ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Adaptive Image Enhancement Model for the Robot Vision System / K. Smelyakov, A. Chupryna. // Proceedings of the 14th International Scientific and Practical Conference. – 2023. – №3. – С. 246–251.

2. PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing [Електронний ресурс] / C.Barnes, S. Eli, A. Finkelstein, D. Goldman // ACM Transactions on Graphics. – 2009. – Режим доступу до ресурсу: https://gfx.cs.princeton.edu/pubs/Barnes_2009_PAR/index.php.

3. Image Completion using Planar Structure Guidance [Електронний ресурс] / J.Huang, S. Bing Kang, N. Ahuja, J. Kopf // ACM Transactions on Graphics. – 2014 – Режим доступу до ресурсу: https://dl.acm.org/doi/10.1145/2601097.2601205.

4. Generative Image Inpainting with Contextual Attention [Електронний ресурс] / [J. Yu, Z. Lin, J. Yang та ін.]. – 2018. – Режим доступу до ресурсу: https://arxiv.org/abs/1801.07892.

5. Yu J. Free-Form Image Inpainting with Gated Convolution [Електронний ресурс] / J. Yu, Z. Lin, J. Yang. – 2019. – Режим доступу до ресурсу: https://arxiv.org/abs/1806.03589.

6. Newson A. Video Inpainting of Complex Scenes [Електронний ресурс] / A. Newson, A. Almansa, M. Fradet. – 2015. – Режим доступу до ресурсу: https://arxiv.org/abs/1503.05528.

7. Foster D. Generative Deep Learning: Teaching Machines to Paint, Write, Compose, and Play / David Foster., 2023. – (Oreilly & Associates Inc). – (2).

8. Rashid T. Make Your First GAN With PyTorch / Tariq Rashid., 2020. – 207 с.

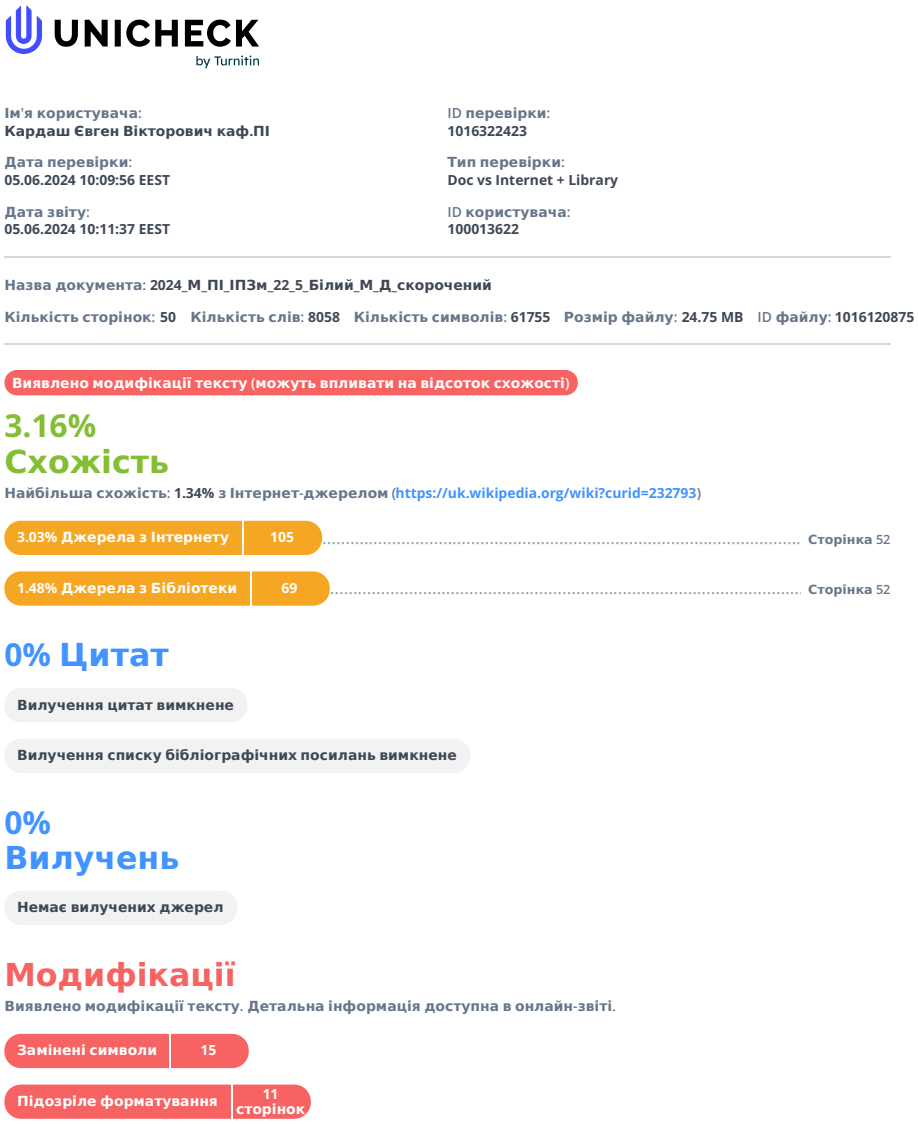
ДОДАТОК А

Перелік джерел посилання за науковими напрямами керівника та науковців кафедри програмної інженерії

1. Adaptive Image Enhancement Model for the Robot Vision System / K. Smelyakov, A. Chupryna. // Proceedings of the 14th International Scientific and Practical Conference. – 2023. – №3. – С. 246–251.

ДОДАТОК Б

Звіт результатів перевірки на унікальність тексту в базі



UNICHECK
by Turnitin

Ім'я користувача:
Кардаш Євген Вікторович каф.ПІ

ID перевірки:
1016322423

Дата перевірки:
05.06.2024 10:09:56 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
05.06.2024 10:11:37 EEST

ID користувача:
100013622

Назва документа: 2024_М_ПІ_ІПЗм_22_5_Білий_М_Д_скорочений

Кількість сторінок: 50 Кількість слів: 8058 Кількість символів: 61755 Розмір файлу: 24.75 MB ID файлу: 1016120875

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

3.16%
Схожість

Найбільша схожість: 1.34% з Інтернет-джерелом (https://uk.wikipedia.org/wiki?curid=232793)

3.03% Джерела з Інтернету 105 ..... Сторінка 52

1.48% Джерела з Бібліотеки 69 ..... Сторінка 52

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

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

Замінені символи 15

Підозріле форматування 11 сторінок

ДОДАТОК В

Слайди презентації

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

NURE ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Дослідження методів обробки зображень. Методи реалістичного заміщення об’єктів

Білий М. Д.
Науковий керівник: доц. Каук В.І.

SE software engineering

21 червня 2024

Дослідження

- **Актуальність**

Сучасні технології обробки зображень є надзвичайно важливими для багатьох сфер, включаючи розваги та безпеку. Штучний інтелект і машинне навчання значно підвищили якість обробки зображень, забезпечуючи реалістичне заміщення об'єктів та покращення візуального контенту.

- **Напрям дослідження**

Дослідження присвячене аналізу та оцінці сучасних методів обробки зображень для реалістичного заміщення об'єктів. Розглядаються технології PatchMatch, планарна структура, Contextual Attention, Gated Convolution, DeepFillV2 та Stable Diffusion.

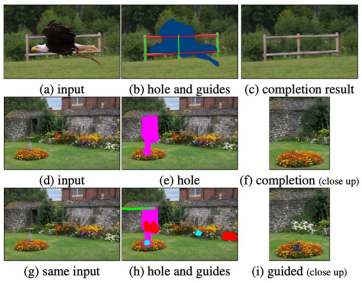
- **Об’єкт дослідження**

Методи заміщення об'єктів на зображеннях за допомогою алгоритмів штучного інтелекту, зокрема **DeepFillV2** (Celeba-HQ та Places2), **Stable Diffusion**.

SE software engineering

Огляд літератури (аналогів)

PatchMatch - це алгоритм для швидкого знаходження відповідностей між ділянками зображень.

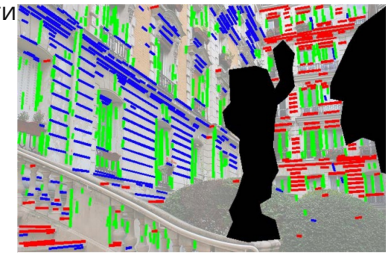


Основні кроки алгоритму:

- Ініціалізація випадковими зміщеннями.
- Ітеративне оновлення відповідностей.
- Випадковий пошук для покращення точності.

Огляд літератури (аналогів)

Метод завершення зображення за допомогою навігації планарної структури дозволяє виявляти та відновлювати плоскі поверхні на зображеннях. Використовується коли необхідно враховувати геометричні особливості сцени.



Основні кроки методу:

- Виявлення країв та ліній на зображенні.
- Оцінка точок сходу (vanishing points) для виявлення орієнтації площин.
- Локалізація площинних областей та відновлення зображення з урахуванням перспективи.

Огляд літератури (аналогів)

Метод контекстуальної уваги, який використовується в генеративних змагальних мережах (GANs), дозволяє відновлювати зображення, враховуючи інформацію з оточуючих областей. Це забезпечує реалістичне заповнення відсутніх частин зображень.

Основні кроки:

- Витягнення патчів із фонових областей зображення.
- Оцінка подібності між фоновими та передніми патчами.
- Використання зваженої оцінки для відновлення відсутніх частин зображення на основі найбільш релевантних патчів.

Метод	loss1	loss2	PSNR	TV Loss
PatchMatch	16.1%	3.9%	16.62	25.0%
Contextual Attention	8.6%	2.1%	18.91	25.3%

Огляд літератури (аналогів)

Gated Convolution дозволяє моделі автоматично вивчати оптимальну маску для кожного пікселя, забезпечуючи точне та реалістичне заповнення відсутніх частин зображення.

	Прямокутна маска		Вільної форми маск	
	l1	l2	l1	l2
PatchMatch	16.1%	3.9%	11.3%	2.4%
Global&Loca	9.3%	2.2%	21.6%	7.1%
ContextAttention	8.6%	2.1%	17.2%	4.7%
PartialConv	9.8%	2.3%	10.4%	1.9%
GatedConvolution	8.6%	2.0%	9.1%	1.6%

Огляд літератури (аналогів)

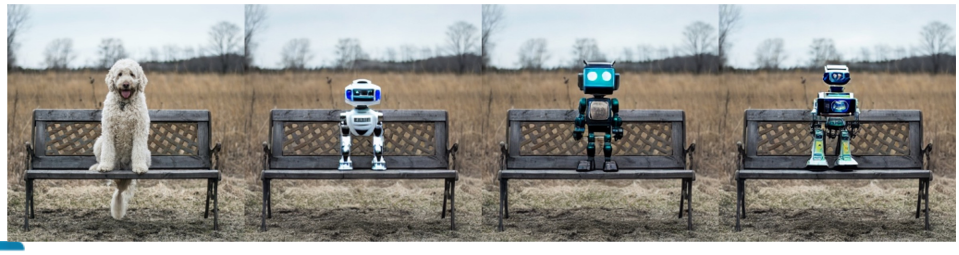
DeepFillV2 - це вдосконалений метод інпейнтингу, який використовує генеративні змагальні мережі (GANs) та:

- Contextual Attention
- Gated Convolution

Огляд літератури (аналогів)

Stable Diffusion - це передова модель генерації зображень, яка використовується для:

- Генерації зображень з тексту
- Відновлення зображень



Постановка задачі

- **Проблема**

Існуючі методи не завжди забезпечують необхідну точність та реалістичність у різних ситуаціях.

- **Очікувані результати**

Розробка системи, яка забезпечує високу якість та реалістичність заміщення об'єктів на зображеннях.

Система дозволить вибирати найбільш релевантний метод обробки для конкретних завдань, підвищуючи точність, зменшуючи артефакти та забезпечуючи гнучкість у використанні різних методів.

Методологія

Методи дослідження

- Аналіз літератури та існуючих методів: PatchMatch, Contextual Attention, Gated Convolution, DeepFillV2 та Stable Diffusion.
- Проведення експериментів для порівняння ефективності різних методів у завданнях заміщення об'єктів на зображеннях.
- Вимірювання точності та якості обробки зображень за допомогою кількісних метрик, таких як PSNR, SSIM та MSE.
- Інтеграція різних методів у єдину програмну систему для забезпечення гнучкості та ефективності обробки.

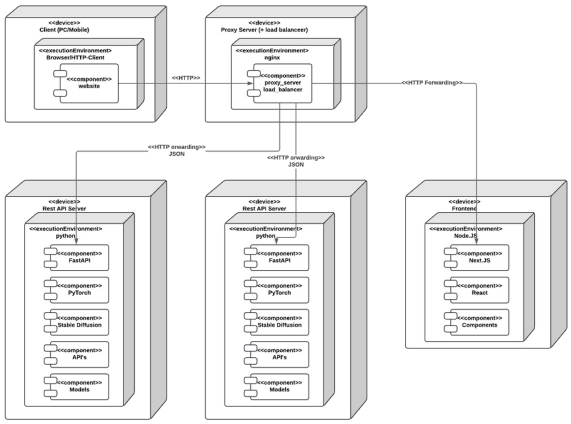
Інструментарій та технології

- Python, PyTorch, FastAPI, DeepFillV2, StableDifusion
- Next.JS, React

Архітектура система для проведення експериментального дослідження

Ключові компоненти

- Rest API Server
  - DeepFillV2 Celeba
  - DeepFillV2 Places2
  - Stable Diffusion
- Fronted
  - Next.JS
  - React



Опис програмного забезпечення, що було використано у дослідженні

●Процес розробки

Процес розробки включав теоретичний аналіз, проектування архітектури, інтеграцію методів заміщення об’єктів, тестування та оптимізація.

●Вибрані мови програмування та фреймворки

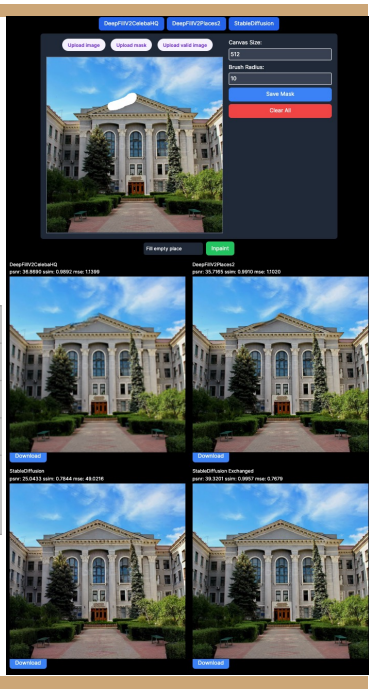
- Python, JavaScript
- FastAPI, PyTorch, Next.JS, React

Зміст проведеного експерименту

- **Методи**
  - DeepFillV2 Celeba-HQ
  - DeepFillV2 Places2
  - Stable Diffusion
  - Stable Diffusion із заміною області
  - Stable Diffusion заміна об’єкта
- **Вхідні дані**
  - Набір тестових зображень
  - Маски
  - Текстові описи для Stable Diffusion
- **Критерії**
  - PSNR (пікове відношення сигналу до шуму)
  - SSIM (індекс структурної подібності)
  - MSE
- **Послідовність**
  - Проектування
  - Впровадження
  - Тестування
  - Оптимізація
  - Підготовка даних
  - Запуск алгоритмів
  - Збирання результатів
  - Аналіз
- **Вимірювання**
  - Візуальна якість
  - PSNR
  - SSIM
  - MSE

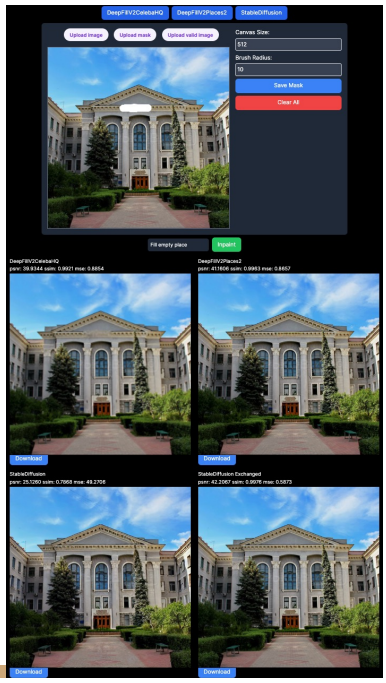
Результати експерименту

Метод	PSNR	SSIM	MSE
DeepFillV2CelebaHQ	36.869	0.9892	1.1399
DeepFillV2Places2	35.7165	0.9919	1.1020
StableDiffusion	25.0433	0.7844	49.0216
StableDiffusion Exchanged	39.3201	0.9957	0.7679



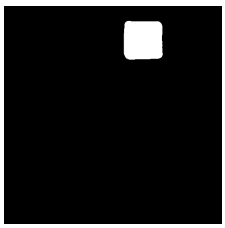
Результати експерименту

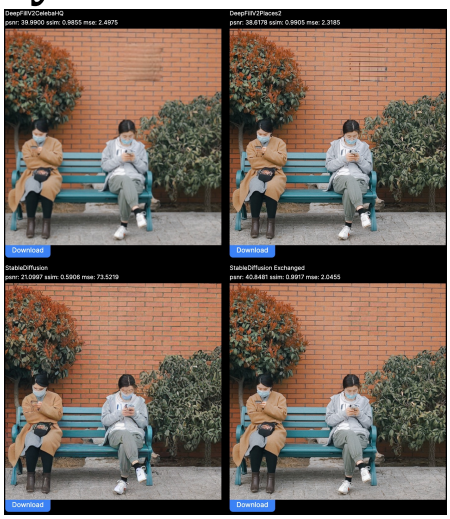
Метод	PSNR	SSIM	MSE
DeepFillV2CelebaHQ	39.9344	0.9921	0.8854
DeepFillV2Places2	41.1606	0.9963	0.8657
StableDiffusion	25.1260	0.7868	49.2706
StableDiffusion Exchanged	42.2067	0.9976	0.5873



Результати експерименту

	PSNR	SSIM	MSE
DeepFillV2 CelebaHQ	39.9900	0.9855	2.4975
DeepFillV2 Places2	38.6178	0.9905	2.3185
Stable Diffusion	21.0997	0.5906	73.5219
Stable Diffusion Exchanged	40.8481	0.9917	2.0455





Результати експерименту

- Експеримент показав, що метод Stable Diffusion ефективно справляється із завданням заміщення об'єктів, забезпечуючи високу якість та реалістичність зображень.
- Однак, додаткова пост-обробка не завжди є корисною і в даному випадку призвела до погіршення кінцевого результату.

Оригінальне | Маска

Stable Diffusion | Stable Diffusion with post-processing

Аналіз отриманих результатів

- **DeepFill V2 Celeba-HQ**

Гарні результати тільки у найпростіших завданнях, що логічно, оскільки він навчений на зображеннях розміром 256x256 пікселів.

- **DeepFill V2 Places2**

Показав себе дещо краще та забезпечив більш реалістичні результати порівняно з DeepFill V2 Celeba-HQ, завдяки навчанню на більш різноманітних та складних зображеннях.

- **Stable Diffusion з пост-обробкою**

Дав найкращі результати, забезпечивши високу якість та реалістичність заміщення об'єктів. Пост-обробка з використанням зміни області значно покращила вигляд відновлених зображень, роблячи їх максимально природними.

- **Stable Diffusion**

Показав хороші результати при заміні об'єктів на інші за текстовим описом, забезпечуючи високу якість і реалістичність відновлених зображень.

SE software engineering

Публікація результатів

УДК: 004.89 DOI: https://doi.org/10.30837/ITSSI.2024.472

ДОСЛІДЖЕННЯ МЕТОДІВ ОБРОБКИ ЗОБРАЖЕНЬ. МЕТОДИ РЕАЛІСТИЧНОГО ЗАМІЩЕННЯ ОБ'ЄКТІВ

Білий М. Д.
Науковий керівник – к.т.н, доц. Каук В. І.
Харківський національний університет радіоелектроніки, каф. ПІ
м. Харків, Україна
e-mail: mykhailo.bilyi@nure.ua

This study explores advanced methods for realistic object substitution in digital images through machine learning techniques. Focusing on convolutional neural networks and generative models, it evaluates various algorithms including PatchMatch, GANs, DeepFill V2, and Stable Diffusion Inpainting, among others, for their effectiveness in seamless object integration. The analysis demonstrates the high efficiency of these methods in processing complex contextual relationships, ensuring that the integration of objects into original images appears natural and convincing.

Це дослідження розглядає передові методи реалістичної заміни об'єктів у цифрових зображеннях за допомогою технік машинного навчання. Зосереджуючись на згорткових нейронних мережах та генеративних моделях, воно оцінює різні алгоритми, включаючи PatchMatch, GANs, DeepFill V2 та Stable Diffusion Inpainting, серед інших, за їх ефективність у безшовній інтеграції об'єктів.

Метод PatchMatch та генеративні змагальні мережі (GANs) взаємодіють у сфері обробки зображень, доповнюючи один одного. PatchMatch ефективно вирішує задачі редагування зображень шляхом швидкого знаходження відповідностей між фрагментами, тоді як GANs забезпечують створення високореалістичних зображень, що важко відрізнити від справжніх. Вони навчаються відтворювати реалістичні текстури та деталі, заміщаючи пропущені сегменти так, що відновлене зображення здається цілісним і неперервним. Це досягається шляхом "навчання" однієї частини мережі (генератора) виробляти зображення, в той час як інша частина (дискримінатор) оцінює їх якість, покращуючи з часом результат (див. рис. 1).

Однією з проблемою є те що згорткові нейронні мережі обробляють особливості зображення з локальним згортковим ядром шар за шаром, отже, не ефективні для запозичення ознак із віддалених просторових місць. Щоб подолати обмеження, ми розглядаємо механізм уваги та вводимо новий контекстний рівень уваги в глибокій генеративній мережі.

Рівень контекстної уваги дізнається, де запозичити або скопіювати інформацію про функції з відомих фонових латок, щоб створити відсутні латки. Він диференційований, тому його можна навчити в глибоких моделях, і повністю згортковий, що дозволяє тестувати на довільних роздільностях.

Підсумки

- Висока реалістичність:
  - Методи обробки зображень, такі як Stable Diffusion з пост-обробкою, показали найкращі результати, забезпечивши високу якість та реалістичність заміщення об'єктів.
  - Інші методи, включаючи DeepFill V2 (Celeba-HQ та Places2), також продемонстрували добрі результати, але з обмеженнями залежно від складності завдань.
- Корисність:
  - Отримані результати можуть бути корисними в різних галузях, включаючи медицину, розваги, безпеку та дизайн.
  - Можливість точного та реалістичного заміщення об'єктів на зображеннях відкриває нові перспективи для використання в комерційних та наукових цілях.

ДОДАТОК Г

Апробація результатів роботи

УДК: 004.89 DOI: https://doi.org/10.30837/IYF.IIS.2024.472

ДОСЛІДЖЕННЯ МЕТОДІВ ОБРОБКИ ЗОБРАЖЕНЬ. МЕТОДИ РЕАЛІСТИЧНОГО ЗАМІЩЕННЯ ОБ'ЄКТІВ

Білий М. Д.
Науковий керівник – к.т.н, доц. Каук В. І.
Харківський національний університет радіоелектроніки, каф. ПІ
м. Харків, Україна
e-mail: mykhailo.bilyi@nure.ua

This study explores advanced methods for realistic object substitution in digital images through machine learning techniques. Focusing on convolutional neural networks and generative models, it evaluates various algorithms including PatchMatch, GANs, DeepFill V2, and Stable Diffusion Inpainting, among others, for their effectiveness in seamless object integration. The analysis demonstrates the high efficiency of these methods in processing complex contextual relationships, ensuring that the integration of objects into original images appears natural and convincing.

Це дослідження розглядає передові методи реалістичної заміни об'єктів у цифрових зображеннях за допомогою технік машинного навчання. Зосереджуючись на згорткових нейронних мережах та генеративних моделях, воно оцінює різні алгоритми, включаючи PatchMatch, GANs, DeepFill V2 та Stable Diffusion Inpainting, серед інших, за їх ефективність у безшовній інтеграції об'єктів.

Метод PatchMatch та генеративні змагальні мережі (GANs) взаємодіють у сфері обробки зображень, доповнюючи один одного. PatchMatch ефективно вирішує задачі редагування зображень шляхом швидкого знаходження відповідностей між фрагментами, тоді як GANs забезпечують створення високореалістичних зображень, що важко відрізнити від справжніх. Вони навчаються відтворювати реалістичні текстури та деталі, заміщаючи пропущені сегменти так, що відновлене зображення здається цілісним і неперервним. Це досягається шляхом "навчання" однієї частини мережі (генератора) виробляти зображення, в той час як інша частина (дискримінатор) оцінює їх якість, покращуючи з часом результат (див. рис. 1).

Однією з проблемою є те що згорткові нейронні мережі обробляють особливості зображення з локальним згортковим ядром шар за шаром, отже, не ефективні для запозичення ознак із віддалених просторових місць. Щоб подолати обмеження, ми розглядаємо механізм уваги та вводимо новий контекстний рівень уваги в глибокій генеративній мережі.

Рівень контекстної уваги дізнається, де запозичити або скопіювати інформацію про функції з відомих фонових латок, щоб створити відсутні латки. Він диференційований, тому його можна навчити в глибоких моделях, і повністю згортковий, що дозволяє тестувати на довільних роздільностях.

Зіставлення та спостереження. Ми розглядаємо проблему, у якій ми хочемо зіставити характеристики відсутніх пікселів (передній план) з оточенням (фон). Як показано на рисунку 2, ми спочатку витягуємо патчі (3 × 3) у фоновому режимі та змінюємо їх як згорткові фільтри. Щоб зіставити патчі переднього плану $\{f_{x,y}\}$ із фоновими $\{b_{\acute{x},\acute{y}}\}$, ми вимірюємо нормалізований внутрішній добуток (косинус подібності).

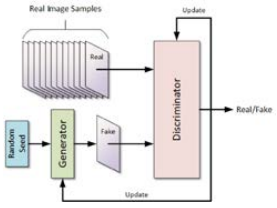


Рисунок 1 – GAN

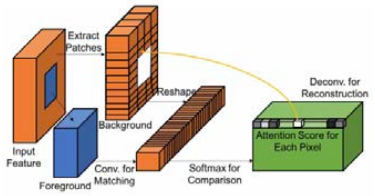


Рисунок 2 – Contextual Attention

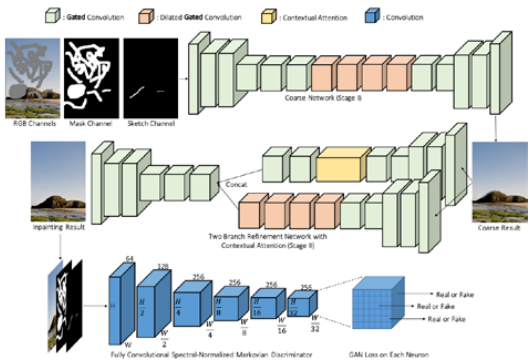


Рисунок 3 – Gated Convolution

Рисунок 4 – DeepFill V2

Gated Convolution вносить інновації в обробку зображень, вирішуючи проблему жорсткості звичайних згорткових мереж, які однаково обробляють всі частини зображення. Він додає "ворота" до згорткових операцій, дозволяючи мережі адаптивно змінювати вагу кожного пікселя на основі його контексту.

Таблиця 2 – Порівняння Gated Convolution з іншими методами.

Метод	Прямокутна маска		Вільної форми маск	
	ℓ_1 помилка	ℓ_2 помилка	ℓ_1 помилка	ℓ_2 помилка
PatchMatch	16.1%	3.9%	11.3%	2.4%
Global&Loca	9.3%	2.2%	21.6%	7.1%
ContextAttention	8.6%	2.1%	17.2%	4.7%
PartialConv	9.8%	2.3%	10.4%	1.9%
GatedConvolution	8.6%	2.0%	9.1%	1.6%

Gated Convolution ефективно вирішує проблеми з непередбачуваними або змінними областями зображення, значно підвищуючи точність та адаптивність моделі. Архітектуру можна побачити на рисунку 3.

DeepFill v2 розширює можливості інтегруючи Contextual Attention разом із Gated Convolution. Це поєднання дозволяє моделі більш точно відновлювати зображення, звертаючи увагу на контекстуальні відносини всередині пропущених або видалених ділянок (рис. 4).

Передовою моделлю генеративного машинного навчання є Stable Diffusion, розробленою для створення деталізованих зображень з текстових описів. Ця технологія використовує глибоке навчання для аналізу великих наборів даних із зображеннями та відповідними описами, навчаючись генерувати нові зображення, що відповідають заданим текстовим підказкам. Stable Diffusion здатна створювати високоякісні візуальні зображення з широкого спектру категорій, від реалістичних сцен до фантастичних ілюстрацій, відкриваючи нові можливості для творчості та дизайну.

В результаті дослідження, було встановлено, що кожен метод обробки зображень має свої переваги в певних умовах. Створення програмної системи, яке дозволяє завантажити зображення, нанести маску та вибрати метод обробки, стане важливим інструментом для фахівців. Це (рис. 5) дозволить не тільки вибрати найбільш підходящий метод для конкретної задачі inpainting, але й забезпечить можливість порівняння результатів, сприяючи вибору найкращого рішення для реалістичного заміщення об'єктів.

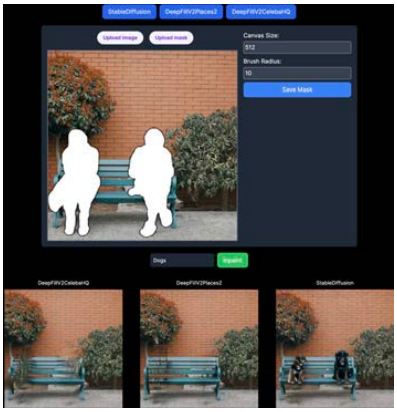


Рисунок 5 – Програмна система

Список використаних джерел:

1. Adaptive Image Enhancement Model for the Robot Vision System Vide. Tehnologija. Resursi – Environment, Technology, Resources, 2023, 3, с. 246–251.

2. Generative Image Inpainting with Contextual Attention [Електронний ресурс] / [J. Yu, Z. Lin, J. Yang та ін.]. – 2018. – Режим доступу до ресурсу: https://arxiv.org/abs/1801.07892.

3. Yu J. Free-Form Image Inpainting with Gated Convolution [Електронний ресурс] / J. Yu, Z. Lin, J. Yang. – 2019. – Режим доступу до ресурсу: https://arxiv.org/abs/1806.03589.

ДОДАТОК Д

Експертний висновок результатів перевірки кваліфікаційної роботи на відповідність оформлення вимогам ДСТУ 3008: 2015

1

Експертний висновок результатів перевірки кваліфікаційної роботи

студент (посада) | програмної інженерії (кафедра) | ІПЗм-22-5 (група)

Білий Михайло Дмитрович
(прізвище, ім'я, по батькові)

Зауваження

Пункт ДСТУ 3008-2015	Зміст пункту	Сторінка кваліфікаційної роботи
1	2	3
	7.1 Загальні положення	
	7.3 Нумерація сторінок звіту	
	7.4 Нумерація розділів, підрозділів, пунктів, підпунктів	
	7.5 Рисунки	
	7.6 Таблиці	
	7.7 Переліки	
	7.8 Примітки	
	7.9 Виноски	
	7.10 Формули та рівняння	
	7.11 Посилання	
	7.13 Список авторів	
	7.14 Скорочення та умовні познаки	
	7.15 Додатки	

зауважень немає

Експерт _____ (підпис) Олена ОЛІЙНИК (прізвище, ініціали)

12.06.2024

ДОДАТОК Е

Код базового InPaintPipeline

```
import cv2
import numpy as np
import json
from io import BytesIO

from pydantic import parse_obj_as
from PIL import Image
from skimage.metrics import structural_similarity, peak_signal_noise_ratio

from models.inpaint.base_inpainter import BaseInPainter
from models.inpaint.deepfill.deepfill_v2 import DeepFillV2InPainter
from models.inpaint.dto import Config, ModelType, InPaintResult
from models.inpaint.stable_diffusion import StableDiffusionInPainter


MODEL_TYPE_MAPPING = {
    ModelType.STABLE_DIFFUSION: StableDiffusionInPainter,
    ModelType.DEEP_FILL_V2: DeepFillV2InPainter,
}


class InPaintPipeline:
    def __init__(self, config_path: str) -> None:
        self.config_path = config_path
        self.config: Config = {}
        self.models: dict[str, BaseInPainter] = {}

    def get_available_models(self) -> list[str]:
        return list(self.models.keys())

    def load(self) -> None:
        with open(self.config_path, "r") as f:
            self.config = parse_obj_as(Config, json.load(f))
        for name, model_config in self.config.items():
            model_cls = MODEL_TYPE_MAPPING[model_config.type]
            self.models[name] = model_cls(
```

```
                name=name,
                device=model_config.device,
                preload=model_config.preload,
                **model_config.params
            )

    def inpaint(
            self, model_name: str, pil_image: Image.Image, pil_mask_image:
Image.Image, prompt: str = None
    ) -> list[InPaintResult]:
        inpainter = self.models[model_name]
        return inpainter.inpaint(pil_image, pil_mask_image, prompt=prompt)

    @staticmethod
    def pil_image_to_bytes(pil_image: Image, extension: str = "PNG") ->
bytes:
        with BytesIO() as buf:
            pil_image.save(buf, format=extension)
            buf.seek(0)
            return buf.read()

    @staticmethod
    def calculate_metrics(image: Image, valid_image: Image) -> dict:
        image,          valid_image          =          image.convert("RGB"),
valid_image.convert("RGB")
        valid_image = valid_image.resize(image.size)
        image, valid_image = np.array(image), np.array(valid_image)
        mse = np.mean((image - valid_image) ** 2)

        image_gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
        valid_image_gray = cv2.cvtColor(valid_image, cv2.COLOR_RGB2GRAY)
        mssim = structural_similarity(image_gray, valid_image_gray)
        psnr = peak_signal_noise_ratio(valid_image_gray, image_gray)
        return {
            "psnr": psnr,
            "ssim": mssim,
            "mse": mse,
        }
```

ДОДАТОК Ж

Код BaseInPainter

```
from abc import ABC, abstractmethod

import torch
from PIL.Image import Image

from models.inpaint.dto import InPaintResult


class BaseInPainter(ABC):
    def __init__(self, name: str, device: torch.device | str = None,
preload: bool = False, **kwargs):
        self.name = name
        if device is None:
            device = self.get_available_gpu_or_cpu()
        self.device = device
        self.is_loaded = False
        self.model = None
        if preload:
            self.load()

    @abstractmethod
    def load(self) -> None:
        pass

    @abstractmethod
    def inference(self, image: Image, mask: Image, prompt: str = None) ->
list[InPaintResult]:
        pass

    def pre_process(self, image: Image, mask: Image, prompt: str = None) ->
(Image, Image, str):
        pil_image = image.convert("RGB")
        pil_mask_image = mask.quantize(colors=256)
        return pil_image, pil_mask_image, prompt
```

```
    def inpaint(self, pil_image: Image, pil_mask_image: Image, prompt: str
= None) -> list[InPaintResult]:
        if not self.is_loaded:
            self.load()
        image, mask, prompt = self.pre_process(pil_image, pil_mask_image,
prompt)
        result = self.inference(image, mask, prompt)
        return result

    def free(self) -> None:
        self.model = None
        self.is_loaded = False

    @staticmethod
    def get_available_gpu_or_cpu():
        device = torch.device("cpu")
        if torch.cuda.is_available():
            device = torch.device("cuda")
        elif torch.backends.mps.is_available():
            device = torch.device("mps")
        return device
```