     l_tensor: torch.Tensor,
14     min_hint_size: int = 2,
15     max_hint_size: int = 16,
16     num_hints_val: int = 3,
17     patch_size_val: int = 15,
18     training: bool = True
19 ) -> torch.Tensor:
20     _, h, w = ab_tensor.shape
21     device = ab_tensor.device
22     mask = torch.zeros((1, h, w), dtype=torch.float32, device=device)
23
24     if training:
25         dist = torch.distributions.Geometric(probs=torch.tensor([0.2]))
26         num_hints = int(dist.sample().item())
27
28         points = saliency_sampler.sample_points(l_tensor, num_hints * 3) if num_hints > 0 else []
29
30         if points and num_hints > 0:
31             placed_points = []
32             min_dist_sq = (max_hint_size * 1.5) ** 2
33
34             for y, x in points:
35                 too_close = any((y - py)**2 + (x - px)**2 < min_dist_sq for py, px in placed_points)
36                 if too_close:
37                     continue
38
39             placed_points.append((y, x))
40
41             pad = torch.randint(min_hint_size, max_hint_size, (1,), device=device).item()
42             patch_size = pad * 2 + 1
43             base_patch = get_gaussian_patch(patch_size, device=device) # type: ignore
44
45             intensity = torch.rand(1, device=device).item() * 0.8 + 0.2
46             patch = base_patch * intensity
47
48             y1, y2 = max(0, y - pad), min(h, y + pad + 1)
49             x1, x2 = max(0, x - pad), min(w, x + pad + 1)
50
51             py1, py2 = max(0, pad - y), patch_size - max(0, y + pad + 1 - h)
52             px1, px2 = max(0, pad - x), patch_size - max(0, x + pad + 1 - w)
53
54             mask[0, y1:y2, x1:x2] = torch.maximum(mask[0, y1:y2, x1:x2], patch[py1:py2, px1:px2])
55
56             if len(placed_points) >= num_hints:
57                 break
58
59     else:
60         pad = patch_size_val // 2
61         inhibition_radius = patch_size_val * 2
62
63         base_patch = get_gaussian_patch(size=patch_size_val, device=device)
64
65         mag = ab_tensor.pow(2).sum(dim=0)
66
67         for _ in range(num_hints_val):
68             idx = mag.argmax().item()
69             y = idx // w
70             x = idx % w
71
72             y1, y2 = max(0, y - pad), min(h, y + pad + 1)
73             x1, x2 = max(0, x - pad), min(w, x + pad + 1)
74
75             py1, py2 = max(0, pad - y), patch_size_val - max(0, y + pad + 1 - h)
76             px1, px2 = max(0, pad - x), patch_size_val - max(0, x + pad + 1 - w)
77

```

```

78         mask[0, y1:y2, x1:x2] = torch.maximum(mask[0, y1:y2, x1:x2], base_patch[py1:py2, px1:px2])
79
80         iy1, iy2 = max(0, y - inhibition_radius), min(h, y + inhibition_radius)
81         ix1, ix2 = max(0, x - inhibition_radius), min(w, x + inhibition_radius)
82         mag[iy1:iy2, ix1:ix2] = -1.0
83
84     hint_ab = ab_tensor * mask
85     hints = torch.cat([hint_ab, mask], dim=0)
86     return hints

```

А.3 Багатоваріантність

У цьому підрозділі наведено програмний код, який відповідає за формування множини альтернативних кольорових рішень для одного вхідного зображення. Перший лістинг містить класи для просторового семплювання координат нових підказок на основі градієнтної значущості та виділення семантичних центрів об'єктів. Другий лістинг демонструє їх інтеграцію у базовий клас моделі для ітеративної генерації варіантів колоризації.

Лістинг 6: Класи просторового семплювання координат (PyTorch, OpenCV)

```

1  import cv2
2  import numpy as np
3  import torch
4  import torch.nn.functional as F
5
6  class FastSaliencySampler:
7      """
8      Клас для генерації координат підказок на основі градієнтної значущості.
9      Оптимізовано для використання у PyTorch DataLoader (CPU) та на GPU.
10     """
11     def __init__(self, blur_kernel_size: int = 15, uniform_prior: float = 0.15):
12         self.kernel_size = blur_kernel_size
13         self.uniform_prior = uniform_prior
14
15         self.sobel_x = torch.tensor([
16             [-1., 0., 1.],
17             [-2., 0., 2.],
18             [-1., 0., 1.]
19         ], dtype=torch.float32).view(1, 1, 3, 3)
20
21         self.sobel_y = torch.tensor([
22             [-1., -2., -1.],
23             [ 0., 0., 0.],
24             [ 1., 2., 1.]
25         ], dtype=torch.float32).view(1, 1, 3, 3)
26
27     def get_pdf(self, l_tensor: torch.Tensor) -> torch.Tensor:
28         """Обчислює 2D функцію щільності ймовірності (PDF) для зображення."""
29         device = l_tensor.device
30         # [C, H, W] -> [1, 1, H, W]
31         x = l_tensor.unsqueeze(0) if l_tensor.dim() == 3 else l_tensor
32
33         sx = self.sobel_x.to(device)
34         sy = self.sobel_y.to(device)
35
36         gx = F.conv2d(x, sx, padding=1)
37         gy = F.conv2d(x, sy, padding=1)
38
39         magnitude = torch.sqrt(gx**2 + gy**2 + 1e-6)
40
41         padding = self.kernel_size // 2
42         saliency = F.avg_pool2d(magnitude, kernel_size=self.kernel_size, stride=1, padding=padding)
43         saliency = saliency.squeeze(0).squeeze(0) # [H, W]
44
45         saliency_norm = saliency / (saliency.sum() + 1e-8)
46         uniform_dist = torch.ones_like(saliency_norm) / saliency_norm.numel()
47
48         pdf = (1.0 - self.uniform_prior) * saliency_norm + self.uniform_prior * uniform_dist

```

```

49     return pdf
50
51 def sample_points(self, l_tensor: torch.Tensor, num_points: int) -> list[tuple[int, int]]:
52     """
53     Повертає список координат (y, x) на основі карти значущості.
54     """
55     if num_points <= 0:
56         return []
57
58     pdf = self.get_pdf(l_tensor)
59     h, w = pdf.shape
60
61     # [H, W] -> [H*W]
62     pdf_flat = pdf.view(-1)
63
64     sampled_indices = torch.multinomial(pdf_flat, num_points, replacement=False)
65
66     # Відновлюємо 2D координати
67     points = []
68     for idx in sampled_indices:
69         y = (idx // w).item()
70         x = (idx % w).item()
71         points.append((y, x))
72
73     return points
74
75 class SemanticCenterSampler:
76     """
77     Генерує координати підказок у центрах семантичних об'єктів' (Distance Transform).
78     Уникає розміщення нових точок в об'єктах', які вже мають підказки.
79     """
80     def __init__(self, tolerance: int = 15):
81         # tolerance для floodFill. Визначає однорідність "" об'єкта' різниці (яскравості 0-255).
82         self.tolerance = tolerance
83
84     def sample_points(self, l_tensor: torch.Tensor, num_points: int, occupied_mask: torch.Tensor) -> list[tuple[int, int]]:
85         """
86         l_tensor: [1, 1, H, W]
87         occupied_mask: [1, 1, H, W]
88         """
89         l_np = l_tensor.squeeze().cpu().numpy()
90         l_uint8 = ((l_np + 1.0) / 2.0 * 255.0).clip(0, 255).astype(np.uint8)
91         h, w = l_uint8.shape
92
93         v = np.median(l_uint8)
94         lower = int(max(0, (1.0 - 0.33) * v))
95         upper = int(min(255, (1.0 + 0.33) * v))
96         edges = cv2.Canny(l_uint8, lower, upper)
97
98         inv_edges = cv2.bitwise_not(edges)
99         dist_map = cv2.distanceTransform(inv_edges, cv2.DIST_L2, 5)
100
101         mask_ff = np.zeros((h + 2, w + 2), dtype=np.uint8)
102         occ_np = occupied_mask.squeeze().cpu().numpy()
103
104         ys, xs = np.where(occ_np > 0)
105         for y, x in zip(ys, xs):
106             if mask_ff[y + 1, x + 1] == 0:
107                 cv2.floodFill(
108                     l_uint8, mask_ff, (int(x), int(y)), 0,
109                     loDiff=self.tolerance, upDiff=self.tolerance,
110                     flags=4 | (255 << 8) | cv2.FLOODFILL_MASK_ONLY
111                 )
112
113         valid_dist_map = dist_map.copy()
114         valid_dist_map[mask_ff[1:-1, 1:-1] == 255] = 0
115
116         points = []
117         for _ in range(num_points):
118             _, max_val, _, max_loc = cv2.minMaxLoc(valid_dist_map)
119
120             # Якщо максимальна відстань до краю < 3 пікселів, великих порожніх об'єктів' не лишилось
121             if max_val < 3.0:
122                 break
123
124             x, y = max_loc
125             points.append((y, x))
126
127             # Заливаємо знайдений об'єкт', щоб він більше не брав участь у виборі
128             cv2.floodFill(
129                 l_uint8, mask_ff, (int(x), int(y)), 0,
130                 loDiff=self.tolerance, upDiff=self.tolerance,
131                 flags=4 | (255 << 8) | cv2.FLOODFILL_MASK_ONLY
132             )
133
134             valid_dist_map[mask_ff[1:-1, 1:-1] == 255] = 0
135
136         return points

```

Лістинг 7: Методи стохастичної генерації варіантів колоризації

```

1 from abc import ABC, abstractmethod
2 import random
3 import torch
4 import torch.nn as nn
5
6 from colorization_engine.utils.patches import get_gaussian_patch_circle as get_gaussian_patch
7 from colorization_engine.utils.saliency import FastSaliencySampler, SemanticCenterSampler
8
9 SEMANTIC_SAMPLER = SemanticCenterSampler(tolerance=20)
10 SALIENCY_SAMPLER = FastSaliencySampler(blur_kernel_size=15, uniform_prior=0.15)
11
12 class BaseColorizer(nn.Module, ABC):
13     """
14     Абстрактний базовий клас для всіх моделей колоризації.
15     Забезпечує єдиний інтерфейс для інференсу та валідації.
16     """
17     @abstractmethod
18     def forward(self, l_channel: torch.Tensor, hints: torch.Tensor | None = None) -> torch.Tensor:
19         """
20         l_channel: Tensor форми [B, 1, H, W] [-1, 1] (L)
21         hints: Tensor форми [B, 3, H, W] (ab + mask) or None
22         Returns: Tensor форми [B, 2, H, W] [-1, 1] (ab)
23         """
24     pass
25
26     def _generate_random_color(self, device: torch.device = torch.device('cpu')) -> torch.Tensor:
27         """
28         Генерує випадкові кольори у просторі CIELAB з фокусом на природніприглушені/ тони.
29         """
30         # std_dev = 0.35
31         std_dev = 0.25 # контролює насиченість
32
33         color_ab = torch.randn((2, 1, 1), device=device) * std_dev # нормальний розподіл, де центр у 0
34
35         # warm_bias = torch.tensor([[[0.15]], [[0.20]]], device=device)
36         # color_ab = color_ab + warm_bias
37
38         color_ab = torch.clamp(color_ab, min=-1.0, max=1.0)
39
40         return color_ab
41
42     def _generate_random_gaussian_hints(self, l_channel: torch.Tensor, base_hints: torch.Tensor | None, num_hints: int = 5):
43         """
44         Генерує тензор випадкових підказок у семантичних центрах незайнятих об'єктів'.
45         """
46         B, _, H, W = l_channel.shape
47         device = l_channel.device
48
49         new_hints = torch.zeros((B, 3, H, W), device=device)
50
51         if base_hints is not None:
52             occupied = base_hints[:, 2:3, :, :] > 0.05
53         else:
54             occupied = torch.zeros((B, 1, H, W), dtype=torch.bool, device=device)
55
56         min_dist_sq = (max_hint_size * 1.5) ** 2
57
58         for b in range(B):
59             if num_hints <= 0:
60                 continue
61
62             points = SALIENCY_SAMPLER.sample_points(l_channel[b:b+1], num_hints * 3) if random.random() < 0.5 else SEMANTIC_SAMPLER.sample_points(l_channel[b:b+1], num_hints * 3)
63             # points = SALIENCY_SAMPLER.sample_points(l_channel[b:b+1], num_hints * 3) + SEMANTIC_SAMPLER.sample_points(l_channel[b:b+1], num_hints * 3)
64             # random.shuffle(points)
65
66             if not points:
67                 continue
68
69             placed_points = []
70
71             for y, x in points:
72                 if occupied[b, 0, y, x]:
73                     continue
74
75                 too_close = any((y - py)**2 + (x - px)**2 < min_dist_sq for py, px in placed_points)
76                 if too_close:
77                     continue
78
79                 placed_points.append((y, x))
80
81             pad = int(torch.randint(min_hint_size, max_hint_size, (1,)).item())
82             patch_size = pad * 2 + 1
83             base_patch = get_gaussian_patch(patch_size, device=device) # type: ignore
84
85             intensity = torch.rand(1, device=device).item() * 0.8 + 0.2
86             patch = base_patch * intensity
87
88             y1, y2 = max(0, y - pad), min(H, y + pad + 1)

```

```

89         x1, x2 = max(0, x - pad), min(W, x + pad + 1)
90
91         py1, py2 = max(0, pad - y), patch_size - max(0, y + pad + 1 - H)
92         px1, px2 = max(0, pad - x), patch_size - max(0, x + pad + 1 - W)
93
94         current_mask_patch = patch[py1:py2, px1:px2]
95         new_hints[b, 2, y1:y2, x1:x2] = torch.maximum(new_hints[b, 2, y1:y2, x1:x2], current_mask_patch)
96
97         random_color = self._generate_random_color(device=device)
98         colored_patch = random_color * current_mask_patch
99         new_hints[b, 0:2, y1:y2, x1:x2] = new_hints[b, 0:2, y1:y2, x1:x2] + colored_patch
100
101         if len(placed_points) >= num_hints:
102             break
103
104     return new_hints
105
106 @torch.no_grad()
107 def sample(self, l_channel: torch.Tensor, hints: torch.Tensor | None = None, num_samples: int = 5, color_intensity: fl
108     """
109     Генерує варіанти колоризації, додаючи випадкові кольорові плями в порожні зони.
110     """
111     self.eval()
112     variants = []
113
114     clean_pred = self(l_channel, hints)
115     variants.append(clean_pred)
116
117     for _ in range(num_samples - 1):
118         gen_hints = self._generate_random_gaussian_hints(l_channel=l_channel, base_hints=hints, min_hint_size=min_hint
119         gen_hints[:, :2, :, :] *= color_intensity
120         gen_hints[:, :2, :, :] = torch.clamp(gen_hints[:, :2, :, :], -1.0, 1.0)
121
122         if hints is not None:
123             user_color = hints[:, :2, :, :]
124             user_mask = hints[:, 2:3, :, :]
125
126             gen_color = gen_hints[:, :2, :, :]
127             gen_mask = gen_hints[:, 2:3, :, :]
128
129             combined_mask = torch.maximum(user_mask, gen_mask)
130
131             combined_color = user_color + gen_color
132             combined_color = torch.clamp(combined_color, -1.0, 1.0)
133             # is_user = user_mask > 0
134             # combined_color = torch.where(is_user, user_color, gen_color)
135
136             combined_hints = torch.cat([combined_color, combined_mask], dim=1)
137         else:
138             combined_hints = gen_hints
139
140         variant_pred = self(l_channel, combined_hints)
141         variants.append(variant_pred)
142
143     return variants

```

A.4 Комбінована функція втрат

Наступний лістинг демонструє кастомну функцію втрат, яка оптимізує модель одночасно за трьома критеріями: попиксельна точність L1, перцептивна реалістичність LPIPS та точність виконання інтерактивних підказок користувача.

Лістинг 8: Реалізація гібридної функції втрат (PyTorch)

```

1 import torch
2 import torch.nn as nn
3 import torch.nn.functional as F
4 from torchmetrics.image import LearnedPerceptualImagePatchSimilarity
5
6 from colorization_engine.loss import BaseLoss
7 from colorization_engine.factory.registry import register_loss
8 import kornia
9
10

```

```

11 @register_loss("colorization")
12 class ColorizationLoss(BaseLoss):
13     def __init__(self, lpips_weight: float, l1_weight: float, hints_weight: float):
14         super().__init__()
15         self.lpips_weight = lpips_weight
16         self.l1_weight = l1_weight
17         self.hints_weight = hints_weight
18
19         self.l1_loss = nn.L1Loss()
20
21         _lpips_module = LearnedPerceptualImagePatchSimilarity(net_type="vgg", normalize=True)
22         self.lpips_net = _lpips_module.net
23         self.lpips_net.eval()
24         self.lpips_net.requires_grad_(False)
25
26     def forward(
27         self,
28         pred: torch.Tensor,
29         target: torch.Tensor,
30         l_channel: torch.Tensor,
31         hint_mask: torch.Tensor | None = None
32     ) -> tuple[torch.Tensor, dict[str, torch.Tensor]]:
33         loss_l1 = self.l1_loss(pred, target) * self.l1_weight
34
35         loss_hints = torch.zeros_like(loss_l1)
36         if hint_mask is not None and hint_mask.any():
37             loss_hints_raw = F.l1_loss(pred * hint_mask, target * hint_mask, reduction='none')
38
39             loss_hints_per_image = loss_hints_raw.sum(dim=[1, 2, 3])
40             hint_pixels_per_image = hint_mask.sum(dim=[1, 2, 3]).clamp(min=1e-8)
41
42             loss_hints_avg = loss_hints_per_image / hint_pixels_per_image
43             loss_hints = loss_hints_avg.mean() * self.hints_weight
44
45         l_unnorm = (l_channel + 1.0) * 50.0
46         ab_pred_unnorm = pred * 110.0
47         ab_target_unnorm = target * 110.0
48
49         lab_pred = torch.cat([l_unnorm, ab_pred_unnorm], dim=1)
50         lab_target = torch.cat([l_unnorm, ab_target_unnorm], dim=1)
51
52         rgb_pred = kornia.color.lab_to_rgb(lab_pred).clamp(0.0, 1.0)
53         rgb_target = kornia.color.lab_to_rgb(lab_target).clamp(0.0, 1.0)
54
55         rgb_pred_norm = rgb_pred * 2.0 - 1.0
56         rgb_target_norm = rgb_target * 2.0 - 1.0
57
58         loss_lpips = self.lpips_net(rgb_pred_norm, rgb_target_norm).mean() * self.lpips_weight
59
60         total_loss = loss_l1 + loss_hints + loss_lpips
61
62         return total_loss, {
63             "loss_total": total_loss.detach(),
64             "loss_l1": loss_l1.detach(),
65             "loss_hints": loss_hints.detach(),
66             "loss_lpips": loss_lpips.detach(),
67         }

```

ДОДАТОК Б ПРОТОКОЛ УЗГОДЖЕННЯ

● **Хто проводить дослідження та відповідає за дані:** дослідження проводить Климентьев Максим Андрійович, студент 4-го курсу бакалаврату на кафедрі ММАД НН ФТІ КПІ ім. Ігоря Сікорського.

● **Мета дослідження:** суб'єктивна оцінка візуальної якості зображень для дипломної роботи бакалавра, присвяченої інтерактивній багатоваріантній колоризації за допомогою моделі глибокого навчання.

● **Які дані збираються:** збираються виключно Ваші відповіді в цьому опитуванні. Опитування є повністю анонімним – ми не збираємо Ваші імена, електронні адреси, IP-адреси чи іншу конфіденційну інформацію.

● **Що відбуватиметься під час дослідження:** опитування розділене на три короткі етапи. Спочатку Вам буде запропоновано оцінити реалістичність окремих зображень, потім – обрати серед кількох варіантів той, який найточніше виконав підказки, а наприкінці – оцінити природність різних версій розфарбування одного й того ж фото. Це займе до 10 хвилин.

● **Використання та поширення результатів:** узагальнені результати будуть використані в тексті дипломної роботи, під час її передзахисту та захисту перед екзаменаційною комісією та в обговореннях із науковим керівником. Жодні індивідуальні відповіді не будуть публікуватися окремо.

● **Добровільність та право на відмову:** Ваша участь є абсолютно добровільною. Ви можете припинити участь і закрити цю форму в будь-який момент. Оскільки дані анонімні, після натискання кнопки "Надіслати" відкликати конкретно Ваші відповіді буде технічно неможливо.

● **Термін зберігання даних:** анонімні результати опитування зберігатимуться до завершення захисту дипломної роботи, після чого будуть назавжди видалені з бази даних дослідника.

● **Спостереження та запис сесії:** дослідження проходить у зручний для Вас час онлайн. За процесом проходження ніхто не спостерігає, відео- або аудіозапис Вашого екрана чи пристрою не ведеться.

● **Ваші права та контакти:** якщо у Вас виникнуть запитання щодо

дослідження або Ви захочете подати скаргу, Ви можете зв'язатися з дослідником за адресою: maksklym-ipt27@lil.kpi.ua.

- **Інші організації, що обробляють дані:** збір даних здійснюється виключно через платформу Google Forms. Жодні інші сторонні агенції чи сервіси не мають доступу до результатів.