Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

ДОСЛІДЖЕННЯ МЕТОДІВ ПЕРЕНЕСЕННЯ ХАРАКТЕРИСТИК З ОДНОГО ЗОБРАЖЕННЯ НА ІНШЕ
(тема)

Виконав:
студент 2 курсу, групи ІНФм-22-2

Туркін М.Д.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник проф. Машталір В.П.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти другий (магістерський)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Туркіну Микиті Дмитровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів перенесення характеристик з одного зображення на інше

затверджена наказом по університету від 3 листопада 2023 року № 1280Ст

2. Термін подання студентом роботи до екзаменаційної комісії 22 грудня 2023р.

3. Вихідні дані до роботи моделі перенесеня стилю з одного зобрадення на інше на основі NST та GAN підходів.

4. Перелік питань, що потрібно опрацювати в роботі

1. Огляд методів перенесення стилю зображення.
2. Огляд вихідних даних шарів CNN моделей.
3. Розробка моделей перенесення стилю зображення.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) актуальність проблеми обраної теми, постановка задачі, обрані методи для перенесення стилю зображень, тестові зображення.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	03.11.2023	
2	Аналіз завдання, підбір літератури	4.11.23-6.11.23	
3	Аналіз літератури з досліджуваної проблеми	7.11.23-13.11.23	
4	Аналіз методів вибору ключових точок	13.11.23-16.11.23	
5	Розробка системи перенесення стилю	17.11.23-22.11.23	
6	Програмна реалізація	22.11.23-27.11.23	
7	Оформлення пояснювальної записки	27.11.23-2.12.23	
8	Перевірка на плагіат	06.12.2023	
9	Рецензування	10.12.2023	
10	Підготовка презентації та доповіді	24.12.2023	
11	Занесення роботи в електронний архів	01.01.2024	
12	Попередній захист кваліфікаційної роботи	03.01.2024	

Дата видачі завдання 3 листопада 2023 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Машталір В.П.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 68 с., 27 рис., 41 джерело.

КОНВОЛЮЦІЙНІ НЕЙРОННІ МЕРЕЖІ, ПЕРЕНЕСЕННЯ СТИЛЮ, ФУНКЦІЯ ВТРАТИ, ГЕНЕРАТИВНІ НЕЙРОННІ МЕРЕЖІ, CYCLEGAN, PYTORCH.

Об’єктом дослідження є методи перенесення стилів з одного зображення на інше.

Метою дослідження є розробка методів, що базуються на використанні конволюційних і генеративних нейронних мереж.

Перенесення стилю – це техніка комп’ютерного зору, яка використовує попередньо навчену згорткову нейронну мережу (CNN) для перетворення вмісту зображення в стиль іншого зображення. Вона передбачає три вхідні дані: зображення вмісту, зображення стилю та зображення білого шуму. Програма спрямована на мінімізацію втрат контенту, які вимірюють відповідність вихідного зображення високорівневим ознакам, і втрат стилю, які вимірюють відповідність низькорівневим ознакам. Нейронне перенесення стилю пов’язане з проблемами балансування змісту і стилю, збереження семантичної цілісності оригінального зображення, а також контролю рівня і позиції перенесення стилю.

У результаті дослідження здійснена програмна реалізація методів перенесення стилю з одного зображення на інше.

CONVOLUTIONAL NEURAL NETWORKS, STYLE TRANSFER, LOSS FUNCTION, GENERATIVE NEURAL NETWORKS, CYCLEGAN, PYTORCH.

The object of research is methods of transferring styles from one image to another.

The purpose of research is to develop methods based on the use of convolutional and generative neural networks.

Style transfer is a computer vision technique that uses a pre-trained convolutional neural network (CNN) to convert the content of an image into the style of another image. It takes three inputs: a content image, a style image, and a white noise image. The programme aims to minimise the content loss, which measures the correspondence of the original image to high-level features, and the style loss, which measures the correspondence to low-level features. Neural style transfer is associated with the problems of balancing content and style, preserving the semantic integrity of the original image, and controlling the level and position of style transfer.

The research resulted in the software implementation of style transfer methods from one image to another.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

CNN – Convolutional Neural Network

NST – Neural Style Transfer

GAN – Generative Adversarial Network

ШІ – Штучний інтелект

ВСТУП

Перенесення стилю – це техніка комп'ютерного зору, яка дозволяє трансформувати зміст зображення у стиль іншого зображення. Кожен, хто коли-небудь уявляв, як би виглядала фотографія, намальована відомим художником, може втілити це бачення в реальність.

Нейронна передача стилю використовує попередньо навчену згорткову нейронну мережу (CNN), яка була навчена на великому наборі даних зображень. CNN [1] складається з декількох шарів, які охоплюють різні рівні абстракції, від низькорівневих країв і текстур до високорівневих об'єктів і сцен. Для створення стилізованого зображення необхідні три вхідні дані: зображення контенту, зображення стилю та зображення білого шуму. Зображення білого шуму слугує початковим виходом, який буде модифіковано алгоритмом. Програма має на меті зменшити дві втрати: втрату вмісту та втрату стилю. Втрата вмісту вимірює відповідність вихідного зображення високорівневим ознакам, таким як форми та об'єкти, що зображені на еталонному зображенні. Втрата стилю вимірює відповідність низькорівневих характеристик, таких як кольори і текстури вихідного зображення еталонному зображенню. Програма продовжує ітерації доти, доки обидві втрати не будуть задовільненими.

Нейронне перенесення стилю – це захоплююча і творча методологія, але вона має певні проблеми та обмеження. Основною проблемою є балансування між втратою змісту та стилю, оскільки вони можуть суперечити одне одному. Прикладом цього є ситуація, коли стильовий образ має кілька деталей і контрастів, що потенційно може домінувати над змістовним образом і призвести до того, що він стане неідентифікованим. Збереження семантичної цілісності вихідного зображення є значним викликом, оскільки алгоритм може спотворити або розмити критичні аспекти зображення, які сприяють формуванню сенсу, наприклад, обличчя або текст. Крім того, нейронна

передача стилю вимагає значних обчислювальних ресурсів і часу через численні ітерації та вимоги до градієнтного спуску. Також, складно керувати рівнем і позицією перенесення стилю, оскільки він залежить від бажаних шарів CNN і ваги втрат.

Перенесення характеристик з одного зображення на інше має неабиякий потенціал в індустрії: провідні бренди використовують його для створення сучасних творів мистецтва, які пропонують художній погляд на творчість, нові модні тенденції, дизайн одягу та візуально привабливу рекламу. Тим не менш, існують певні обмеження у його використанні. Наприклад, вона може призвести до викривлених і нереалістичних зображень, змінити оригінальні естетичні аспекти мистецтва або спотворити і затуманити значення зображень, символів, об’єктів чи сцен. У деяких випадках це може навіть вплинути на емоційне або контекстуальне значення твору, що може призвести до приниження системи переконань або мати згубний вплив на релігійні вірування чи давню історичну цінність твору мистецтва.

Актуальність дослідження полягає у тому, що перенесення характеристик з одного зображення на інше активно застосовується як в індустріях, які наразі активно розвиваються, таких як доповнена та віртуальна реальність, так і у сферах геймінгу та графічних інструментів які вже мають досить широкий ринок. Хоча застосунки, що використовують передачу стилю у віртуальній реальності, перебувають переважно на стадії дослідження, перспективи є багатообіцяючими та інтригуючими. Facebook просуває потенціал перенесення стилю, щоб суттєво змінити методи, за допомогою яких розробники VR [2] передають візуальні наративи у своїх застосунках, іграх та фільмах, серед інших засобів масової інформації. І перші демонстрації показують, що перенесення стилю має потенціал для посилення занурення у світи віртуальної реальності. У Google заявили, що їхні зусилля спрямовані на те, щоб дати можливість розробникам виражати свою творчість через ігри та інші застосунки з перенесенням стилів.

1 ОГЛЯД ОСНОВНИХ МЕТОДІВ ДЛЯ ПЕРЕНЕСЕННЯ СТИЛЮ

1.1 Базові підходи перенесення стилю

Перенесення стилю є прикладом стилізації зображень. Це техніка обробки та маніпулювання зображеннями, яка вивчалася десятиліттями в рамках ширшої галузі нефотореалістичного рендерингу. Якщо розглядати розв’язання цієї задачі за допомогою традиційного підходу до навчання під контролем, то для передачі вивченого стилю необхідно мати дует вхідних зображень, початкове зображення та художнє відтворення цього початкового зображення. Згодом модель машинного навчання засвоїть вивчену трансформацію і перенесе її на нові оригінальні зображення.

На жаль, ця методика не може бути застосована, оскільки такі пари зображень зустрічаються рідко. Однак в останні роки з’явилася нова методика під назвою Нейронне перенесення стилю (NST) [3]. NST використовує глибокі нейронні мережі для полегшення цих перетворень. Без необхідності в явних парах зображень нейронні мережі аналізують статистичні особливості, пов’язані з контентом і стилем, що дозволяє точно вимірювати ефективність перенесення стилю. Завдяки цьому вдосконаленому підходу нейромережі потрібно лише одне зображення-еталон стилю, щоб застосувати стиль до оригінальних зображень контенту.

Ранні версії NST, хоча й не позбавлені недоліків, підходили до завдання як до оптимізаційної задачі, вимагаючи сотні або навіть тисячі ітерацій для перенесення стилю на одне зображення. Для усунення цієї неефективності вчені створили так зване швидке нейронне перенесення стилів [4]. Цей метод також використовує глибокі нейронні мережі, але тренує окрему модель для перетворення будь-якого зображення за один прохід вперед. За допомогою однієї ітерації через мережу, а не тисяч, навчені моделі можуть стилізувати будь-яке зображення.

1.2 Шляхи та засоби нейронної передачі стилю

Як правило, архітектури глибокого навчання, придатні для передачі стилю, покладаються на згорткові нейронні мережі (CNN). Простежуючи еволюцію досліджень у галузі перенесення стилю, можна чітко розрізнити різні підходи. Від методів, що використовують одну або кілька варіантів стилю, до методів, що реалізують довільні варіанти стилю, вивчення постійного вдосконалення цих методів і оптимізацій може прояснити, чого можна досягти за допомогою перенесення стилю.

Для навчання моделі перенесення стилю потрібні дві мережі: попередньо навчений екстрактор ознак і мережа перенесення. Попередньо навчений екстрактор ознак використовується для того, щоб уникнути необхідності використовувати парні навчальні дані. Його корисність випливає з цікавої тенденції окремих шарів глибоких згорткових нейронних мереж, навчених класифікації зображень, спеціалізуватися на розумінні специфічних особливостей зображення. Деякі шари вчаться виокремлювати зміст зображення (форму собаки або положення автомобіля), тоді як інші вчаться фокусуватися на текстурі (дрібних мазках пензля художника або фрактальних візерунках природи) [5]. Передача стилю використовує цю перевагу, пропускаючи два зображення через попередньо навчену нейронну мережу, дивлячись на вихід попередньо навченої мережі на декількох рівнях і порівнюючи їхню схожість. Зображення, які дають схожі результати на одному шарі попередньо навченої моделі, ймовірно, мають схожий вміст, тоді як збіг результатів на іншому шарі сигналізує про схожий стиль [6].

Навчена модель дозволяє нам порівняти зміст і стиль двох зображень, але не допоможе нам створити стилізовану картинку. Це робота другої нейронної мережі, яку ми назвемо трансферною мережею. Трансферна мережа – це мережа перетворення зображень, яка отримує на вхід одне зображення, а на виході – інше. Зазвичай, архітектура мережі перетворення – це кодер-декодер.

На початкових етапах попередньо навчений екстрактор ознак обробляє одне або кілька зображень стилів і зберігає їхні результати на декількох стильових рівнях для подальшого використання. Згодом в систему подаються зображення контенту. Кожне зображення контенту обробляється попередньо навченим екстрактором ознак, в результаті чого його результати зберігаються на різних рівнях контенту. Потім зображення контенту передається в мережу передачі, яка створює стилізоване зображення на виході. Стилізоване зображення також пропускається через екстрактор ознак, і виходи з шарів вмісту та стилю зберігаються. Якість стилізованого зображення визначається спеціальною функцією втрат, яка враховує як особливості вмісту, так і особливості стилю. Ознаки вмісту, витягнуті зі стилізованого зображення, порівнюються з оригінальним зображенням вмісту, тоді як ознаки стилю порівнюються з тими, що отримані з еталонного зображення (зображень) стилю. Наприкінці кожного кроку оновлюється лише мережа перенесення, використовуючи ваги попередньо навченого екстрактора ознак, що залишаються фіксованими протягом усього процесу (рис. 1.1).

Рисунок 1.1 – Приклад перенесення стилю

Налаштовуючи різні компоненти функції втрат, ми можемо навчити моделі генерувати вихідні зображення з різним ступенем стилізації [7]. Існує багато різних архітектур нейронних мереж, і хоча неможливо охопити всі з

них, має сенс зробити огляд надійних архітектур, які можуть слугувати відправною точкою для подальших досліджень.

1.3 Моделі перенесення художніх стилів

Метод використовує зображення одного стилю для перетворення будь-якого вхідного зображення у бажаний художній стиль. У 2016 році Джонсон та ін. [8] стали першими, хто навчив незалежну нейронну мережу стилізувати зображення за один прямий прохід. Вони використовували великі моделі VGG16 (рис. 1.2), попередньо навчені на ImageNet, для вилучення ознак, тоді як відносно невелика мережа кодерів-декодерів слугувала мережею перетворення в цьому процесі [9]. У цьому підході мережа перетворення навчається для кожного потрібного стилю.

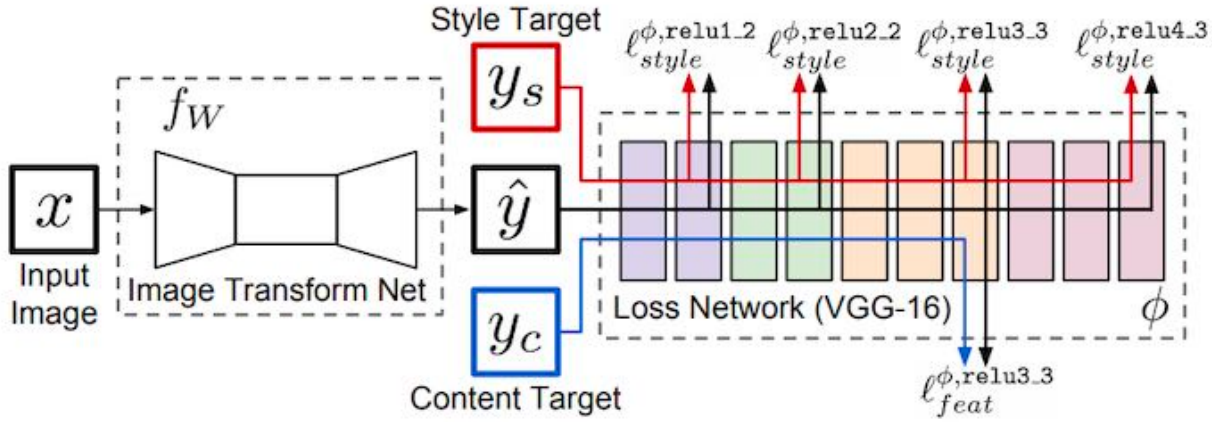


Рисунок 1.2 – Архітектура нейронної мережі

Мережа перетворення зображень – це згорткова нейронна мережа з глибокими залишками, яка навчена розв’язувати задачу оптимізації, запропоновану Гатісом [10]. Отримавши вхідне зображення x, ця мережа перетворює його у вихідне зображення $\hat{y}$. Ваги цієї мережі W визначаються за допомогою втрат, розрахованих на основі вихідного зображення $\hat{y}$, і

порівняння їх з зображеннями стилю y_s та контенту y_c, у випадку передачі стилю. Мережа перетворення зображення навчається за допомогою стохастичного градієнтного спуску, щоб отримати ваги W, які мінімізують зважену суму всіх функцій втрат.

$$W = \arg minE_{x,y_i}[\sum_{i=1} \lambda_i l_i(f_W(x), y_i], \tag{1.1}$$

де з ключових аспектів: використовується стохастичний градієнтний спуск для оновлення ваг мережі перетворення зображень f_w; мета – мінімізувати зважену суму функцій втрат з усіх шарів і, з яких береться представлення втрат; втрати обчислюються між вихідним зображенням $f_w(x)$ та цільовим представленням зображення y_i для кожного шару і; λ_i – скалярне значення, що множить втрати з шару і для отримання зваженої суми всіх втрат.

Мережа втрат – це попередньо навчена VGG16 на наборі даних ImageNet. Мережа втрат використовується для отримання представлень вмісту та стилю з зображень. Представлення вмісту беруться з шару relu3_3. Представлення стилю беруться з шарів relu1_2, relu2_2, relu3_3 та relu4_3. Ці представлення використовуються для визначення двох типів втрат: втрата реконструкції ознак з вихідним зображенням $\hat{y}$ та представленням вмісту з шару relu3_3, використовуючи наступну функцію втрат для мінімізації відстані між представленнями вмісту вихідного та цільового зображення.

$$l_{content}(y, \hat{y}) = \frac{1}{C_j H_j W_j} \left\| \phi_j(\hat{y}) - \phi_j(y) \right\|_2^2 . \tag{1.2}$$

З вихідним зображенням $\hat{y}$ та представленнями стилю з шарів relu1_2, relu2_2, relu3_3 та relu4_3, використовуючи наступну функцію втрат з зображення [11]. Загальна втрата зазвичай є зваженою сумою втрати реконструкції ознак та втрати реконструкції стилю у випадку перенесення

стилю і просто зваженим добутком втрати реконструкції ознак для надрозрізнення.

У 2017 році, через рік після публікації оригінального методу швидкої стилізації, дослідники з Google розширили методику [12], щоб дозволити одній мережі стилізації генерувати фотографії з різними стилями, а також сумішшю стилів. Їхній основний внесок полягав в інтеграції страт «умовної нормалізації екземплярів» всередині мережі, що дозволило впливати на стилізовану фотографію за допомогою додаткової вхідної моделі. Ці мережі потребують зображення контенту та додаткового вектора для застосування різних стилів до зображення.

Під час стилізації зображення користувач може вказати ступінь застосування кожного стилю. Наприклад, модель можна навчити, використовуючи картини Ван Гога, Пікассо та Матісса як референси. Наприклад, [1, 0, 0] для Ван Гога, [0, 1, 0] для Пікассо або [0,33, 0,33, 0,33] для поєднання всіх трьох стилів. Такий зручний підхід усуває необхідність мати кілька моделей для різних стилів і надає користувачам гнучкість у комбінуванні різних стилів. У цій статті пропонується метод навчання однієї нейронної мережі для виконання передачі художнього стилю для декількох стилів. Перенесення стилю передбачає трансформацію зображення відповідно до художнього стилю іншого зображення зі збереженням вмісту.

Попередні методи вимагали навчання окремої мережі для кожного стилю. Вони демонструють це, навчаючи мережі з 32 різними художніми стилями. Результати відповідають якості мереж перенесення окремих стилів, але вимагають значно меншої кількості параметрів. Перевага полягає в тому, що нові стилі можна швидко додавати, навчаючи лише невеликі специфічні для стилю параметри, а не всю мережу. Мережа також дозволяє довільно змішувати/інтерполювати різні художні стилі. Загалом, цей метод дозволяє навчати модель передачі художнього стилю, яка узагальнює великий словник стилів живопису. Це також дає уявлення про те, як художні стилі можуть бути представлені та поєднані в моделі на основі нейронної мережі.

Довільні стилі для кожної моделі. Одно та багато стильові моделі перенесення обмежені в тому, що вони можуть створювати зображення лише в тих стилях, на яких вони були навчені. Якщо модель була навчена на роботах Ван Гога, вона не може створювати зображення, подібні до картин Пікассо, без повного перенавчання мережі. Однак довільне перенесення стилю Хуана та ін. змінює цю ситуацію [13]. Модель дозволяє вводити зображення контенту та зображення стилю і виконує перенесення стилю за один прохід у прямому напрямку. По суті, модель навчається виокремлювати та застосовувати будь-який стиль до зображення за один прохід. Це досягаєтеся за допомогою адаптивної нормалізації екземплярів (AdaIN) [14].

Адаптивна нормалізація екземплярів – це метод нормалізації, який вирівнює середнє та дисперсію ознак вмісту з ознаками стилю. Нормалізація екземплярів нормалізує вхідні дані до одного стилю, визначеного за допомогою афінних параметрів. Адаптивна нормалізація екземплярів є розширенням. В AdaIN отримується вхідні дані контенту і вхідні дані стилю і просто вирівнюємо середнє значення та дисперсію по каналах щоб вони відповідали значенням. На відміну від пакетної нормалізації, нормалізації екземплярів або умовної нормалізації екземплярів, AdaIN не має афінних параметрів, які можна вивчити. Натомість він адаптивно обчислює афінні параметри на основі вхідних даних стилю.

$$AdaIN(x, y) = \sigma(y)\left(\frac{x-\mu(x)}{\sigma(x)}\right) + \mu(y). \quad (1.3)$$

AdaIN відносно простий, але ефективний підхід для передачі довільного стилю в реальному часі, запропонований Хуангом і Белонгі [15]. В основі AdaIN лежить адаптивний рівень нормалізації екземплярів, який вирівнює статистику ознак між зображенням контенту та зображенням довільного нового стилю. Зокрема, мережа передачі стилю спочатку витягує представлення ознак із пари зображень контенту та стилю за допомогою

попередньо навченої мережі кодерів VGG, яка фіксує інформацію про сприйняття. Потім шар AdaIN приймає ці характеристики контенту та стилю як вхідні дані. Для кожного каналу він нормалізує ознаки контенту до нульового середнього значення та одиничної дисперсії, а потім масштабує та зсуває ці нормалізовані ознаки, використовуючи середнє значення та дисперсію, розраховані на основі ознак стилю. Математично, якщо *c* і *s* – це характеристики контенту і стилю, AdaIN обчислює вихідні характеристики наступним чином:

$$t = \sigma(s) * \big(c - \mu(c)\big)/\sigma(c) + \mu(s), \tag{1.4}$$

де μ та σ позначають операції середнього та дисперсії у просторових вимірах. Ця проста операція дозволяє перенести статистику ознак, а саме моменти першого та другого порядку, з ознак стилю на ознаки змісту. У статті показано, що зіставлення цих статистичних даних ефективно фіксує інформацію про стиль [16]. Після передачі AdaIN у просторі ознак мережа декодерів навчається інвертувати вихідні ознаки назад у піксельний простір для створення остаточного стилізованого зображення. Декодер оптимізовано на основі втрат при відтворенні стилю та втрат при відтворенні контенту, обчислених за допомогою попередньо навченої мережі VGG, так що він зберігає структуру контенту, відображаючи при цьому стильові патерни.

Перевагою AdaIN є те, що він може передавати довільні нові стилі, не помічені під час навчання, динамічно обчислюючи статистику ознак. Оперуючи простором ознак, AdaIN забезпечує ефективний і гнучкий підхід до перенесення стилів, який досягає швидкості, порівнянної з попередніми методами зворотного зв’язку, обмеженими фіксованим набором стилів. Експерименти показують прискорення на порядок порівняно з попередніми методами перенесення стилів, а також можливість перенесення в реальному часі наданих користувачем стилів, невидимих під час навчання. Також

природно включені додаткові елементи керування для компромісів між стилями контенту, інтерполяції стилів, збереження кольорів (рис. 1.3).

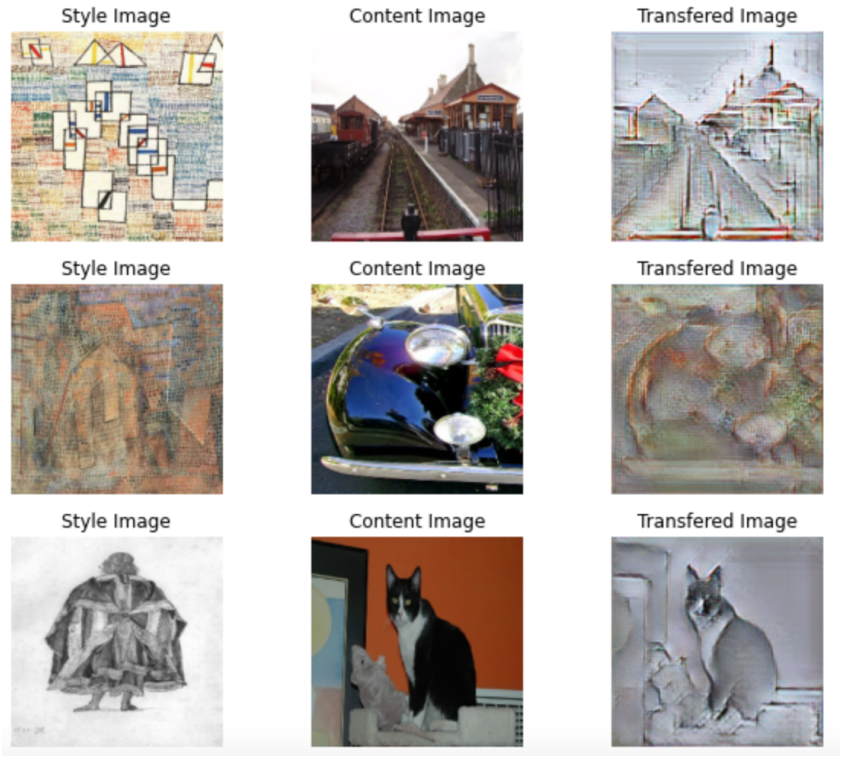
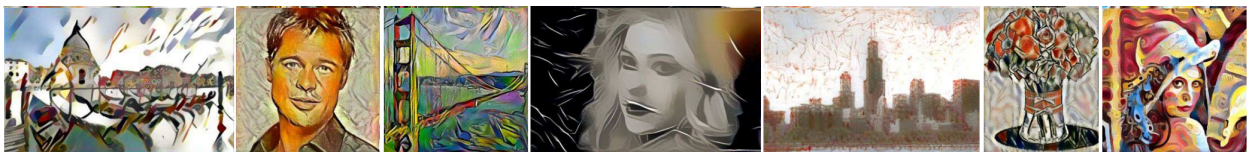


Рисунок 1.3 – Приклади роботи AdaIN

Розширенням підходу AdaIN є AvatarNet. AvatarNet – це багатомасштабний метод перенесення стилів, запропонований Шенгом та ін., який дозволяє передавати довільні нові стилі в реальному часі за один прохід у зворотному напрямку. В основі AvatarNet лежить модуль декоратора стилів, який поширює шаблони стилів із зображення довільного стилю на елементи зображення контенту, зберігаючи при цьому семантичну структуру [17]. Мережа реконструкції зображень з архітектурою пісочного годинника та багатомасштабним злиттям стилів потім перетворює ці декоровані елементи назад у простір зображення. Зокрема, декоратор стилю спочатку проектує вміст і стильові особливості в загальний вибілений простір за допомогою AdaIN. Це видаляє текстурні характеристики і приводить об’єкти до спільного масштабу. Потім виконується узгодження патчів між нормалізованим контентом і елементами стилю за допомогою ефективної згортки на основі

патчів. Це дозволяє зіставити кожен патч контенту з найбільш корельованим патчем стилю у спосіб, інваріантний до текстури. Потім патчі стилів знову збираються в макет контенту для створення декорованих елементів. Нарешті, розфарбовування перетворює їх назад у область елементів стилю. Цей модуль декоратора стилів передає цілісний розподіл стилів, одночасно отримуючи детальні шаблони стилів без спотворень або упередженості. Потім AvatarNet використовує кодер-декодер у вигляді пісочного годинника з багатомасштабними модулями злиття стилів, які дозволяють об'єднати шаблони стилів із закодованих стильових елементів у проміжні декодовані елементи в різних масштабах. Це забезпечує багатомасштабний шлях для цілісної інтеграції стилів (рис. 1.4).

а)



б)

в)

Рисунок 1.4 – Порівняння підходів:
а) нейронне перенесення стилю Гатіс та інші; б) AdaIN;
в) AvatarNet

Весь конвеєр дозволяє отримати довільне зображення стилю, декорувати елементи контенту, а потім згенерувати стилізований вихідний результат у швидкому циклі прямого перетворення. На відміну від

оптимізаційного методу перенесення стилів NST, який ітеративно оновлює пікселі відповідно до статистики матриці Грама [18], AvatarNet працює повністю в просторі об’єктів, безпосередньо поширюючи шаблони. Це дає змогу на порядки прискорити процес узагальнення для нових стилів. AvatarNet також відрізняється від AdaIN, який просто зіставляє статистику каналу, а не шаблони. Модулі декоратора стилів і злиття забезпечують більш складну стратегію трансформації та інтеграції функцій. Експерименти показують, що AvatarNet забезпечує кращу стилізацію у порівнянні з попередніми методами для довільних стилів у реальному часі, забезпечуючи при цьому гнучкий контроль з боку користувача.

Можливість прикрашати об’єкти явними стильовими патернами в поєднанні з багатомасштабною інтеграцією видається ключовим фактором для якісного узагальнення.

Оптимізація та розширення перенесення стилів. Існує декілька оптимізацій та розширень перенесення стилів, які заслуговують на обговорення. По-перше, існує стабільне перенесення стилів, яке вирішує проблему мерехтіння у відеокадрах. Застосування натренованої моделі передачі стилю до відеокадрів часто призводить до змін і шуму, що призводить до відволікаючих часових невідповідностей. Наприклад, футбольний м’яч може змінити колір посеред польоту. Щоб вирішити цю проблему, моделі стабільного перенесення стилю включають новий термін втрат, що стосується «часової когерентності». Це передбачає стилізацію та порівняння двох послідовних кадрів, щоб навчити модель створювати послідовну стилізацію об’єкта, коли він рухається в кадрі [19].

Іншим аспектом передачі стилю є збереження оригінальних кольорів зображення під час передачі мазків пензля художника. Для цього існує кілька методів, зокрема перетворення представлення зображення з RGB в інший колірний простір і застосування передачі стилю лише до каналу яскравості, або застосування алгоритму передачі кольору до кінцевого зображення, перетвореного стилем. І, нарешті, хоча більшість згаданих вище способів

передачі стилю стосувалися застосування художнього стилю до фотозображення, між двома фотореалістичними зображеннями можна переносити й інші фактори, наприклад, погоду або час доби. Передача фотореалістичного стилю часто вимагає коригування мережі передачі кодера-декодера, щоб мінімізувати артефакти, які можуть з'являтися внаслідок глибокої згортки та підвищеної дискретизації шарів.

1.4 Застосування перенесення стилю в реальному житті

Інтеграція з програмним забезпеченням для редагування фото та відео – одне з найочевидніших застосувань Style Transfer. Можливість додавати відомі художні стилі до зображень і відеокліпів обіцяє безпрецедентну потужність цим інструментам для творчості – від обміну стилізованими селфі до доповнення створених користувачем музичних відео і не тільки.

Завдяки гнучкості та продуктивності сучасних підходів до глибокого навчання моделі перенесення стилів можна вбудовувати в периферійні пристрої, такі як мобільні телефони. Це дозволяє програмам обробляти і трансформувати зображення і відео в режимі реального часу, роблячи інструменти для редагування фото і відео професійної якості більш доступними і простими у використанні, ніж будь-коли раніше.

Доступно кілька чудових інструментів, які включають передачу стилю як частину свого інструментарію, в тому числі різні цікаві проекти, програми та демонстраційні версії для експериментів [20]. Наприклад: Prisma (iOS, Android) (рис. 1.5), Instapainting's AI Painter (Desktop) (рис. 1.6), Video Star (Art Studio, iOS), Painter's Lens (iOS), Looq (iOS, Android) (рис. 1.7).

Рисунок 1.5 – Prisma, приклад інтерфейсу

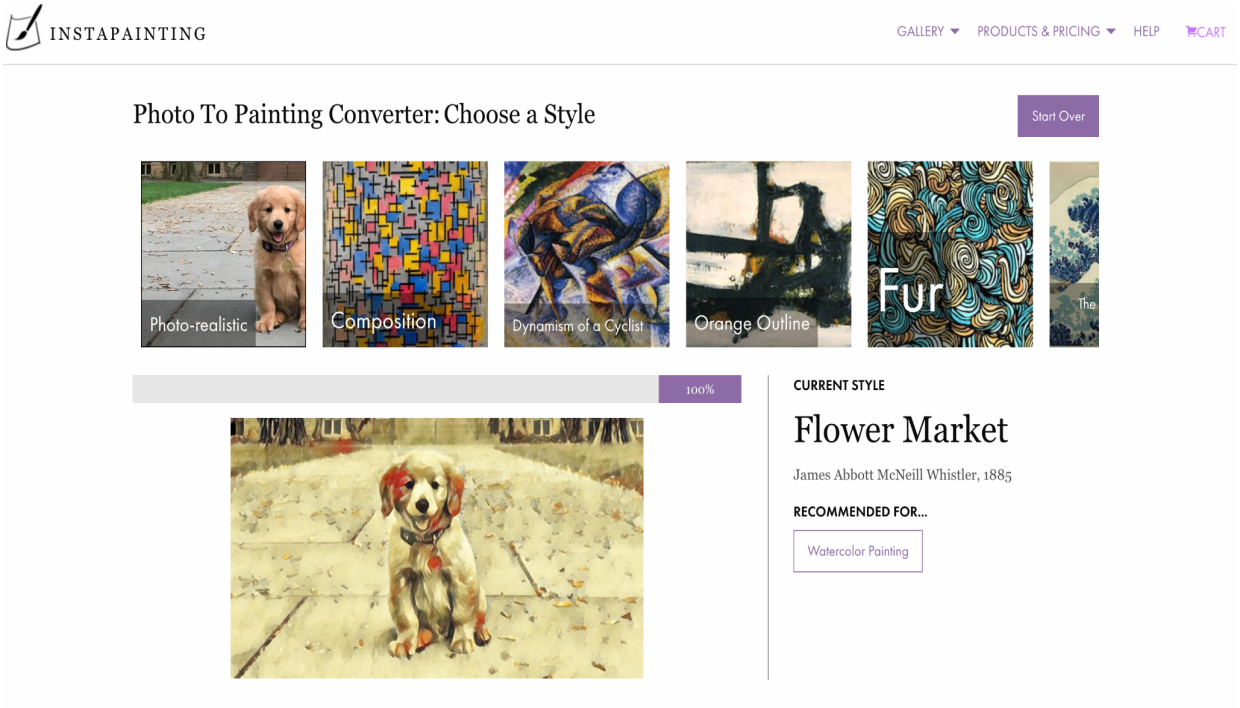


Рисунок 1.6 – Instapainting, приклад інтерфейсу

Рисунок 1.7 – Looq, приклад інтерфейсу

Перенесення стилю – це техніка комп'ютерного зору, яка дозволяє створювати твори мистецтва за допомогою ШІ. Незалежно від того, чи продається твір мистецтва на аукціоні високого класу, чи художники-початківці шукають нові способи поділитися своєю естетикою зі світом, перенесення стилю обіцяє змінити способи сприйняття мистецтва, те, що становить оригінальність, і те, як мистецтво представлене у фізичному світі. Ми можемо уявити, як реалізувати передачу стилю для створення тиражованих, першокласних принтів, призначених для офісних будівель і великих рекламних кампаній. Це лише кілька можливих каналів, за допомогою яких трансфер стилю може революціонізувати наше уявлення про комерційний вплив мистецтва.

Нейронне перенесення стилів має різні особисті та професійні застосування і переваги. На особистому рівні нейронне перенесення стилів може покращувати фотографії та створювати унікальні шпалери й аватарки, дозволяючи людям виражати своє художнє бачення. Експерименти з різними стилями та жанрами також можуть допомогти у вивченні історії мистецтва та культури. Нейронне перенесення стилів можна використовувати у професійних цілях для створення анімації, логотипів, плакатів, флаєрів та інших графічних дизайнів. Крім того, він має можливість генерувати контент для соціальних мереж, блогів, веб-сайтів та онлайн-курсів. Використання нейронного перенесення стилів може допомогти привернути увагу глядачів, передати емоції, передати повідомлення та надихнути на творчість. Саме в цьому полягає трансформаційна сила перенесення стилю. Він дозволяє митцям ділитися своєю творчою естетикою, дозволяючи новим та інноваційним репрезентаціям мистецьких стилів співіснувати з оригінальними шедеврами.

Крім того, перенесення стилю дає можливість людям по всьому світу експериментувати з власною творчістю. Ця техніка також відіграє важливу роль у комерційній арт-індустрії. Нещодавно на аукціоні Christie’s було продано картину, створену штучним інтелектом, за понад 430 000 доларів [21]. Постійний розвиток апаратного забезпечення зі штучним інтелектом, як у хмарі, так і на периферії, тепер дозволяє передавати стиль у знятому відео та відео в реальному часі. Ця нова здатність створює можливості для дизайну, створення контенту та розробки інструментів для творчості. Універсальність перенесення стилів передбачає, що його можна використовувати в різних застосунках, наприклад, у таких сферах, як: фото та відеоредактори; взаємодія художника та громади; комерційне мистецтво; ігри; віртуальна реальність.

Хоча цей список не є вичерпним, він висвітлює деякі з найважливіших сфер, на які впливає перенесення стилів.

1.5 Постановка задачі дослідження

Процес передачі стилю в комп’ютерному зорі дозволяє перетворити вміст зображення в стиль іншого зображення. Проте збереження семантичної цілісності оригінального зображення може бути складним завданням, оскільки важливі компоненти, такі як обличчя або текст, можуть бути спотворені або розмиті. Крім того, нейронна передача стилю вимагає значних обчислювальних ресурсів і часу. Крім того, може бути складно регулювати ступінь і місце перенесення стилю.

Об’єктом дослідження є методи перенесення стилів з одного зображення на інше.

Метою дослідження є розробка методів, що базуються на використанні конволюційних і генеративних нейронних мереж.

Для досягнення мети необхідно вирішити такі завдання:

– провести аналіз існуючих методів та оцінити ефективність наявних підходів до автоматичного перенесення стиля з одного зображення на інше;

– зробити експериментальне дослідження з використанням реальних або синтетичних даних для оцінки підходів;

– реалізувати програмний застосунок для порівняння підходів покращення перенесення характеристик між зображеннями.

2 ПІДХОДИ ДО ПОБУДОВИ НЕЙРОННИХ МЕРЕЖ ДЛЯ ПЕРЕНЕСЕННЯ СТИЛЮ З ОДНОГО ЗОБРАЖЕННЯ НА ІНШЕ

2.1 CNN для перенесення стилю з одного зображення на інше

Згорткові нейронні мережі (CNN) працюють на основі фундаментальної концепції згортки. Наприклад, коли на вхід подається зображення і фільтр, фільтр ковзає по зображенню, створюючи вихід, який ілюструє суму зважених вхідних даних, охоплених фільтром. Наприклад, коли на зображенні є фільтр, фільтр ковзає по зображенню, створюючи вихід, який ілюструє суму зважених входів, охоплених фільтром. Наприклад, при наявності зображення і фільтра, фільтр ковзає по зображенню, створюючи вихід, який ілюструє суму зважених входів, охоплених фільтром. Сума зазнає нелінійного перетворення, наприклад, сигмоїд, ReLU або тангенс. Під час процесу згортки кожен фільтр має свій власний набір незмінних ваг.

Згортка – це комбінована інтеграція двох функцій, яка показує, як одна функція модифікує іншу. У цьому процесі є три важливі компоненти: вхідне зображення, детектор ознак і карта ознак. Важливо зазначити, що мова, яка використовується в цьому поясненні, є об’єктивною, чіткою та формальною. Крім того, технічні скорочення (наприклад, 3×3) пояснюються при першому використанні, а текст є граматично точним і не містить зайвих слів. Вхідне зображення – це зображення, яке потрібно розпізнати. Детектор ознак, як правило, являє собою матрицю 3×3, але також може бути матрицею 7×7, яку також називають ядром або фільтром [22].

Матричне представлення вхідного зображення потім поелементно перемножується з детектором ознак для отримання карти ознак, також відомої як згорнута карта ознак або карта активації. Метою цього кроку є зменшення розміру зображення для прискорення обробки. Ці унікальні особливості дозволяють ідентифікувати конкретні об’єкти, наприклад, характерні ознаки

тварин. Щоб запобігти втраті інформації про зображення, використовуються численні карти ознак, які визначають місцезнаходження конкретних ознак на зображенні. Щоб запобігти втраті інформації про зображення, використовуються численні карти ознак, які визначають місцезнаходження певних об'єктів на зображенні. Хоча деякі елементи зображення приносяться в жертву, ключові елементи, необхідні для розпізнавання зображення, залишаються недоторканими. Щоб запобігти втраті інформації про зображення, використовуються численні карти ознак для визначення розташування певних елементів на зображенні [23]. Щоб запобігти втраті інформації про зображення, використовуються численні карти ознак, які визначають розташування певних елементів на зображенні (рис. 2.1).

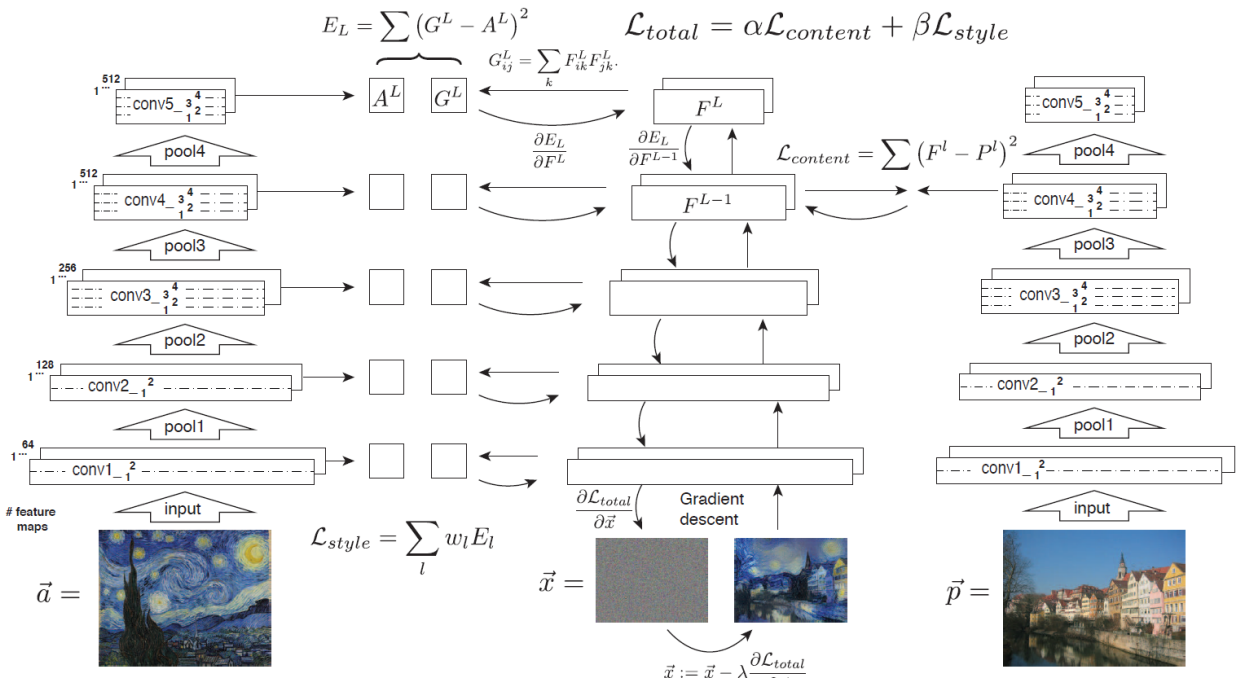


Рисунок 2.1 – Алгоритм переносу стилю на основі CNN

Відокремлення змісту від стилю в природних зображеннях є складним завданням. Однак останні досягнення в галузі глибоких згорткових нейронних мереж дозволили розробити потужні системи комп'ютерного зору, які отримують високорівневу семантичну інформацію з природних зображень. Згорткові нейронні мережі, які навчаються на великій кількості мічених даних

для конкретних завдань, таких як розпізнавання об'єктів, можуть навчитися витягувати високорівневий вміст зображень і узагальнювати його в наборах даних. Ці мережі також можуть бути застосовані до інших завдань обробки візуальної інформації, таких як розпізнавання текстур і класифікація художніх стилів.

Щодо нейронного алгоритму художнього стилю, алгоритму для виконання передачі стилю зображення. Концептуально, цей алгоритм полегшує передачу текстури, використовуючи представлення ознак із сучасних згорткових нейронних мереж для обмеження методу синтезу текстури. Оскільки модель текстури базується на глибоких уявленнях зображення, метод передачі стилю стає проблемою оптимізації в одній нейронній мережі. Нові зображення створюються шляхом виконання попереднього пошуку зображень на відповідність представленням функцій у прикладах зображень. Цей підхід раніше використовувався в контексті синтезу текстур та для покращення розуміння глибоких зображень. Алгоритм передачі стилю інтегрує параметричну модель текстури, засновану на згорткових нейронних мережах, з технікою зворотного перетворення їхніх зображень [24].

CNN може повторювати й оптимізувати ключові кроки в уніфікованій структурі та вивчати ієрархічні репрезентації безпосередньо з необроблених зображень. При використанні згорткової нейронної мережі, яка вже навчена розпізнавати об'єкти на зображеннях, вона розробляє незалежні внутрішні уявлення про зміст і стиль даного зображення. Ось демонстрація ієрархії CNN від мережі VGG. Початкові шари набувають ознак низького рівня, тоді як глибші згорткові шари можуть виявляти більш масштабні ознаки, що призводить до більш високого рівня представлення вмісту зображення.

Зображення можна створювати за допомогою карт особливостей на певному шарі згортки, що відповідає шару згортки зображення з заданим вмістом. Мета полягає в тому, щоб обидва зображення мали однаковий вміст, хоча їх текстура і стиль можуть відрізнятися. Втрата вмісту для вибраного

шару встановлюється шляхом вимірювання середньоквадратичної похибки між картою ознак, зображення вмісту і картою ознак згенерованого зображення. При мінімізації втрат контенту на змішаному зображенні спостерігається активація ознак у зазначених шарах, що дуже нагадує активацію на контент-зображенні (рис. 2.2).

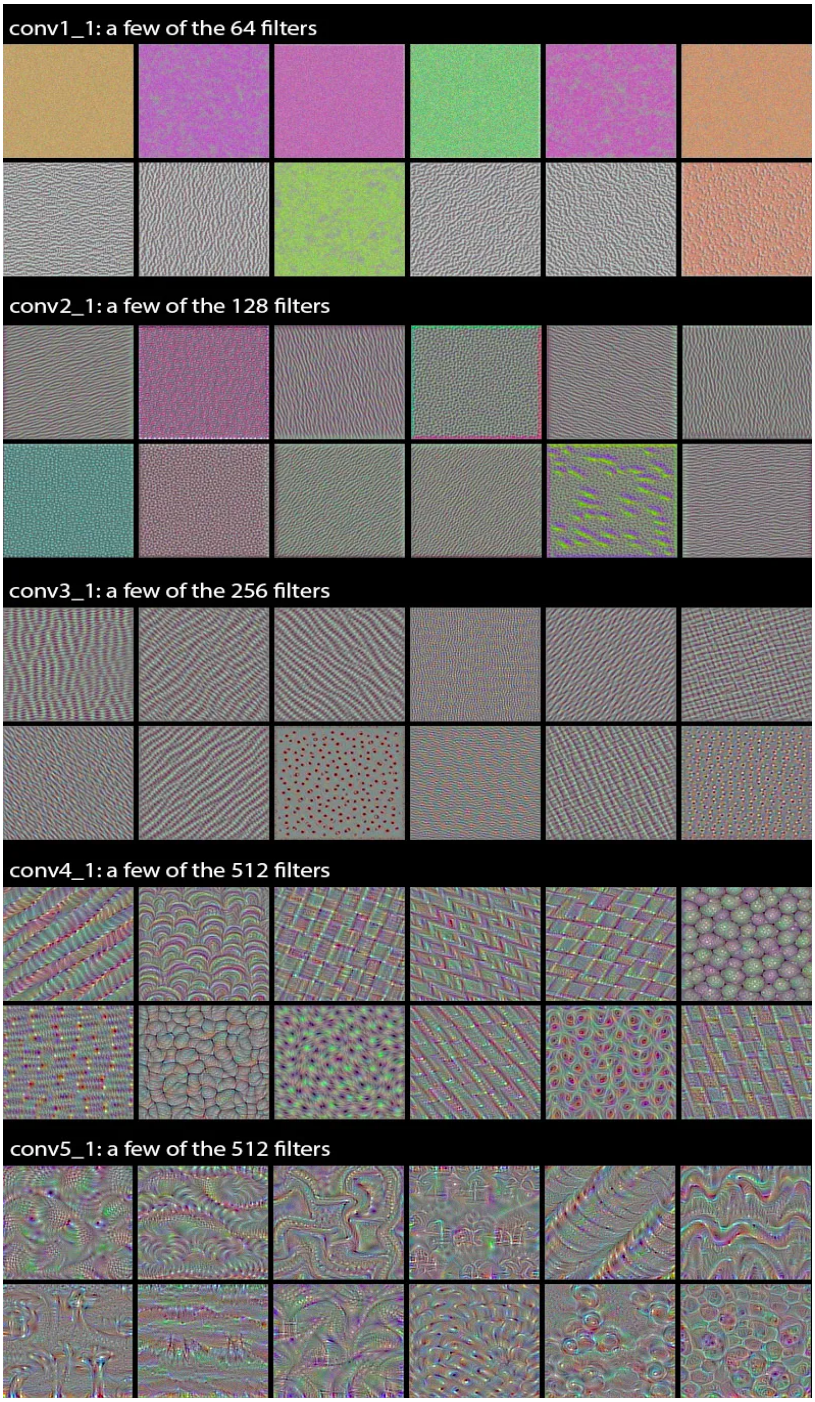


Рисунок 2.2 – Демонстрація ієрархії CNN від мережі VGG

Це полегшує перенесення контурів із зображення-контенту на змішане зображення, яке залежить від обраних шарів [25]. Використовуючи попередньо навчену нейронну мережу, таку як VGG-19, можна мінімізувати втрати в мережі, ввівши зображення, яке забезпечує вміст, картину з яскраво вираженими стильовими елементами як зображення стилю, а вихідне зображення – як випадкове зображення. Така мінімізація гарантує, що втрати стилю (втрати між стилем вихідного зображення і стилем зображення-стилю), втрати контенту (втрати між зображенням контенту і вихідним зображенням) і загальні варіаційні втрати (які забезпечують попіксельну гладкість) будуть мінімальними. У таких випадках зображення, отримане в результаті роботи мережі, нагадує вхідне зображення, а також містить стилістичні елементи зображення-еталона [26].

Потенційним підходом є обчислення матриці Грама (матриці, що включає корельовані ознаки) для тензорів, згенерованих стильовими шарами. Матриця Грама – це, по суті, матриця точкових добутків для векторів активацій ознак стильового шару. Якщо елемент Грам-матриці має значення, близьке до нуля, це означає, що дві ознаки в даному шарі не активуються в тандемі для відповідного стилю-образу. Якщо значення в Грам-матриці велике, то це означає, що для даного стиль-образу активуються одночасно дві ознаки. Наступним кроком є генерація змішаного зображення, яке відтворює цю схему активації стилю-образу. Функція втрати стилю дуже схожа на нашу функцію втрати вмісту, за винятком того, що вона обчислює середньоквадратичну похибку для матриць Грама замість вихідних тензорів шарів. Коли два зображення створюють однакову матрицю Грама в даному шарі, очікується, що вони мають однаковий стиль, але не обов’язково однаковий вміст (рис. 2.3). Застосування цього методу до ранніх шарів дає змогу виявити дрібніші текстури зображення, тоді як застосування його до глибших шарів дає змогу виявити більш високорівневі елементи стилю зображення.

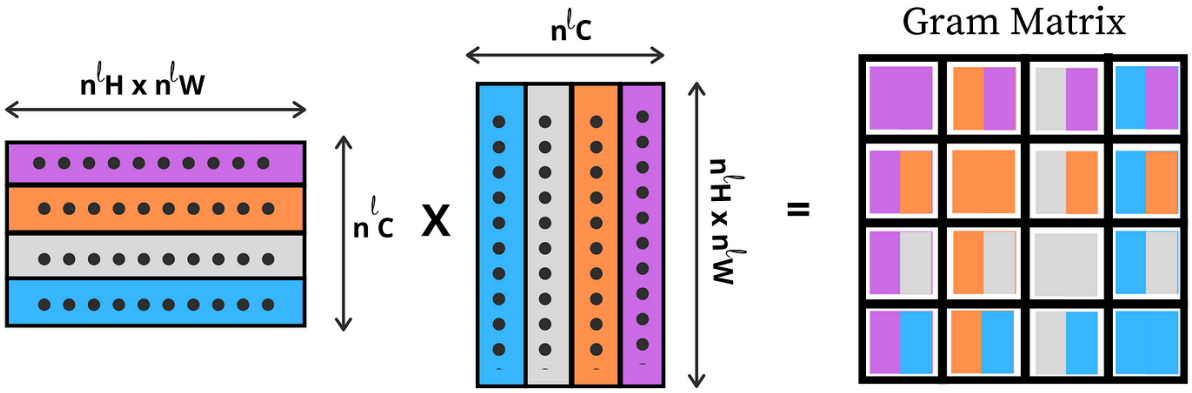


Рисунок 2.3 – Грам-матриця

2.2 GAN для перенесення стилю з одного зображення на інше

У 2014 році Гудфелоу і його колеги [27] представили модель неконтрольованого навчання для генеративної змагальної мережі (GAN), що складається з генеративної та дискримінативної мереж. Генеративна модель *G*, яка відображає розподіл даних. *G* приймає на вхід випадковий вектор шуму *z* і зіставляє його з вибіркою *G(z)*. Дискримінативна модель *D*, яка оцінює ймовірність того, що вибірка була отримана з навчальних даних, а не з *G*. *D* навчається для максимізації ймовірності присвоєння правильної мітки як навчальним прикладам, так і вибіркам з *G*. Процедура навчання *G* полягає в тому, щоб максимізувати ймовірність того, що *D* зробить помилку – тобто *G* прагне генерувати зразки, які максимально наближені до розподілу навчальних даних, «обманом» змушуючи *D* класифікувати свої виходи як реальні дані (рис. 2.4). Це представляється як мінімаксна гра двох гравців між *G* і *D*. У випадку, коли *G* і *D* є багатошаровими персептронами, вся система може бути навчена за допомогою зворотного поширення і стохастичного градієнтного спуску, без використання ланцюгів Маркова або наближеного виведення. Експерименти демонструють, що GAN можуть генерувати реалістичні зразки на базах даних MNIST, Toronto Face Database та CIFAR-10.

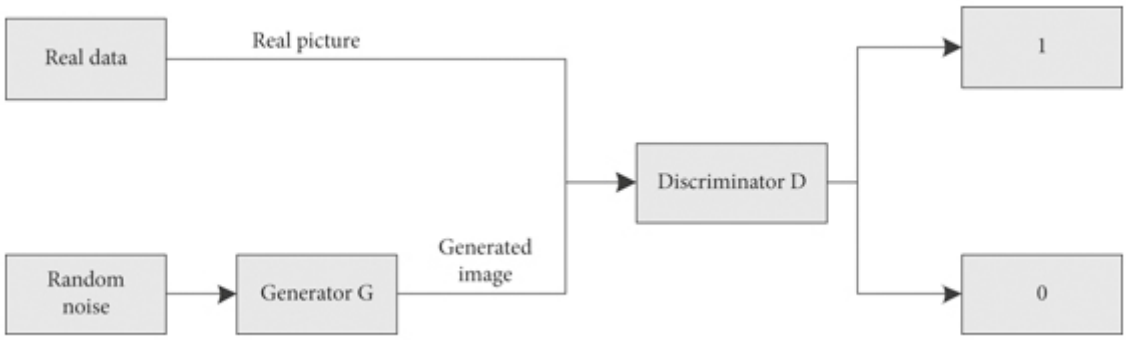


Рисунок 2.4 – Схема роботи мережі GAN

Серед ключових переваг GAN можна виділити наступні: уникнення наближеного виводу або нерозв’язних функцій розбиття під час навчання; ланцюги Маркова ніколи не потрібні для вибірки; сумісність з широким спектром архітектур генераторів/дискримінаторів; можна представляти різкі генеративні розподіли. Виклики включають в себе: відсутність явного представлення ймовірності; складність синхронізації *G* і *D* під час навчання.

У 2016 році Ізола та ін. представили модель pix2pix як репрезентативну роботу з перекладу зображення в зображення [28]. Модель використовує умовну генеративну змагальну мережу для зіставлення вхідного зображення з відповідним вихідним зображенням. Основна увага моделі зосереджена на навчанні функції відображення з необроблених пар вхід-вихід у неконтрольований спосіб. Результати показують, що модель перевершує різні сучасні методи в задачах перекладу зображення в зображення. Алгоритм використовує велику кількість парних зображень для керованого навчання і генерує мережу перекладу зображень «один до одного». Ця мережа виконує передачу стилю зображення з винятковими результатами. Хоча pix2pix дозволяє реалістично перетворювати зображення, його навчання потребує значної кількості даних парних зображень. Це обмеження обмежує просування та застосування моделі через дефіцит доступних наборів даних парних зображень у реальності (рис. 2.5).

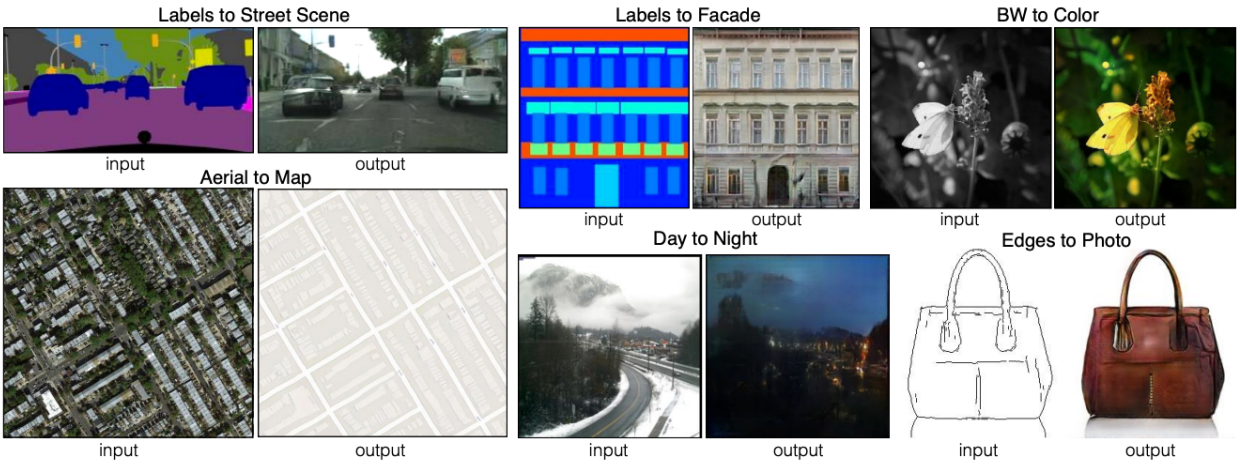


Рисунок 2.5 – Приклади роботи мережі pix2pix

У 2017 році Чжу та його колеги запропонували некеровану змагальну мережу CycleGAN [29], засновану на GAN, щоб подолати обмеження pix2pix. Передача стилю зображення в GAN передбачає перетворення одного типу зображення в інший, подібно до перекладу мови. CycleGAN усуває потребу в навчанні парних наборів даних, покладаючись виключно на генератор і дискримінатор для виконання перетворення домену зображення. Для кращого збереження структури вмісту зображень використовується послідовність циклів, яка обмежує і забезпечує їхню точність. CycleGAN отримує навчальну модель, навчаючись лише на двох типах вхідних зображень, що свідчить про широкий спектр її застосування.

CycleGAN – це мережева модель, заснована на GAN. На відміну від звичайної моделі GAN, де на вхід подається випадковий шум, CycleGAN використовує зображення як вхідний сигнал і має два GAN. Вона складається з двох генераторних мереж і двох дискримінаторних мереж D_X та D_Y. На відміну від моделі pix2pix, яка навчає лише одне парне зображення, CycleGAN досягає взаємного відображення між двома доменами зображень за допомогою генератора і дискримінатора, що дозволяє взаємне перетворення між двома типами стилізованих зображень (рис. 2.6).

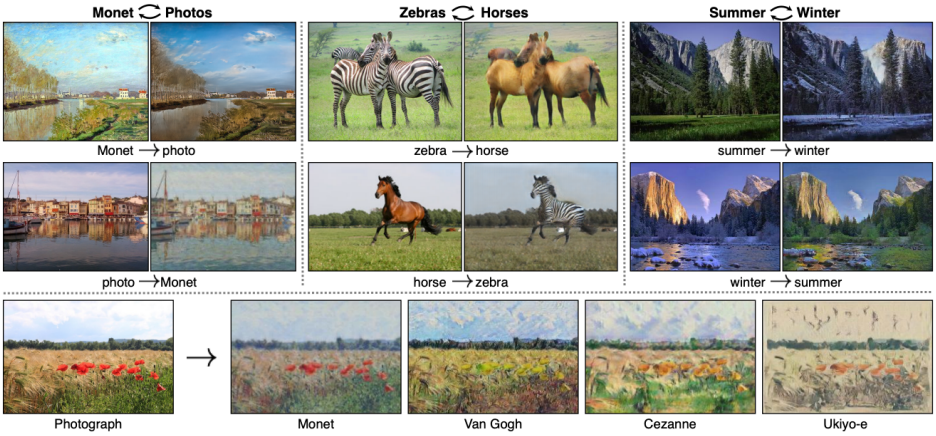


Рисунок 2.6 – Приклади роботи мережі CycleGAN

Припускаючи, що набір даних, який потрібно перетворити, включає типи X і Y, генератор перетворить зображення X-типу в зображення Y-типу і навпаки [30]. Дискримінатор D_X оцінює приналежність зображення до типу X, а дискримінатор D_Y оцінює приналежність зображення до типу Y (рис. 2.7).

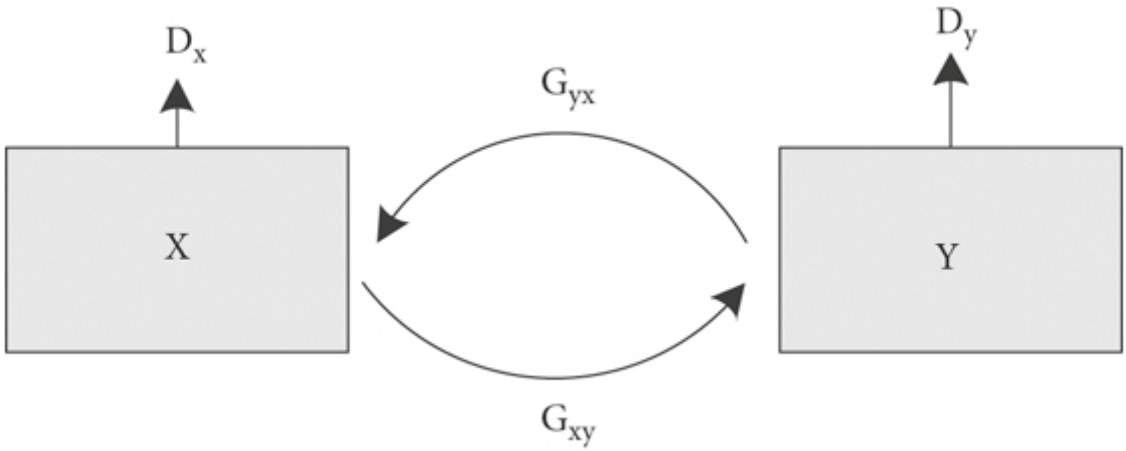


Рисунок 2.7 – Схема роботи мережі CycleGAN

Ключова ідея полягає в тому, щоб навчитися відображенням між двома візуальними областями X і Y, маючи непарні навчальні дані з кожної області. Мета полягає в тому, щоб навчити відображення G, яке переводить зображення з області X в область Y, і зворотне відображення F, яке переводить з Y в X. Щоб досягти цього, CycleGAN застосовує втрати, які навчають G і F

переводити зображення, які неможливо відрізнити від цільової області. Це допомагає ще більше впорядкувати відображення.

Повна навчальна задача поєднує в собі втрати суперника і втрати узгодженості циклу, щоб спонукати *G* і *F* транслювати зображення зі збереженням ключових атрибутів між входом і виходом. Демонструються переконливі візуальні результати перенесення стилю колекції, трансформації об'єктів, перенесення пори року та покращення фотографій. Важливо, що для цих завдань не існує парних навчальних даних. Кількісна оцінка на спарених даних для семантичного маркування та перекладу аерофотознімків показує, що CycleGAN досягає високої продуктивності, перевершуючи попередні методи перекладу без спарених даних.

Дослідження підтверджують важливість як втрат, пов'язаних з конкуренцією, так і втрат узгодженості циклу. Таким чином, CycleGAN пропонує ефективну загальну основу для непарного перекладу від зображення до зображення, що базується на змагальному навчанні з урахуванням втрат узгодженості циклу. Якісні та кількісні результати демонструють її можливості для навчання складних відображень без парних навчальних даних. CycleGAN може виконувати перенесення стилю зображення, навчаючись відображенням між двома різними візуальними областями – наприклад, картинами Моне і пейзажними фотографіями використовуючи непарні навчальні дані [31]. Він навчає два відображення: *G* для перекладу картин на фотографії та *F* для перекладу фотографій на стиль картин. Втрати в змаганні спонукають до того, щоб переклади не відрізнялися від реальних зображень у цільовій предметній області. Водночас, втрати узгодженості циклу призводять до того, що при перекладі в одну область і з неї відновлюється початковий вхідний код. Комбіновані втрати дозволяють навчати модель без парних даних, зберігаючи при цьому ключові атрибути між доменами, такі як колірна композиція. Для відображення картини на фото *G* використовується додаткова втрата ідентичності для подальшого збереження колірних характеристик. Навчений *G* діє як генератор передачі стилю

зображення – перетворюючи вхідні картини на пейзажні фотографії в стилі Моне. По суті, CycleGAN використовує змагальне та циклічне навчання для вивчення перенесення стилю з однієї візуальної області в іншу, що забезпечує потужне перенесення стилю без необхідності використання парних навчальних прикладів. Функція втрат CycleGAN складається з двох компонентів. Початковий компонент – це конкурентні втрати у звичайній мережі GAN. Другий компонент – це втрата узгодженості циклу, яка передбачає відображення X в Y, а потім реконструкцію назад до X, також відома як втрата при реконструкції (рис. 2.8).

Рисунок 2.8 – Перенесення стилю за допомогою CycleGAN

Крім CycleGAN, GAN має різні типові архітектури для передачі стилю зображень, такі як StarGAN і AttGAN. Pix2Pix вирішує проблему перетворення зображень один до одного, тоді як CycleGAN займається перетворенням між доменами. При перетворенні декількох доменів необхідно перенавчати модель для кожного з них.

У 2017 році Чой та його колеги представили свою інноваційну та адаптивну методику, відому як StarGAN [32], яка використовує єдину модель для проведення перетворень зображень між кількома доменами. Це стосується і перетворення атрибутів обличчя. Поряд зі StarGAN, Хе та ін. представили метод AttGAN для багатодоменної передачі атрибутів обличчя [33]. Метод AttGAN реалізує уніфікований фреймворк для передачі атрибутів подібний до StarGAN.

2.3 Функції втрат сприйняття

Функції втрат сприйняття (з англійської perceptual loss functions) – це термін у функції втрат, який заохочує природні та приємні для сприйняття результати [34]. Основна ідея використання втрат сприйняття для передачі стилю полягає в тому, що згорткові нейронні мережі, попередньо навчені класифікації зображень, вже навчилися кодувати семантичну та перцептивну інформацію в ознаках, витягнутих їхніми згортковими шарами. Замість того, щоб заохочувати вихідні зображення відповідати цільовим зображенням на рівні необроблених пікселів, втрати сприйняття дозволяють порівнювати зображення з точки зору цих попередньо навчених представлень ознак.

Зокрема, Джонсон та ін. визначають дві такі втрати сприйняття – втрати при реконструкції ознак і втрати при реконструкції стилю. Втрати на реконструкцію ознак вимірюють відмінності у змісті між зображеннями, порівнюючи їхні активації ознак з певного згорткового шару фіксованої мережі VGG16, попередньо навченої на ImageNet. Вона обчислює

нормалізовану евклідову відстань між картами ознак, витягнутими з вихідного і цільового зображень. Мінімізація цих втрат призводить до того, що вихідні зображення є дуже близькими до цільового, зберігаючи загальну просторову структуру та вміст, але допускаючи варіації кольору та текстури. Втрати при реконструкції стилю спрямовані на узгодження стилю або текстурних патернів між зображеннями [35].

Для цього порівнюються матриці Грама, розраховані на основі карт ознак вихідного і стильового зображень. Матриця Грама фіксує кореляції між каналами ознак, що представляють ті ознаки, активації яких мають тенденцію до співпадіння. Зіставляючи матриці Грама, мережа повинна синтезувати текстури і візерунки на виході, які призводять до подібних кореляцій ознак, що і зображення стилю.

$$G_j^{\phi}(x)_{c,c'} = \frac{1}{C_j H_j W_j} \sum_{h=1}^{H_j} \sum_{w=1}^{W_j} \phi_j(x)h,w,c\phi_j(x)h,w,c' \,. \qquad (2.1)$$

Це передає стилістичні характеристики без обмеження просторового розташування. Разом ці втрати сприйняття дозволяють відокремлювати і рекомбінувати інформацію про зміст і стиль з природних зображень.

Втрати при реконструкції функцій зберігають ключову структуру контенту цільового зображення, тоді як втрати при реконструкції стилю переносять патерни текстур із зображення стилю. Налаштовуючи їхні відносні ваги, вихідне зображення може гнучко інтерполювати між цільовими зображеннями контенту та стилю в просторі сприйняття ознак.

Порівняно з попередніми підходами на основі оптимізації, які ітеративно оновлювали пікселі для мінімізації втрат сприйняття, ключова перевага полягає в тому, що Джонсон та ін. навчають мережу прямого перетворення безпосередньо вирішувати цю оптимізацію за один прямий прохід. Це майже так само ефективно узгоджує статистику ознак, але на кілька порядків прискорює роботу. Точне налаштування перцептивних втрат для

передачі стилю дозволяє передавати семантичні знання з мережі втрат для керування процесом синтезу зображення (рис. 2.9).

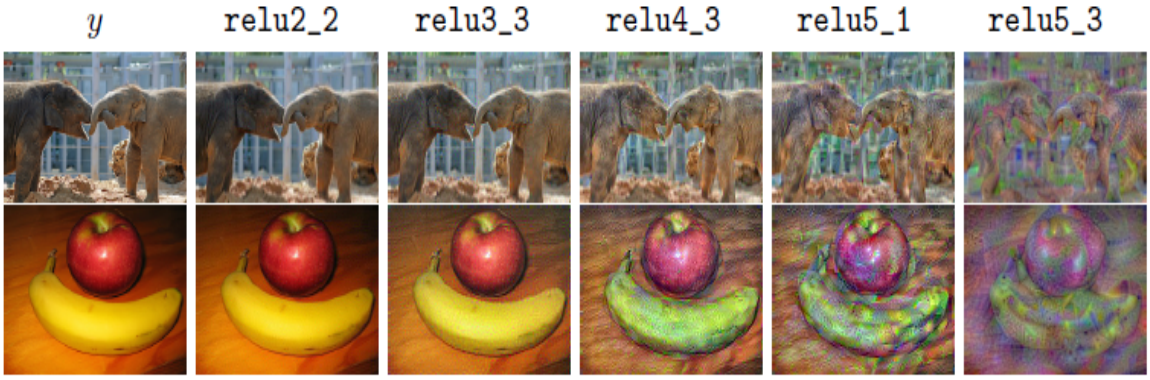


Рисунок 2.9 – Оптимізація для мінімізації втрат при реконструкції ознак

Під час багатошарової реконструкції зображень зберігається основний зміст і загальна просторова структура зображення, але не зберігаються колір, текстура і точна форма [36].

3 РЕАЛІЗАЦІЯ МОДЕЛЕ ДЛЯ ПЕРЕНЕСЕННЯ СТИЛЮ

3.1 Вибір технологій для розробки застосунку

Для розробки моделі нейронного стильового трансферу було обрано мову програмування Python та такі ключові бібліотеки.

OpenCV – популярна бібліотека комп'ютерного зору з відкритим кодом, що містить функції для обробки зображень та відео. Вона буде використана для завантаження вхідних зображень, перетворення їх розміру та візуалізації результатів.

PyTorch – бібліотека машинного навчання для побудови та тренування нейронних мереж. Вона використовує динамічні графи обчислень, що спрощує побудову складних архітектур з циклами і гілками. Всі дані подані у PyTorch у вигляді тензорів – багатовимірних масивів, над якими легко проводити обчислення. Також бібліотека автоматично обчислює градієнти для зворотнього поширення помилки. PyTorch добре інтегрується з екосистемою Python та має відкритий код і активну спільноту. Завдяки цим властивостям, PyTorch дозволяє швидко прототипувати й масштабувати нейронні мережі будь-якої архітектури та складності. Дозволить реалізувати архітектуру моделі стиль-трансферу на базі VGG-19 [37].

Torchvision – колекція попередньо навчених моделей та наборів даних на базі PyTorch. З її допомогою можна легко ініціалізувати VGG-19 з вагами, навченими на ImageNet.

NumPy – популярна бібліотека для роботи з багатовимірними масивами та матрицями, що часто використовується в задачах машинного навчання разом з PyTorch.

Отже, поєднання цих бібліотек надасть усі необхідні інструменти для швидкої розробки та тренування глибокої нейронної мережі для перенесення стилю між зображеннями.

3.2 Художнє перенесення стилю за допомогою CNN

Для реалізації перенесення стилю будемо будувати модель на основі VGG-19 (рис. 3.1).

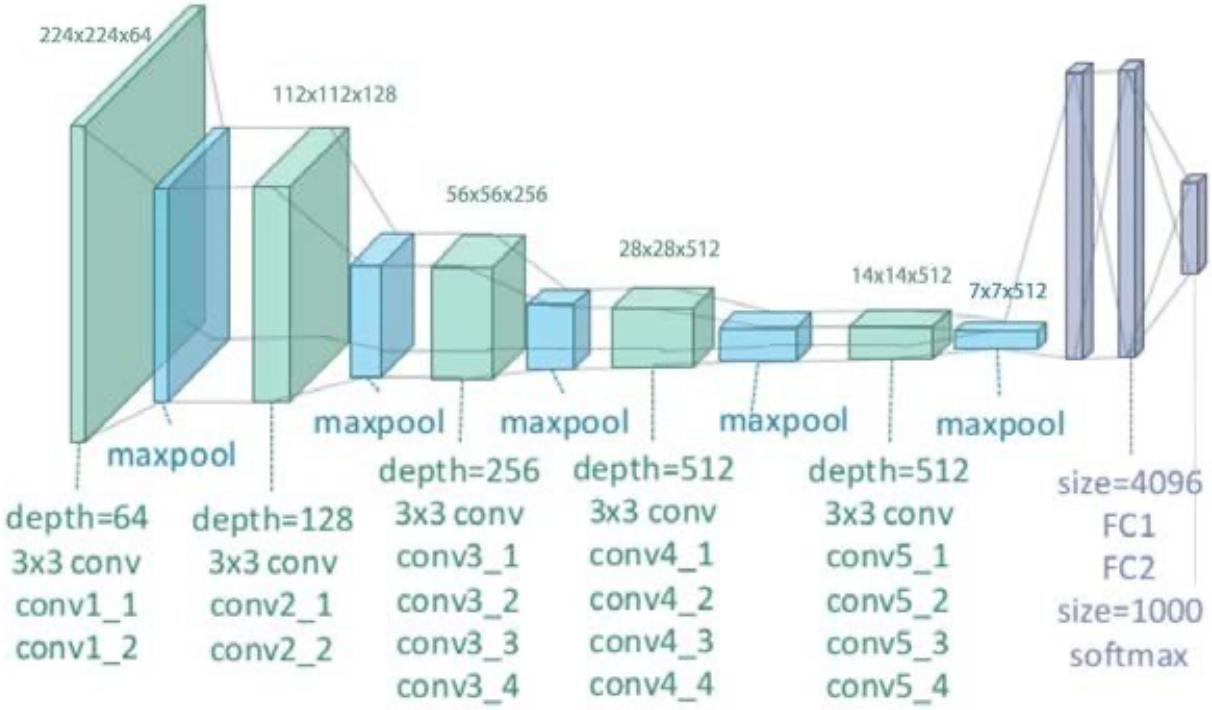


Рисунок 3.1 – Архітектура VGG-19

Попередні згорткові шари в архітектурі VGG-19, що використовуються для нейронного перенесення стилів, відповідають за вилучення інформації про стиль із вхідного зображення. Зокрема, шари Conv1_1, Conv2_1, Conv3_1, Conv4_1 і Conv5_1 захоплюють низькорівневі представлення характеристик, пов’язані з текстурами, кольорами і візуальними патернами. Вони реагують на характеристики стилю, такі як мазки пензля або геометричні фігури. Зіставляючи проміжні результати цих згорткових фільтрів із зображенням стилю, нейронна модель перенесення стилю може кодувати художній стиль, знайдений у прикладі зображення. Пізніші, глибші згорткові шари VGG-19 реагують на вміст вищого рівня, наприклад, об’єкти та сцени. Саме ці шари використовуються для представлення вмісту вхідного зображення: Conv4_2 і

Conv5_2. Використовуючи представлення стилю і контенту, отримані на різних етапах обробки VGG-19, модель може окремо маніпулювати стилем і контентом зображень, щоб створювати художньо візуалізовані результати, які поєднують обидва ці елементи. Налаштування внеску шарів стилю та контенту є ключовим для досягнення бажаного ефекту перенесення.

Завантажимо картинки за допомогою OpenCV. Після цього треба конвертувати зображення з формату OpenCV BGR в RGB. Також має сенс зменшити зображення, тому реалізована зміна висоти та ширини.

Лістинг 3.1 Функція завантаження зображення:

```
def load_image(img_path, target_shape=None):
    # [:, :, ::-1] BGR (opencv format) into RGB
    img = cv.imread(img_path)[:, :, ::-1]
    if target_shape is not None:
        if isinstance(target_shape, int) and target_shape != -1:
            current_height, current_width = img.shape[:2]
            new_height = target_shape
            new_width = int(current_width * (new_height / current_height))
            img = cv.resize(img, (new_width, new_height),
interpolation=cv.INTER_CUBIC)
        else:
            img = cv.resize(img, (target_shape[1], target_shape[0]),
interpolation=cv.INTER_CUBIC)
    img = img.astype(np.float32)
    img /= 255.0
    return img
```

Оскільки архітектура нейронної мережі VGG-19 була навчена на зображеннях фіксованого розміру 256 на 256 пікселів, то перед подачею вхідних зображень в модель стиль-трансферу необхідно попередньо виконати

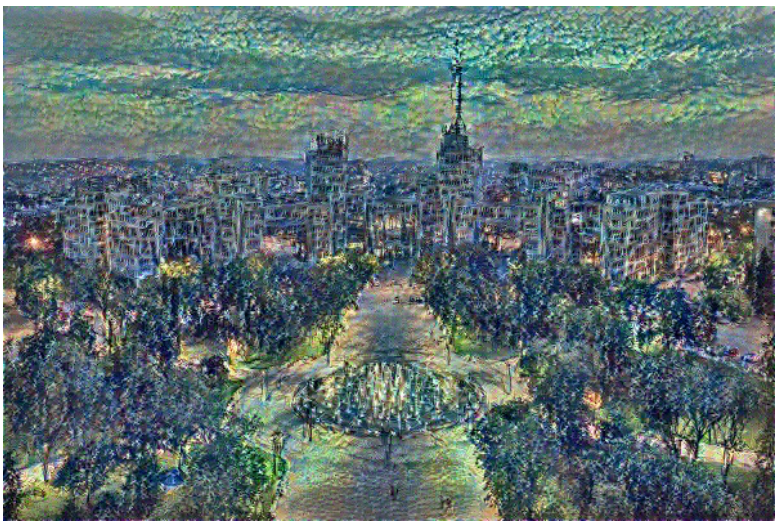
їх масштабування до цього розміру за допомогою відповідних методів комп’ютерного зору. Це дозволить коректно екстрагувати ознаки стилю та контенту з внутрішніх шарів нейронної мережі та досягти якісного перенесення стилю на цільове зображення.

Лістинг 3.2 Функція нормалізації зображення:

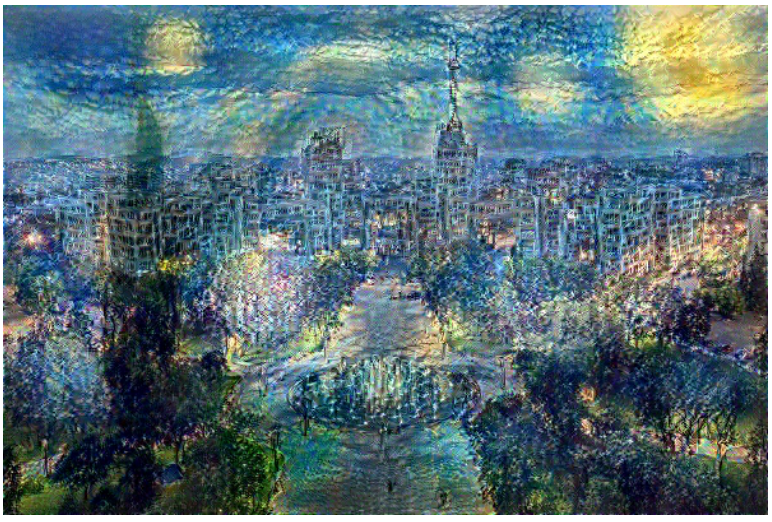
```
def prepare_img(img_path, target_shape, device):
    img = load_image(img_path, target_shape=target_shape)

    transform = transforms.Compose([
        transforms.ToTensor(),
        transforms.Lambda(lambda x: x.mul(255)),
        transforms.Normalize(mean=IMAGENET_MEAN_255,
std=IMAGENET_STD_NEUTRAL)
    ])

    img = transform(img).to(device).unsqueeze(0)

    return img
```

Далі оголошується змінна, яка буде зберігати зображення яке буде оптимізуватись за допомогою нейронного перенесення стилю. Тензор містить початкове зображення. Початковим зображення може бути як випадкове шумове зображення, так і зображення контента або стилю, яке буде ітеративно оновлюватися. Як показують експерименти найкраще зображення виходить якщо відштовхуватися від зображення контенту (рис. 3.2). Також обернувши його у змінну і встановивши *requires_grad* у True, ми додамо цьому зображенню градієнти, які можуть сприяти оптимізації. Зокрема, прапорець *requires_grad* встановлює вимогу градієнтів.

а)



б)



в)

Рисунок 3.2 – Результат роботи з різним базовим зображення:

а) контент; б) білий шум; в) стиль

При передачі у нейронну мережу для перенесення стилю PyTorch автоматично обчислюватиме градієнти відносно цього вхідного зображення. Оновлюється значення пікселів, щоб мінімізувати втрати вмісту і стилю, тому вхідне зображення є «параметром, що навчається», який градієнтний спуск може налаштовувати так само, як ваги моделі. Зрештою, значення пікселів у optimizing_img будуть ітеративно зміщуватися, щоб відповідати бажаним представленням стилю та контенту з шарів VGG19 за допомогою порівняння втрат і зворотного поширення. Перетворення вхідного зображення на змінну зі значенням *requires_grad=True* дозволяє оптимізувати його для передачі художнього стилю.

Переходячи до вираховування цільового представлення контенту та стилю з вхідних зображень для подальшого трансферу стилю. *content_feature_maps* та *style_feature_maps* – це функції активації з різних шарів нейромережі VGG19 відповідно для контенту і стилю. Далі *target_content_representation* витягує представлення контенту з відповідного шару, стискаючи його в вектор. А *target_style_representation* обчислює грам-матриці для шарів, які відповідають за стиль – в них міститься інформація про стиль зображення. Наприкінці зберігаємо разом цільові ознаки контенту і стилю в *target_representations*, які далі використовуватимуться при оптимізації перенесення стилю за допомогою VGG19.

Лістинг 3.3 Вираховування цільового представлення контенту:

```
content_feature_maps = neural_net(content_img)
style_feature_maps = neural_net(style_img)

target_content_representation = content_feature_maps[content_feature_maps_index_name[0]].squeeze(axis=0)
target_style_representation = [utils.gram_matrix(x) for cnt, x in enumerate(style_feature_maps) if cnt in style_feature_maps_indices_names[0]]
```

target_representations = [target_content_representation, target_style_representation]

Грам матриця містить інформацію про кореляції між функціями активації різних позицій карт ознак. Це дозволяє оцінити текстурні і стилістичні патерни зображення.

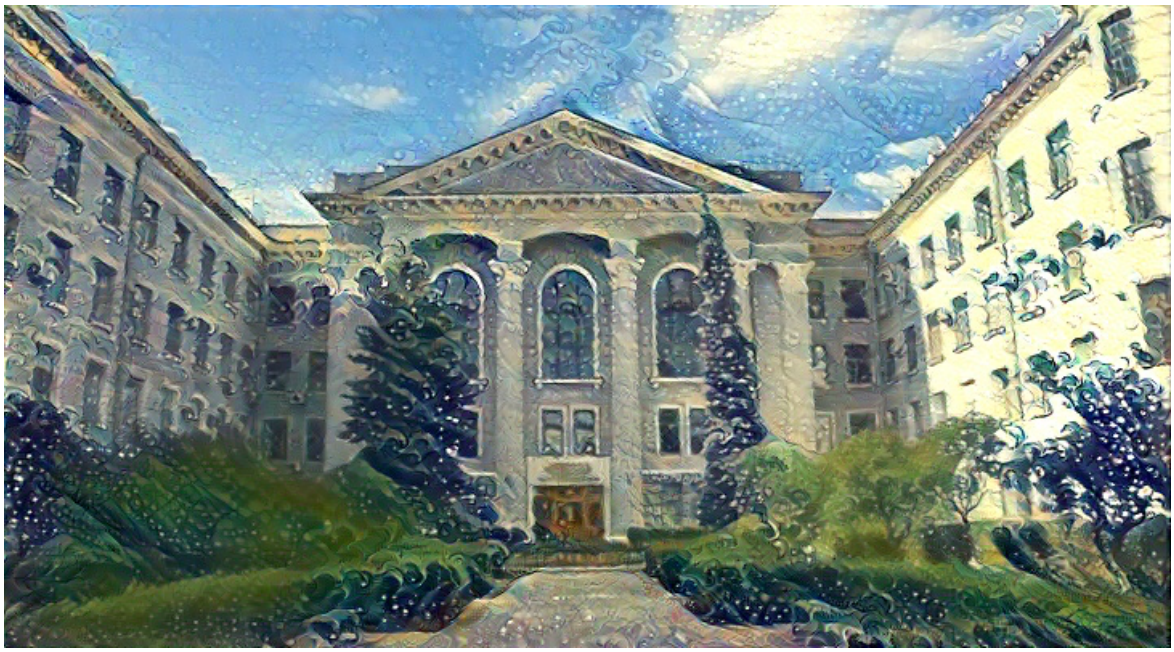
Лістинг 3.4 Функція реалізації грам матриці:

```
def gram_matrix(x, should_normalize=True):
    (b, ch, h, w) = x.size()
    features = x.view(b, ch, w * h)
    features_t = features.transpose(1, 2)
    gram = features.bmm(features_t)
    if should_normalize:
        gram /= ch * h * w
    return gram
```

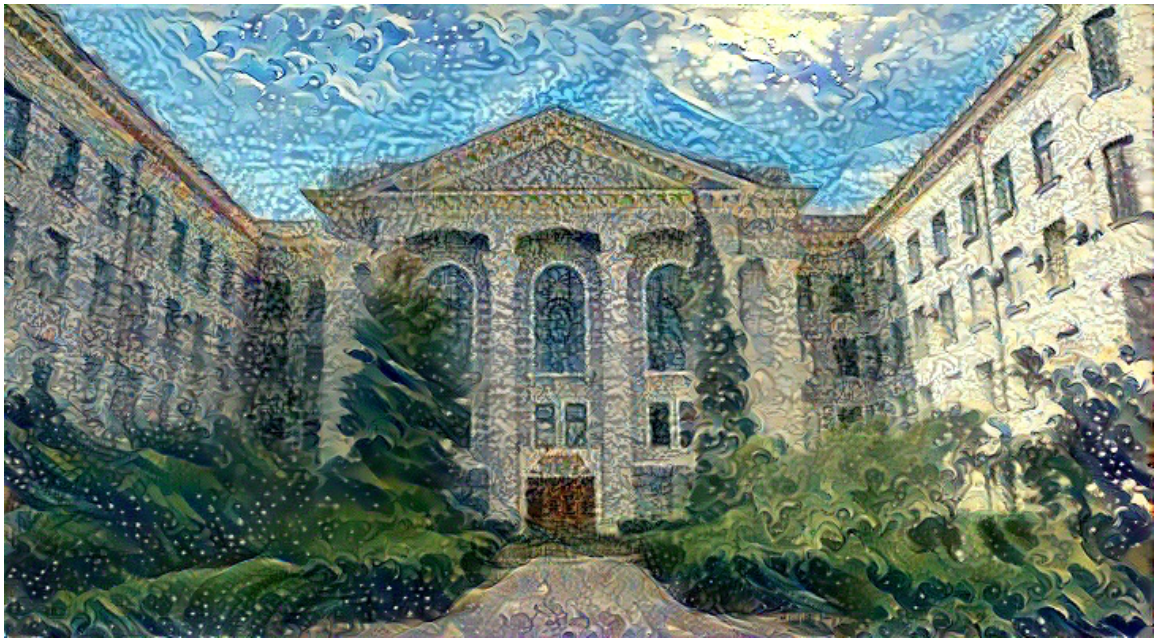
Порівнюючи грам матриці стильового зображення і зображення на яке переносимо стиль, ми можемо кількісно оцінити наскільки гарно перенесено текстури та стиль.

В якості оптімайзера було порівняно LBFGS [38] та Adam [39]. Оптимізатор LBFGS належить до класу методів квазі-Ньютона, які використовують інформацію про градієнт для пошуку локального мінімуму функції втрат. Він добре працює на невеликих наборах даних і потребує мало пам’яті, але не підходить для великих задач. З іншого боку, Adam – це адаптивний градієнтний спуск з вбудованою підтримкою моментів для прискорення навчання. Він краще масштабується на великі об’єми даних, але може бути менш стабільним. Для нейронного перенесення стилю на високоякісні зображення зазвичай краще підходить Adam – він швидше збігається за прийнятну кількість ітерацій. Але LBFGS все ще може давати

кращі результати на невеликих зображеннях, де його недоліки проявляються менше. Отже обидва оптимізатори мають свої переваги залежно від даних. Експерименти показують, візуально оптимайзер LBFGS дає більш приємні зображення (рис. 3.3).



а)



б)

Рисунок 3.3 – Результат роботи з різними оптимайзерами:
а) LBFGS; б) Adam

Лістинг 3.5 Робота за оптимайзером Adam:

```
optimizer = Adam((optimizing_img,), lr=1e1)

tuning_step = make_tuning_step(
        neural_net, optimizer, target_representations,
        content_feature_maps_index_name[0],
        style_feature_maps_indices_names[0],
        config
)

for cnt in range(num_of_iterations[config['optimizer']]):
    total_loss, content_loss, style_loss, tv_loss = tuning_step(optimizing_img)
```

Лістинг 3.6 Робота за оптимайзером LBFGS:

```
optimizer = LBFGS(
        (optimizing_img,),
        max_iter=num_of_iterations['lbfgs'],
        line_search_fn='strong_wolfe'
)
cnt = 0

def closure():
    nonlocal cnt
    if torch.is_grad_enabled():
        optimizer.zero_grad()
    total_loss, content_loss, style_loss, tv_loss = build_loss(
        neural_net,
        optimizing_img,
```

```
            target_representations,
            content_feature_maps_index_name[0],
            style_feature_maps_indices_names[0],
            config
        )
        if total_loss.requires_grad:
            total_loss.backward()
        cnt += 1
        return total_loss

    optimizer.step(closure)
```

Проводячи експерименти можна побачити, що навчання з допомогою Adam оптимайзера відбувається повільніше, аніж LBFGS (рис. 3.4).

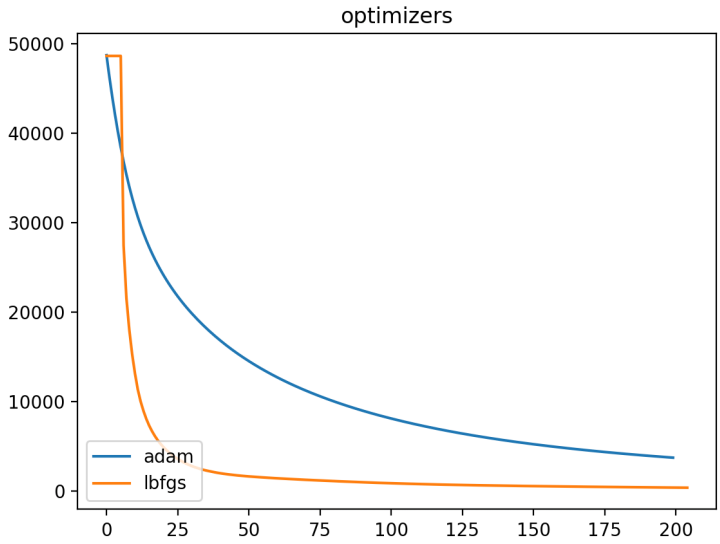


Рисунок 3.4 – Графік навчання нейромережі з різними оптимайзерами

Також можна порівняти яка різниця при реконструкції контенту при використанні архітектур VGG-16 та VGG-19 (рис. 3.5).

а) б)

Рисунок 3.5 – Результат роботи різних архітектур при однакових втратах: а) VGG-16; б) VGG-19

Як показують експерименти VGG-16 відпрацьовує краще під час реконструкції зображення з білого шуму.

Розглянемо відмінності між використанням convolutional та relu шарів нейронної мережі VGG19 для обчислення грам-матриць в методах перенесення стилю. Convolutional шари безпосередньо екстрагують просторові ознаки з вхідного зображення за допомогою великої кількості фільтрів. Вони вловлюють низькорівневі текстурні характеристики та кольорові переходи. А функції активації ReLU, які йдуть після convolutional шарів, вносять нелінійність та відсікають негативні значення. Отже, використання саме convolutional шарів для побудови грам-матриць дає більш безпосереднє уявлення про розподіл низькорівневих ознак стилю по зображенню. А relu шари зсувають це представлення у позитивну область, втрачаючи частину інформації (рис. 3.6). Однак на практиці, іноді використання саме relu шарів для грам-матриць призводить до кращого перенесення глобальних стилістичних особливостей. А convolutional представлення часто підходить для локального перенесення текстур (рис. 3.7).

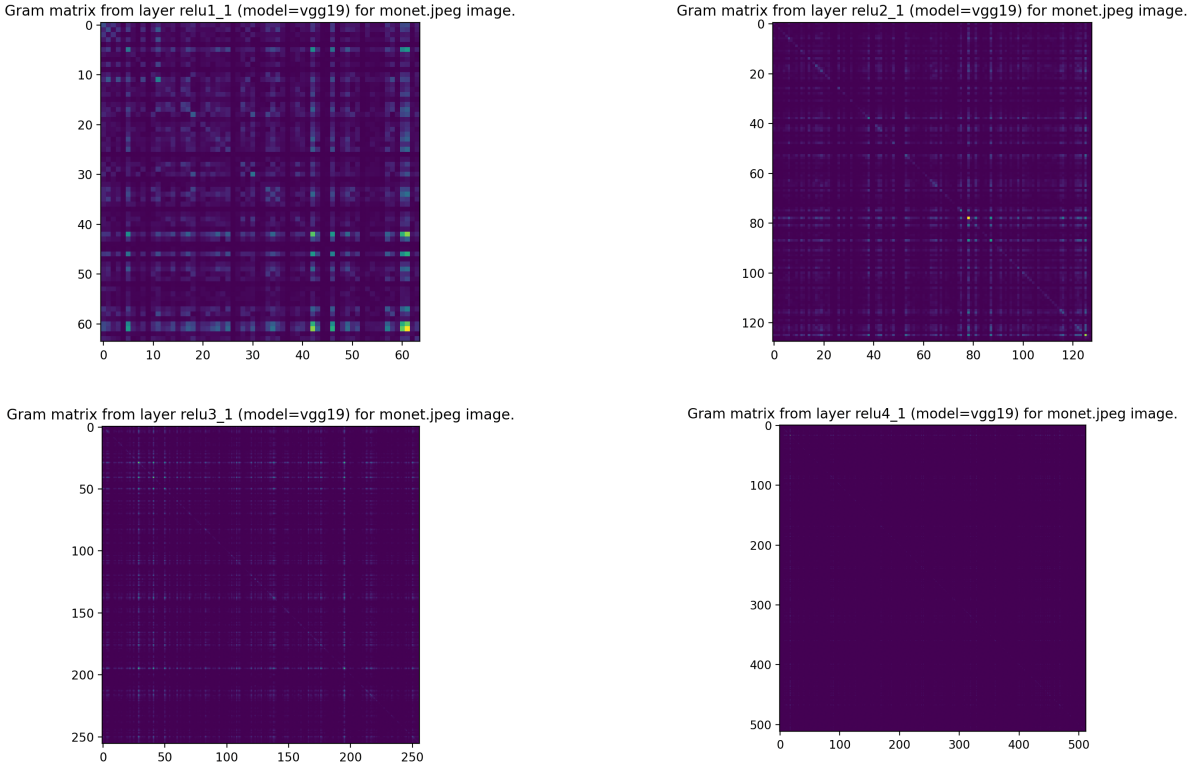


Рисунок 3.6 – Грам-матриці для relu шарів

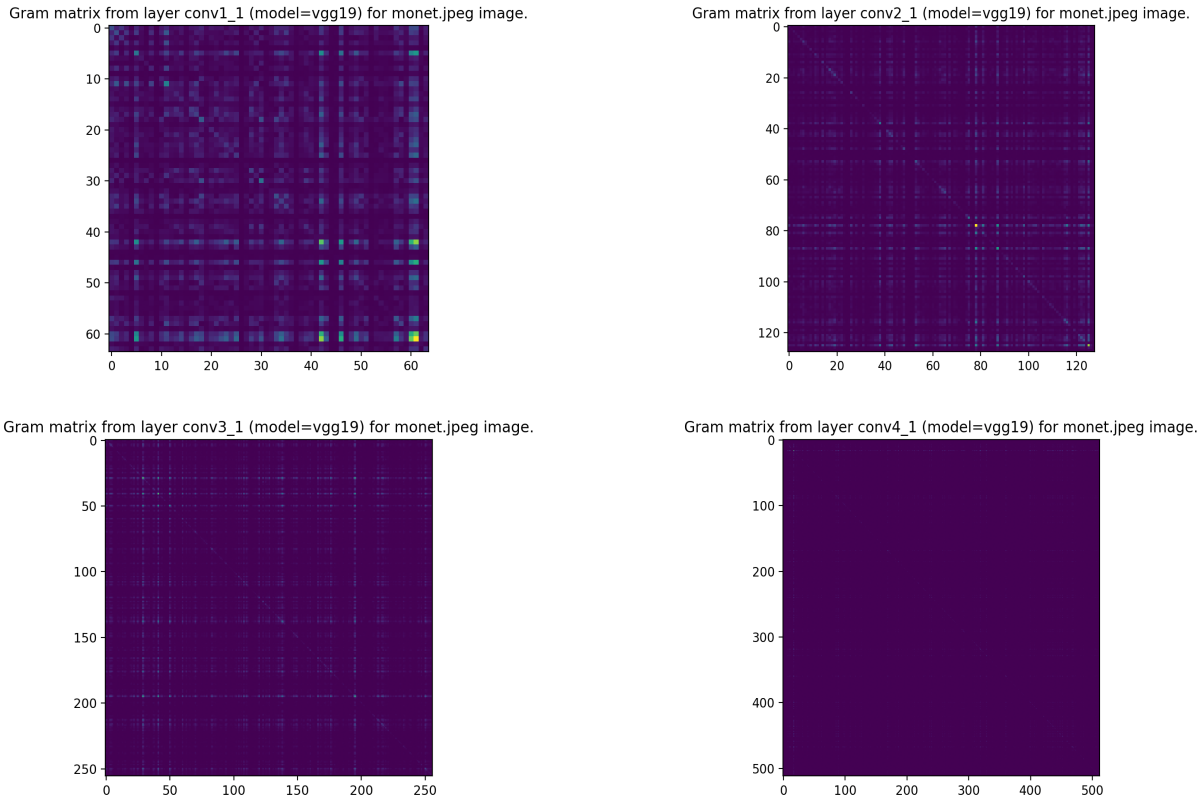
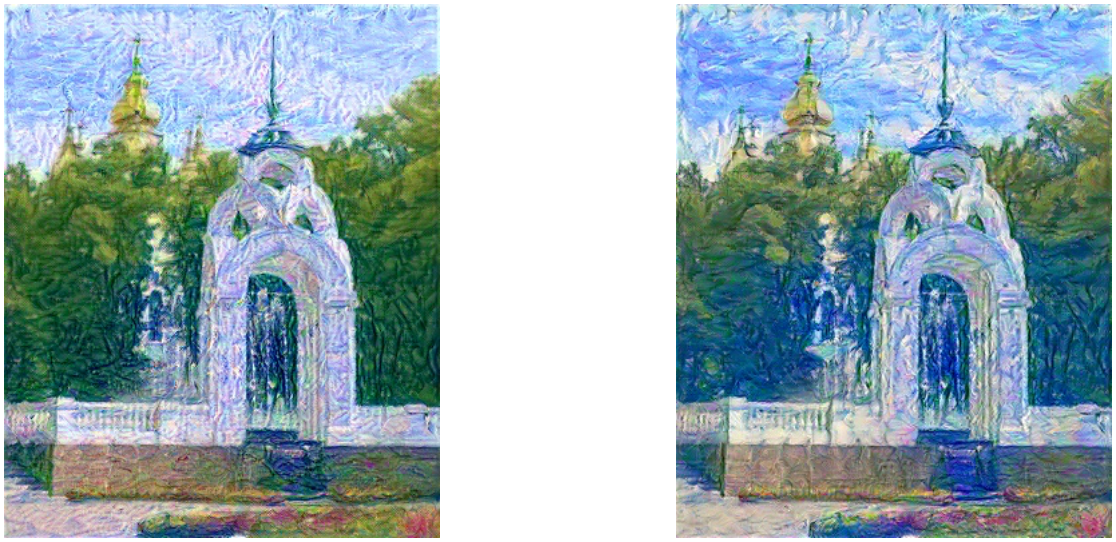


Рисунок 3.7 – Грам-матриці для conv шарів

При тому що грам матриці виходять досить схожими – вибір які шари використовувати впливає на фінальне зображення (рис. 3.8).



а) б)

Рисунок 3.8 – Застосування стилю з використанням різних шарів мережі: а) conv; б) relu

Як можна побачити при використанні relu шарів на зображення більше впиливає кольорова гамма стилю. Відповідно якщо при накладанні стилю ми хочемо залишити кольорову гамму оригінального зображення краще буде вибір conv шарів.

Функція *build_loss* відіграє ключову роль в алгоритмі нейронного перенесення стилю, адже саме вона визначає цільову функцію, мінімізація якої і призводить до трансферу стилю з одного зображення на інше. Спочатку через нейронну мережу *neural_net* пропускається поточна версія оптимізованого зображення *optimizing_img*, отримуючи набір активацій проміжних шарів – карт ознак. Це необхідно для подальшого розрахунку втрат контенту та стилю. Далі з поточних карт ознак витягується інформація про контент у вигляді вектору *current_content_representation*. Порівнюючи його з цільовим вектором контенту *target_content_representation* через функцію втрат *MSELoss*, отримуємо значення *content_loss*. Чим ближчі ці вектори – тим

менша втрата контенту. Аналогічно з карт ознак обчислюються поточні грам–матриці *style_representation* і порівнюються з цільовими грам–матрицями *style_representation*. Знову ж за допомогою *MSELoss* рахується *style_loss*, який треба мінімізувати. І наприкінці всі складові втрат зважуються коефіцієнтами і сумуються в *total_loss* – саме її оптимізація методом градієнтного спуску і призводить до перенесення стилю на зображення.

Лістинг 3.7 Функція build_loss:

```
def build_loss(neural_net, optimizing_img, target_representations, content_feature_maps_index, style_feature_maps_indices, config):
    target_content_representation = target_representations[0]
    target_style_representation = target_representations[1]

    current_set_of_feature_maps = neural_net(optimizing_img)

    current_content_representation = current_set_of_feature_maps[content_feature_maps_index].squeeze(axis=0)
    content_loss = torch.nn.MSELoss(reduction='mean')(target_content_representation, current_content_representation)

    style_loss = 0.0
    current_style_representation = [utils.gram_matrix(x) for cnt, x in enumerate(current_set_of_feature_maps) if cnt in style_feature_maps_indices]
    for gram_gt, gram_hat in zip(target_style_representation, current_style_representation):
        style_loss += torch.nn.MSELoss(reduction='sum')(gram_gt[0], gram_hat[0])
    style_loss /= len(target_style_representation)
```

```
tv_loss = utils.total_variation(optimizing_img)

total_loss = config['content_weight'] * content_loss +
config['style_weight'] * style_loss + config['tv_weight'] * tv_loss

return total_loss, content_loss, style_loss, tv_loss
```

3.3 Фотореалістичне перенесення стилю за допомогою GAN

Вибір архітектури GAN для перенесення стилю між зображеннями значною мірою залежить від характеру та обсягу навчальних даних. Якщо є достатня кількість пар зображень відповідних стилів для навчання, можна використати підхід pix2pix з U-Net архітектурою генератора та патч-базованою дискримінацією. Це забезпечить стабільне навчання та детальне перенесення стилю. Але в багатьох випадках таких пар не існує і GAN доводиться тренувати на непозначених даних – окремих наборах зображень у потрібних стилях. Тут краще підійде CycleGAN з двома генераторами та обмеженням циклічності. Його можна реалізувати на базі ResNet для кращого поширення градієнтів.

Для навчання було обрано датасет Summer2Winter Yosemite. Датасет Summer2Winter Yosemite містить пари зображень парку Йосеміті в Каліфорнії, зроблені влітку та взимку. Він часто використовується для тренування генеративно-змагальних мереж, зокрема CycleGAN, для перенесення стилю між порами року. Датасет містить 1273 пари зображень, де одне зображення з пари було зроблене влітку, а друге – того ж ракурсу, але взимку. Роздільна здатність зображень становить 256×256 пікселів. Така пара зображень дозволяє навчити CycleGAN перетворювати літні пейзажі парку Йосеміті у зимові і навпаки. Мережі навчаються захоплювати візуальні особливості різних пір року – сніг, оголені дерева, кольори неба тощо. Завдяки

достатньому об’єму даних та чітким відмінностям між класами, датасет добре підходить для демонстрації можливостей CycleGAN в перенесенні стилю між зображеннями (рис. 3.9).

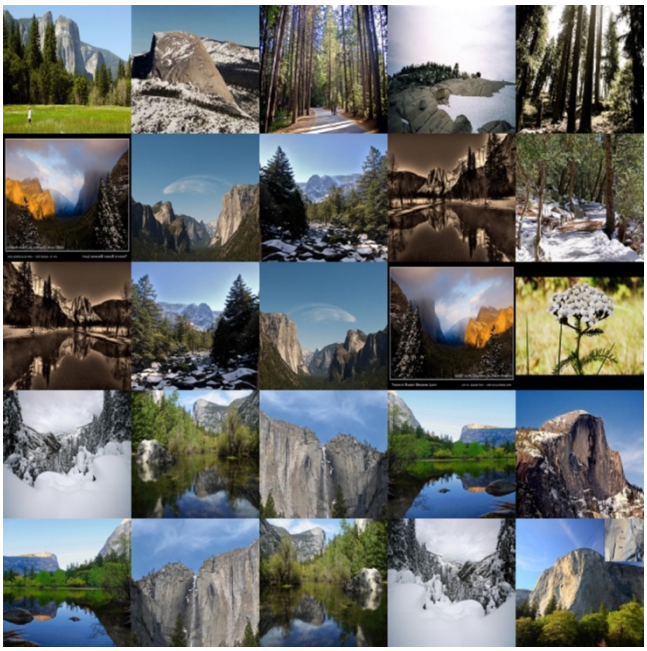


Рисунок 3.9 – Датасет Summer2Winter Yosemite

CycleGAN складається з двох конкуруючих генераторів та дискримінаторів, які тренуються разом. Перший генератор *G* перетворює зображення з домену *X* (наприклад, літні фото) у зображення домену *Y* (зимові фото). Другий генератор *F* робить зворотне перетворення – з зимових фото у літні. Дискримінатор *DX* намагається відрізнити реальні зимові зображення від штучно згенерованих *G*. Аналогічно дискримінатор *DY* класифікує літні зображення як справжні або штучні (створені *F*). Особливістю CycleGAN є наявність циклічних зв’язків між генераторами, які примушують *G* і *F* бути оборотними. Наприклад, перетворивши взимку в літо за допомогою *G*, ми можемо повернути зимовий стиль через *F*. Це покращує якість перенесення стилю. Таким чином архітектура CycleGAN дозволяє за допомогою непозначених даних навчити мережі переходу між різними візуальними доменами.

ResNet архітектуру (рис. 3.10) можна успішно застосувати як базу для генератора в CycleGAN. Використання залишкових з'єднань дозволяє побудувати більш глибоку мережу, полегшуючи поширення градієнтів під час тренування [40]. Наявність шорткатів сприяє збереженню просторової інформації на глибоких рівнях, що важливо для генерації реалістичних зображень.

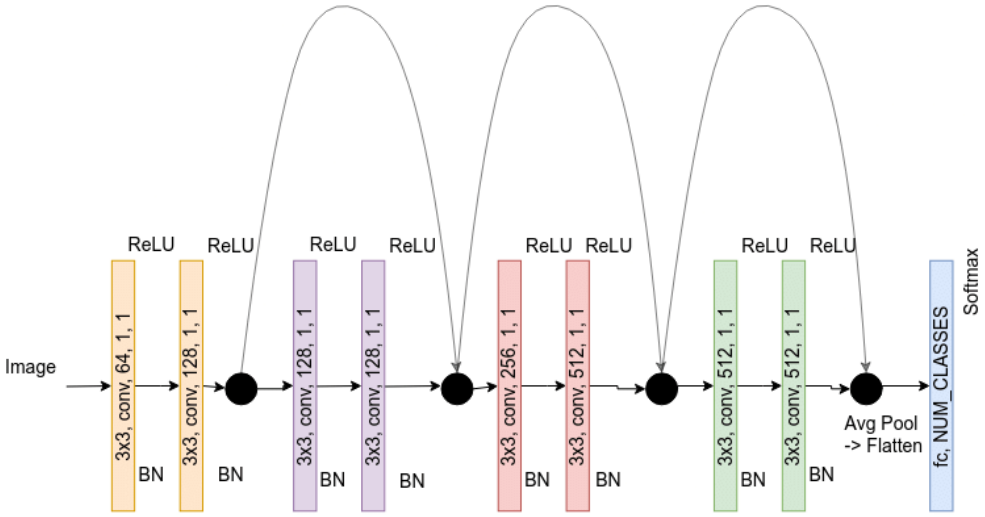


Рисунок 3.10 – Архітектура ResNet 9

Крім того, ResNet має менше параметрів порівняно з іншими мережами подібної глибини, отже швидше оптимізується. Можна ініціалізувати генератор вагами з попередньо навченої ResNet для подальшого дооптимізування під конкретне перенесення стилю. Особливо ефективним є використання багатоканальних шарів на вході для вилучення просторових ознак зображень. Таким чином ResNet надає гарну основу для потужного генератора в CycleGAN.

Лістинг 3.8 ResNet генератор:

```
class ResnetGenerator(nn.Module):
    def __init__(self, input_nc=3, output_nc=3, ngf=64,
norm_layer=nn.BatchNorm2d, use_dropout=True, num_blocks=6):
```

```
        super(ResnetGenerator, self).__init__()

        if type(norm_layer) == functools.partial:
            use_bias = norm_layer.func == nn.InstanceNorm2d
        else:
            use_bias = norm_layer == nn.InstanceNorm2d

        res_model = [nn.ReflectionPad2d(3),
                     conv_norm_relu(input_nc, ngf * 1, 7, norm_layer=norm_layer,
bias=use_bias),
                     conv_norm_relu(ngf * 1, ngf * 2, 3, 2, 1,
norm_layer=norm_layer, bias=use_bias),
                     conv_norm_relu(ngf * 2, ngf * 4, 3, 2, 1,
norm_layer=norm_layer, bias=use_bias)]
for i in range(num_blocks):
            res_model += [ResidualBlock(ngf * 4, norm_layer, use_dropout,
use_bias)]

        res_model += [dconv_norm_relu(ngf * 4, ngf * 2, 3, 2, 1, 1,
norm_layer=norm_layer, bias=use_bias),
                      dconv_norm_relu(ngf * 2, ngf * 1, 3, 2, 1, 1,
norm_layer=norm_layer, bias=use_bias),
                      nn.ReflectionPad2d(3),
                      nn.Conv2d(ngf, output_nc, 7),
                      nn.Tanh()]
        self.res_model = nn.Sequential(*res_model)

    def forward(self, x):
        return self.res_model(x)
```

Ця модель реалізує генератор зображень на основі архітектури ResNet. Спочатку відбувається ініціалізація базових шарів: Convolution-BatchNorm-ReLU, які виконують екстракцію низькорівневих ознак Ngf – це кількість фільтрів в шарах, що зростає від 64 до 256, тобто створюється воріння для подальшої обробки ознак Далі йде послідовність з 6 Residual блоків (за замовчуванням). Вони реалізують основну обробку ознак та використовують шорткати для полегшення оптимізації. Після Residual блоків відбувається зворотня трансформація за допомогою деконволюційних шарів В кінці екстраговані ознаки перетворюються в RGB зображення бажаного розміру за допомогою Conv2d та функції активації Tanh.

Дискримінатор є важливою складовою GAN, оскільки саме він навчає генератор створювати реалістичні зображення. Архітектура N Layer дозволяє побудувати дискримінатор із довільною кількістю шарів. Спочатку визначається базова послідовність шарів: Convolution-LeakyReLU, яка екстрагує примітивні ознаки.

Лістинг 3.9 N Layered дискрімінатор:

```
class NLayerDiscriminator(nn.Module):
   def __init__(self, input_nc, ndf=64, n_layers=3,
norm_layer=nn.BatchNorm2d, use_bias=False):
      super(NLayerDiscriminator, self).__init__()
      dis_model = [nn.Conv2d(input_nc, ndf, kernel_size=4, stride=2,
padding=1), nn.LeakyReLU(0.2, True)]
      nf_mult = 1
      nf_mult_prev = 1
      for n in range(1, n_layers):
         nf_mult_prev = nf_mult
         nf_mult = min(2**n, 8)
```

```
            dis_model += [conv_norm_lrelu(ndf * nf_mult_prev, ndf * nf_mult,
        kernel_size=4, stride=2, norm_layer= norm_layer, padding=1,
        bias=use_bias)]
            nf_mult_prev = nf_mult
            nf_mult = min(2**n_layers, 8)
            dis_model += [conv_norm_lrelu(ndf * nf_mult_prev, ndf * nf_mult,
        kernel_size=4, stride=1, norm_layer= norm_layer, padding=1,
        bias=use_bias)]
            dis_model += [nn.Conv2d(ndf * nf_mult, 1, kernel_size=4, stride=1,
        padding=1)]

            self.dis_model = nn.Sequential(*dis_model)
        def forward(self, input):
            return self.dis_model(input)
```

В циклі додаються шари, що подвоюють кількість фільтрів кожен раз, реалізуючи ідею перетворення. NF_mult обмежує максимальну кількість фільтрів значенням 8. У такий спосіб реалізується багатошарова архітектура, яка може навчитися виявляти ознаки різного ступеню складності. Останній шар дискримінатора робить бінарне рішення: зображення реальне чи штучне. Такий підхід дає можливість гнучко контролювати глибину дискримінатора в GAN.

Маючи архітектуру для генератора і дискримінатора можна створити модель для CycleGAN.

Лістинг 3.10 Модель CycleGAN:

```
class CycleGAN(object):
    def __init__(self,args):
        self.Gab = define_Gen(input_nc=3, output_nc=3, ngf=args.ngf,
    netG=args.gen_net, norm=args.norm,
```

```
                                    use_dropout= not args.no_dropout,
gpu_ids=args.gpu_ids)
        self.Gba = define_Gen(input_nc=3, output_nc=3, ngf=args.ngf,
netG=args.gen_net, norm=args.norm,
                                    use_dropout= not args.no_dropout,
gpu_ids=args.gpu_ids)
        self.Da = define_Dis(input_nc=3, ndf=args.ndf, netD= args.dis_net,
n_layers_D=3, norm=args.norm, gpu_ids=args.gpu_ids)
        self.Db = define_Dis(input_nc=3, ndf=args.ndf, netD= args.dis_net,
n_layers_D=3, norm=args.norm, gpu_ids=args.gpu_ids)

        utils.print_networks([self.Gab,self.Gba,self.Da,self.Db],
['Gab','Gba','Da','Db'])
        self.MSE = nn.MSELoss()
        self.L1 = nn.L1Loss()

        self.g_optimizer =
torch.optim.Adam(itertools.chain(self.Gab.parameters(),self.Gba.parameters()
), lr=args.lr, betas=(0.5, 0.999))
        self.d_optimizer =
torch.optim.Adam(itertools.chain(self.Da.parameters(),self.Db.parameters()),
lr=args.lr, betas=(0.5, 0.999))

        self.g_lr_scheduler =
torch.optim.lr_scheduler.LambdaLR(self.g_optimizer,
lr_lambda=utils.LambdaLR(args.epochs, 0, args.decay_epoch).step)
        self.d_lr_scheduler =
torch.optim.lr_scheduler.LambdaLR(self.d_optimizer,
lr_lambda=utils.LambdaLR(args.epochs, 0, args.decay_epoch).step)
```

Спочатку ініціалізуються два генератори *Gab* і *Gba*, що відповідають за прямий і зворотній напрямки перенесення стилю. Також створюються два дискримінатори *Da* і *Db* для кожного домену зображень окремо. Далі для цих модулів задаються окремі оптимізатори на базі Adam з різними швидкостями навчання та розкладами зменшення швидкості навчання (рис. 3.11).

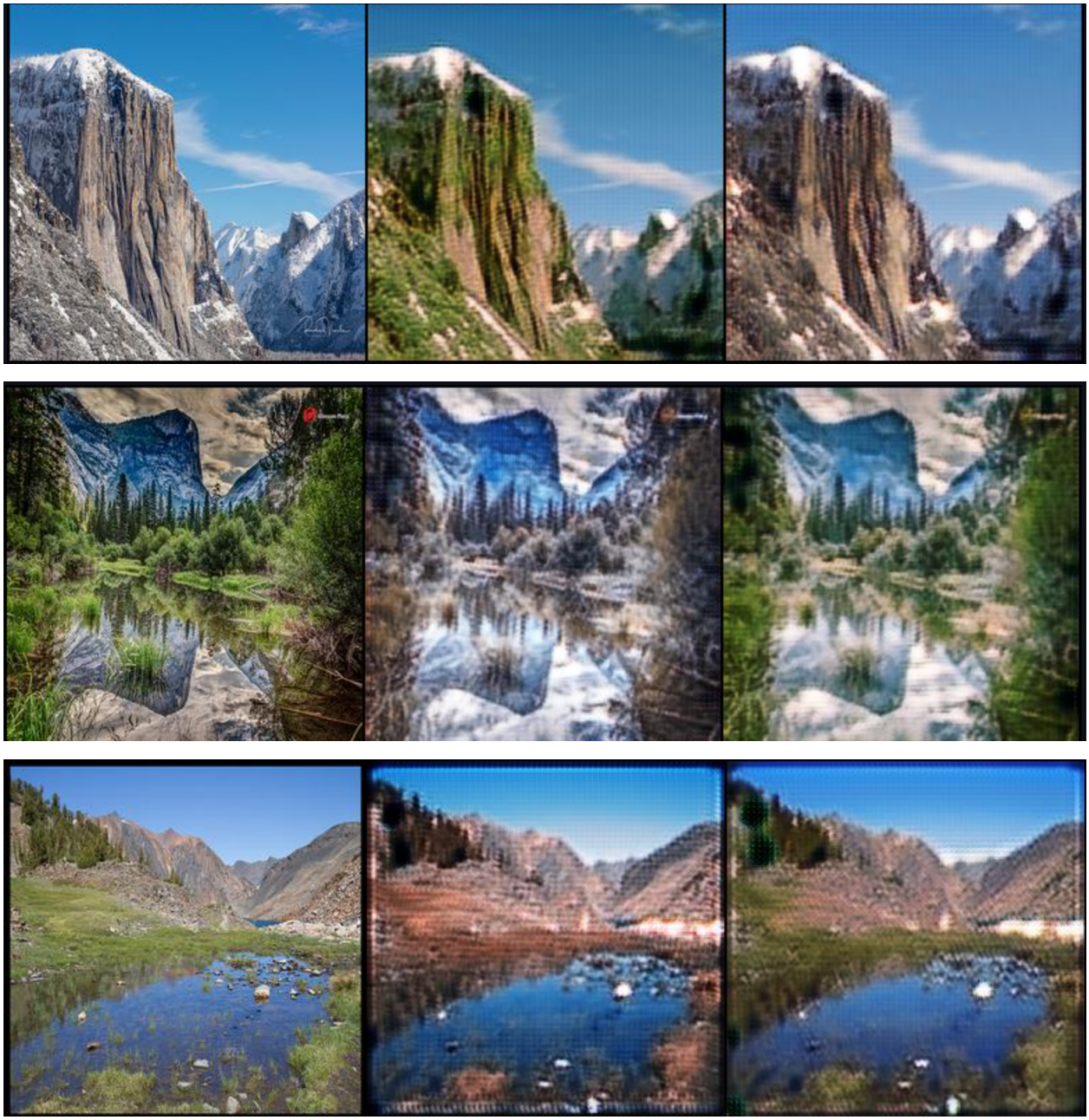


Рисунок 3.11 – Результати роботи описаної реалізації CycleGAN (оригінал; трансформація з початкового домена в другий; трансформація з згенерованої фотографії в початковий домен)

ВИСНОВКИ

У рамках кваліфікаційної роботи були реалізовані методи перенесення характеристик з одного зображення на інше.

Процес передачі стилю в комп'ютерному зорі дозволяє перетворити вміст зображення в стиль іншого зображення. Проте збереження семантичної цілісності оригінального зображення може бути складним завданням, оскільки важливі компоненти, такі як обличчя або текст, можуть бути спотворені або розмиті. Крім того, нейронна передача стилю вимагає значних обчислювальних ресурсів і часу. Крім того, може бути складно регулювати ступінь і місце перенесення стилю. Хоча перенесення стилю має потенційне застосування в таких галузях, як сучасне мистецтво, дизайн одягу та реклама, його використання має певні обмеження. Воно може спотворювати або створювати нереалістичні образи, змінювати оригінальні естетичні аспекти мистецтва або змінювати значення зображень, символів, об'єктів чи сцен. У деяких випадках це може навіть вплинути на емоційне або контекстуальне значення твору мистецтва, потенційно підриваючи системи вірувань або історичну цінність. Дослідження в цій галузі є цінними завдяки їх практичному застосуванню в таких нових секторах, як віртуальна та доповнена реальність, ігри та графічний дизайн. Застосунки для перенесення стилю у віртуальну реальність наразі перебувають на стадії дослідження, однак вони демонструють багатообіцяючу здатність підвищувати ступінь занурення. Facebook і Google досліджують можливості використання технології передачі стилю для трансформації візуальних оповідань у застосунках віртуальної реальності (VR), що дозволить розробникам продемонструвати свій творчий потенціал в іграх та інших застосунках. Такий підхід може розширити доступ до художнього самовираження для тих, кого зазвичай не визнають митцями.

Перенесення візуальних характеристик між зображеннями є активною сферою досліджень, в якій розроблено низку методів для таких застосувань,

як перенесення стилю, розфарбування, надвисока роздільна здатність і міждоменний переклад зображень. Одна з найвідоміших нещодавніх парадигм навчає глибокі нейронні мережі передавати атрибути між зображеннями, використовуючи втрати сприйняття та перетворення у просторі ознак, а не оперуючи безпосередньо у піксельному просторі.

Перенесення стилю означає рендеринг зображення, який переймає стиль або візуальні патерни текстури іншого зображення, зберігаючи при цьому ключові структури контенту. Фундаментальна робота Gatys та ін. започаткувала підхід на основі оптимізації, який ітеративно оновлює значення пікселів відповідно до багатошарових статистичних даних, таких як матриці Грама, витягнуті з попередньо навчених згорткових мереж. Незважаючи на гнучкість, ця ітеративна оптимізація виявилася дуже дорогою в обчислювальному плані. Подальші методи були спрямовані на підвищення ефективності. Вони включають навчання мереж прямого перенесення стилю з використанням втрат сприйняття, застосування шаблонів стилю з обміном патчів або перетворенням ознак, а також багатомасштабну обробку. Поточні виклики в галузі перенесення стилів включають ефективну обробку довільних нових стилів і зображень контенту, оцінку якості стилізації та визначення того, які просторові подібності найкраще зберігають структуру.

Під час розфарбовування вхідні зображення у відтінках сірого алгоритмічно розфарбовуються на основі еталонних зображень. Методи варіюються від поширення кольорів між відповідними ділянками до використання глибинних ознак і змагального навчання для перетворення колекцій напівтонових зображень у кольорові. Для надвисокої роздільної здатності створення реалістичних деталей з високою роздільною здатністю, які відповідають, але не представлені безпосередньо у вхідних даних з низькою роздільною здатністю, залишається складним завданням. Використання глибокої перцептивної схожості є більш перспективним, ніж покладання лише на реконструкцію пікселів. Для різних застосувань важливе

значення має адаптація моделей і втрат, щоб зосередитися на особливостях, що мають відношення до сприйняття.

Генеративні змагальні мережі можуть переводити зображення з однієї візуальної області в іншу, керуючись навчальними прикладами з кожної області, в таких застосунках, як сезонні зміни та передача архітектурного стилю рендерингу. А умовні нейронні мережі можуть передавати структурні та семантичні атрибути, такі як розташування, поза, форма та ідентичність, керуючись прикладами зображень.

Майбутні напрямки передачі атрибутів зображень включають розширення сфер використання, таких як відео та 3D-рендеринг, покращення контролю користувача над переданими атрибутами та розробку оптимізованих моделей, придатних для практичного розгортання. Також залишаються відкриті питання оцінки якості, поглиблення розуміння того, як моделі переносять атрибути, обробки різноманітних стилів і контенту, а також дослідження того, які просторові подібності найкраще зберігають структуру сприйняття під час перенесення. Простір для інновацій в алгоритмах, застосунках і теорії глибокого перекладу зображень на основі людського сприйняття залишається багатим.

Результати дослідження апробовано у вигляді тез доповідей під час VI Міжнародна науково-практична конференція «Methodical and practical methods of creating inventions» [41].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Tammina, S. (2019). Transfer learning using vgg-16 with deep convolutional neural network for classifying images. International Journal of Scientific and Research Publications (IJSRP), 9(10), 143-150.

2. Ruder, M., Dosovitskiy, A., & Brox, T. (2018). Artistic style transfer for videos and spherical images. International Journal of Computer Vision, 126(11), 1199-1219.

3. Jing, Y., Yang, Y., Feng, Z., Ye, J., Yu, Y., & Song, M. (2019). Neural style transfer: A review. IEEE transactions on visualization and computer graphics, 26(11), 3365-3385.

4. Gatys, L. A., Ecker, A. S., Bethge, M., Hertzmann, A., & Shechtman, E. (2017). Controlling perceptual factors in neural style transfer. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3985-3993).

5. Nguyen, A., Yosinski, J., & Clune, J. (2019). Understanding neural networks via feature visualization: A survey. Explainable AI: interpreting, explaining and visualizing deep learning, 55-76.

6. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

7. Singh, A., Jaiswal, V., Joshi, G., Sanjeeve, A., Gite, S., & Kotecha, K. (2021). Neural style transfer: A critical review. IEEE Access, 9, 131583-131613.

8. Johnson, J., Alahi, A., & Fei-Fei, L. (2016). Perceptual losses for real-time style transfer and super-resolution. In Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14 (pp. 694-711). Springer International Publishing.

9. Qassim, H., Verma, A., & Feinzimer, D. (2018, January). Compressed residual-VGG16 CNN model for big data places image recognition. In 2018 IEEE 8th annual computing and communication workshop and conference (CCWC) (pp. 169-175). IEEE.

10. Gatys, L. A., Ecker, A. S., & Bethge, M. (2015). A neural algorithm of artistic style. arXiv preprint arXiv:1508.06576.

11. Agarap, A. F. (2018). Deep learning using rectified linear units (relu). arXiv preprint arXiv:1803.08375.

12. Dumoulin, V., Shlens, J., & Kudlur, M. (2016). A learned representation for artistic style. arXiv preprint arXiv:1610.07629.

13. Huang, X., & Belongie, S. (2017). Arbitrary style transfer in real-time with adaptive instance normalization. In Proceedings of the IEEE international conference on computer vision (pp. 1501-1510). Goodfellow, I. J., Mirza, M., Xu, B., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative Adversarial Networks. ArXiv. /abs/1406.2661

14. Karras, T., Laine, S., & Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 4401-4410).

15. Deng, Y., Tang, F., Dong, W., Huang, H., Ma, C., & Xu, C. (2021, May). Arbitrary video style transfer via multi-channel correlation. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 35, No. 2, pp. 1210-1217).

16. Chandran, P., Zoss, G., Gotardo, P., Gross, M., & Bradley, D. (2021). Adaptive convolutions for structure-aware style transfer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 7972-7981).

17. Sheng, L., Lin, Z., Shao, J., & Wang, X. (2018). Avatar-net: Multi-scale zero-shot style transfer by feature decoration. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 8242-8250).

18. De Almeida, M. C., Asada, E. N., & Garcia, A. V. (2008). On the use of Gram matrix in observability analysis. IEEE Transactions on Power Systems, 23(1), 249-251.

19. Vulimiri, P. S., Deng, H., Dugast, F., Zhang, X., & To, A. C. (2021). Integrating geometric data into topology optimization via neural style transfer. Materials, 14(16), 4551.

20. Vishwakarma, D. K. (2019, March). A state-of-the-arts and prospective in neural style transfer. In 2019 6th International Conference on Signal Processing and Integrated Networks (SPIN) (pp. 244-247). IEEE.

21. Joseph, M., Richard, J., Halim, C. S., Faadhilah, R., & Qomariyah, N. N. (2021, August). Recreating Traditional Indonesian Batik with Neural Style Transfer in AI Artistry. In 2021 International Conference on ICT for Smart Society (ICISS) (pp. 1-8). IEEE.

22. Pomazan V., Tvoroshenko I., and Gorokhovatskyi V. (2023) Development of an application for recognizing emotions using convolutional neural networks, International Journal of Academic Information Systems Research, 7(7), pp. 25-36.

23. Luan, F., Paris, S., Shechtman, E., & Bala, K. (2017). Deep photo style transfer. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4990-4998).

24. Elad, M., & Milanfar, P. (2017). Style transfer via texture synthesis. IEEE Transactions on Image Processing, 26(5), 2338-2351.

25. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., Gadetska S., and Al-Dhaifallah M. (2023) Statistical data analysis models for determining the relevance of structural image descriptions, IEEE Access, 11, pp. 126938-126949.

26. Li, Y., Fang, C., Yang, J., Wang, Z., Lu, X., & Yang, M. H. (2017). Universal style transfer via feature transforms. Advances in neural information processing systems, 30.

27. Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1125-1134).

28. Zhu, J. Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In Proceedings of the IEEE international conference on computer vision (pp. 2223-2232).

29. Choi, Y., Choi, M., Kim, M., Ha, J. W., Kim, S., & Choo, J. (2018). Stargan: Unified generative adversarial networks for multi-domain image-to-image

translation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 8789-8797).

30. Harms, J., Lei, Y., Wang, T., Zhang, R., Zhou, J., Tang, X., ... & Yang, X. (2019). Paired cycle-GAN-based image correction for quantitative cone-beam computed tomography. Medical physics, 46(9), 3998-4009.

31. Manandhar, S., Naikodi, S., VL, M. V., & AN, M. Z. (2023). Generating Monet Style Images Using DCGAN Based CycleGAN.

32. He, Z., Zuo, W., Kan, M., Shan, S., & Chen, X. (2019). Attgan: Facial attribute editing by only changing what you want. IEEE transactions on image processing, 28(11), 5464-5478.

33. Nikulin, Y., & Novak, R. (2016). Exploring the neural algorithm of artistic style. arXiv preprint arXiv:1602.07188.

34. Johnson, J., Alahi, A., & Fei-Fei, L. (2016). Perceptual losses for real-time style transfer and super-resolution. In Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14 (pp. 694-711). Springer International Publishing.

35. Liu, H., Michelini, P. N., & Zhu, D. (2018, August). Artsy-gan: A style transfer system with improved quality, diversity and performance. In 2018 24th International Conference on Pattern Recognition (ICPR) (pp. 79-84). IEEE.

36. Gatys, L. A., Ecker, A. S., & Bethge, M. (2016). Image style transfer using convolutional neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2414-2423).

37. Subramanian, V. (2018). Deep Learning with PyTorch: A practical approach to building neural network models using PyTorch. Packt Publishing Ltd.

38. Ge, F., Ju, Y., Qi, Z., & Lin, Y. (2018). Parameter estimation of a Gaussian mixture model for wind power forecast error by Riemann L-BFGS optimization. Ieee Access, 6, 38892-38899.

39. Zhang, Z. (2018, June). Improved adam optimizer for deep neural networks. In 2018 IEEE/ACM 26th international symposium on quality of service (IWQoS) (pp. 1-2). Ieee.

40. Woldegiorgis, S., Enqvist, A., & Baciak, J. (2021). ResNet and CycleGAN for pulse shape discrimination of He-4 detector pulses: Recovering pulses conventional algorithms fail to label unanimously. Applied Radiation and Isotopes, 176, 109819.

41. Туркін, М. Д. (2023, October). ДОСЛІДЖЕННЯ МЕТОДІВ ПЕРЕНЕСЕННЯ ХАРАКТЕРИСТИК З ОДНОГО ЗОБРАЖЕННЯ НА ІНШЕ. InThe 6th International scientific and practical conference «Methodical and practical methods of creating inventions» (October 24–27, 2023) Sofia, Bulgaria. International Science Group. 2023. 282 p. (p. 278).