ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка комп'ютерного додатку генерації зображень з використанням мереж глибокого навчання»

на здобуття освітнього ступеня бакалавра
зі спеціальності 122 Комп'ютерні науки
(код, найменування спеціальності)
освітньо-професійної програми Штучний інтелект
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Нікіта КУБАР
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач вищої освіти гр.ШІД-41

Нікіта КУБАР

Керівник: Тетяна КИСІЛЬ
старший викладач

Рецензент:
науковий ступінь, вчене звання _____
Ім'я, ПРІЗВИЩЕ

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Штучного інтелекту

Ступінь вищої освіти Бакалавр

Спеціальність 122 Комп'ютерні науки

Освітньо-професійна програма Штучний інтелект

ЗАТВЕРДЖУЮ
Завідувач кафедрою Штучного інтелекту

_____ Ольга ЗІНЧЕНКО
«_____» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Кубар Нікіті Максимовичу
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Розробка комп'ютерного додатку генерації зображень з використанням мереж глибокого навчання

керівник кваліфікаційної роботи Тетяна КИСІЛЬ *старший викладач*
(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» 02.2024р. № 36

2. Строк подання кваліфікаційної роботи «31» травня 2024р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, вивчення фото та відео редакторів, дослідження фізичного моделювання.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

- Дослідження фізичного моделювання
- Розробка програмного продукту та демонстрація результатів

– Аналіз наявної науково-технічної літератури

5. Перелік графічного матеріалу: *презентація*
   1. Опис процесу роботи програми
   2. Характеристики VGG19
   3. Опис використаних інструментів
   4. Висновки

6. Дата видачі завдання«27» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	28.02-03.03.24	
2	Вивчення фото та відео редакторів	04.03-17.03.24	
3	Підготовка дата-сету із зображеннями	18.03-27.03.24	
4	Тестування даних на VGG19	29.02-10.03.24	
5	Навчання нейронної мережіVGG19 на 80% підготовленого дата-сету	11.03-25.03.24	
6	Розробка програмного продукту та демонстрація результатів	26.03-13.05.24	
7	Тестування нейронної мережі VGG 19 на 20% дата-сету	27.04-10.05.24	
8	Підготовка та розробка демонстраційних матеріалів	14.05-15.05.24	
8	Оформлення роботи: вступ, висновки, реферат	15.05-16.05.24	

Здобувач вищої освіти _____ Нікіта КУБАР
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Тетяна КИСІЛЬ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 56 стор., 26 рис., 20 джерел.

Мета роботи — розробити комп'ютерний додаток для генерації зображень з використанням мереж глибокого навчання.

Об'єкт дослідження — мережі глибокого навчання.

Предмет дослідження — методи обробки зображень за допомогою нейронних мереж, зокрема художня обробка, покращення та відновлення зображень.

Короткий зміст роботи: У роботі проведено огляд існуючих технологій розробки апаратно-програмного комплексу для обробки зображень. Зокрема, досліджено методи художньої обробки, покращення та відновлення зображень за допомогою сучасних технологій машинного навчання та нейронних мереж. Проведено аналіз різних застосувань цих методів у сферах фото та відео редакторів, ігор та віртуальної реальності, маркетингу та дизайну, а також фізичного моделювання. Особлива увага приділена стилізації зображень, збільшенню роздільної здатності, зменшенню шуму та відновленню втрачених частин зображень. Розглянуто можливості використання бібліотек PyTorch, NumPy, PIL та інших інструментів Python для реалізації цих методів.

КЛЮЧОВІ СЛОВА: Neural Style Transfer, Модель VGG19, Згорткові нейронні мережі (ЗНМ).

ЗМІСТ

ВСТУП

У сучасному світі комп'ютерна обробка зображень та глибокі нейронні мережі стали невід'ємною частиною багатьох сфер людської діяльності. Розвиток технологій у цих областях сприяє створенню новаторських рішень у відтворенні та трансформації зображень.

Мета даної дипломної роботи — розробити комп'ютерний додаток для генерації зображень з використанням мереж глибокого навчання. Додаток повинен забезпечувати можливість застосування різноманітних стилів до вихідного зображення, що відкриває широкі можливості для творчості та естетичного експерименту.

Обґрунтування актуальності проблеми. За останні десятиліття глибокі нейронні мережі виявилися надзвичайно потужними інструментами у багатьох областях, зокрема в обробці зображень. Вони можуть використовуватися для генерації зображень, перенесення стилів, зміни контенту та багатьох інших завдань, що вимагають високого рівня розуміння зображень.

Розробка комп'ютерного додатку для генерації зображень з використанням мереж глибокого навчання є актуальною задачею, оскільки такий додаток може бути корисним для художників, дизайнерів, фотографів та всіх, хто цікавиться творчістю та обробкою зображень.

Мета та завдання роботи. Метою даної дипломної роботи є розробка комп'ютерного додатку для генерації зображень з використанням мереж глибокого навчання, зокрема для застосування різних стилів до вихідних зображень. Для досягнення цієї мети необхідно вирішити наступні завдання:

– Огляд і аналіз методів генерації зображень з використанням глибоких нейронних мереж.

– Розробка алгоритму для застосування стилів до вихідних зображень.

– Реалізація комп'ютерного додатку на основі вибраного алгоритму.

– Тестування та апробація розробленого додатку на різноманітних зображеннях.

Обгрунтування вибору засобів реалізації. Для реалізації комп'ютерного додатку буде використана мова програмування Python, оскільки вона є популярною серед розробників та має багато бібліотек для обробки зображень та роботи з глибокими нейронними мережами.

Для створення графічного інтерфейсу користувача (GUI) буде використана бібліотека Tkinter, яка є стандартним інструментом для розробки GUI в Python. Дана бібліотека надає зручні засоби для створення вікон, кнопок, полів введення та інших елементів інтерфейсу.

Для реалізації алгоритму генерації зображень з використанням мереж глибокого навчання буде використана бібліотека PyTorch, яка дає змогу використовувати багато можливостей для роботи з нейронними мережами, включаючи підтримку обчислень на графічних процесорах (GPU). Модель VGG19 буде використана для витягування ознак зображення, алгоритм оптимізації Adam буде застосований для навчання моделі.

Структура роботи. Дипломна робота складатиметься з наступних розділів:

– Вступ: у цьому розділі буде визначено мету та актуальність роботи, а також сформульовані основні завдання та вибрані засоби реалізації.

– Обзор літератури та аналіз методів: розділ присвячений огляду і аналізу існуючих методів генерації зображень з використанням глибоких нейронних мереж. Будуть розглянуті основні підходи та їх переваги та недоліки.

– Реалізація комп'ютерного додатку: у цьому розділі буде описано процес розробки комп'ютерного додатку для генерації зображень з використанням мереж глибокого навчання. Буде надано детальний опис алгоритму та його реалізації з використанням мови програмування Python та бібліотеки PyTorch.

– Тестування та апробація: у даному розділі будуть проведені тести розробленого додатку на різних зображеннях з метою оцінки його ефективності та якості генерації.

– Висновки: після кожного розділу будуть підведені підсумки проведеного дослідження у та розглянуті можливості подальшого розвитку та вдосконалення розробленого додатку.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність застосування мереж глибокого навчання у генерації зображень

Застосування мереж глибокого навчання у генерації зображень є актуальним, оскільки це відкриває нові можливості в різних областях, таких як фото та відео редактори, ігри та віртуальна реальність, маркетинг та дизайн.

1.1.1 Фото та відео редактори

В сучасному світі фото та відео редактори стають все більш популярними серед користувачів, які хочуть створювати унікальний контент для соціальних мереж, блогів або особистого використання. Застосування мереж глибокого навчання у таких редакторах дозволяє автоматизувати та поліпшити процес обробки зображень, шляхом застосування фільтрів, покращення роздільної здатності, видалення шумів, реставрації зіпсованих або пошкоджених зображень та багато іншого. Мережі глибокого навчання можуть ефективно виконувати завдання ретуші та стилізації фотографій, що дозволяє користувачам створювати вражаючі зображення з мінімальними зусиллями.

1.1.2 Ігри та віртуальна реальність

У галузі ігор та віртуальної реальності мережі глибокого навчання використовуються для різних цілей, включаючи генерацію реалістичних об'єктів, створення вражаючих світів та пейзажів, реалістичної анімації персонажів та взаємодії з ними, а також для покращення графіки та ефектів спеціальних візуальних ефектів. Мережі глибокого навчання дозволяють створювати більш іммерсивні та захоплюючі ігрові досвіди, що привертають увагу гравців та забезпечують більшу емоційну залученість.

1.1.3 Маркетинг та дизайн

У сфері маркетингу та дизайну застосування мереж глибокого навчання є надзвичайно важливим. Мережі можуть бути використані для автоматизації процесу генерації креативних зображень для рекламних кампаній, створення унікальних дизайнів для продуктів та брендів, а також для аналізу трендів та передбачення майбутніх напрямків у дизайні та маркетингу. Крім того, мережі глибокого навчання можуть допомагати в розпізнаванні та аналізі зображень з соціальних мереж або Інтернету для виявлення нових тенденцій та популярних стилів, що допомагає компаніям підтримувати своє конкурентне перевагу.

1.1.4 Фізичне моделювання

В сфері фізичного моделювання мережі глибокого навчання використовуються для створення реалістичних симуляцій об'єктів, процесів та явищ. Наприклад, в аерокосмічній промисловості мережі можуть використовуватися для моделювання повітряних потоків навколо літаків або ракет, а в автомобільній промисловості — для тестування та оптимізації аеродинамічних властивостей автомобілів. Мережі глибокого навчання дозволяють отримувати більш точні та реалістичні результати моделювання, що допомагає вдосконалювати та вдосконалювати продукти та технології в різних галузях.

1.2 Опис предметної області проєкту

Обробка зображень за допомогою методів машинного навчання є активно розвиваючою галуззю, яка включає в себе широкий спектр технік та методів для аналізу, модифікації та використання візуальної інформації. Вона застосовується в різних сферах життя, включаючи медицину, автомобільну промисловість, медіа, маркетинг і багато інших.

Завдяки розвитку глибокого навчання, особливо глибоких нейронних мереж,

досягнуті значні досягнення в обробці зображень. Методи, такі як згорткові нейронні мережі (ЗНМ), рекурентні нейронні мережі (РНМ) та генеративні моделі, надають засоби для розв'язання широкого спектру завдань, таких як класифікація, сегментація, виявлення об'єктів, генерація зображень та багато .

Проєкт спрямований на розробку системи обробки зображень, яка використовуватиме передові методи машинного навчання для рішення різноманітних завдань. Система буде включати в себе модулі для передобробки даних, навчання та тестування моделей, а також інструменти для візуалізації результатів та аналізу ефективності. Вона буде призначена для використання в наукових дослідженнях, промислових застосуваннях та реальних проєктах, що вимагають обробки зображень з високою точністю та швидкістю.

1.3 Основні завдання та методи обробки зображень

1.3.1 Художня обробка

Художня обробка зображень є ключовою в індустрії дизайну та реклами. Вона використовується для створення привабливих та ефектних зображень, які привертають увагу споживачів і залучають їх до продуктів або послуг. Наприклад, художня обробка може використовуватися для створення стильних рекламних

У сфері мистецтва художня обробка зображень відкриває безмежні можливості для художників та творців. Вони можуть експериментувати з різними техніками та стилями, перетворюючи звичайні фотографії на унікальні художні шедеври. Наприклад, застосування ефектів та фільтрів може допомогти художникам створити абстрактні або фантастичні зображення, які вражають своєю оригінальністю та креативністю.

Крім того, художня обробка зображень знаходить широке застосування в медіа-індустрії. Вона використовується для покращення якості фотографій та відео, ретушування зображень, створення спеціальних ефектів у фільмах та відеоіграх, а також для створення анімацій та візуальних ефектів.

Отже, як приклад, давайте візьмемо фотографію затишного пейзажу з гірськими вершинами та зеленими деревами. Ми можемо застосувати стиль картини "Зіркова ніч" Вінсента ван Гога до цієї фотографії, щоб створити нове зображення, яке поєднує елементи реальності з естетикою відомого мистецтва (рис. 1.1).

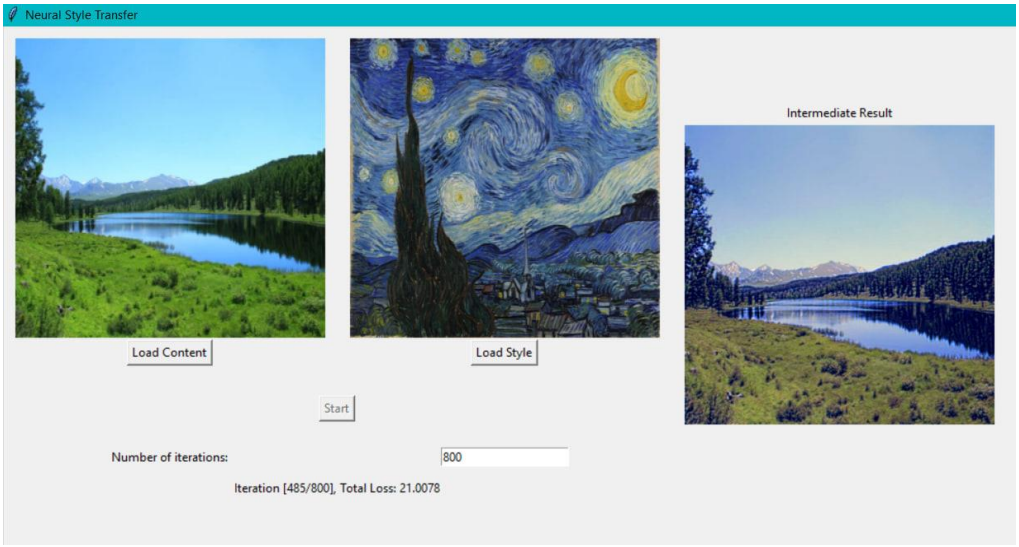


Рис. 1.1 Обробка зображень нейронною мережею.

1.3.2 Покращення зображення

Покращення зображення є ключовим етапом у вирішенні численних завдань, включаючи поліпшення якості фотографій, відео, медичних зображень, а також відновлення пошкоджених або розмитих зображень. Методи машинного навчання дозволяють автоматизувати цей процес і забезпечити високу якість покращення зображень.

Одні з найпоширеніших методів машинного навчання для покращення зображень включають в себе глибокі згорткові нейронні мережі (Deep Convolutional Neural Networks, CNNs) та генеративні згорткові мережі (Generative Adversarial Networks, GANs). Ці моделі можуть бути навчені аналізувати зображення та виконувати різноманітні операції з покращення, такі як підвищення різкості, підвищення контрастності, видалення шумів, відновлення деталей тощо [7].

Наприклад, з допомогою глибоких нейронних мереж можна автоматично видаляти шуми з фотографій, що зроблені в умовах поганої освітленості або використовуючи неякісну техніку зйомки. Також можна автоматично підвищувати різкість та деталізацію зображення, щоб отримати більш якісні результати.

Застосування методів машинного навчання для покращення зображень відкриває широкі перспективи у таких сферах, як: медицина, наука, мистецтво, дизайн, реклама. Відновлення та поліпшення зображень за допомогою машинного навчання може сприяти підвищенню якості діагностики у медицині, поліпшенню візуального сприйняття у мистецтві та дизайні, а також покращенню якості відео та фотоматеріалів у медіаіндустрії.

В цій сфері дослідження активно ведуться багатьма відомими компаніями та дослідницькими групами, що свідчить про важливість та перспективність цього напрямку.

Наприклад, компанія NVIDIA розробила інноваційну технологію під назвою NVIDIA Deep Learning Super Sampling, яка використовує глибокі нейронні мережі для підвищення роздільної здатності графічних зображень у відеоіграх (рис. 1.2). Ця технологія дозволяє підвищити якість зображення та підвищити швидкість кадрів у грі за рахунок штучного інтелекту, що демонструє потенціал методів машинного навчання у поліпшенні візуального досвіду [12].

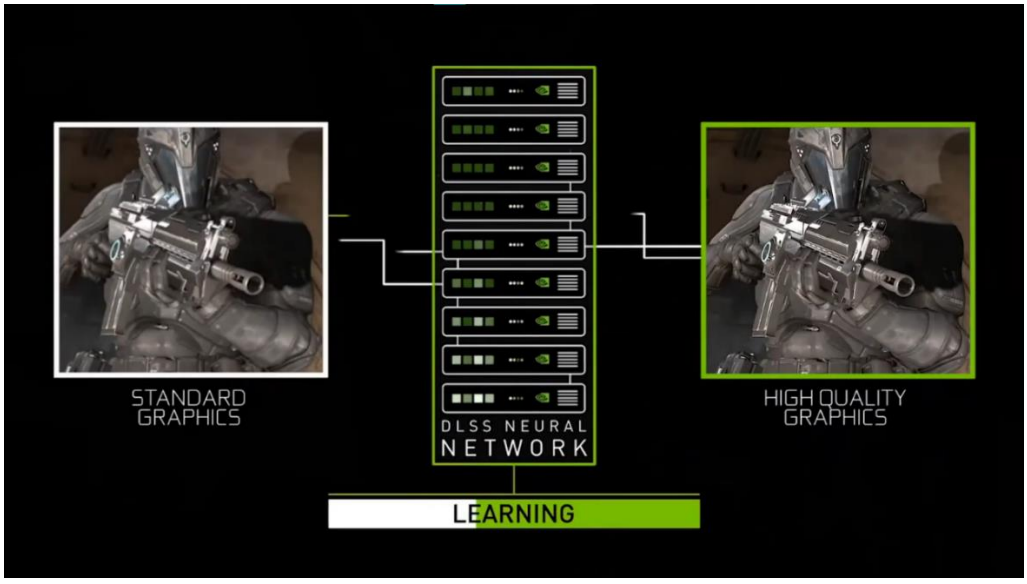


Рис. 1.2 Підвищення роздільної здатності зображень

Також, відомі дослідницькі групи, такі як Google Research та OpenAI, активно

працюють над розвитком нових алгоритмів та моделей машинного навчання для покращення зображень. Їхні роботи включають в себе створення більш ефективних та потужних алгоритмів обробки зображень, які можуть застосовуватися в різних галузях, від медичної діагностики до редакції фотографій.

Ці приклади демонструють значущість та перспективи досліджень у сфері покращення зображень за допомогою методів машинного навчання. Вони показують, що цей напрямок знаходить широке застосування і привертає увагу провідних компаній та наукових установ.

1.3.3 Відновлення зображення

Завдання відновлення зображень включає в себе відновлення пошкоджених або втрачених даних на зображеннях. Це важливий процес, який знаходить застосування в багатьох областях, таких як медицина, супутникова зйомка, реставрація мистецьких творів та багато інших. Відновлення зображень може включати усунення шумів, виправлення розмитих ділянок, відновлення втрачених фрагментів або видалення артефактів.

Методи машинного навчання, такі як автокодери [8] та глибокі згорткові мережі (CNN), можуть бути використані для ефективного відновлення пошкоджених зображень. Автокодери навчаються стискати та декодувати зображення, що дозволяє їм відновлювати структуру та деталі зображень навіть після значних пошкоджень. Глибокі згорткові мережі, в свою чергу, здатні виявляти та виправляти складні візуальні дефекти завдяки своїй багатошаровій архітектурі [7].

Основні методи відновлення зображень:

1. *Автокодери (Autoencoders*): автокодери складаються з двох основних частин: енкодера, який стискає вхідне зображення до меншого розміру (коду), і декодера, який відновлює зображення з коду. Навчаючись на великій кількості зображень, автокодери можуть навчитися відновлювати оригінальні зображення з пошкоджених версій.

– Глибокі згорткові мережі (CNN): NN використовують згорткові шари для аналізу зображень, що дозволяє їм виділяти важливі ознаки та виправляти дефекти. CNN можуть бути навчені на великих наборах даних, щоб розпізнавати та виправляти різні типи пошкоджень.

– Generative Adversarial Networks (GANs): GANs складаються з двох моделей: генератора, який створює зображення, і дискримінатора, який оцінює їх якість. Навчання GANs може допомогти в створенні реалістичних відновлених зображень з пошкоджених або втрачених даних.

– Приклади застосування генеративних систем:

2. *Медицина*:

– Відновлення медичних зображень, таких як рентгенівські знімки або МРТ, для поліпшення діагностичної точності. Наприклад, зображення МРТ можуть бути пошкоджені шумами, і методи машинного навчання можуть допомогти їх усунути, що забезпечить більш точне діагностування.

3. Супутникова зйомка:

– Відновлення зображень, отриманих з супутників, які можуть бути пошкоджені через атмосферні явища або технічні проблеми. Це важливо для точного аналізу географічних даних та моніторингу навколишнього середовища.

4. Реставрація мистецьких творів:

– Відновлення старовинних картин та фотографій, які з часом зазнали пошкоджень. Машинне навчання може бути використане для заповнення відсутніх частин зображень, відновлення кольорів та деталей.

5. Покращення якості відео:

– Відновлення та покращення якості старих відео, які можуть мати низьку роздільну здатність або пошкодження. Це може бути корисним для архівів та кіностудій, що бажають відновити старі фільми.

1.3.4 Сегментація зображення

Сегментацією зображень називають процес розділення зображення на окремі частини або об'єкти, що мають схожі характеристики або властивості. Цей процес допомагає комп'ютерам розуміти структуру та вміст зображення, що в свою чергу може бути використано для багатьох цілей, таких як розпізнавання об'єктів, візуальний пошук, аналіз зображень та інші.

1. Методи сегментації зображень [3]:

– Порогова сегментація: Зображення розділяється на певні зони на основі заданого порогу інтенсивності пікселів. Пікселі з інтенсивністю вище порогу вважаються однією областю, а з меншою — іншою. Наприклад, для чорно-білих зображень, можна використовувати поріг яскравості для розділення областей.

– Методи розташування країв: Вони використовують властивості країв об'єктів для їх сегментації. Краї можуть бути визначені шляхом виявлення різниці в інтенсивності між сусідніми пікселями або з використанням фільтрів, які виділяють краї.

– Сегментація на основі кольору: Цей підхід використовує інформацію про колір пікселів для їх групування в об'єкти. Методи можуть бути основаними на розділенні кольорів у просторі кольорів або на використанні моделей кольору для виділення областей зі схожими кольорами.

– Сегментація з використанням машинного навчання: Вона використовує класифікатори, які були навчені на основі попередньо сегментованих зображень для автоматичного сегментування нових зображень. Це може бути як навчання з учителем, так і без нього.

2. Приклади застосування сегментації зображень:

– Медичне зображення: Сегментація допомагає лікарям виділити та аналізувати різні структури на зображеннях, такі як органи або пухлини, що є важливим для діагностики та лікування [4].

– Автомобільні системи помічника водія: Системи ADAS [15] (Advanced Driver Assistance Systems) використовують сегментацію зображень для виявлення та

класифікації дорожніх знаків, пішоходів, транспортних засобів тощо.

– Автоматична обробка зображень: Сегментація допомагає відокремити об'єкти від тла для подальшого аналізу, такого як вимірювання розмірів об'єктів або виявлення аномалій.

– Реконструкція 3D-моделей: Сегментація може використовуватися для відокремлення об'єктів на зображеннях і після цього для реконструкції їх 3D-моделей.

У нашому коді сегментація зображень не використовується безпосередньо, але модель VGG19, яка завантажується із бібліотеки torchvision, використовується для витягування ознак зображення, що є одним з етапів в процесі стилю перенесення. Витягнуті ознаки використовуються для обчислення контенту та стилю зображення.

Ось декілька моментів, які стосуються сегментації зображень:

1. Завантаження та підготовка зображень: Функції load_content_image та load_style_image завантажують та підготовлюють зображення, що є важливим етапом обробки зображень. Хоча це не чиста сегментація, але зображення обробляються для подальшого використання.
2. Виділення ознак: Функція get_features використовує модель VGG19 для виділення ознак зображення. Це включає різні шари моделі, які можна розглядати як різні рівні деталізації, подібно до того, як працюють деякі методи сегментації.
3. Матриця Грама: Функція gram_matrix використовується для обчислення матриці Грама, яка використовується для оцінки стилю зображення. Це можна розглядати як певний аспект аналізу структури зображення, який нагадує певні аспекти сегментації.

Хоча сегментація зображень не є центральним елементом цього коду, але певні аспекти цього процесу можна помітити в контексті обробки зображень для досягнення певних цілей, таких як стиль перенесення.

1.3.5 Стиснення зображення

Стиснення зображення — це процес зменшення обсягу даних зображення за рахунок видалення або зменшення кількості інформації, що зберігається в файлі зображення. Це може бути зроблено різними способами, такими як видалення непотрібної інформації, використання методів стиснення без втрат або стиснення з втратами.

1. Безвтратне стиснення: При безвтратному стисненні зображення зменшується без втрат якості. Такі методи стиснення використовують алгоритми, що дозволяють зменшити обсяг даних, не втрачаючи жодної інформації. Наприклад, алгоритм стиснення без втрат, такий як алгоритм стиснення зображень з використанням кодування Хаффмана, може зменшити розмір файлу без втрати якості.
2. Стиснення з втратами: При стисненні з втратами деяка інформація втрачається з метою зменшення обсягу даних. Такі методи стиснення зазвичай використовуються в тих випадках, коли невелике зниження якості прийнятне. Прикладами таких методів є JPEG-стиснення або алгоритми стиснення з втратами у форматі WebP.

У нашому коді стиснення зображення використовується при завантаженні та відображенні зображення. Коли ми завантажуєте зображення за допомогою **Image.open()** з бібліотеки PIL (Python Imaging Library), ми можемо застосовувати різні методи стиснення, наприклад, JPEG-стиснення, залежно від параметрів, які використовуємо при збереженні зображення.

Наприклад:

image.save("compressed_image.jpg", quality=85) # Збереження зображення з JPEG-стисненням і якістю 85%

Також, при відображенні зображення в інтерфейсі за допомогою ImageTk.PhotoImage(), ми можемо використовувати різні методи стиснення, щоб зменшити розмір відображення зображення в інтерфейсі.

1.3.6 Маніпулювання зображенням

Маніпулювання зображенням — це процес зміни або модифікації властивостей зображення для досягнення певних цілей або виконання певних завдань. Це може включати в себе зміну розміру, обрізку, обертання та зміну кольору. Методи машинного навчання можуть використовуватися для автоматизації цих операцій та розробки моделей, які відповідають на різні завдання маніпулювання зображенням.

1. Зміна розміру зображення:

Зміна розміру зображення — це процес зміни фізичних розмірів зображення. Це може бути корисним для пристосування зображення до певних вимог або обмежень. Наприклад, зменшення розміру зображення може бути корисним для швидкого завантаження в мережі або для підготовки даних для моделі машинного навчання (рис1.3).

```
from PIL import Image

# Відкриття зображення
image = Image.open("image.jpg")

# Зміна розміру зображення
resized_image = image.resize((new_width, new_height))
resized_image.show()
```

Рис. 1.3 Приклад коду зміни розміру зображення

2. Обрізка зображення:

Обрізка зображення — це процес видалення частини зображення, залишаючи лише певну область. Це може бути корисним для виділення певних об'єктів або скорочення непотрібної інформації зображення (рис1.4).

```python
from PIL import Image

# Відкриття зображення
image = Image.open("image.jpg")

# Обрізка зображення
cropped_image = image.crop((left, top, right, bottom))
cropped_image.show()
```

Рис. 1.4 Приклад коду обрізки зображення

3. Обертання зображення:

Обертання зображення — це процес повороту зображення на певний кут. Це може бути корисним для вирішення проблеми неправильного орієнтування зображення або для створення спеціальних ефектів (рис1.5).

```python
from PIL import Image

# Відкриття зображення
image = Image.open("image.jpg")

# Обертання зображення на 90 градусів
rotated_image = image.rotate(90)
rotated_image.show()
```

Рис. 1.5 Приклад коду обертання зображення

4. Зміна кольору зображення:

Зміна кольору зображення — це процес модифікації кольорової палітри зображення. Це може включати зміну яскравості, контрасту, насиченості або кольорового тону (рис1.6).

```python
from PIL import ImageEnhance

# Відкриття зображення
image = Image.open("image.jpg")

# Збільшення насиченості кольорів
enhancer = ImageEnhance.Color(image)
color_enhanced_image = enhancer.enhance(2)  # Параметр 2 підвищує насиченість вдвічі
color_enhanced_image.show()
```

Рис. 1.6 Приклад коду зміни кольору зображення

1.3.7 Генерація зображень

Генерація зображень — це процес створення нових зображень за допомогою алгоритмів машинного навчання. Ця техніка використовується в різних галузях, включаючи комп'ютерне зорове розпізнавання, медичне зображення, графічний дизайн та багато інших. Основними методами генерації зображень є глибокі генеративні моделі, такі як генеративні згорткові мережі (GANs), автокодери та варіаційні автокодери.

1. Глибокі генеративні моделі (GANs) [16]:

   – GANs — це клас нейронних мереж, які використовують два конкуруючих нейронних мережі — генератор і дискримінатор. Генератор створює нові зображення, які намагаються імітувати вхідні дані, в той час як дискримінатор намагається відрізнити справжні зображення від згенерованих.

   – Наприклад, GAN може використовуватися для створення реалістичних зображень облич, пейзажів або навіть мистецьких творів.

2. Автокодери:

   – Автокодери - це нейронні мережі, які використовуються для автоматичного відтворення вхідних даних. Вони вчаться кодувати вхідні дані в якісний вектор і потім декодувати цей вектор, щоб відновити оригінальні дані.

   – Наприклад, автокодер може використовуватися для створення реалістичних зображень, що були затемнені або пошкоджені.

3. Варіаційні автокодери:

   – Варіаційні автокодери — це спеціальний тип автокодерів, які використовуються для генерації нових зображень, шляхом генерації нових прикладів з вхідного простору даних.

   – Наприклад, варіаційний автокодер може використовуватися для створення нових зображень облич, з врахуванням певних властивостей, які були вивчені з вхідних зображень.

   Приклади використання генерації зображень у реальних компаніях:

DeepArt [5] — це онлайн-сервіс, який використовує глибокі нейронні мережі для перетворення звичайних зображень в мистецькі шедеври, імітуючи стилі відомих художників. Користувачі можуть завантажити свої фотографії та обрати стиль малюнку, який хочуть застосувати. Потім система використовує глибокі генеративні моделі, щоб створити нове зображення, яке поєднує стиль художника з вмістом фотографії.

GANPaint Studio — це інтерактивний інструмент, розроблений дослідниками з MIT та IBM Watson AI Lab, який використовує генеративні згорткові мережі (GANs) для створення нових зображень та редагування вже існуючих. Цей інструмент дозволяє користувачам додавати, видаляти або змінювати об'єкти на зображеннях, змінювати їх стилі або кольори.

DeepDream [6] — це проєкт, розроблений в Google, який використовує нейронні мережі для генерації психоделічних зображень. Він використовується для вивчення та візуалізації внутрішніх репрезентацій нейронних мереж. DeepDream може бути використаний як творчий інструмент для створення унікальних мистецьких творів.

1.4 Огляд та застосування Neural Style Transfer

1.4.1 Фото та відео редактори

Neural Style Transfer (NST) [11] — це техніка машинного навчання, що дозволяє комбінувати стиль одного зображення з контентом іншого, створюючи таким чином нове зображення, яке поєднує в собі естетику стилю з вмістом оригінального зображення. Основна ідея полягає в тому, щоб витягнути стилістичні характеристики одного зображення (наприклад, картини або фотографії) і застосувати їх до іншого зображення (зазвичай фотографії), зберігаючи при цьому вміст останнього.

Одним з відомих методів NST є використання глибоких нейронних мереж, таких як глибокі згорткові мережі (CNN). У навчальному процесі NST нейронна

мережа навчається мінімізувати втрати, які виникають в результаті відмінностей між змістовими та стилістичними характеристиками оригінальних зображень. Цей процес відбувається шляхом використання двох функцій втрат: втрати контенту і втрати стилю.

Застосування NST у фото- та відеоредакторах надає користувачам можливість експериментувати з різноманітними художніми ефектами та фільтрами. Наприклад, користувач може застосувати стиль відомого художника до своєї фотографії або навпаки — застосувати стиль своєї улюбленої фотографії до відомої картини. Це дає безліч можливостей для творчості та самовираження.

Окрім того, NST може бути використаний для редагування відео, де він може застосовуватися до кожного кадру, щоб створити вражаючі ефекти та трансформації стилю протягом усього відео. Такі редагувальні можливості дозволяють створювати унікальні інтерактивні шедеври, які вражають глядачів своєю красою та оригінальністю.

У сучасному світі Neural Style Transfer стає все більш популярним інструментом для творчих та художніх застосувань, від фотографії до мистецтва та реклами. Його потужність і гнучкість роблять його корисним інструментом для будь-якого, хто бажає виразити себе через зображення.

Приклади застосування Neural Style Transfer у фото- та відеоредакторах:

1. Арт-фільтри в соціальних мережах: Багато популярних соціальних мереж надають функціонал для застосування різноманітних фільтрів до фотографій. Наприклад, Instagram має функцію "Фільтри", яка дозволяє користувачам вибирати між різними стилями, що надають фотографіям художній вигляд.
2. Фоторедактори з функцією "Авто-стиль": Деякі фоторедактори, такі як Adobe Photoshop та Snapseed, мають функції "Авто-стиль" або "Автоматичний фільтр", які застосовують набір ефектів до фотографій на основі їхнього змісту і стилю.
3. Мобільні додатки для редагування відео: Додатки для редагування відео на мобільних пристроях, такі як TikTok та Adobe Premiere Rush, можуть використовувати Neural Style Transfer для застосування ефектів стилю до відеороликів. Наприклад, користувач може змінити стиль відео на анімацію чи

живопис, щоб зробити його більш ефектним.

4. Відеоредактори з функцією "Зміна стилю": Деякі відеоредактори, такі як Adobe After Effects та DaVinci Resolve, мають функції "Зміна стилю", які дозволяють користувачам застосовувати різноманітні ефекти стилю до відеороликів.

5. Онлайн-сервіси для редагування фото та відео: Існують онлайн-сервіси, такі як Canva та Pixlr, які мають функціонал для редагування фотографій та відео з використанням Neural Style Transfer. Ці сервіси дозволяють користувачам легко застосовувати різноманітні стилі та ефекти до своїх зображень та відеороликів прямо в браузері.

1.4.2 Ігри та віртуальна реальність

Віртуальна реальність (VR) і ігри є одними з найперспективніших областей застосування Neural Style Transfer (NST). Використання NST у цих сферах може принести значні переваги, такі як поліпшення візуальної естетики, підвищення іммерсивності для гравців, а також створення унікальних візуальних стилів.

У сфері віртуальної реальності, NST може використовуватися для створення реалістичних текстур, оточення і атмосфери. Наприклад, за допомогою NST можна застосувати художній стиль до віртуального середовища, щоб створити ефекти, які доповнюють настрій або жанр гри. Це може включати стилізацію оточення під певний історичний період або мистецький стиль, що додає глибину іммерсії для користувачів VR.

У відеоіграх NST може використовуватися для створення унікальних візуальних ефектів, які роблять гру більш привабливою для гравців. Наприклад, NST може бути використаний для застосування художнього стилю до текстур персонажів, архітектури та оточення гри. Це дозволяє розробникам створювати ігри з унікальним візуальним стилем, який вирізняється серед інших і привертає увагу гравців.

Одним з прикладів використання NST у відеоіграх є застосування стилю відомих художників до візуальних елементів гри. Наприклад, гра може

використовувати стиль Рембрандта або Вінсента ван Гога для стилізації оточення чи персонажів. Це не лише надає грі унікальний візуальний стиль, але й робить її більш культурно цікавою та навчальною для гравців.

Отже, застосування NST у віртуальній реальності та іграх відкриває широкі можливості для створення іммерсивних та захоплюючих візуальних досвідів, які покращують взаємодію користувачів з віртуальними середовищами та гральними світами.

1.4.3 Маркетинг та дизайн

Neural Style Transfer (NST) відкриває широкі можливості у сфері маркетингу та дизайну, дозволяючи створювати унікальні, естетично привабливі зображення та рекламні матеріали, що виділяються на тлі конкуренції. Завдяки здатності NST поєднувати стиль та контент зображень, компанії можуть розробляти креативні та ефективні візуальні стратегії для своїх маркетингових кампаній.

Приклади застосування NST у маркетингу та дизайні:

1. Брендові візуальні матеріали

NST може бути використаний для створення унікальних брендових візуальних матеріалів. Наприклад, компанії можуть застосовувати стилі відомих художників або унікальні візуальні ефекти до своїх логотипів, банерів та інших рекламних матеріалів. Це допомагає створити неповторний бренд, який легко впізнається і запам'ятовується споживачами.

Розглянемо конкретний приклад застосування NST у маркетингу. Компанія Coca-Cola може використовувати NST для створення рекламних плакатів, стилізованих під роботи таких художників, як Енді Воргол. Це допомогло б привернути увагу до їхньої продукції за рахунок використання впізнаваних художніх стилів, які додають креативності та унікальності рекламним матеріалам [18].

2. Створення контенту для соціальних мереж. У соціальних мережах візуальний контент відіграє ключову роль у залученні аудиторії. Використання NST дозволяє

створювати оригінальні зображення, які можуть стати вірусними. Наприклад, бренди можуть перетворювати звичайні фотографії своїх продуктів у витвори мистецтва, що допомагає виділитися серед великої кількості контенту в соціальних мережах.

Розглянемо конкретний приклад застосування NST у маркетингу. Модний бренд Gucci може використовувати NST для стилізації фотографій своїх моделей та аксесуарів у стилі відомих художників або художніх напрямків, таких як імпресіонізм чи поп-арт. Це приверне увагу до їхніх постів та збільшить залученість аудиторії.

3. Дизайн упаковки

NST може бути використаний для розробки унікальних дизайнів упаковок для продуктів. Замість звичайних шаблонів компанії можуть створювати художні шедеври, які підкреслюють унікальність їхньої продукції. Це може значно підвищити привабливість продукту в очах споживачів та виділити його на полиці магазинів.

Приклад розробки унікальних дизайнів може бути розглянутим у компанії Nestlé яка може застосувати NST для створення упаковок для своїх шоколадних батончиків, стилізованих під класичні картини, такі як "Зоряна ніч" Вінсента ван Гога. Така упаковка буде не лише привабливою, але й здатною викликати емоційний відгук у покупців, асоціюючись з високим мистецтвом та культурою.

4. Рекламні відео

NST можна використовувати і у відеомаркетингу. З його допомогою можна створювати рекламні ролики з унікальними візуальними ефектами, що дозволяє створювати запам'ятовувані та креативні відеоматеріали.

Прикладом такого застосування може бути компанія Nike яка може створити рекламний ролик, де відео спортсменів, що займаються спортом, буде стилізоване під різні художні напрямки, такі як кубізм або футуризм. Це не лише приверне увагу до бренду, але й підкреслить його креативність та інноваційність.

Переваги використання NST у маркетингу та дизайні:

1. Унікальність та впізнаваність: Використання NST дозволяє створювати

унікальні візуальні матеріали, які виділяють бренд на тлі конкурентів.

2. Естетична привабливість: Зображення, створені з використанням NST, часто мають високу естетичну цінність, що підвищує їх привабливість для аудиторії.

3. Емоційний зв'язок: Стилізовані зображення можуть викликати сильні емоції у споживачів, що сприяє кращому запам'ятовуванню бренду та продукції.

4. Креативність та інноваційність: Використання передових технологій, таких як NST, демонструє креативність та інноваційність бренду, що позитивно відображається на його іміджі.

В результаті виконаного дослідження можна зробити висновок, що Neural Style Transfer є потужним інструментом у руках маркетологів та дизайнерів. Його використання дозволяє створювати унікальні, запам'ятовувані та естетично привабливі візуальні матеріали, які можуть значно підвищити ефективність маркетингових кампаній та допомогти брендам виділитися на тлі конкуренції. З розвитком технологій NST та зростанням інтересу до візуального контенту, його значення у сфері маркетингу та дизайну буде лише зростати.

1.4.4 Фізичне моделювання

Фізичне моделювання передбачає створення реалістичних симуляцій фізичних процесів, таких як рух рідин, динаміка частинок або поведінка матеріалів під впливом зовнішніх сил. У цій галузі методи машинного навчання, зокрема Neural Style Transfer (NST), можуть бути використані для покращення візуальної складової моделей та створення більш реалістичних і візуально привабливих симуляцій.

Застосування NST у фізичному моделюванні: NST дозволяє перенести стиль одного зображення на інше, зберігаючи при цьому основні структурні особливості оригіналу. Це може бути використано для візуалізації фізичних процесів, створюючи реалістичні та художньо привабливі зображення симуляцій.

Дослідження: У дослідженні "Deep Fluids: A Generative Network for Parameterized Fluid Simulations", яке було опубліковане на конференції з

комп'ютерного бачення та обробки зображень (CVPR), автори пропонують новий підхід до генерації параметризованих симуляцій рідин за допомогою глибоких нейронних мереж. Вони використовують мережу глибокого навчання для генерації візуальних ефектів, які доповнюють фізично точні симуляції.

Ідея полягає в тому, щоб використовувати NST для стилізації симуляцій рідин, додавання до них художньої привабливості та естетичності. Наприклад, шляхом застосування стилю відомих картин або фотографій до симуляцій рідин можна створити унікальні візуальні ефекти, які привертають увагу та роблять симуляції більш привабливими для сприйняття.

Цей підхід не лише покращує естетичний аспект симуляцій, але й може мати практичне застосування в різних галузях, включаючи візуальні ефекти у кіноіндустрії, комп'ютерних іграх, наукових візуалізаціях та багато інших.

1.5 Постановка задачі та формальна постановка

Центральною метою даного проєкту є розробка комп'ютерного додатку, який використовує методи машинного навчання для зміни стилю зображень. Головною задачею є створення програмного продукту, який дозволить користувачам застосовувати різноманітні стилі до своїх зображень шляхом використання передових алгоритмів глибокого навчання, зокрема Neural Style Transfer (NST). Основні функції додатку будуть включати в себе можливість завантаження зображень, вибір стилів та налаштування параметрів стилю, а також перегляд результатів та збереження оброблених зображень.

Для досягнення цієї мети передбачається реалізація наступних завдань:

1. Розробка алгоритму для використання методу NST з метою зміни стилю зображення.
2. Створення інтерфейсу користувача, який буде зручним у використанні та дозволить легко взаємодіяти з програмою.
3. Імплементація функціоналу для завантаження та збереження зображень, вибору стилів та налаштування параметрів обробки.

4. Оптимізація алгоритмів для швидкої та ефективної обробки зображень на різних пристроях.

Ми плануємо детально проаналізувати існуючі підходи до вирішення цієї задачі, а також реалізувати власні варіанти алгоритмів з метою покращення якості та ефективності обробки. В результаті цей проєкт має стати потужним інструментом для зміни стилю зображень, який буде корисним для широкого кола користувачів у різних областях, таких як мистецтво, дизайн, маркетинг, освіта та інші.

Цей проєкт може бути корисним для таких груп людей та компаній:

– Митці та дизайнери: Даний додаток надасть їм можливість швидко та ефективно експериментувати з різними стилями та впливати на вигляд своїх творів.

– Маркетологи та рекламні агентства: Здатність швидко створювати естетично привабливі зображення може бути великим перевагою при розробці рекламних матеріалів та кампаній.

– Освітні установи: Використання цього додатку може сприяти у створенні навчальних матеріалів та презентацій, що привертають увагу та полегшують засвоєння інформації.

– IT-компанії та розробники програмного забезпечення: Розробка та оптимізація алгоритмів для цього додатку може бути інтересною задачею для IT-спеціалістів, що спеціалізуються на машинному навчанні та обробці зображень.

– Користувачі різних соціальних мереж: Здатність створювати унікальні та привабливі зображення для публікації в соціальних мережах може зацікавити широке коло користувачів, що активно спілкуються в онлайн-середовищі.

1.6 Висновки до розділу 1

В цьому розділі ми розглянули важливі застосування методу Neural Style Transfer (NST) в різних сферах, таких як віртуальна реальність, маркетинг та дизайн, а також фізичне моделювання. Ми дізналися, що NST може значно поліпшити візуальні ефекти у віртуальних середовищах, надаючи їм більшу реалістичність та

іммерсивність. У маркетингу та дизайні він відкриває нові можливості для створення унікальних та привабливих візуальних матеріалів, які привертають увагу аудиторії. Крім того, в фізичному моделюванні NST може бути використаний для покращення візуальної складової симуляцій, забезпечуючи їх більшою реалістичністю та естетичністю. Загалом, ми побачили, що NST є потужним інструментом у руках різних галузей, який може допомогти покращити візуальний досвід користувачів та ефективність комунікації з аудиторією.

2 АНАЛІЗ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ

2.1 Згорткові нейронні мережі (ЗНМ)

Згорткові нейронні мережі — ефективне вирішення широкого спектру завдань, таких як розпізнавання об'єктів, класифікація зображень, виявлення облич та інші. Давайте розглянемо кожен аспект згорткових нейронних мереж більш детально:

Згорткові шари (Convolutional Layers):Шари які виконують основну роботу з виявлення ознак на зображенні. Вони використовують фільтри або ядра для просторової згортки зображення з метою виділення різних візуальних ознак, таких як границі, текстури та форми. Згорткові шари дозволяють моделі вчитися виявляти важливі особливості на різних рівнях абстракції.

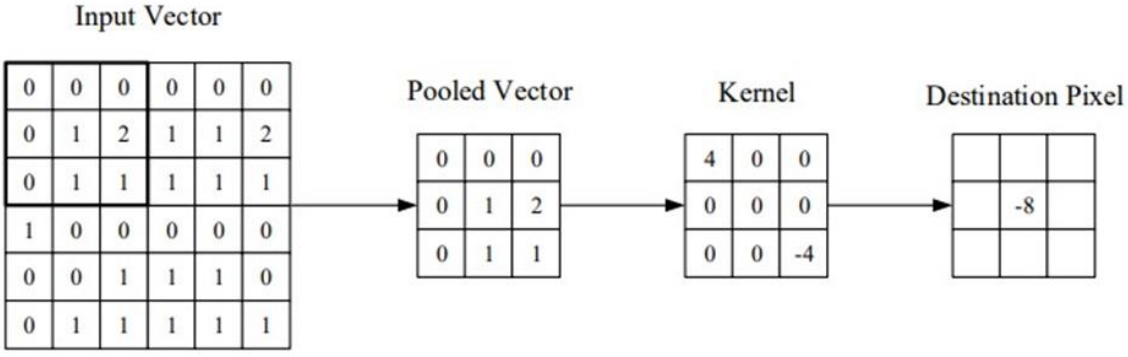


Рис. 2.1 Візуальне представлення згорткового шару [1]

Шари пулінгу (Pooling Layers): Об'єднання шарів в мережах глибокого навчання має на меті зменшення просторових розмірів даних зі згорткового шару, щоб зменшити кількість параметрів і обчислювальну складність моделі. Це досягається шляхом масштабування розмірності карти активації кожного шару за допомогою функції пулінгу, такої як Max Pooling або Average Pooling.

Найпоширенішою функцією пулінгу є максимальне об'єднання, яке обирає максимальне значення з кожного підблоку карти активації. Це забезпечує певну інваріантність перекладу, що означає, що об'єкт буде розпізнаний незалежно від

його місця на зображенні.

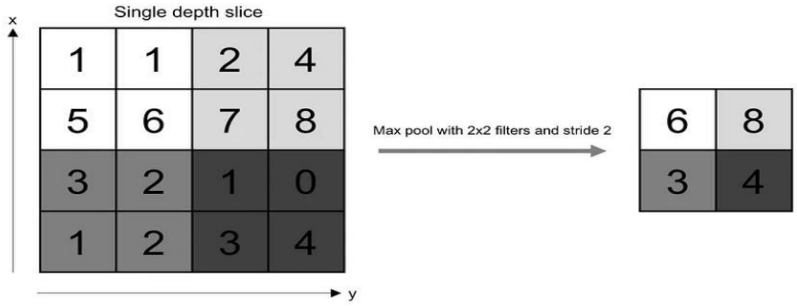


Рис. 2.2 Візуалізація операції максимального об’єднання [9]

Об’єднання забезпечує певну стійкість перекладу, тобто забезпечує, що об’єкт буде розпізнаним незалежно від його розташування на кадрі.

Повністю зв'язані шари (Fully Connected Layers): Ці шари зазвичай розміщені в кінці моделі та використовуються для класифікації об'єктів на зображенні або вирішення інших завдань. Вони приймають на вхід вектор атрибутів, отриманих після попередніх шарів, та генерують вектор вихідних значень, які інтерпретуються як ймовірності належності до різних класів.

2.2 Архітектура Neural Style Transfer (NST)

Для реалізації NST потрібні три основних зображення: зображення вмісту (яке ми хочемо зберегти), зображення стилю (з якого ми хочемо взяти стиль) та вихідне зображення, яке представляє собою поєднання вмісту та стилю.

NST використовує попередньо навчену модель ЗНМ, таку як VGG, навчену на наборі даних ImageNet, у фреймворку PyTorch. Для навчання моделі NST необхідні дві мережі: попередньо навчений екстрактор ознак та мережа передачі стилів.

Зображення, які використовуються для NST, спочатку перетворюються на необроблені пікселі та передаються моделі. Модель обробляє ці дані за допомогою ЗНМ, перетворюючи їх на набір функцій, які відповідають різним аспектам стилю та вмісту зображень.

Приховані шари моделі виступають як екстрактори ознак, які використовуються для опису вмісту та стилю вхідних зображень. Для отримання представлення стилю еталонного зображення використовується кореляція між різними відповідями фільтрів.

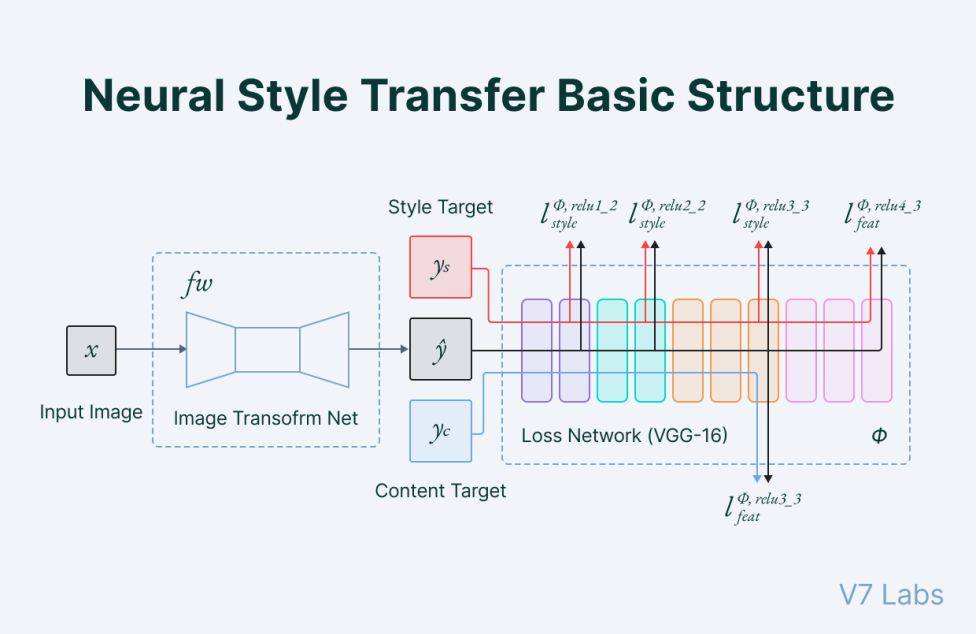


Рис. 2.3 Архітектура Neural Style Transfer [13]

2.2.1Функція втрати вмісту

Функція втрати вмісту гарантує, що активації на вищих рівнях мережі будуть схожі між вихідним зображенням і згенерованим зображенням з врахуванням вмісту. Згорткові нейронні мережі отримують інформацію про вміст на більш високих рівнях абстракції, тоді як на нижчих рівнях увага сконцентрована на окремих пікселях.

2.2.2 Функція втрати стилю

Використовується для забезпечення передачі стилю еталонного зображення на створене зображення, зберігаючи його вміст незмінним. Основна ідея полягає в тому, щоб визначити кореляції між різними фільтрами у вищих рівнях активацій мережі та використати ці кореляції для відтворення стилістичних особливостей

зображення.

У функції втрати стилю акцент здійснюється на збереженні взаємозв'язків між активаціями різних фільтрів у вищих рівнях згорткових шарів мережі. Це допомагає захопити абстрактні поняття стилю, такі як текстури, форми та колірні простори, які виражаються через активації в цих рівнях.

Функція втрати стилю обчислюється за допомогою матриці Грама, яка відображає кореляції між різними каналами активацій вищих рівнів мережі. Ця матриця обчислюється шляхом перемноження матриці активацій на її транспоновану матрицю. Це дозволяє виразити кореляції між фільтрами, які активуються одночасно, і відображає схожість стилів у вихідному та згенерованому зображеннях.

Матриця Грама — це інструмент аналізу, що дозволяє отримати уявлення про стиль у зображенні шляхом вимірювання взаємозв'язків між різними функціями у певному шарі нейронної мережі. Вона використовується для виявлення загального розподілу ознак та стильових елементів, присутніх у зображенні.

Матриця Грама обчислюється шляхом знаходження ступеня кореляції між картами об'єктів у вказаному шарі. Цей процес дозволяє отримати інформацію про те, які об'єкти та ознаки активуються разом, що відображається у взаємозв'язках між ними.

Використання матриці Грама у методі перенесення стилю полягає в тому, що вона дозволяє захопити стилістичні характеристики зображення, такі як текстури та форми, шляхом аналізу кореляцій між їхніми функціями. Ця інформація використовується для визначення функції втрати стилю, що допомагає зберегти стиль еталонного зображення під час генерації нового зображення. Таким чином, матриця Грама стає ключовим інструментом у процесі аналізу та передачі стилю між зображеннями.

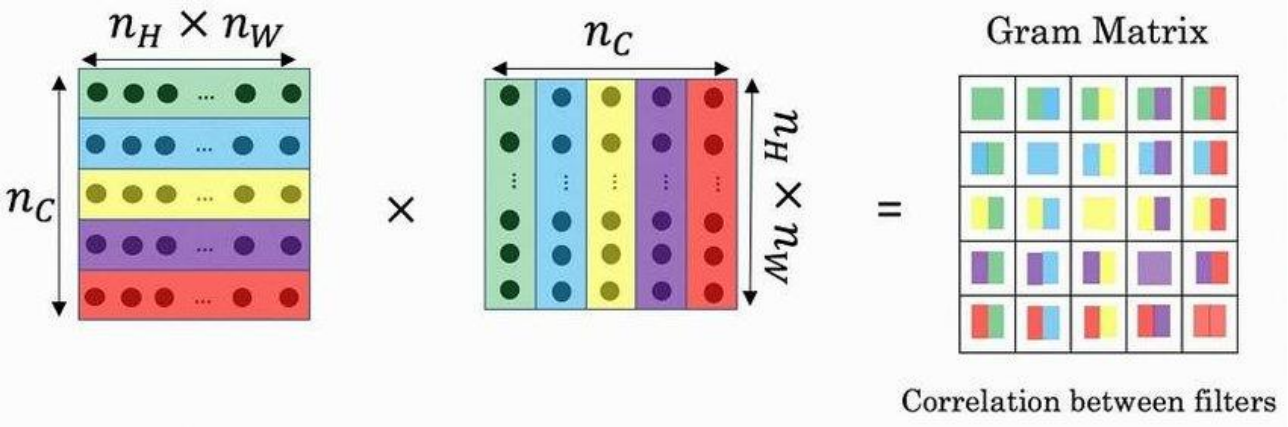


Рис. 2.4 Матриця Грама [14]

2.2.3 Оптимізатори

Одним із ключових аспектів глибокого машинного навчання є оптимізатори, які налаштовують параметри моделі, такі як ваги та швидкість навчання, для мінімізації функції втрат або максимізації функції винагороди. Ці алгоритми допомагають знаходити оптимальні значення параметрів, щоб модель ефективно прогнозувала або класифікувала дані.

Існує різноманіття оптимізаторів з власними перевагами та недоліками. Деякі з найбільш поширених включають:

- Стохастичний градієнтний спуск: це варіант градієнтного спуску, який оновлює параметри моделі з частішими оновленнями.
- Градієнтний спуск: це один з найбільш популярних алгоритмів оптимізації першого порядку, що використовує похідні функції втрат для визначення зміни ваг з метою досягнення мінімуму функції.
- Adam (Adaptive Moment Estimation): це адаптивний оптимізатор, який комбінує ідеї зі зваженого середнього моменту і корекції зміщення для ефективного адаптування швидкості навчання для кожного параметра окремо.
- Adagrad (Adaptive Gradient): адаптивний оптимізатор, який враховує історію градієнтів для адаптації швидкості навчання для кожного параметра.
- Adadelta: метод стохастичного градієнтного спуску, який використовує

адаптивну швидкість навчання для кожного виміру для усунення деяких недоліків інших алгоритмів.

- Стохастичний градієнтний спуск (SGD Nesterov) з імпульсом Нестерова: це розширення традиційного алгоритму SGD, яке використовує імпульс для прискорення конвергенції.
- RMSprop (Root Mean Square Propagation): цей алгоритм також адаптує швидкість навчання для кожного параметра, використовуючи експоненційно згладжене середнє квадратичне значення градієнтів.

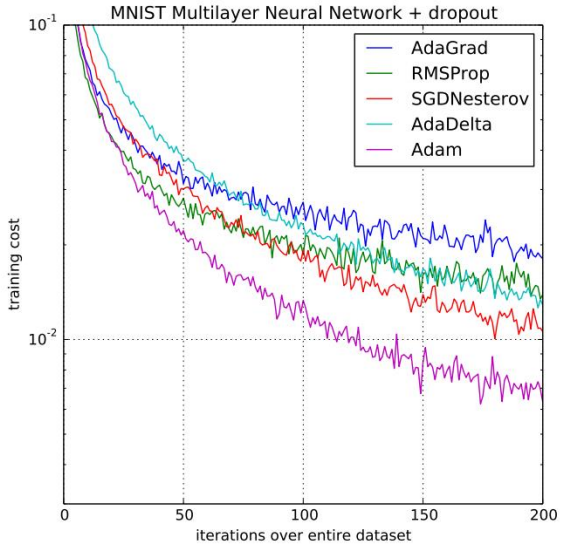


Рис. 2.5 Порівняльний аналіз деяких оптимізаторів на задачі класифікації цифр [2]

2.2.4 Оптимізатор Adam

Оптимізатор Adam (Adaptive Moment Estimation) є одним з найефективніших та широко використовуваних алгоритмів оптимізації в глибокому навчанні. Його використовують для налаштування параметрів моделі, таких як ваги, з метою мінімізувати функції втрат або максимізації функції винагороди.

Основні компоненти алгоритму Adam включають в себе:

– Швидкість навчання (Learning Rate): Adam використовує адаптивну швидкість навчання, яка автоматично адаптується для кожного параметра. Це дозволяє уникнути проблем зі занадто великими або занадто малими значеннями швидкості навчання.

– Експоненційне згладжування середнього моменту (Exponential Moving Average of Gradients): Adam обчислює зважений середній момент градієнтів. Це дозволяє алгоритму враховувати минулі градієнти і враховувати їх в оптимізаційному процесі.

– Корекція зміщення (Bias Correction): В початкових ітераціях навчання, коли ваги ще не стабілізувалися, може виникнути ситуація, коли значення середнього моменту та градієнтів будуть дуже близькі до нуля. Це може призвести до великих значень корекції. Adam вирішує цю проблему за допомогою корекції зміщення, що підраховується в кожній ітерації.

Алгоритм Adam дозволяє ефективно навчати моделі глибокого навчання з великою кількістю параметрів, забезпечуючи стійкий та швидкий процес оптимізації. Він демонструє високу ефективність та здатність адаптуватися до різних умов навчання, що робить його дуже популярним серед дослідників та практиків глибокого навчання.

2.4 Попередня обробка зображень

Попередня обробка зображень - це важлива складова процесу підготовки даних перед їх використанням у моделі глибокого навчання. Цей процес включає в себе ряд операцій, які мають на меті покращити якість та коректність вхідних даних, що подаються на вхід моделі. Основні кроки попередньої обробки зображень включають:

- Завантаження зображень: Початкові зображення завантажуються з джерела даних, такого як файлова система або база даних.
- Зменшення розміру зображень: У деяких випадках великі зображення можуть бути зменшені до менших розмірів для прискорення обробки та зменшення обсягу використовуваної пам'яті.
- Нормалізація: Зображення можуть бути нормалізовані шляхом зміни діапазону значень пікселів, зазвичай від 0 до 1 або від -1 до 1. Це допомагає моделі краще збагачувати зображення та прискорює процес навчання.

- Аугментація даних: Цей крок включає застосування різноманітних трансформацій до зображень, таких як обертання, зміщення, масштабування тощо. Це допомагає розширити набір даних та покращити загальну роботу моделі, зменшуючи перенавчання.
- Видалення шуму: У деяких випадках може бути застосована фільтрація для видалення шуму або інших артефактів зображення, які можуть вплинути на точність моделі.

Попередня обробка зображень відіграє важливу роль у підготовці даних для глибокого навчання, допомагаючи забезпечити, що модель отримує високоякісні та коректні вхідні дані для ефективного навчання.

2.5 Модель VGG19

VGG19, або Visual Geometry Group 19, представляє собою глибоку згорткову нейронну мережу, розроблену ученими з Visual Geometry Group при Оксфордському університеті. Ця модель є покращеною версією попередньої моделі VGG16 і має 19 шарів, включаючи як згорткові, так і повнозв'язані шари. Мережа навчалася на даних ImageNet, яка включає понад 14 мільйонів зображень і може класифікувати їх за 1000 категоріями.

Особливістю VGG19 є її архітектура, що складається з послідовних блоків згорткових шарів із згортками та пулінгом. Кожен блок має два або чотири згорткових шари, за якими слідує пулінговий шар. Ця структура дозволяє мережі ефективно виявляти різноманітні особливості на різних рівнях абстракції.

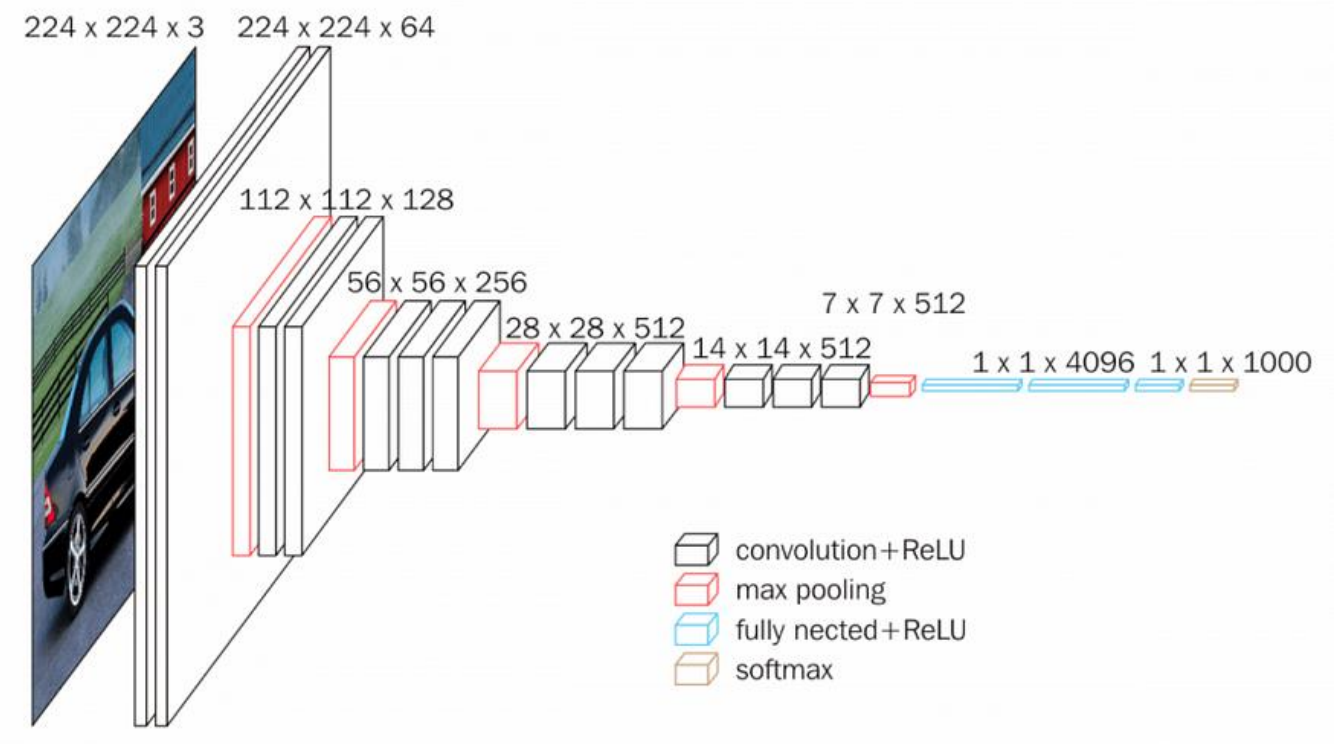


Рис. 2.6 Послідовність шарів VGG19 [1]

Архітектура VGG19 [10]:

– Зображення RGB фіксованого розміру (224 * 224) було подано як вхідні дані для цієї мережі, що означає, що матриця має форму (224,224,3).

– Була виконана попередня обробка зображень, яка полягає в відніманні середнього значення RGB з кожного пікселя, обчислене для всього навчального набору.

– Використані ядра розміром (3 * 3) з розміром кроку в 1 піксель, це дозволило охопити все поняття зображення.

– Просторове заповнення використовується для збереження просторової роздільної здатності зображення.

– Максимальне об’єднання виконано для вікон розміром 2 * 2 пікселя за допомогою sride 2.

– Після цього використана функція активації ReLU для введення нелінійності, щоб зробити модель краще класифікованою та покращити час обчислення. Попередні моделі VGG використовували функції гіперболічного тангенсу або сигмоїду, тоді як ReLU виявилася набагато кращою.

– Реалізовано три повністю пов’язані рівні, з яких перші два мають розмір 4096, а потім шар з 1000 каналами для 1000-сторонньої класифікації ILSVRC , а останній рівень – це функція softmax.

VGG19 є важливою моделлю у сфері комп'ютерного зору, вона використовується для попереднього навчання глибоких нейронних мереж в різноманітних завданнях. Її популярність полягає в простій, але ефективній архітектурі та здатності глибоко аналізувати складність зображень. Завдяки широкому застосуванню в класифікації, виявленні об'єктів і навіть у стилізації зображень за допомогою Neural Style Transfer, VGG19 стала відомою як потужний інструмент для рішення завдань комп'ютерного зору. Її глибокий стек згорткових шарів дозволяє моделі розпізнавати навіть найскладніші аспекти зображень та забезпечувати високу точність класифікації. Більш того, завдяки великій кількості попередньо навчених параметрів, VGG19 може швидко адаптуватися до нових даних і навчатися з високою ефективністю.

2.6 Висновки до розділу 2

У даному розділі ми оглянули ключові математичні концепції, які є фундаментом нашого проєкту. Ми розглянули згорткові нейронні мережі (ЗНМ), які є основою багатьох завдань у сфері обробки зображень, а також детально проаналізували архітектуру Neural Style Transfer (NST), яка дозволяє переносити стиль одного зображення на інше. Ми також дослідили оптимізатори, які грають важливу роль у навчанні моделей глибокого навчання, а також важність попередньої обробки зображень для покращення якості та швидкості навчання моделей. Нарешті, ми описали модель VGG19, яка є потужним інструментом для класифікації зображень та використовується в багатьох дослідженнях та практичних застосуваннях. Ці концепції і техніки будуть використовуватися в подальшому дослідженні та розробці нашого проєкту.

3 РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ ТА ДЕМОНСТРАЦІЯ ПРИНЦИПІВ РОБОТИ

3.1 Вибір мови програмування для проєктування системи обробки зображень

Python:

- Простота використання: Python відомий своєю простотою і читабельністю, що дозволяє швидко розробляти і тестувати нові ідеї. Його синтаксис інтуїтивно зрозумілий, що знижує час на навчання і дозволяє зосередитися на логіці алгоритмів.
- Широкий спектр бібліотек: Python має величезну кількість бібліотек, що робить його ідеальним для розробки комплексних програм. Особливо це стосується бібліотек для роботи з машинним навчанням та науковими обчисленнями, такими як NumPy, PyTorch, PIL/Pillow та багато інших.

PyCharm:

- Інтегроване середовище розробки (IDE): PyCharm є потужним IDE для Python, яке забезпечує розробників широким спектром інструментів для редагування коду, налагодження, тестування та інтеграції з системами контролю версій.
- Поліпшена продуктивність: PyCharm пропонує інтелектуальні підказки, автоматичне завершення коду, аналіз коду в реальному часі, що значно підвищує продуктивність розробника.
- Інструменти для роботи з науковими даними: PyCharm має вбудовану підтримку Jupyter Notebook, інтеграцію з бібліотеками для машинного навчання та наукових обчислень, що робить його ідеальним вибором для даного проєкту.

PyTorch:

- Реалізація та навчання нейронних мереж: PyTorch є одним з

найпопулярніших фреймворків для машинного навчання, завдяки своїй гнучкості і зручності у використанні. Він дозволяє легко реалізовувати, навчати і налагоджувати нейронні мережі.

- Динамічні обчислювальні графи: PyTorch підтримує динамічні обчислювальні графи, що значно спрощує налагодження моделей і експерименти з ними. Це дозволяє змінювати модель на льоту, не перезавантажуючи обчислювальний граф.
- Підтримка GPU: PyTorch надає прості інтерфейси для використання обчислювальних ресурсів GPU, що значно прискорює обробку даних і навчання моделей.

NumPy:

- Обробка даних та матриць: NumPy є основною бібліотекою для роботи з масивами даних у Python. Вона забезпечує швидке і ефективне оброблення числових даних, що є основою для багатьох алгоритмів машинного навчання.
- Широкий набір функцій: NumPy пропонує багатий набір функцій для математичних обчислень, статистики, лінійної алгебри і багатьох інших задач, що робить її незамінною у наукових і інженерних дослідженнях.

PIL (Python Imaging Library) / Pillow:

- Обробка та маніпуляція зображеннями: PIL, а точніше її сучасний форк Pillow, використовується для завантаження, обробки і збереження зображень. Вона дозволяє легко маніпулювати зображеннями: змінювати розмір, обрізати, конвертувати між різними форматами і багато іншого.
- Функції зображення: Pillow надає потужний набір інструментів для обробки зображень, що є важливим для підготовки даних перед передачею їх у нейронні мережі.

Опис коду

Огляд програми Neural Style Transfer

Код реалізує програму для переносу стилю з одного зображення на інше за допомогою нейронних мереж. Програма використовує інструменти Python, PyTorch, NumPy та PIL/Pillow для обробки зображень, екстракції ознак і

оптимізації результату. Для створення графічного інтерфейсу використовується бібліотека Tkinter.

Основні компоненти програми:

Інтерфейс користувача (GUI):

- Tkinter: Tkinter використовується для створення простого і зручного графічного інтерфейсу. Інтерфейс дозволяє користувачам завантажувати зображення, вказувати параметри і запускати процес переносу стилю.
- Компоненти інтерфейсу: GUI складається з кнопок для завантаження контентних і стилевих зображень, поля для введення кількості ітерацій, кнопки запуску процесу і відображення проміжних результатів.

Завантаження та обробка зображень:

- PIL/Pillow: Ця бібліотека використовується для завантаження зображень з файлової системи та їх попередньої обробки.
- Torchvision.transforms: Використовується для перетворення зображень у формат, придатний для обробки нейронною мережею (зміна розміру, перетворення у тензор, нормалізація).

Модель та обчислення:

- VGG19: Використовується попередньо навчена модель VGG19 з PyTorch для екстракції ознак зображень. Ця модель добре підходить для задач переносу стилю завдяки своїй глибокій архітектурі.
- Gram Matrix: Використовується для обчислення стилевих ознак зображень. Це дозволяє порівнювати стиль різних зображень шляхом обчислення їх кореляційних матриць.

Перенесення стилю:

- Процес переносу стилю включає кілька ітерацій оптимізації зображення, щоб воно набуло стилю стилевого зображення, зберігаючи при цьому вміст контентного зображення.
- Оптимізація виконується за допомогою методу градієнтного спуску (Adam optimizer). На кожній ітерації розраховується втрата (loss), яка складається з втрат контенту і стилю, після чого ваги моделі оновлюються для зменшення

цієї втрати.

Відображення результатів:

- Tkinter використовується для відображення проміжних результатів у процесі переносу стилю, а також фінального результату. Це дозволяє користувачам в реальному часі спостерігати за процесом і бачити, як зображення змінюється з кожною ітерацією.
-

3.2 Архітектура розробленої програми

Для розробки програми використано об'єктно-орієнтований підхід, що дозволяє розділити функціонал на різні класи та модулі, забезпечуючи модульність, підтримуваність і розширюваність коду.

Класова структура

Основним класом програми є NeuralStyleTransferApp, який включає всі необхідні компоненти для виконання переносу стилю. Ось детальний опис архітектури та функціоналу кожного методу і функції.

Клас NeuralStyleTransferApp

Клас NeuralStyleTransferApp відповідає за реалізацію графічного інтерфейсу користувача (GUI), завантаження зображень, запуск процесу переносу стилю і відображення результатів. Він містить такі основні методи:

1. init:

- Ініціалізує графічний інтерфейс і компоненти, такі як кнопки, мітки, поля введення та інші елементи.
- Завантажує попередньо навчену модель VGG19 і переводить її у режим оцінки.
- Налаштовує пристрій для обчислень (GPU або CPU).

```
def __init__(self, master):
    self.master = master
    master.title("Neural Style Transfer")
    self.content_image = None
    self.style_image = None
    self.device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    self.model = models.vgg19(pretrained=True).features.to(self.device).eval()
    for param in self.model.parameters():
        param.requires_grad_(False)
    self.mean = torch.tensor([0.485, 0.456, 0.406]).to(self.device)
    self.std = torch.tensor([0.229, 0.224, 0.225]).to(self.device)
    self.setup_ui()
```

Рис. 3.1 Приклад коду

2. load_content_image:

– Відповідає за завантаження контентного зображення через файловий діалог і підготовку його для подальшої обробки.

```
def load_content_image(self):
    content_path = filedialog.askopenfilename()
    if content_path:
        self.content_image = self.load_image(content_path)
        self.display_image(self.content_image, self.content_image_label)
```

Рис. 3.2 Приклад коду

3. load_style_image:

– Відповідає за завантаження стилевого зображення через файловий діалог і підготовку його для подальшої обробки.

```
def load_style_image(self):
    style_path = filedialog.askopenfilename()
    if style_path:
        self.style_image = self.load_image(style_path)
        self.display_image(self.style_image, self.style_image_label)
```

Рис. 3.3 Приклад коду

4. load_image:

- Виконує завантаження і попередню обробку зображень (зміна розміру, перетворення у тензор і нормалізація).

```
def load_image(self, path):
    image = Image.open(path).convert('RGB')
    image = transforms.Compose([
        transforms.Resize((300, 300)),
        transforms.ToTensor(),
        transforms.Normalize(self.mean.cpu(), self.std.cpu())
    ])(image).unsqueeze(0)
    return image.to(self.device, torch.float)
```

Рис. 3.4 Приклад коду

5. display_image:

- Відповідає за відображення зображень у графічному інтерфейсі.

```
def display_image(self, image, label):
    image = image.squeeze(0).detach().cpu().numpy()
    image = np.transpose(image, (1, 2, 0))
    image = image * np.array(self.std.cpu()) + np.array(self.mean.cpu())
    image = np.clip(image, 0, 1)
    image = Image.fromarray((image * 255).astype(np.uint8))
    image = ImageTk.PhotoImage(image)
    label.configure(image=image)
    label.image = image
```

Рис. 3.5 Приклад коду

6. get_features:

- Використовує модель VGG19 для екстракції ознак зображень на певних шарах.

```
def get_features(self, image, model, layers=None):
    if layers is None:
        layers = {'0': 'conv1_1',
                  '5': 'conv2_1',
                  '10': 'conv3_1',
                  '19': 'conv4_1',
                  '21': 'conv4_2',
                  '28': 'conv5_1'}
    features = {}
    x = image
    for name, layer in model._modules.items():
        x = layer(x)
        if name in layers:
            features[layers[name]] = x
    return features
```

Рис. 3.6 Приклад коду

7. gram_matrix:

- Обчислює матрицю Грама для заданого тензора ознак, що використовується для визначення стилевих ознак.

```
def gram_matrix(self, tensor):
    _, d, h, w = tensor.size()
    tensor = tensor.view(d, h * w)
    gram = torch.mm(tensor, tensor.t())
    return gram
```

Рис. 3.7 Приклад коду

8. start_neural_style_transfer:

- Запускає процес переносу стилю, виконує оптимізацію параметрів зображення для зменшення загальної втрати.
- Виконує ітерації оптимізації, оновлюючи зображення і відображаючи проміжні результати.

9.Завантаження готового результату:

- Додано кнопку "Load Result", яка дозволяє завантажувати готовий результат

переносу стилю.

- При кліку на цю кнопку викликається функція load_result_image, яка завантажує зображення та відображає його на інтерфейсі.

```
def load_result_image(self):
    result_path = filedialog.askopenfilename()
    if result_path:
        result_image = self.load_image(result_path)
        self.display_image(result_image, self.content_image_label)
```

Рис. 3.8 Приклад коду

3.3 Демонстрація результатів роботи

Під час розробки програми для перенесення стилю я зрозумів, наскільки важливою є врахування якості та характеристик вихідних фотографій. Ось кілька ключових відомостей, які я зробив:

1. Якість вихідних фотографій: Початкові фотографії повинні бути якісними та добре освітленими. Чим краща якість фото, тим більше деталей та текстур можна відтворити під час перенесення стилю.
2. Характер стилю та вибір моделі: Вибір стилю впливає на передачу кольору та текстури. Деякі моделі можуть бути кращі у відтворенні певних аспектів стилю, тому важливо вибрати відповідну модель для конкретного завдання.
3. Параметри оптимізатора та ваги: Налаштування параметрів оптимізатора та ваг, що використовуються у процесі навчання, є критичними для досягнення високої якості результату. Вони впливають на швидкість збіжності та стабільність результату.
4. Експерименти та корекції: Під час розробки програми я експериментував з різними налаштуваннями та проводив корекції для покращення якості результату. Це включало в себе зміну параметрів оптимізатора, ваг, вибір інших моделей та врахування характеристик вихідних фотографій.

Програма використовує фотографію кімнати як вхідне (основне) зображення та дозволяє користувачеві вибрати різні стилі для застосування до цього зображення. Для кожного обраного стилю ми налаштовуємо ваги, що визначають важливість кожного шару у втраті стилю. Наприклад, для деяких стилів важливіше лише перші кілька шарів мережі, тоді як для інших стилів враховуються всі шари.

Після застосування кожного стилю ми оцінюємо результат та аналізуємо його, налаштовуючи ваги та параметри оптимізатора Adam за потреби. Наприклад, ми можемо змінювати швидкість навчання (learning rate) для досягнення більш стабільних або швидких збіжностей, а також можемо регулювати інші параметри оптимізатора для оптимізації процесу.

Цей ітеративний процес дозволяє нам експериментувати з різними комбінаціями ваг та параметрів оптимізатора з метою знайти оптимальні налаштування, які найкраще підкреслять стиль та зміст зображення кімнати, щоб створити кінцевий результат, що відповідає нашим потребам і вимогам.

На першому зображенні ми можемо побачити ліворуч оригінальне зображення кімнати, а праворуч — його оброблену версію, (рис. 3.9) до якої застосований обраний стиль. Це дозволяє нам порівняти, яким чином обробка впливає на вигляд початкового зображення.

На другому зображенні відображений сам стиль, який ми намагалися нанести на вхідне зображення. Проте, можна помітити, що програма на даному етапі працює неідеально через некоректні налаштування ваг, що відповідають за врахування текстур, а також через невірно вибрані параметри оптимізатора Adam. (рис. 3.10)

Ця ситуація вказує на необхідність відкоригувати налаштування програми, а саме - переглянути та відкоригувати ваги, які визначають важливість різних аспектів стилю, а також налаштувати параметри оптимізатора Adam для досягнення кращих результатів.

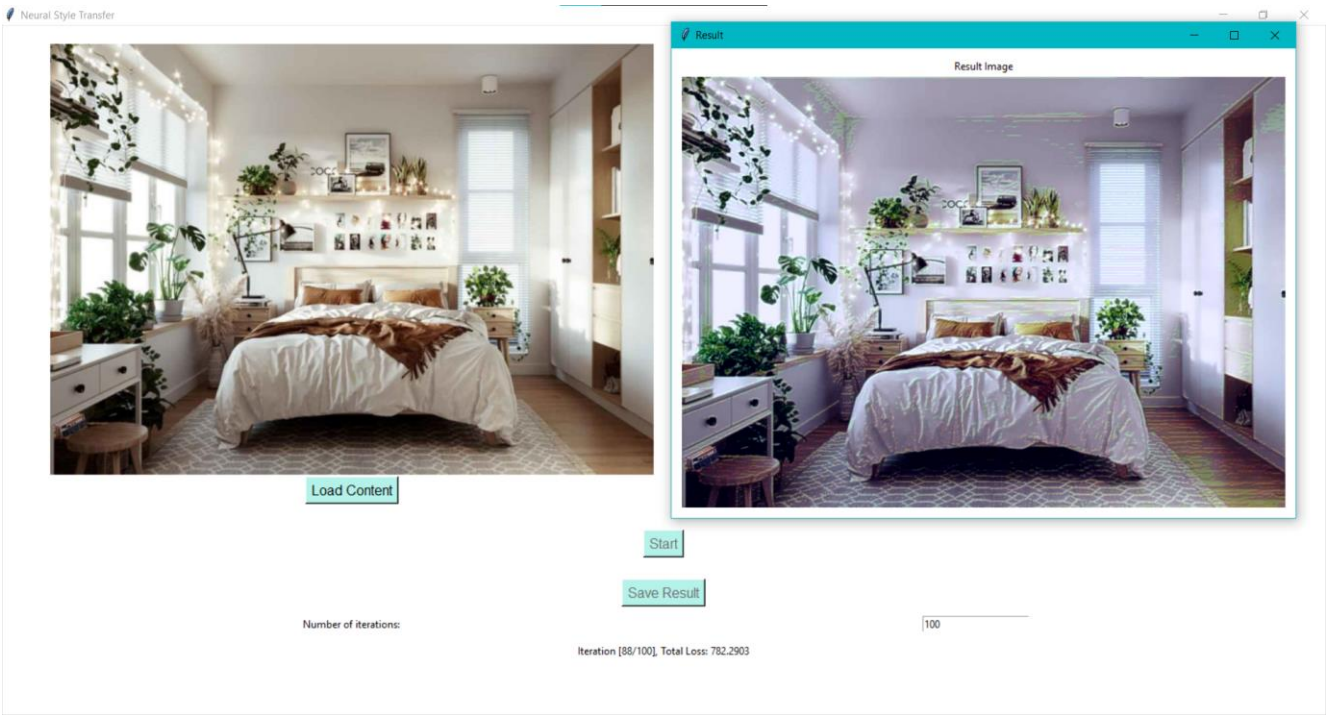


Рис. 3.9 Приклад роботи програми

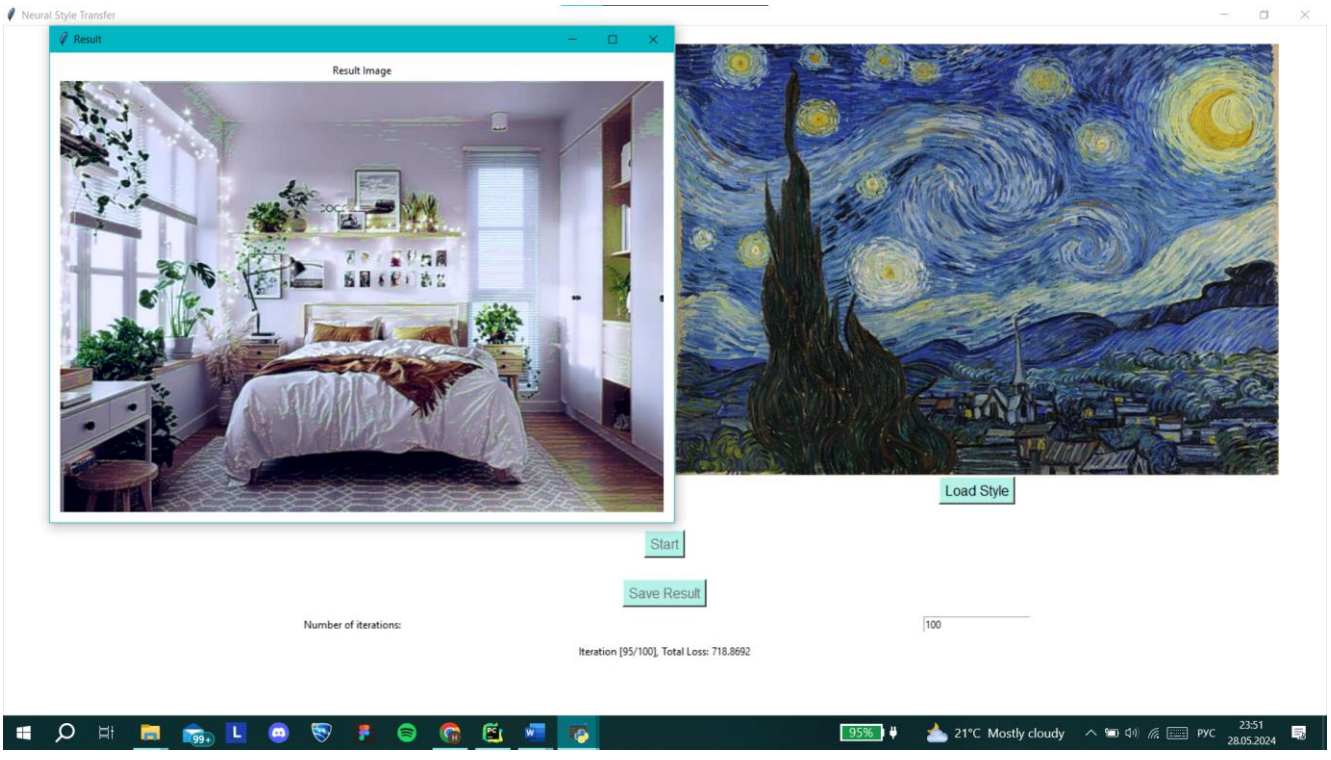


Рис. 3.10 Приклад роботи програми

На наступних фотографіях ми бачимо, як нейронна система майстерно доповнила наші зображення відповідним стилем (рис. 3.11-3.13). Часто для досягнення цього було потрібно знизити швидкість оптимізатора Adam та

відповідно змінити параметри ваг, які відповідають за кольорову гаму. Деякі ваги, що відповідають за текстуру, на цьому етапі могли бути зменшені або навіть виключені з процесу оптимізації. Також часто для досягнення оптимального результату було достатньо від 200 до 300 ітерацій, після яких зображення набувало бажаний стиль та вигляд.

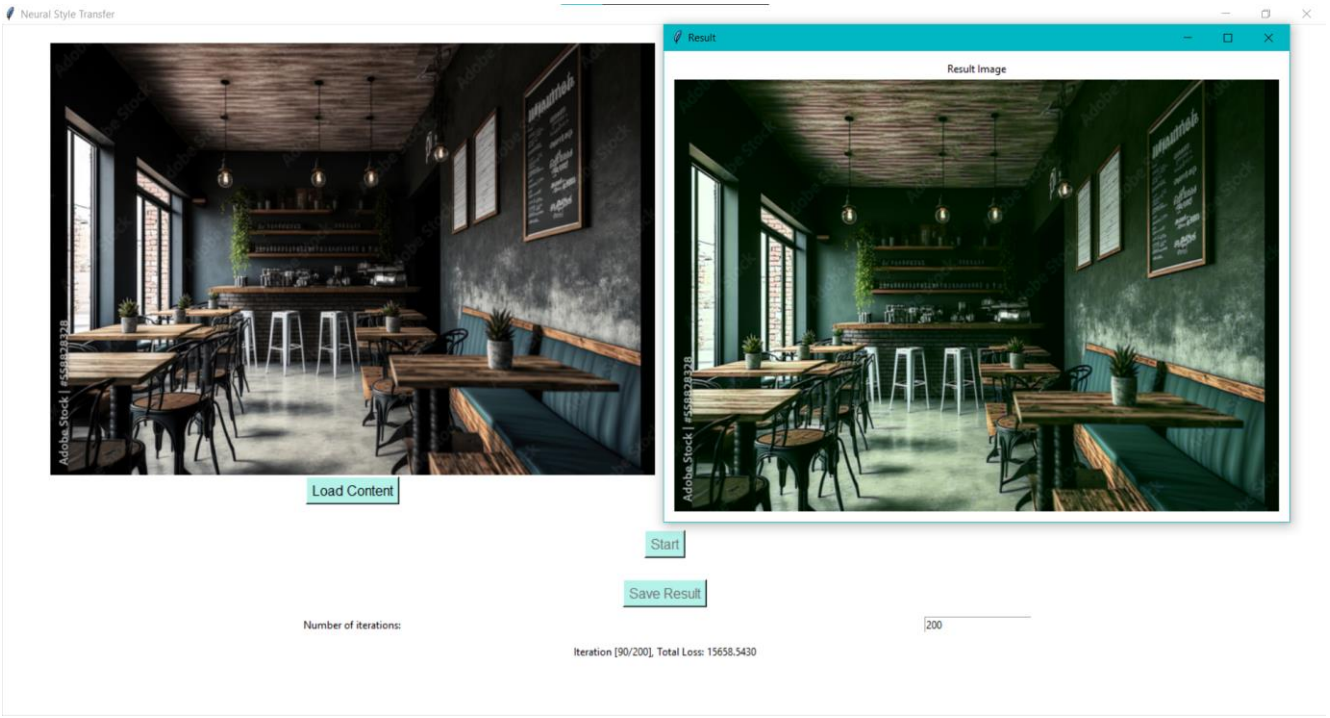


Рис. 3.11 Приклад роботи програми

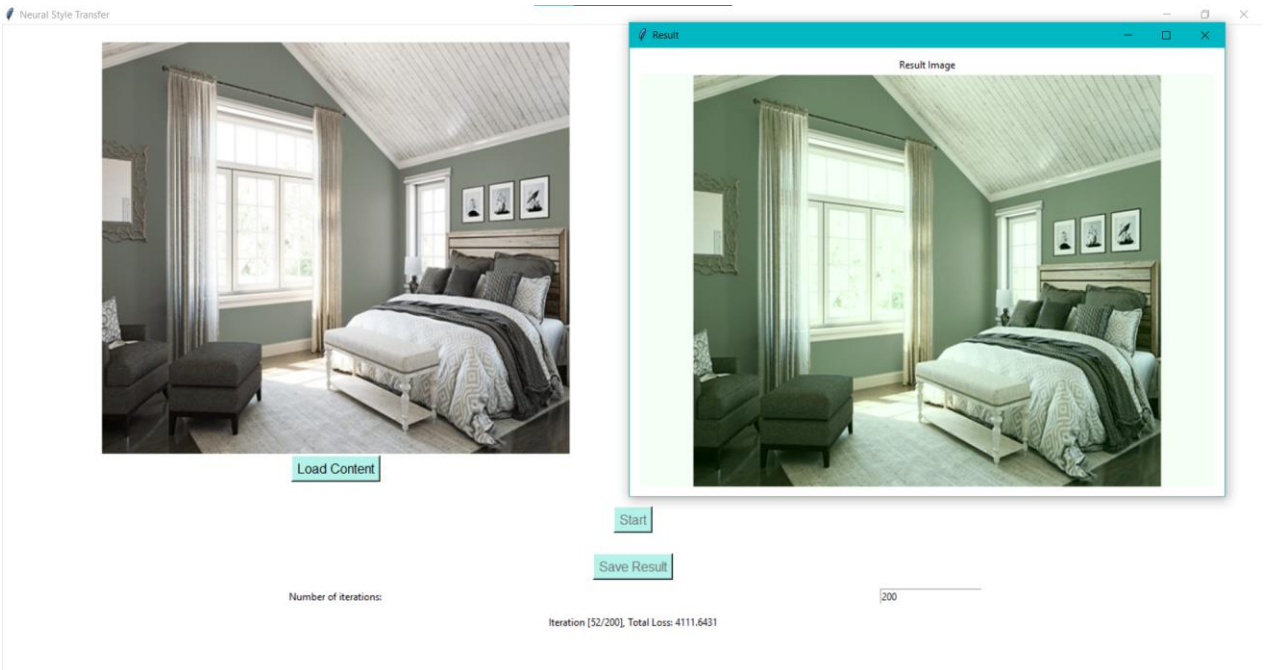


Рис. 3.12 Приклад роботи програми

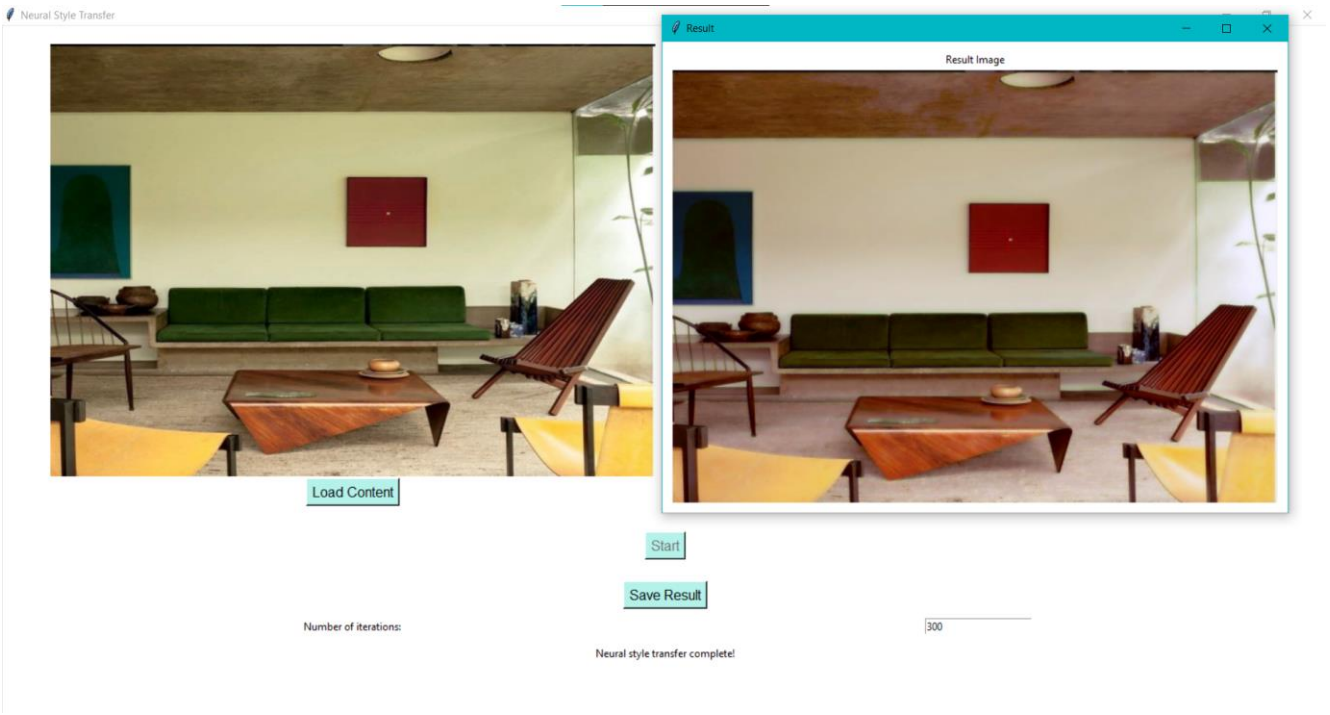


Рис. 3.13 Приклад роботи програми

3.4 Приклад застосування програми

Користувачі в Instagram можуть використовувати програму для створення цікавих та естетичних фільтрів для своїх фотографій. За допомогою цієї програми вони можуть застосовувати різноманітні стилі до своїх зображень, щоб надати їм унікального вигляду та виразності.

Наприклад, користувач може застосувати фільтр, який переносить стиль відомих художників або фотографічних технік до своїх фотографій. Це може створити захоплюючі та запам'ятовувальні зображення, які відзначаються оригінальністю та творчістю.

Приклад застосування програми може включати генерацію зображень у різних стилях та їх подальше використання у фото та відео редакторах, маркетингових матеріалах та ігрових додатках. Наприклад, ці зображення можуть бути використані для створення унікального контенту для соціальних мереж, рекламних кампаній або ілюстрацій для ігор. Крім того, користувачі можуть використовувати програму для створення зображень, які відображають їхні унікальні інтереси, стиль та особистість. Завдяки різноманітним фільтрам та

стилям, вони можуть експериментувати з естетикою своїх зображень, надаючи їм особливий шар креативності.

Це також може стати потужним інструментом для брендів та маркетингових агентств. Вони можуть використовувати програму для створення унікального візуального контенту, який відображає їхню брендову ідентичність та привертає увагу аудиторії. Такий контент може бути використаний для створення привабливих рекламних постів, інформаційних графіків та іншого рекламного матеріалу.

Загалом, програма надає користувачам можливість створювати не лише красиві зображення, але й вражаючий візуальний контент, що відображає їхню унікальну творчість та стиль.

3.5 Висновки до розділу 3

У розділі 3 ми провели розгляд розробки програмного продукту для обробки зображень за допомогою методів машинного навчання. Було надано опис архітектури програми та її функціоналу, що дозволяє користувачам застосовувати різноманітні стилі до своїх фотографій. Ми також продемонстрували результати роботи програми, які включали у себе створення зображень у різних стилях та їхнє подальше використання у фото та відео редакторах, маркетингових матеріалах та ігрових додатках.

Результати роботи показали, що програма здатна ефективно застосовувати стилі до фотографій, що дозволяє користувачам створювати вражаючі та ефектні зображення, які привертають увагу та відзначаються оригінальністю. Крім того, програма може бути використана для створення унікального візуального контенту для соціальних мереж, рекламних кампаній та ігор, що робить її потужним інструментом для користувачів з різноманітними потребами та цілями.

Загалом ми краще розібрали в процесі розробки програмного продукту для обробки зображень за допомогою методів машинного навчання, а також продемонстрували його потенціал та можливості.

ВИСНОВКИ

Застосування глибокого навчання у генерації зображень стає все більш актуальним, відкриваючи нові можливості у таких сферах, як фото- та відеоредакція, ігрова індустрія, віртуальна реальність, маркетинг і дизайн. В даному дослідженні ми оглянули техніки обробки зображень, зосереджуючись на Neural Style Transfer (NST), що дозволяє комбінувати стиль одного зображення з вмістом іншого. Ми також розглянули різноманітні сфери застосування NST та сформулювали задачу його застосування у контексті генерації зображень за допомогою глибокого навчання.

У другому розділі ми описали математичні основи роботи, включаючи архітектуру згорткових нейронних мереж, використання оптимізаторів та попередню обробку зображень. Особлива увага була приділена моделі VGG19, яка використовується для виявлення ознак у NST.

У третьому розділі ми представили розроблений програмний продукт, описали його архітектуру та вибір інструментів розробки. Також була проведена демонстрація результатів роботи програми, що включала приклади застосування та згенеровані зображення, що підтверджує ефективність та потенціал використання NST у практичних завданнях.

ПЕРЕЛІК ПОСИЛАНЬ

1. A visual representation of a convolutional layer [Електронний ресурс]. Режим доступу: https://www.researchgate.net/figure/A-visual-representation-of-aconvolutional-layer-The-centre-element-of-the-kernel_is_fig6_285164623 (date of access: 24.04.2024).

2. Adaptive Moment Estimation (Adam), [Електронний ресурс]. Режим доступу: https://ml-explained.com/blog/adam-explained (date of access: 20.04.2024).

3. Ai-tern, Сегментація зображень засобами OpenCV [Електронний ресурс]. Режим доступу: https://ai-tern.in.ua/Segmentation.html (date of access: 28.04.2024).

4. Bandyopadhyay R., Kundu R., Elaziz M. A., & Sarkar, R. (2021). Segmentation of brain MRI using an altruistic Harris Hawks' Optimization algorithm. Knowledge Based Systems, 232, 107468. https://doi.org/10.1016/j.knosys.2021.107468 (date of access: 14.04.2024).

5. Deepai, Documentation [Електронний ресурс]. Режим доступу: https://deepai.org/docs (date of access: 24.04.2024).

6. DEEP DREAM GENERATOR, About Deep Dream Generator (DDG) [Електронний ресурс]. Режим доступу: https://deepdreamgenerator.com/about (date of access: 20.04.2024).

7. DOU.ua, Використовуємо CNN для обробки зображень. Частина перша [Електронний ресурс]. Режим доступу: https://dou.ua/forums/topic/48368/ (date of access: 24.04.2024).

8. Guru99, Що таке Autoencoder у Deep Learning? [Електронний ресурс]. Режим доступу: https://www.guru99.com/uk/autoencoder-deep-learning.html (date of access: 24.04.2024).

9. Max pooling in Convolutional neural network [Електронний ресурс]. Режим доступу: https://matlab1.com/max-pooling-in-convolutional-neural-network/ (date of access: 28.04.2024).

10. Medium, VGG Net Architecture [Електронний ресурс]. Режим доступу: https://medium.com/@sajals1146/vgg-net-architecture-880df59c37ea (date of access: 20.04.2024).

11. Neural Style Transfer [Електронний ресурс]. Режим доступу: https://www.v7labs.com/blog/neural-style-transfer (date of access: 28.04.2024).

12. NVIDIA Developer [Електронний ресурс]. Режим доступу: https://developer.nvidia.com/rtx/dlss (date of access: 25.04.2024).

13. Old Photo Restoration Online with AI [Електронний ресурс]. Режим доступу: https://vanceai.com/old-photo-restoration/ (date of access: 18.04.2024).

14. ResearchGate, [Електронний ресурс]. Режим доступу: https://www.researchgate.net/figure/The-Gram-Matrix-is-created-from-a-target-image-and-a-reference-image_fig4_356667127 (date of access: 20.04.2024).

15. Tehkontrol, Система ADAS сучасна допомога водію [Електронний ресурс]. Режим доступу: https://tehkontrol.com/ua/adas-system/ (date of access: 20.04.2024).

16. The Complete Guide to Generative Adversarial Networks [Електронний ресурс]. Режим доступу: https://www.v7labs.com/blog/generative-adversarial-networksguide (date of access: 20.04.2024).

17. VGG Net Architecture, [Електронний ресурс]. Режим доступу: https://medium.com/@sajals1146/vgg-net-architecture-880df59c37ea (date of access: 24.04.2024).

18. YourArt, Coca-Cola випустила рекламу з роботами Воргола, Мунка й Ван Гога [Електронний ресурс]. Режим доступу: https://supportyourart.com/news/coca-cola-vypustyla-reklamu-z-vorgola-munka-j-van-goga/ (date of access: 24.04.2024).

19. СТРАТЕГІЇ КІБЕРСТІЙКОСТІ: УПРАВЛІННЯ РИЗИКАМИ ТА БЕЗПЕРЕРВНІСТЬ БІЗНЕСУ / V Науково-технічна конференція «Сучасний стан та перспективи розвитку IoT» Збірник тез. – К.: ДУІКТ, 2024р., с. 51-53

20. АВТОМАТИЗАЦІЯ БІЗНЕС-ПРОЦЕСІВ: ПРАКТИЧНИЙ ДОСВІД ВПРОВАДЖЕННЯ РІШЕНЬ HEWLETT PACKARD ENTERPRISE В КОНТЕКСТІ

ПІДПРИЄМНИЦЬКОЇ ЕФЕКТИВНОСТІ ВІДЕО / Науково-практична конференція «Сучасні досягнення компанії HEWLETT PACKARD ENTERPRISE в галузі ІТ та нові можливості їх вивчення і застосування». Збірник тез. – К.: ДУІКТ, 2023р., с. 20-23+

ДОДАТОК А. ПРОГРАМНИЙ КОД ПРОЄКТОВАНОЇ СИСТЕМИ

```
import tkinter as tk
from tkinter import filedialog, messagebox
from PIL import Image, ImageTk
import numpy as np
import torch
import torch.optim as optim
from torchvision import transforms, models

class ResultWindow(tk.Toplevel):
    def __init__(self, master=None):
        super().__init__(master)
        self.title("Result")
        self.configure(bg="#FFFFFF")

        # Frame for displaying the result image
        self.result_frame = tk.Frame(self, bg="#FFFFFF")
        self.result_frame.pack(padx=10, pady=10)

        self.result_image_label = tk.Label(self.result_frame,
text="Result Image", bg="#FFFFFF")
        self.result_image_label.pack()

        self.result_image_canvas = tk.Label(self.result_frame,
bg="#FFFFFF")
        self.result_image_canvas.pack()

class NeuralStyleTransferApp:
    def __init__(self, master):
        self.master = master
        master.title("Neural Style Transfer")
        master.configure(bg="#FFFFFF")  # Set background color to
white

        # Frame for loading images
        self.input_frame = tk.Frame(master, bg="#FFFFFF")
        self.input_frame.pack(padx=10, pady=10)

        # Content image frame and components
        self.content_image_frame = tk.Frame(self.input_frame,
bg="#FFFFFF")
```

```
        self.content_image_frame.grid(row=0, column=0, padx=10,
pady=10)

        self.content_image_label =
tk.Label(self.content_image_frame, text="Content Image",
bg="#FFFFFF")
        self.content_image_label.pack()

        self.load_content_button =
tk.Button(self.content_image_frame, text="Load Content",

command=self.load_content_image, bg="#B5F2EA", font=("Arial", 12))
        self.load_content_button.pack()

        # Style image frame and components
        self.style_image_frame = tk.Frame(self.input_frame,
bg="#FFFFFF")
        self.style_image_frame.grid(row=0, column=1, padx=10,
pady=10)

        self.style_image_label = tk.Label(self.style_image_frame,
text="Style Image", bg="#FFFFFF")
        self.style_image_label.pack()

        self.load_style_button = tk.Button(self.style_image_frame,
text="Load Style", command=self.load_style_image,
                                           bg="#B5F2EA",
font=("Arial", 12))
        self.load_style_button.pack()

        # Start button
        self.start_button = tk.Button(self.input_frame,
text="Start", command=self.start_neural_style_transfer,
                                      bg="#B5F2EA", font=("Arial",
12))
        self.start_button.grid(row=1, column=0, columnspan=2,
pady=20)

        # Save Result button
        self.save_result_button = tk.Button(self.input_frame,
text="Save Result", command=self.save_result_image,
                                            bg="#B5F2EA",
font=("Arial", 12))
```

```
        self.save_result_button.grid(row=2, column=0, columnspan=2,
pady=5)
        self.save_result_button.config(state="disabled")  #
Initially disable the button

        # Iterations label and entry
        self.iterations_label = tk.Label(self.input_frame,
text="Number of iterations:", bg="#FFFFFF")
        self.iterations_label.grid(row=3, column=0, pady=5)

        self.iterations_entry = tk.Entry(self.input_frame)
        self.iterations_entry.grid(row=3, column=1, pady=5)
        self.iterations_entry.insert(0, "800")  # Default value

        # Progress label
        self.progress_label = tk.Label(self.input_frame, text="",
bg="#FFFFFF")
        self.progress_label.grid(row=4, column=0, columnspan=2,
pady=5)

        # Initialize image variables
        self.content_image = None
        self.style_image = None
        self.result_image = None  # Variable to store the processed
image

        # Device (GPU/CPU)
        self.device = torch.device("cuda" if
torch.cuda.is_available() else "cpu")

        # Load pretrained VGG19 model
        self.model =
models.vgg19(pretrained=True).features.to(self.device).eval()
        for param in self.model.parameters():
            param.requires_grad_(False)

        # Means and standard deviations for image normalization
        self.mean = torch.tensor([0.485, 0.456,
0.406]).to(self.device)
        self.std = torch.tensor([0.229, 0.224,
0.225]).to(self.device)

    def load_content_image(self):
        content_path = filedialog.askopenfilename()
```

```
        if content_path:
            self.content_image = self.load_image(content_path)
            self.display_image(self.content_image,
self.content_image_label)

    def load_style_image(self):
        style_path = filedialog.askopenfilename()
        if style_path:
            self.style_image = self.load_image(style_path)
            self.display_image(self.style_image,
self.style_image_label)

    def load_image(self, path):
        image = Image.open(path).convert('RGB')
        image = transforms.Compose([
            transforms.Resize((500, 700)),
            transforms.ToTensor(),
            transforms.Normalize(self.mean.cpu(), self.std.cpu())
        ])(image).unsqueeze(0)
        return image.to(self.device, torch.float)

    def display_image(self, image, label):
        image = image.squeeze(0).detach().cpu().numpy()
        image = np.transpose(image, (1, 2, 0))
        image = image * np.array(self.std.cpu()) +
np.array(self.mean.cpu())
        image = np.clip(image, 0, 1)
        image = Image.fromarray((image * 255).astype(np.uint8))
        image = ImageTk.PhotoImage(image)
        label.configure(image=image)
        label.image = image

    def get_features(self, image, model, layers=None):
        if layers is None:
            layers = {'0': 'conv1_1',
                      '5': 'conv2_1',
                      '10': 'conv3_1',
                      '19': 'conv4_1',
                      '21': 'conv4_2',
                      '28': 'conv5_1'}
        features = {}
        x = image
        for name, layer in model._modules.items():
            x = layer(x)
```

```
            if name in layers:
                features[layers[name]] = x
        return features

    def gram_matrix(self, tensor):
        _, d, h, w = tensor.size()
        tensor = tensor.view(d, h * w)
        gram = torch.mm(tensor, tensor.t())
        return gram

    def start_neural_style_transfer(self):
        print("Starting neural style transfer...")
        self.progress_label.config(text="Starting neural style
transfer...")
        self.master.update()
        try:
            num_iterations = int(self.iterations_entry.get())
        except ValueError:
            messagebox.showerror("Error", "Number of iterations must
be an integer.")
            return

        self.start_button.config(state="disabled")  # Disable the
Start button during processing

        # Start the style transfer process
        if self.content_image is None or self.style_image is None:
            messagebox.showerror("Error", "Please load both content
and style images.")
            self.start_button.config(state="normal")  # Return
button to its original state
            return

        content_features = self.get_features(self.content_image,
self.model)
        style_features = self.get_features(self.style_image,
self.model)
        style_grams = {layer:
self.gram_matrix(style_features[layer]) for layer in style_features}

        target =
self.content_image.clone().requires_grad_(True).to(self.device)
        optimizer = optim.Adam([target], lr=0.001)
```

```
        style_weights = {'conv1_1': 0.3, 'conv2_1': 0.0, 'conv3_1':
0.0, 'conv4_1': 0.0, 'conv5_1': 0.0}

        # Create a new ResultWindow to display the result image
        self.result_window = ResultWindow(self.master)

        def style_transfer_iteration(iteration):
            if iteration < num_iterations:
                target_features = self.get_features(target,
self.model)
                content_loss =
torch.mean((target_features['conv4_2'] -
content_features['conv4_2']) ** 2)

                style_loss = 0
                for layer in style_weights:
                    target_feature = target_features[layer]
                    _, d, h, w = target_feature.shape
                    target_gram = self.gram_matrix(target_feature)
                    style_gram = style_grams[layer]
                    layer_style_loss = style_weights[layer] *
torch.mean((target_gram - style_gram) ** 2) / (d * h * w)
                    style_loss += layer_style_loss

                total_loss = content_loss + 1e2 * style_loss

                optimizer.zero_grad()
                total_loss.backward()
                optimizer.step()

                self.progress_label.config(text=f"Iteration
[{iteration}/{num_iterations}], Total Loss:
{total_loss.item():.4f}")
                self.display_image(target,
self.result_window.result_image_canvas)  # Display image in the
result window
                self.master.update()
                self.master.after(1, style_transfer_iteration,
iteration + 1)
            else:
                self.result_image = target  # Store the final result
                self.progress_label.config(text="Neural style
transfer complete!")
```

```
                self.save_result_button.config(state="normal")  #
Enable Save Result button
                print("Neural style transfer complete!")

        style_transfer_iteration(0)  # Start the iteration

    def save_result_image(self):
        if self.result_image is not None:
            save_path =
filedialog.asksaveasfilename(defaultextension=".jpg")
            if save_path:
                result_image =
self.result_image.squeeze(0).detach().cpu().numpy()
                result_image = np.transpose(result_image, (1, 2, 0))
                result_image = result_image *
np.array(self.std.cpu()) + np.array(self.mean.cpu())
                result_image = np.clip(result_image, 0, 1)
                result_image = Image.fromarray((result_image *
255).astype(np.uint8))
                result_image.save(save_path)
                messagebox.showinfo("Info", "Result image saved
successfully.")
        else:
            messagebox.showerror("Error", "No result image to
save.")

root = tk.Tk()
app = NeuralStyleTransferApp(root)
root.mainloop()
```

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «**Розробка комп'ютерного додатку генерації зображень з використанням мереж глибокого навчання**»

Виконав: здобувач вищої освіти гр. ШІД-41
Кубар Нікіта Максимович
Керівник: Кисіль Тетяна

2024 рік

Постановка задачі

Мета роботи: підвищення якості системи по генерації зображень на основі методів машинного навчання та покращення якості, ефективності обробки.

Об'єкт дослідження: процеси генерації цифрових зображень.

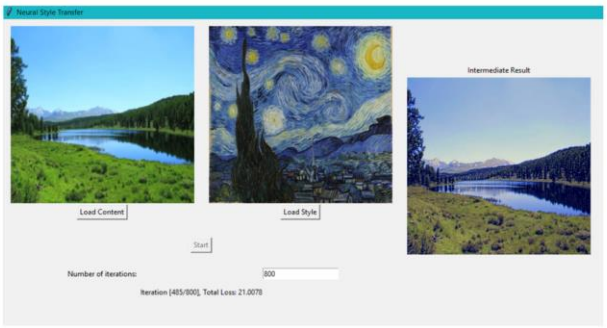
Предмет дослідження: методи глибокого машинного навчання на основі згорткової нейронної мережі VGG19.

Основні завдання кваліфікаційної роботи:

1.Реалізація алгоритму перенесення стилю зображень.
2.Розробка інтерактивного інтерфейсу користувача.
3.Оптимізація та розширення функціональності програми.

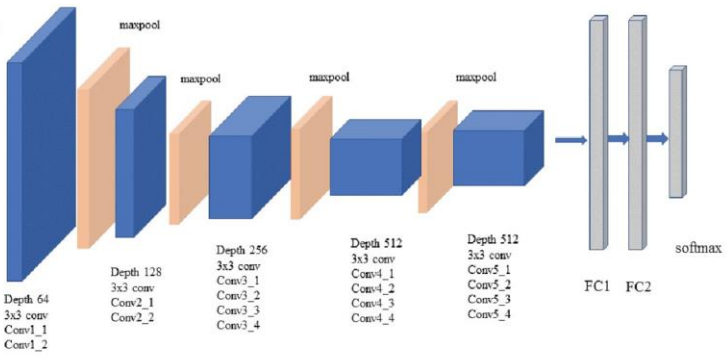
Опис процесу роботи програми

1.Завантажуються контентне та стильове зображення.
2.Вилучаються характеристики зображень за допомогою попередньо навченої моделі VGG19.
3.Обчислюються матриці Грама для шарів стильового зображення.
4.Ініціалізується цільове зображення (копія контентного), яке буде оптимізоване для мінімізації втрат контенту та стилю.
5.Задається функція втрат, включаючи втрати контенту та стилю.
6.Виконується оптимізація цільового зображення за допомогою градієнтного спуску для мінімізації загальної втрати.
7.Зображення оновлюється на кожній ітерації, і процес повторюється до досягнення заданої кількості ітерацій.



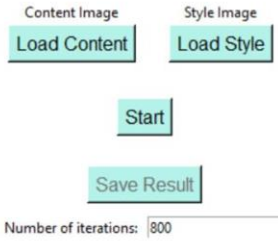
Опис нейронної мережі VGG19

VGG19 - це глибока згорткова нейронна мережа, розроблена командою Visual Geometry Group в Оксфордському університеті. Вона є покращеною версією моделі VGG16 і складається з 19 шарів, включаючи згорткові та повнозв'язні. Навчена на базі даних ImageNet, мережа здатна класифікувати зображення за 1000 категоріями. Основна особливість VGG19 - її архітектура, яка має послідовні блоки згорткових шарів із об’єднуючими нормалізаційними шарами, що дозволяє ефективно виявляти особливості на різних рівнях абстракції.

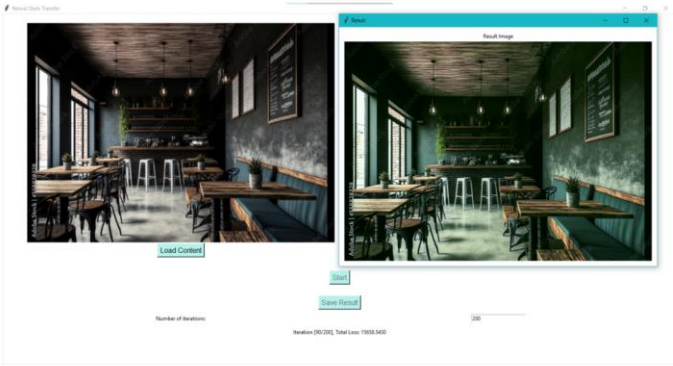


Принцип роботи розробленої програми

1.Користувач бачить вікно інтерфейсу з мітками "Content Image" та "Style Image" для завантаження відповідних зображень.
2.Натискаючи кнопку "Load Content" чи "Load Style", відкривається вікно вибору файлу для обрання зображення з комп'ютера користувача.
3.Обрані зображення відображаються у відповідних мітках "Content Image" та "Style Image".
4.Користувач вводить бажану кількість ітерацій перенесення стилю у відповідне поле.
5.Натиснувши кнопку "Start", програма обчислює втрати контенту та стилю та відображає повідомлення про прогрес та втрати на кожній ітерації.
6.Після завершення процесу користувач отримує повідомлення про успішне завершення роботи та може переглянути результати перенесення стилю.

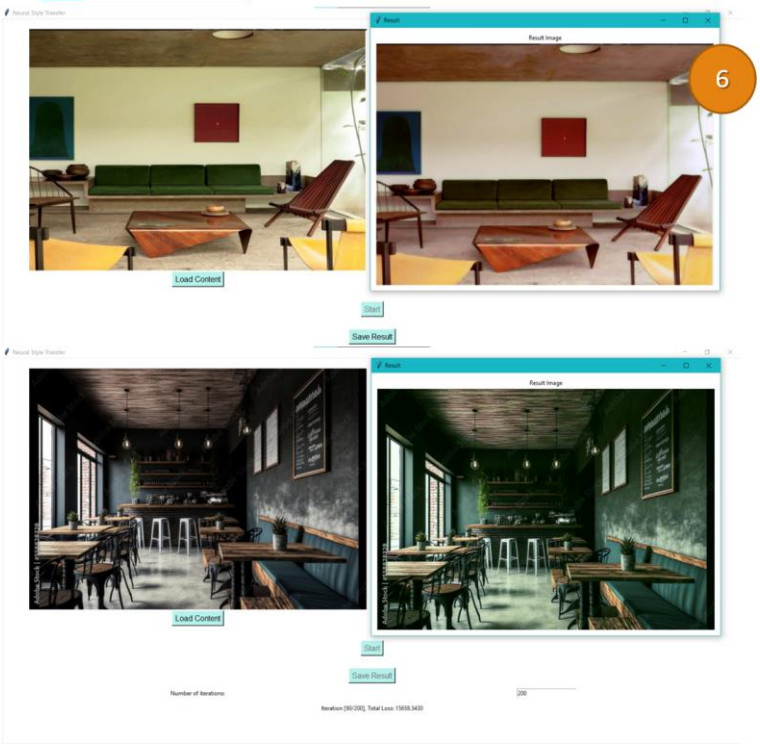


5



Приклад застосування розробленої програми

Приклад застосування програми може включати генерацію зображень у різних стилях та їх подальше використання у фото та відео редакторах, маркетингових матеріалах та ігрових додатках. Наприклад, ці зображення можуть бути використані для створення унікального контенту для соціальних мереж, рекламних кампаній або ілюстрацій для ігор. Крім того, користувачі можуть використовувати програму для створення зображень, які відображають їхні унікальні інтереси, стиль та особистість. Завдяки різноманітним фільтрам та стилям, вони можуть експериментувати з естетикою своїх зображень, надаючи їм особливий шар креативності.



6

Використані інструменти та програми

Python:
•Простота використання та читабельність, що сприяє швидкому розвитку та тестуванню нових ідей.

PyCharm:
•Інтегроване середовище розробки (IDE), яке забезпечує широкий спектр інструментів для редагування коду, налагодження, тестування та інтеграції з системами контролю версій.

PyTorch:
•Фреймворк для машинного навчання, що надає можливості для реалізації та навчання нейронних мереж.

NumPy:
•Бібліотека для роботи з масивами даних у Python, яка забезпечує швидке та ефективне оброблення числових даних.

PIL/Pillow:
•Бібліотека для обробки та маніпуляції зображеннями, що дозволяє легко маніпулювати зображеннями та виконувати функції обробки зображень.

Висновок

Розроблено програмний додаток для генерації зображень, використовуючи мережі глибокого навчання.

Основними завданнями були реалізація алгоритму перенесення стилю зображень, розробка інтерактивного інтерфейсу користувача та оптимізація функціональності програми.

Використано попередньо навчену модель VGG19 для генерації характеристик зображень та реалізовано алгоритм перенесення стилю.

Створено інтерактивний інтерфейс для зручного взаємодії з програмою.
Отриманий програмний продукт дозволяє ефективно виконувати перенесення стилю зображень та надає зручний інтерфейс для користувачів.