МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

ПАРТИКА Павло Миколайович

«Моделі генерації текстур на основі алгоритмів машинного навчання / Texture generation models based on machine learning algorithms»

спеціальність: 123 - Комп'ютерна інженерія
освітньо-професійна програма - Комп'ютерна інженерія
Кваліфікаційна робота

Виконав студент групи КІзм-21
П.М. Партика
_____

Науковий керівник:
к.т.н., Г. М. Мельник
_____

Кваліфікаційну роботу допущено
до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____Л. О. Дубчак

Тернопіль – 2024

АНОТАЦІЯ

Партика П.М. Моделі генерації текстур на основі алгоритмів машинного навчання. – Рукопис.

Кваліфікаційна робота на здобуття освітнього ступеня «магістр» за спеціальністю 123 «Комп'ютерна інженерія», освітньо-професійна програма. Західноукраїнський національний університет, Тернопіль, 2024.

Робота написана обсягом 70 сторінок і містить 8 ілюстрацій, 1 таблицю, 2 додатки та 38 джерела за переліком посилань. Метою є розробка моделі генерації текстур на основі генеративних змагальних мереж (GAN) для створення реалістичних текстур високої якості та різноманітності.

Методи досліджень: використано методи машинного навчання (GAN, варіаційні автоенкодери), математичний аналіз, методи оптимізації (Adam), а також інструменти для роботи з великими наборами даних (TensorFlow, PyTorch).

Результати дослідження: розроблено модель GAN для генерації текстур, проведено навчання та тестування. Оцінка якості виконувалася за допомогою метрик (Fréchet Inception Distance, Inception Score) та візуального аналізу, що підтвердило здатність моделі створювати реалістичні текстури з високою різноманітністю.

Практичне значення: результати роботи можуть бути застосовані у сфері комп'ютерної графіки, відеоігор, віртуальної та доповненої реальності (VR/AR), архітектурного моделювання, а також у дизайні для автоматизації процесу створення текстур.

Орієнтовні напрямки розвитку досліджень: розширення моделі для генерації динамічних текстур, оптимізація алгоритмів для роботи в реальному часі, інтеграція з хмарними сервісами для масштабування, а також розробка спеціалізованих інструментів для художників і дизайнерів.

КЛЮЧОВІ СЛОВА: ГЕНЕРАТИВНІ ЗМАГАЛЬНІ МЕРЕЖІ, ГЕНЕРАЦІЯ ТЕКСТУР, МАШИННЕ НАВЧАННЯ, FID, INCEPTION SCORE, КОМП'ЮТЕРНА ГРАФІКА.

ANNOTATION

Partyka P.M. Models of Texture Generation Based on Machine Learning Algorithms. – Manuscript.

Qualification work for obtaining the educational degree of "Master" in the specialty 123 "Computer Engineering", educational and professional program. West Ukrainian National University, Ternopil, 2024.

The work consists of 70 pages and includes 8 illustrations, 1 table, 2 appendices, and 37 references. The aim is to develop a texture generation model based on Generative Adversarial Networks (GAN) for creating realistic textures of high quality and diversity.

Research Methods: the study employs machine learning methods (GAN, Variational Autoencoders), mathematical analysis, optimization methods (Adam), and tools for working with large datasets (TensorFlow, PyTorch).

Research Results: a GAN model for texture generation has been developed, and training and testing have been conducted. The quality assessment was performed using metrics (Fréchet Inception Distance, Inception Score) and visual analysis, confirming the model's ability to create realistic textures with high diversity.

Practical Significance: the results can be applied in the fields of computer graphics, video games, virtual and augmented reality (VR/AR), architectural modeling, and design for automating the texture creation process.

Prospective Research Directions: expanding the model for dynamic texture generation, optimizing algorithms for real-time operation, integrating with cloud services for scaling, and developing specialized tools for artists and designers.

KEYWORDS: GENERATIVE ADVERSARIAL NETWORKS, TEXTURE GENERATION, MACHINE LEARNING, FID, INCEPTION SCORE, COMPUTER GRAPHICS.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі комп’ютерна графіка є однією з найбільш затребуваних і динамічно розвиваючихся галузей. Вона знаходить широке застосування в таких сферах, як кіноіндустрія, відеоігри, віртуальна і доповнена реальність (VR/AR), архітектурне моделювання, промисловий дизайн тощо. Одним із ключових елементів, які забезпечують високу якість і реалістичність візуалізації, є текстури. Текстури дозволяють передати фізичні характеристики поверхонь, такі як колір, структура, відбивання світла, що значно впливає на сприйняття об’єктів користувачем.

З розвитком технологій зростають вимоги до якості текстур, а також до швидкості їх створення. Традиційні методи, такі як ручне малювання або процедурна генерація, часто є ресурсозатратними та не завжди здатні забезпечити різноманітність і реалістичність. У таких умовах використання алгоритмів машинного навчання, зокрема генеративних моделей, відкриває нові можливості для автоматизації процесу створення текстур і значного підвищення ефективності.

Машинне навчання, особливо генеративні змагальні мережі (GAN), дозволяє створювати реалістичні текстури, які важко відрізнити від створених вручну. Вивчаючи закономірності у великих наборах даних, ці алгоритми здатні генерувати нові текстури, що зберігають схожість із вихідними зразками. Це відкриває нові горизонти для художників, дизайнерів і розробників, дозволяючи їм зосередитися на творчих аспектах роботи, залишаючи рутинні завдання моделям.

Мета роботи полягає у розробці та оцінці ефективності моделі генерації текстур на основі алгоритмів машинного навчання.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1. аналіз існуючих методів генерації текстур, включаючи традиційні підходи та методи машинного навчання;

2. проаналізувати сучасні алгоритми машинного навчання, які використовуються для генерації текстур.

3. розробити модель генеративної змагальної мережі (GAN) для генерації текстур.

4. провести експерименти із різними параметрами моделі для визначення оптимальних налаштувань.

5. програмно реалізувати розроблені алгоритми.

Об’єкт дослідження: текстури в комп’ютерній графіці та процеси їх генерації. Предмет дослідження: алгоритми машинного навчання та генеративні змагальні мережі (GAN).

Методи дослідження. Для досягнення мети роботи використано методи машинного навчання (зокрема генеративні моделі GAN), методи аналізу зображень, а також метрики оцінки якості текстур (FID, IS). У процесі дослідження застосовувалися нейронні мережі, методи оптимізації, а також інструменти для роботи з великими наборами даних.

Наукова новизна роботи полягає у розробці алгоритму генерації текстур із використанням генеративних змагальних мереж, що дозволяє автоматизувати процес створення текстур, підвищити їх якість та різноманітність.

Практичне значення. Результати дослідження можуть бути застосовані в таких галузях: віртуальна та доповнена реальність (VR/AR) для адаптивної генерації текстур відповідно до потреб користувача; архітектурне моделювання та дизайн для швидкого створення текстур матеріалів.

Апробація роботи. Отримані результати опубліковані в межах всеукраїнської науково-практичної конференції «Інтелектуальні комп’ютерні системи та мережі», опубліковано тези доповіді [5,8].

Робота складається зі вступу, трьох розділів, висновків, списку використаних джерел і додатків.

У першому розділі проведено теоретичний огляд поняття текстур, їх класифікації, різновидів і методів генерації. Також розглянуто основи машинного

навчання, включаючи нейронні мережі, які є основою для побудови генеративних моделей.

У другому розділі проаналізовано алгоритми машинного навчання, зокрема генеративні змагальні мережі (GAN), варіаційні автоенкодери (VAE) та інші підходи, які використовуються для генерації текстур. Також розглянуто сучасні дослідження та тренди у цій галузі.

У третьому розділі описано практичну реалізацію моделі GAN для генерації текстур, включаючи вибір інструментів, підготовку даних, процес навчання, тестування та оцінку результатів за допомогою метрик FID і IS. Також виконано порівняння отриманих результатів із традиційними методами генерації текстур.

У висновках підсумовано основні результати роботи, окреслено перспективи подальших досліджень і запропоновано можливі напрямки практичного застосування отриманих результатів.

1 АНАЛІЗ МЕТОДІВ ТА АЛГОРИТМІВ ГЕНЕРАЦІЇ ТЕКСТУР

1.1 Поняття текстури в комп’ютерній графіці

Текстура в комп’ютерній графіці – це двомірне зображення або патерн, що використовується для надання тривимірним об’єктам реалістичних фізичних і візуальних характеристик. Вона служить для імітації матеріалів, що дозволяє створювати правдоподібні сцени в комп’ютерних іграх, анімаціях і віртуальних середовищах. Текстура накладається на поверхню об’єкта, щоб надати йому деталі, які не можуть бути досягнуті лише за допомогою геометрії [2].

1. Фізичні та візуальні характеристики текстур [3]

Текстури мають різноманітні фізичні характеристики, які впливають на їхнє сприйняття. Основні характеристики включають: колір, візерунок, шорсткість, прозорість.

Колір є одним з найважливіших аспектів текстури. Він визначає, як текстура виглядає візуально і може варіюватися від яскравих до приглушених відтінків. Колір може бути однорідним, але часто включає градієнти або візерунки, що підсилюють реалістичність. Наприклад, текстура трави може мати різні відтінки зеленого, що імітує природний вигляд.

Візерунок текстури визначає її структуру. Це може бути регулярний візерунок (наприклад, плитка) або нерегулярний (наприклад, текстура шкіри). Візерунок може додати глибини і складності, роблячи об’єкт більш привабливим. Наприклад, текстура цегли має чітко виражені межі, що підкреслює її конструктивні особливості.

Шорсткість текстури впливає на те, як вона відображає світло. Гладкі текстури, такі як скло, відображають світло чітко і яскраво, тоді як грубі текстури, такі як бетон, розсіюють світло, створюючи м’якший вигляд. Шорсткість також може впливати на тактильні відчуття, які викликає об’єкт. Наприклад, текстура ворсистого матеріалу може викликати бажання доторкнутися до нього.

Прозорість, деякі текстури можуть бути частково або повністю прозорими, що дозволяє бачити об’єкти, розташовані за ними. Прозорість часто використовується для створення ефектів, таких як вікна, вода або повітря, надаючи додаткову реалістичність. Наприклад, текстура води може відображати навколишнє середовище, створюючи враження глибини.

2. Зв’язок текстури з матеріалами та поверхнями [3]

Текстура тісно пов’язана з матеріалом, з якого виготовлений об’єкт. Кожен матеріал має свої унікальні текстурні властивості, які визначають, як він виглядає під різними умовами освітлення. Наприклад: деревина, метал, камінь, тканини (Рисунок 1.1).

Рисунок 1.1 - Текстури: дерево, метал, камінь, тканина.

Текстура дерева характеризується натуральними візерунками, які створюють відчуття теплоти і органічності. Вона може мати різні кольори та відтінки, залежно від типу дерева. Текстура може включати деталі, такі як сучки, тріщини та інші природні дефекти, що підвищує її реалістичність. Дерев’яні текстури також можуть відображати різні обробки, такі як лакування або фарбування.

Металева текстура зазвичай відображає світло, має глянцевий вигляд і може містити дрібні подряпини або вм’ятини, що додає реалістичності. Метали можуть мати різні текстури, такі як матова, полірована чи шорстка, що впливає на їхнє сприйняття. Наприклад, текстура нержавіючої сталі має холодний блиск, тоді як текстура старого заліза може бути шорсткою і з патиною.

Текстура каменю може бути грубою і нерівною, з різноманітними кольорами та візерунками, що створює враження природності. Текстури каменю часто використовуються в архітектурі та ландшафтному дизайні, оскільки вони можуть передавати відчуття стабільності і довговічності. Наприклад, текстура граніту має чіткі візерунки, які роблять його привабливим для використання в будівництві.

Текстури тканин можуть варіюватися від гладких до ворсистих, від блискучих до матових. Наприклад, текстура шовку має гладкий і блискучий вигляд, тоді як текстура бавовни може бути більш шорсткою і матовою. Використання текстур тканин у візуалізації персонажів або одягу додає виразності та деталізації, підкреслюючи стиль і характер персонажа.

3. Вплив текстури на сприйняття глядача [6]

Текстура грає критичну роль у сприйнятті глядача, оскільки вона може суттєво змінити враження від об’єкта. Реалістичні текстури можуть викликати у глядача відчуття дотику, глибини і матеріальності. Основні аспекти впливу текстури на сприйняття: емоційний вплив, сприйняття простору, реалістичність.

Текстури можуть викликати емоції. Наприклад, м’які та теплі текстури можуть створити відчуття затишку, тоді як холодні і грубі текстури можуть викликати відчуття відчуження або небезпеки. Вибір текстури може суттєво вплинути на атмосферу сцени. Наприклад, текстура старої цегли може викликати ностальгічні почуття, тоді як сучасні гладкі текстури можуть створювати відчуття технологічності.

Текстури допомагають створити ілюзію простору та глибини. Вони можуть акцентувати увагу на певних елементах сцени, ведучи погляд глядача вглиб композиції. Наприклад, використання текстур з перспективою може створити враження, що об’єкти розташовані на різних відстанях. Це особливо важливо в архітектурній візуалізації, де текстури можуть підкреслювати масштаби і пропорції будівель.

Неправильний вибір або застосування текстури може призвести до спотворення сприйняття об’єкта, роблячи його менш реалістичним чи навіть

неприродним. Наприклад, текстура, що не відповідає формі або матеріалу об’єкта, може створити враження, що об’єкт виглядає неправдоподібно. Це підкреслює важливість точного підбору текстур у процесі моделювання, оскільки навіть найкраща геометрія може виглядати невиразно без правильної текстури.

4. Технології створення текстур [7]

Сучасні технології дозволяють художникам і дизайнерам створювати текстури різними способами:

- Фотографічні текстури – одна з найпоширеніших технік – це використання фотографій реальних матеріалів. Художники можуть фотографувати текстури, такі як стіни, підлоги або природні об’єкти, а потім обробляти їх у графічних редакторах. Цей метод дозволяє отримати високу деталізацію і реалістичність.

- Процедурні текстури це текстури, які генеруються математично, без використання реальних зображень. Процедурні текстури можуть бути дуже гнучкими і дозволяють створювати складні візерунки, які легко масштабуються і налаштовуються. Вони особливо корисні для створення текстур, які повинні виглядати безшовно.

- Ручне малювання – деякі художники віддають перевагу ручному малюванню текстур, що дозволяє їм створювати унікальні та індивідуальні стилі. Це може бути особливо корисно в анімації та ігровій графіці, де стиль має велике значення. Ручне малювання також дозволяє досягти ефектів, які важко реалізувати за допомогою автоматизованих методів.

- Текстурні мапи – для досягнення реалістичності часто використовують кілька текстурних мап. Наприклад, мапи нормалей, мапи відбиття та мапи шорсткості можуть бути використані разом, щоб надати об’єкту складніші деталі та ефекти. Це дозволяє створювати багатошарові текстури, що підвищують загальну якість візуалізації.

5. Використання текстур у різних галузях [9]

Текстури використовуються в багатьох галузях, включаючи: відеоігри, анімація, архітектурна візуалізація, мода, віртуальна реальність (VR) і доповнена реальність (AR).

У відеоіграх текстури грають ключову роль у створенні реалістичних персонажів і середовищ. Вони допомагають передати атмосферу гри, створюючи враження занурення. Текстури можуть бути використані для створення об’єктів, таких як будівлі, транспортні засоби та природні елементи.

У анімації текстури використовуються для надання персонажам та об’єктам характеру та стилю. Вони можуть змінюватися в залежності від розвитку сюжету, що додає динаміки та емоційності.

У архітектурній візуалізації текстури використовуються для створення реалістичних зображень будівель і інтер’єрів. Вони допомагають архітекторам та дизайнерам презентувати свої проекти клієнтам, надаючи їм уявлення про майбутній вигляд об’єктів.

У моді текстури використовуються для створення унікальних дизайнів одягу та аксесуарів. Текстури можуть підкреслювати стиль і настрій колекції, впливаючи на сприйняття виробів.

У VR та AR текстури грають важливу роль у створенні інтерактивних і занурювальних середовищ. Вони допомагають створити реалістичний досвід для користувачів, що взаємодіють з віртуальними об’єктами.

Таким чином, текстура є важливим елементом комп’ютерної графіки, що впливає на загальну якість і реалістичність візуалізації. Вона не лише надає об’єктам зовнішній вигляд, але й формує емоційне сприйняття глядача, що робить її ключовим аспектом у створенні візуального контенту. Текстури дозволяють художникам і дизайнерам реалізувати свої ідеї, створюючи вражаючі та правдоподібні світи, які захоплюють глядачів і користувачів.

Текстури можна класифікувати на різні категорії в залежності від їхнього походження та застосування. Основні різновиди текстур включають природні, синтетичні текстури та текстури в контексті різних культур і стилів.

1. Природні текстури [10]

Природні текстури виникають внаслідок природних процесів і характерні для органічних матеріалів. Вони включають: пейзажі, органічні матеріали.

Пейзажні текстури охоплюють різноманітні природні елементи, такі як гори, ліси, поля та водойми. Наприклад:

- Гори - текстури скель можуть бути грубими з вираженими тріщинами та візерунками, що відображають геологічні процеси.

- Ліси - текстури листя можуть варіюватися за кольором і формою, від яскраво-зеленого влітку до червоного і жовтого восени.

- Водойми - текстури води можуть відображати градієнти кольорів, хвилі та відображення навколишніх об’єктів, що створює динамічний ефект.

Органічні матеріали – текстури, що імітують натуральні матеріали, такі як дерево, камінь і вода, використовуються для створення реалістичних зображень. Наприклад:

- Дерево - текстури деревини можуть включати різні візерунки, кольори та шорсткість. Наприклад, текстура червоного дерева може бути насиченою та глибокою, тоді як сосна має світліший відтінок з вираженими смугами.

- Камінь - текстури каменю можуть варіюватися від гладкого мармуру, що використовується в архітектурі, до грубого граніту, що використовується в ландшафтному дизайні. Наприклад, текстура базальту може мати характерні колони, що утворилися внаслідок охолодження лави.

- Вода - текстури води можуть бути різноманітними: спокійна поверхня може відображати небо, тоді як бурхливий потік створює складні візерунки з піни та хвиль.

2. Синтетичні текстури [11]

Синтетичні текстури створюються штучно і часто використовуються для графічних елементів у дизайні та анімації (рисунок 1.2). Вони включають:

- графічні елементи - це текстури, які складаються з векторних або растрових графічних елементів. Вони можуть бути використані для створення логотипів, іконок та інших візуальних компонентів. Наприклад, текстури для веб-дизайну можуть включати градієнти або текстури, що імітують матеріали, такі як

метал або скло. Використання текстур, що імітують глянцеву поверхню, може надати елементам сучасного вигляду.

Рисунок 1.2 -Синтетичні текстури: графічні елементи, абстракції, патерни.

- абстракції - абстрактні текстури можуть включати різноманітні форми, кольори та візерунки, які не мають чіткої прив’язки до реальних об’єктів. Вони часто використовуються для створення фонів, декоративних елементів або художніх проектів. Наприклад, текстури з розмитими кольорами можуть створювати атмосферу спокою або динаміки, відображаючи емоції через кольорові комбінації.

- патерни це регулярні або повторювані візерунки, які можуть бути використані для оформлення, текстилю або графічного дизайну. Патерни можуть бути простими (наприклад, смужки або крапки) або складними (наприклад, геометричні фігури). Вони можуть бути використані для створення інтер’єрів, упаковки продуктів або веб-дизайну. Наприклад, текстура в горошок може використовуватися в текстилі для одягу, тоді як геометричні патерни можуть бути популярними в сучасному графічному дизайні.

3. Текстури в контексті різних культур та стилів [6].

Текстури також можуть відображати культурні особливості та художні стилі. Наприклад:

- національні мотиви - багато культур мають свої унікальні текстури, які відображають традиційні матеріали та техніки. Наприклад, текстури, що

використовують орнаменти української вишивки, можуть включати яскраві кольори та геометричні візерунки, що символізують природу та культуру. Інші приклади включають традиційні японські текстури, такі як «сумі-є», що використовують чорнило для створення текстурованих малюнків.

- архітектурні стилі - текстури можуть варіюватися в залежності від архітектурних стилів, таких як готика, бароко чи сучасний мінімалізм. Кожен стиль має свої характерні риси, які можна відобразити через текстури. Наприклад, готичні собори часто мають текстури з каменю з деталізованими різьбленнями, тоді як сучасні будівлі можуть мати гладкі, металеві або скляні текстури, що підкреслюють простоту та функціональність.

- мистецькі рухи - у мистецтві різні течії, такі як імпресіонізм, експресіонізм чи абстрактний експресіонізм, використовують текстури для передачі емоцій та ідей. Текстури в живопису можуть варіюватися від гладких до грубих, в залежності від стилю художника. Наприклад, імпресіоністи використовували текстури для створення враження руху та зміни світла, тоді як експресіоністи могли застосовувати грубі мазки для передачі емоційної напруги.

Таким чином, різновиди текстур відіграють важливу роль у створенні візуального контенту, допомагаючи передати емоції, атмосферу та культурні особливості. Вони є невід'ємною частиною дизайну, живопису та комп'ютерної графіки, надаючи глибину та різноманітність візуальному сприйняттю.

1.2 Важливість текстур для реалізму візуалізації

Текстури є невід'ємною складовою візуального мистецтва, що надає об'єктам і сценам реалістичності. Вони значно впливають на сприйняття глядача, створюючи емоційний контекст і взаємодіючи з освітленням та матеріалами, що підсилює загальне враження від візуалізації.

Текстури надають об'єктам складність і нюанси, що роблять їх більш

привабливими для глядачів. Наприклад, текстура шкіри персонажів анімаційних фільмів може варіюватися від гладкої до шорсткої, що підкреслює їх вік або емоційний стан. У фільмі «Звериполіс» текстура шкіри різних тварин акцентує їх особливості та характер. Аналогічно, у комп'ютерних іграх, таких як «Cyberpunk 2077», текстури міських будівель, асфальту та тротуарів створюють відчуття реального середовища, з усіма його недоліками та особливостями [10].

Текстури також здатні формувати емоційний контекст і атмосферу сцени. У фільмах жахів текстури тріщин та облущеної фарби на старих стінах можуть викликати відчуття занедбаності та страху, посилюючи напругу. У фільмі «Лев» текстури трави, дерев і води створюють атмосферу безтурботності та краси африканської савани, відображаючи дух пригод. Окрім цього, текстури допомагають глядачам швидко ідентифікувати матеріали та об'єкти. У кулінарних шоу текстури страв, такі як хрустка скоринка або кремова консистенція, сприяють формуванню апетиту. У фільмах костюми персонажів, виконані з різних текстур, можуть відображати їх соціальний статус, характер і навіть розвиток сюжету.

Приклади використання текстур у відомих проектах, таких як ігри та фільми, демонструють їхню важливість. У грі «God of War» текстури пейзажів, персонажів і предметів ретельно розроблені для передачі атмосфери скандинавської міфології. Наприклад, текстура льоду, що відображає світло, створює враження холодного середовища, в якому проходять битви, тоді як текстури персонажів, таких як Кратос, містять деталі, що підкреслюють його бойовий досвід і емоційний стан. У фільмі «Володарі кілець» текстури природних ландшафтів, зокрема текстура лісу Фангорн, створюють відчуття магії та таємниці. Старі дерева з детальною корою передають велич і мудрість природи, а текстури зброї персонажів відображають їхній вік і використання, що підсилює емоційний зв'язок глядача з історією. У анімації «Шрек» текстури персонажів, такі як зелене обличчя Шрека та текстура його одягу, додають комічності та характеру, а фонові текстури, такі як текстура лісу чи болота, формують веселу та казкову атмосферу [13].

Взаємозв’язок текстур з освітленням і матеріалами також є важливим аспектом, що впливає на сприйняття об’єктів у сцені. Типи освітлення, такі як напрямок, інтенсивність та колір, можуть кардинально змінити візуальне сприйняття текстур. М’яке освітлення створює м’які тіні і згладжує деталі текстур, що може використовуватися для романтичних або спокійних сцен, тоді як контрове освітлення підкреслює контури об’єктів, створюючи драматичний ефект, особливо в фільмах жахів. Різні матеріали мають свої характерні текстури, які визначають, як вони виглядають під різними умовами освітлення. Гладкі матеріали, такі як скло чи метал, відбивають світло, створюючи блискучі ефекти, тоді як шорсткі матеріали, такі як бетон або дерево, поглинають світло, надаючи їм більш матовий вигляд.

Таким чином, текстури є основою для створення реалістичних візуалізацій, оскільки вони не лише покращують естетичний вигляд, але й впливають на сприйняття глядачів, створюють атмосферу та взаємодіють з освітленням і матеріалами.

1.3 Класифікація текстур

Текстури, як важливі елементи візуального мистецтва та комп’ютерної графіки, відіграють ключову роль у формуванні сприйняття об’єктів. Їх класифікація є необхідною для розуміння ефективного використання текстур у різних контекстах (рисунок 1.3).

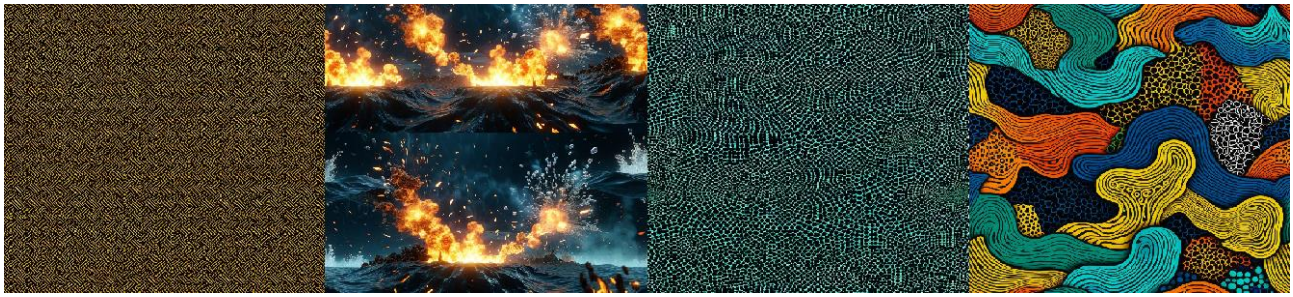

Рисунок 1.3 - Текстури: статичні, динамічні, параметричні, непараметричні

1. Статичні текстури [12]

Статичні текстури залишаються незмінними протягом часу і часто використовуються в статичних зображеннях, таких як живопис, фотографія та графічний дизайн. Вони характеризуються високою деталізацією, що дозволяє передавати тонкі нюанси, а також незмінністю патернів і кольорів, що забезпечує створення впізнаваних образів. Методи їх створення включають фотографічні текстури, які базуються на реальних об’єктах, а також ручне малювання художниками. Статичні текстури використовуються в архітектурному дизайні, ілюстраціях, веб-дизайні та моді, наприклад, для текстур стін, підлог, меблів або тканин.

2. Динамічні текстури [12]

Динамічні текстури змінюються в часі, надаючи їм відчуття руху. Цей тип текстур широко застосовується в анімації, відеоіграх та віртуальній реальності. Основними характеристиками є їх адаптивність, можливість взаємодії з діями користувача та складність реалізації. Методи створення динамічних текстур включають анімацію через ключові кадри та симуляцію фізичних процесів. Приклади використання включають текстури для води, диму та вогню у відеоіграх, а також ефекти в анімаційних фільмах, такі як вибухи.

3. Параметричні текстури [15]

Параметричні текстури генеруються за допомогою математичних функцій, що дозволяє створювати текстури з високою деталізацією без необхідності зберігати великі обсяги даних. Їх гнучкість і алгоритмічна природа забезпечують можливість легкої настройки та генерації нових варіантів без ручного втручання. Методи створення включають генеративну графіку та використання математичних функцій, таких як шум Перліна. Ці текстури активно використовуються в 3D-моделюванні та графічних програмах для створення складних поверхонь.

4. Непараметричні текстури [15]

Непараметричні текстури характеризуються випадковими патернами, які не мають чіткої структури. Вони часто використовуються для створення

враження хаосу та випадковості. Основні характеристики включають випадковість, візуальний інтерес та імітацію природних матеріалів. Методи їх створення включають випадкові алгоритми та застосування фільтрів до зображень. Непараметричні текстури знаходять застосування в графічному дизайні, мистецтві, моді та ігровому дизайні, додаючи унікальності виробам.

Таким чином, класифікація текстур є ключовим елементом у розумінні комп'ютерної графіки та дизайну. Кожен тип текстур має свої унікальні характеристики, методи створення та різноманітні застосування, які суттєво впливають на сприйняття кінцевого продукту. Розуміння цих аспектів дозволяє дизайнерам та художникам ефективно використовувати текстури для досягнення бажаних візуальних ефектів і створення емоційно насичених проектів.

1.4 Методи генерації текстур та постановка завдань кваліфікаційної роботи

Генерація текстур є важливим аспектом комп'ютерної графіки, оскільки вона дозволяє створювати реалістичні та естетично привабливі поверхні для віртуальних об'єктів. Існує безліч методів, які використовуються для цього процесу, починаючи від класичних підходів, таких як пересічення та фрактали, і закінчуючи сучасними технологіями, зокрема нейронними мережами. Розуміння цих методів і їхніх алгоритмів є необхідним для художників і дизайнерів, які прагнуть створювати унікальні текстури, що відповідають вимогам сучасних проектів.

Класичні методи генерації текстур [18]:

1. Метод пересічення базується на використанні простих геометричних форм, які комбінуються для створення текстур. Цей підхід дозволяє генерувати абстрактні візерунки. Наприклад, текстура фону може бути створена шляхом накладання кіл, трикутників і квадратів, формуючи складний патерн. Серед алгоритмів, що застосовуються в цьому методі, можна виділити алгоритм

Вортона, який генерує випадкові точки на площині, формуючи області, що належать найближчій точці. Це дозволяє імітувати природні патерни, такі як острівці на воді. Іншим важливим алгоритмом є алгоритм Брезенхема, що використовується для малювання ліній і форм у піксельній графіці, який також може бути адаптований для текстур.

2. Фрактальні методи використовують математичні функції для створення текстур із самоподібною структурою, що дозволяє отримувати складні текстури з мінімальною кількістю даних. Наприклад, текстура гірського ландшафту може бути згенерована таким чином, що кожен рівень деталізації нагадує попередній. Алгоритм Мандельброта є одним із ключових для створення фрактальних зображень, які можуть слугувати текстурами, подібними до мармуру або кристалів. Алгоритм Сірпінського також генерує фрактальні трикутники, які можуть бути основою для текстур, що імітують сніг або піщані дюни [19].

3. Процедурні методи. Серед процедурних методів виділяється шум Перліна, який генерує плавні переходи між кольорами, що робить його ідеальним для створення природних текстур. Наприклад, текстура води може бути створена за допомогою шуму Перліна, де кольори плавно переходять один в інший, створюючи ефект хвиль. Класичний шум Перліна використовує градієнти та інтерполяцію для створення гладких текстур, тоді як шум Simplex є більш ефективним у обчисленнях. Іншим підходом є алгоритми, що базуються на марковських випадкових полях, які генерують текстури з урахуванням попередніх значень, що дозволяє досягти більшої реалістичності. Наприклад, такі алгоритми можуть бути використані для генерації текстур ландшафтів, де різні ділянки мають різні характеристики, такі як вода, земля та рослинність.

Сучасні методи генерації текстур [20]:

1. Використання нейронних мереж. Сучасні технології, зокрема генеративні змагальні мережі (GAN), дозволяють нейронним мережам навчатися на великих наборах даних для створення нових текстур. Цей підхід забезпечує можливість генерації унікальних текстур, які важко досягти традиційними методами. Наприклад, GAN може бути використаний для генерації текстур

персонажів у відеоіграх, які виглядають максимально реалістично, таких як текстури шкіри або одягу. Серед алгоритмів, що застосовуються, виділяються DCGAN (Deep Convolutional GAN), який генерує зображення на основі навчання з реальних зображень, та StyleGAN, що дозволяє контролювати стиль і деталі текстури, забезпечуючи високу якість генерованих зображень.

2. Адаптивні алгоритми. Адаптивні алгоритми оптимізують процес генерації текстур, підбираючи параметри залежно від контексту, що дозволяє швидше створювати текстури. Наприклад, в динамічних середовищах відеоігор текстури можуть змінюватися в залежності від дій гравця або умов гри. Алгоритми машинного навчання використовуються для адаптації параметрів генерації текстур на основі даних про середовище, а генеративні моделі можуть адаптуватися до специфічних умов, що дозволяє створювати текстури в реальному часі, такі як текстури води, що змінюються в залежності від руху об’єктів у воді [22].

Методи генерації текстур, від класичних до сучасних, забезпечують широкий спектр можливостей для художників і дизайнерів. Розуміння цих методів і алгоритмів дозволяє створювати текстури, які не лише виглядають реалістично, але й відповідають вимогам проекту. Використання новітніх технологій, таких як нейронні мережі, відкриває нові горизонти в генерації текстур, надаючи можливість створювати унікальні та складні візуальні елементи.

1.5 Висновок до розділу

Проведено детальний аналіз основних понять, методів та класифікацій, що стосуються текстур у комп’ютерній графіці. Текстура визначається як ключовий елемент, що надає об’єктам фізичні характеристики, такі як колір, структура та відбивання світла. Важливою частиною дослідження є класифікація текстур, яка включає природні, синтетичні та культурні аспекти.

Розглянуті традиційні методи генерації текстур, такі як ручне малювання та процедурна генерація, продемонстрували свої обмеження в контексті швидкості та різноманітності. Водночас сучасні підходи, зокрема алгоритми машинного навчання, відкривають нові можливості для автоматизації процесу створення текстур.

Проте, незважаючи на досягнення в цій галузі, залишаються нерозв’язані проблеми, які потребують подальшого дослідження. Серед них – необхідність підвищення різноманітності та адаптивності текстур, створених традиційними методами, а також оцінка ефективності алгоритмів машинного навчання, зокрема генеративних змагальних мереж (GAN), у контексті генерації текстур.

2 АЛГОРИТМИ МАШИННОГО НАВЧАННЯ ДЛЯ ГЕНЕРАЦІЇ ТЕКСТУР

2.1 Застосування глибокого навчання для генерації текстур

Глибоке навчання використовує нейронні мережі для моделювання складних залежностей у даних, що дозволяє досягати високої точності в різних задачах. Особливу увагу буде приділено архітектурам нейронних мереж та їх застосуванню в генерації текстур, що відкриває нові можливості в комп'ютерній графіці, дизайні та інших сферах.

Глибоке навчання є потужним підходом у машинному навчанні, що використовує нейронні мережі для моделювання складних залежностей у даних. Нейронні мережі складаються з численних взаємопов'язаних елементів, які імітують роботу біологічних нейронів. Основні компоненти нейронних мереж включають [21]:

- Нейрони - основні одиниці обробки інформації. Кожен нейрон отримує вхідні дані, обробляє їх за допомогою активаційної функції та передає результат на наступний шар. Нейрони можуть бути активовані залежно від значення вхідних сигналів.
- Ваги та зсуви -кожен зв'язок між нейронами має вагу, яка визначає важливість вхідних даних. Зсув (bias) додається для підвищення гнучкості моделі, дозволяючи нейрону активуватися навіть при нульових вхідних значеннях.
- Активаційні функції- використовуються для введення нелінійності в модель, що дозволяє нейронній мережі вивчати складні патерни. Популярні функції включають:
- ReLU (Rectified Linear Unit)-використовується для прискорення навчання, пропускаючи лише позитивні значення. Це допомагає уникнути проблеми затухаючого градієнта.

- Sigmoid- відображає значення в діапазоні від 0 до 1, що корисно для бінарних класифікацій. Однак може призводити до затухання градієнта при великих значеннях.
- Tanh- відображає значення в діапазоні від -1 до 1, що часто використовується в рекурентних нейронних мережах.

Багатошарові нейронні мережі (глибокі нейронні мережі, DNN) складаються з кількох шарів нейронів [23]:

1. Вхідний шар приймає вхідні дані, які можуть бути представлені у вигляді векторів (наприклад, пікселі зображення).

2. Приховані шари - один або кілька шарів, які виконують обробку даних. Кількість прихованих шарів та нейронів у кожному шарі може варіюватися в залежності від задачі. Глибокі мережі можуть мати десятки або навіть сотні прихованих шарів.

3. Вихідний шар - генерує прогноз або класифікацію на основі обробленої інформації. Вихід може бути представлений у вигляді ймовірностей для різних класів або як безпосереднє значення для регресії.

Процес навчання (рисунок 2.1).

Процес навчання нейронних мереж зазвичай включає кілька етапів:

1. Ініціалізація: ваги та зсуви ініціалізуються випадковими значеннями.
2. Прямий прохід: вхідні дані проходять через мережу, і на кожному шарі обчислюються активовані значення нейронів.
3. Обчислення втрат: визначається функція втрат, яка вимірює, наскільки добре модель справляється з поставленою задачею. Наприклад, для класифікації це може бути крос-ентропія.
4. Зворотний прохід: за допомогою алгоритму зворотного розповсюдження градієнта (backpropagation) обчислюються градієнти втрат по відношенню до ваг. Це дозволяє оновити ваги для зменшення втрат.
5. Оновлення ваг: ваги коригуються за допомогою оптимізаторів, таких як SGD (Stochastic Gradient Descent) або Adam, на основі обчислених градієнтів.

6. Повторення: процес повторюється для декількох епох, поки модель не досягне задовільної точності.

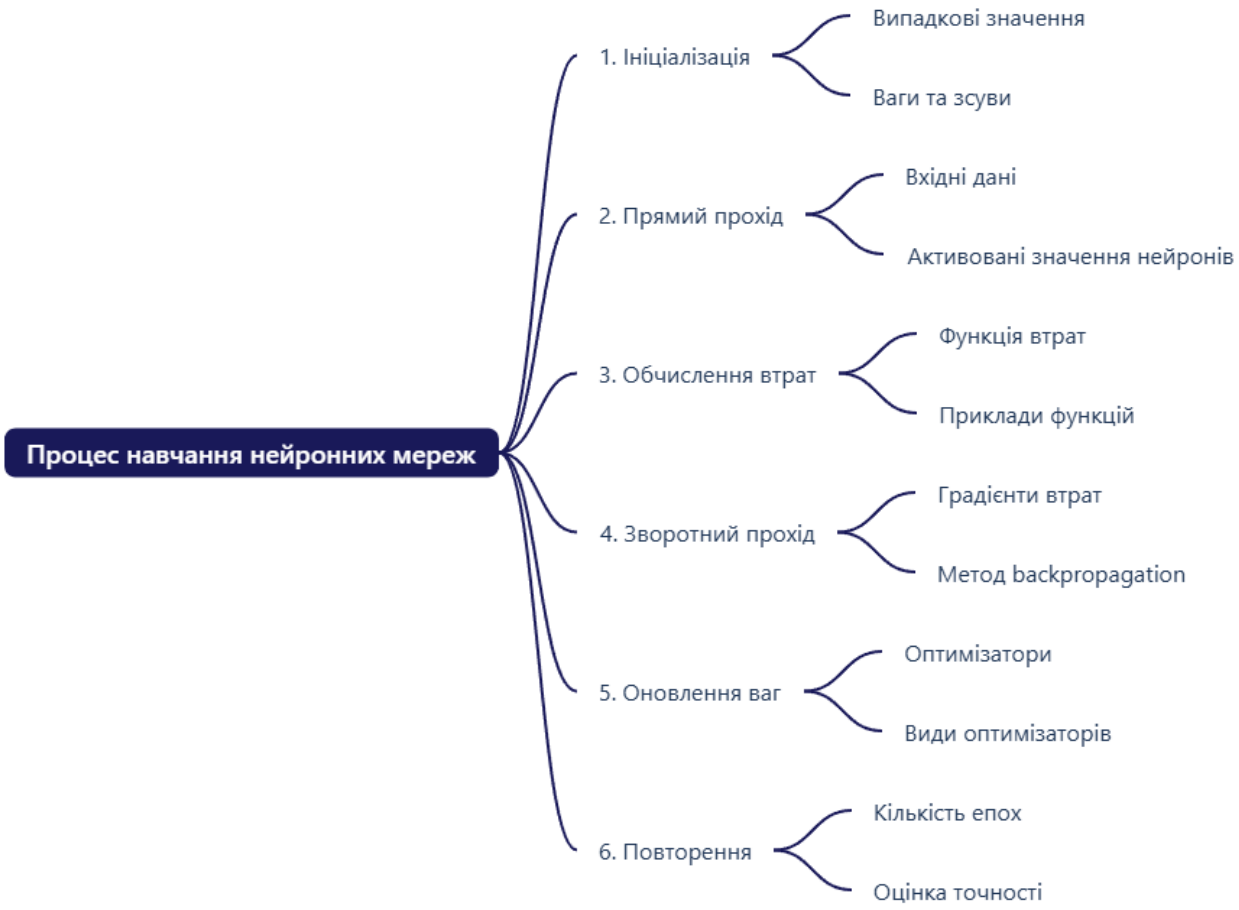

Рисунок 2.1 - Процес навчання нейронних мереж

Використання глибоких нейронних мереж для генерації текстур

Генерація текстур – це процес створення нових текстур на основі існуючих даних. Глибокі нейронні мережі, зокрема генеративні моделі, використовуються для цієї задачі. Основні методи включають:

Генеративні змагальні мережі (GANs) [24]

Переваги

1. Якість текстур: GANs здатні генерувати текстури, які виглядають дуже реалістично, завдяки здатності виявляти складні патерни в даних.

2. Автоматизація: зменшують потребу в ручному створенні текстур, що економить час і ресурси.

3. Адаптивність: можуть навчатися на різноманітних наборах даних, що дозволяє створювати текстури для різних стилів та застосувань.

4. Комбінування стилів: GANs можуть комбінувати різні текстури та стилі, створюючи унікальні результати.

Недоліки:

1. Вимоги до даних -для навчання моделей потрібні великі обсяги даних, що може бути складним у деяких випадках, особливо якщо специфікація текстур є унікальною.

2. Час навчання - процес навчання може займати багато часу та вимагати значних обчислювальних ресурсів.

3. Складність налаштування- налаштування архітектури та параметрів моделі може бути складним і вимагати експериментів.

4. Недостатня інтерпретованість -важко зрозуміти, як модель приймає рішення, що може бути проблемою в критично важливих застосуваннях.

Приклади успішних проектів у галузі генерації текстур [26]:

1. DeepArt проект використовує генеративні змагальні мережі (GAN) для створення художніх текстур, базуючись на стилях відомих художників. Користувачі мають можливість завантажувати свої фотографії, які потім трансформуються в картини у стилі таких митців, як Ван Гог або Пікассо.

2. Prisma програма, що реалізує технології перенесення стилю, дозволяє перетворювати фотографії на художні твори. Вона аналізує стиль оригінального зображення та переносить його на нові зображення, створюючи унікальні текстури.

3. Neural Texture Synthesis - алгоритми, що використовують глибокі нейронні мережі для синтезу нових текстур на основі невеликих зразків. Цей підхід забезпечує створення текстур, які виглядають природно і гармонійно, зберігаючи візуальні характеристики оригінальних зразків.

4. NVIDIA GauGAN -інструмент, що застосовує GAN для перетворення простих ескізів у реалістичні пейзажі. Користувачі можуть малювати

елементарні форми, а система автоматично генерує текстури, такі як дерева, гори та небо.

Інші підходи до генерації текстур (рисунок 2.1.2):

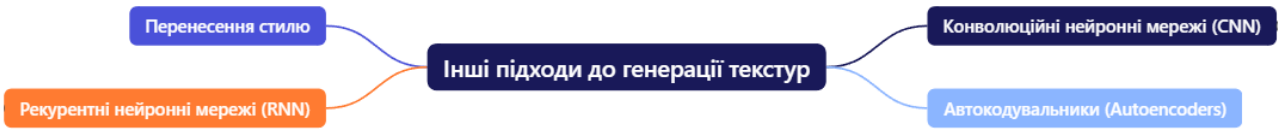

Рисунок 2.2 - Підходи до генерації текстур

1. Конволюційні нейронні мережі (CNN) використовуються для виявлення та генерації текстур. CNN здатні навчатися на зразках текстур і потім застосовувати отримані знання для створення нових текстур на основі вхідних даних. Вони особливо ефективні в обробці зображень завдяки своїй здатності зберігати просторову інформацію. Наприклад, алгоритми, що автоматично генерують текстури на основі вхідних зображень, використовують конволюційні шари для виявлення характерних ознак.

2. Технології перенесення стилю дозволяють переносити текстури з одного зображення на інше, зберігаючи при цьому зміст оригіналу. Це досягається за допомогою глибоких нейронних мереж, які аналізують стиль та зміст. Наприклад, алгоритми, що перетворюють зображення, зберігаючи його структуру, але змінюючи текстуру на основі стилю іншого зображення.

3. Автокодувальники (Autoencoders) є нейронними мережами, які навчаються кодувати вхідні дані в стислий формат, а потім відновлювати їх. Вони можуть бути використані для генерації текстур шляхом навчання на великих наборах даних текстур. Наприклад, автокодувальники можуть створювати нові текстури на основі вже існуючих, зберігаючи основні характеристики.

4. Рекурентні нейронні мережі (RNN)**:** Хоча рекурентні нейронні мережі зазвичай використовуються для обробки послідовностей, їх можна адаптувати для генерації текстур, використовуючи послідовності пікселів. Це

може бути корисним для створення текстур, що мають певну структуру або патерни. Наприклад, генерація текстур, що повторюються, може здійснюватися на основі послідовностей пікселів, вивчених з навчальних даних.

2.2 Варіаційні автоенкодери та альтернативні моделі для генерації текстур

У цій секції ми розглянемо варіаційні автоенкодери (VAE), їх принципи роботи, застосування, а також порівняємо їх з GAN. Додатково обговоримо інші методи, такі як PixelCNN, PixelSNAIL та інші, які можуть бути використані для генерації текстур і даних.

Варіаційні автоенкодери (VAE) являють собою потужні генеративні моделі, що об’єднують концепції автоенкодерів та байєсівської статистики. Основні етапи їх роботи можна розділити на три ключові компоненти: кодування, декодування та обчислення втрат [28].

1. Кодування: вхідні дані, такі як зображення, проходять через кодувальну мережу (енкодер), яка перетворює їх у латентний простір. Вихід енкодера не є простою точкою в латентному просторі; він складається з параметрів розподілу (зазвичай середнього значення та стандартного відхилення), з яких вибирається латентний вектор. Цей латентний вектор дозволяє моделі навчатися більш гнучко та забезпечує регуляризацію, що є критично важливим для контролю над генерацією нових даних.
2. Декодування - латентний вектор передається через декодувальну мережу (декодер), яка генерує нові дані, схожі на вхідні. Декодер намагається відновити вихідні дані з латентного простору, при цьому важливо, що він відтворює ймовірнісні характеристики вхідних даних, а не просто детерміновані значення. Це дозволяє генерувати різноманітні результати, що підвищує якість генерації.

3. Обчислення втрат - VAE використовує дві основні складові для функції втрат. По-перше, втрата реконструкції вимірює, наскільки добре декодер відновлює вихідні дані, зазвичай за допомогою середньоквадратичної помилки. По-друге, втрата регуляризації забезпечує структуру латентного простору, зазвичай у вигляді нормального розподілу. Це досягається через Kullback-Leibler (KL) дивергенцію, яка вимірює різницю між латентним розподілом і нормальним розподілом.

Застосування VAE [29]:

- Генерація зображень - VAE широко використовуються для генерації нових зображень, які мають подібні характеристики до навчальних даних. Наприклад, VAE можуть генерувати нові обличчя, пейзажі або предмети.
- Аналіз даних - VAE можуть бути використані для зменшення розмірності даних і візуалізації складних наборів даних. Це особливо корисно в сценаріях, де дані мають високу вимірність, таких як біологічні дані або дані з соціальних мереж.
- Аномалія виявлення - використання VAE для виявлення аномалій у даних, порівнюючи реконструйовані зразки з оригінальними. Якщо модель не може точно відтворити дані, це може свідчити про аномалію, що корисно, наприклад, у фінансових або медичних застосуваннях.
- Синтез даних -VAE можуть бути використані для створення синтетичних даних, що корисно в ситуаціях, де реальні дані важко отримати, наприклад, у медицині для навчання моделей безпеки.
- Дослідження латентного простору- VAE дозволяють досліджувати латентний простір, що може бути корисним для інтерполяції між даними, виконання маніпуляцій над даними (наприклад, зміна стилю зображення) та інших експериментів.

Інші методи генерації текстур такі як: PixelCNN, PixelSNAIL Autoregressive Models, Flow-based Models, Diffusion Models [30].

Таблиця 2.2 - Порівняння VAE з GAN: переваги та недоліки

Критерій	VAE	GAN
Архітектура	Використовує кодувальник і декодувальник	Використовує генератор і дискримінатор
Стабільність навчання	Зазвичай більш стабільні	Можуть бути нестабільними
Якість генерованих даних	Часто менше деталізації	Висока якість зображень
Регуляризація латентного простору	Включена через втрату регуляризації	Відсутня, що може призводити до режимного колапсу
Застосування	Генерація, зменшення розмірності	Генерація, стилізація
Латентний простір	Структурований, що дозволяє інтерполяцію	Менш структурований, може бути хаотичним
Складність навчання	Простішою для налаштування	Вимагає більше експериментів для стабільності

PixelCNN є ще одним підходом до генерації зображень, який використовує конволюційні нейронні мережі для моделювання піксельних залежностей. Основні принципи роботи PixelCNN включають:

1. Моделювання пікселів: PixelCNN генерує зображення поступово, піксель за пікселем, починаючи з верхнього лівого кута та просуваючись до нижнього правого. Кожен піксель генерується на основі вже згенерованих сусідніх пікселів, що дозволяє моделювати складні залежності між ними. Це робить PixelCNN потужним інструментом для генерації текстур.

2. Умовна генерація: модель може бути адаптована для умовної генерації, що дозволяє генерувати зображення на основі додаткової інформації, такої як клас об’єкта або стиль. Це відкриває можливості для використання PixelCNN у задачах, де необхідно генерувати специфічні типи зображень.

3. Структура: PixelCNN використовує згорткові шари з масками, які забезпечують коректний порядок генерації пікселів, що дозволяє моделювати залежності між ними. Модель може містити кілька шарів, що сприяє захопленню різноманітних рівнів абстракції.

Серед переваг PixelCNN можна виділити високу якість згенерованих зображень, оскільки модель здатна моделювати залежності між пікселями, що особливо помітно в складних текстурах і деталях. Крім того, гнучкість моделі дозволяє адаптувати її для різних типів завдань генерації, включаючи умовну генерацію, що робить PixelCNN універсальним інструментом. Модель також може навчатися на різноманітних наборах даних, що дозволяє їй адаптуватися до різних стилів і характеристик.

Проте, PixelCNN має й недоліки. Генерація зображень піксель за пікселем може бути повільною, особливо для великих зображень, що обмежує практичне використання моделі в реальному часі. Крім того, високі вимоги до обчислювальних ресурсів через необхідність обробки кожного пікселя окремо можуть призвести до значних витрат на апаратне забезпечення.

PixelSNAIL є вдосконаленою версією PixelCNN, яка інтегрує механізм самоуваги для моделювання піксельних залежностей. Основні принципи роботи PixelSNAIL включають:

1. Самоувага: механізм самоуваги в PixelSNAIL дозволяє моделі фокусуватися на різних частинах зображення під час генерації, що сприяє захопленню глобальних залежностей та покращує якість згенерованих зображень.

2. Гнучкість у генерації: як і PixelCNN, PixelSNAIL може бути адаптований для умовної генерації, що робить його придатним для різних завдань.

3. Складні текстури: PixelSNAIL демонструє високі результати у генерації складних текстур і деталей, що робить його корисним для застосувань у мистецтві та дизайні.

Autoregressive Models: моделі, які генерують дані у послідовності, зазвичай використовують рекуррентні нейронні мережі (RNN) або трансформери. Вони моделюють дані по одному елементу за раз, що дозволяє їм захоплювати залежності між елементами.

Flow-based Models: ці моделі, такі як RealNVP і Glow, використовують обернені трансформації для моделювання розподілів даних. Вони дозволяють точно обчислювати ймовірності, що робить їх корисними для задач, де важлива точність.

Diffusion Models: новий клас генеративних моделей, які працюють шляхом поступового «згладжування» шуму до даних. Ці моделі показують обнадійливі результати у генерації зображень.

Отже, варіаційні автоенкодери (VAE) та інші методи, такі як PixelCNN, PixelSNAIL, а також нові підходи, як-от дифузійні моделі, є важливими альтернативами GAN у генерації даних. Кожен підхід має свої переваги та недоліки, що робить їх придатними для різних завдань. VAE забезпечують стабільність та можливість зменшення розмірності, тоді як PixelCNN і PixelSNAIL демонструють високу якість генерації зображень завдяки моделюванню піксельних залежностей.

2.3 Модель GAN для генерації текстур

Генеративні змагальні мережі (GAN) є архітектурою глибокого навчання, що складається з двох нейронних мереж: генератора та дискримінатора. Генератор відповідає за створення нових даних, зокрема текстур, на основі випадкового шуму, тоді як дискримінатор оцінює, чи є дані реальними (з навчального набору) чи згенерованими генератором. Обидві мережі навчаються одночасно: генератор прагне створювати все більш реалістичні текстури, в той

час як дискримінатор вдосконалює свої навички розпізнавання підроблених текстур.

GAN можна описати як гру з нульовою сумою, де генератор намагається мінімізувати помилку дискримінатора, а дискримінатор, у свою чергу, прагне зменшити свою помилку, правильно класифікуючи зображення. Основною метою GAN є досягнення Нешевої рівноваги, при якій генератор створює текстури, що не можуть бути відрізнені дискримінатором від справжніх [33].

Архітектура GAN складається з генератора, дискримінатора та комбінованої моделі (див. Рисунок 2.3). Генератор є нейронною мережею, яка приймає випадковий шум (вектор з латентного простору) і перетворює його в текстуру. Основні функції генератора включають перетворення випадкового шуму у високоякісне зображення, використання транспонованих згорткових шарів для поступового збільшення розміру зображення, а також стабілізацію навчання за допомогою Batch Normalization.

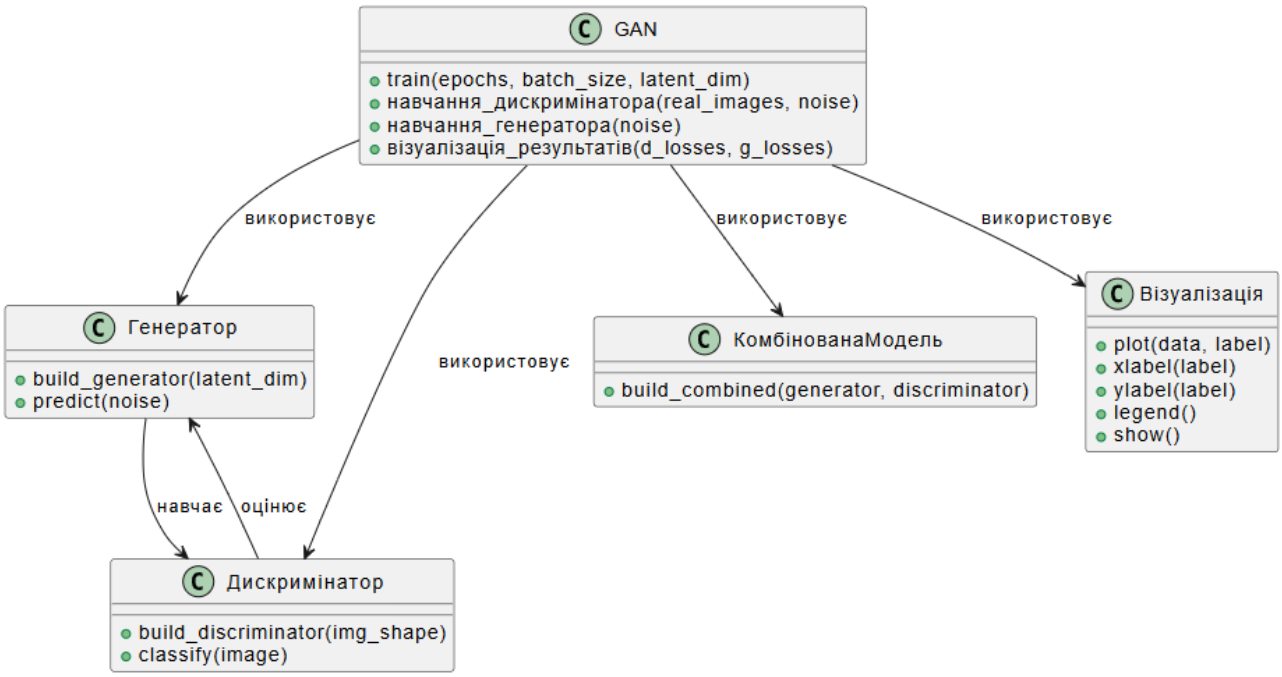

Рисунок 2.3 -Архітектура Генеративної Змагальної Мережі (GAN)

Деталізація архітектури генератора:

1. Вхідний шар: вхідним сигналом є випадковий вектор $z \in R^d$, де d=100 – розмір латентного простору. Цей вектор генерується з нормального розподілу N(0,1).

2. Повнозв'язний шар: перший шар перетворює вхідний вектор у тензор розміру 4×4×512 з використанням функції активації.

3. Транспоновані згорткові шари: енератор поступово збільшує розмір зображення від 4×4 до 64×64. Кожен шар включає фільтри з кількістю 256, 128 та 64, зменшуючи їх кількість на кожному етапі. Використовується stride = 2 для збільшення розмірів, а також Batch Normalization для стабілізації градієнтів. Функцією активації залишається ReLU.

4. Вихідний шар: останній шар генерує зображення розміру 64×64×36 (RGB), використовуючи функцію активації tanh для нормалізації пікселів у діапазоні [-1, 1].

Код генератора:

```
from tensorflow.keras import layers, models
def build_generator(latent_dim):
    model = models.Sequential()
    # Вхідний шар
    model.add(layers.Dense(4 * 4 * 512, input_dim=latent_dim))
    model.add(layers.Reshape((4, 4, 512)))  # Перетворення в тензор
    model.add(layers.BatchNormalization())
    model.add(layers.ReLU())
    # Транспоновані згорткові шари
    model.add(layers.Conv2DTranspose(256, kernel_size=4, strides=2,
padding='same'))
    model.add(layers.BatchNormalization())
    model.add(layers.ReLU())
    model.add(layers.Conv2DTranspose(128, kernel_size=4, strides=2,
padding='same'))
    model.add(layers.BatchNormalization())
    model.add(layers.ReLU())
    model.add(layers.Conv2DTranspose(64, kernel_size=4, strides=2,
padding='same'))
    model.add(layers.BatchNormalization())
    model.add(layers.ReLU())
    # Вихідний шар
    model.add(layers.Conv2DTranspose(3, kernel_size=4, strides=2,
padding='same', activation='tanh'))
    return model
```

Дискримінатор – це згорткова нейронна мережа, яка класифікує зображення як «реальні» або «фейкові».

Функції дискримінатора:

– приймати зображення і визначати ймовірність того, що воно є справжнім;

– використання згорткових шарів для вилучення особливостей з текстури;

– використання LeakyReLU для уникнення проблеми «вмирання нейронів».

Деталізація архітектури дискримінатора:

1. Вхідний шар:

– Вхід – зображення розміру 64×64×36.

2. Згорткові шари:

– Послідовність згорткових шарів, що зменшують просторові розміри зображення:

$$64\times64\rightarrow32\times32\rightarrow16\times16\rightarrow8\times8\rightarrow4\times4.$$

Кожен шар включає:

Фільтри: 64, 128, 256, 512 (кількість фільтрів збільшується на кожному шарі).

Stride = 2 для зменшення розмірів.

LeakyReLU як функцію активації (α=0.2).

3. Повнозв’язний шар:

– Останній шар видає ймовірність того, що зображення є справжнім.

– Функція активації: sigmoid.

Код дискримінатора:

```
def build_discriminator(img_shape):
    model = models.Sequential()
    # Згорткові шари
    model.add(layers.Conv2D(64,      kernel_size=4,      strides=2,
padding='same', input_shape=img_shape))
    model.add(layers.LeakyReLU(alpha=0.2))
    model.add(layers.Conv2D(128,     kernel_size=4,      strides=2,
padding='same'))
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU(alpha=0.2))
    model.add(layers.Conv2D(256,     kernel_size=4,      strides=2,
padding='same'))
```

```
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU(alpha=0.2))
    model.add(layers.Conv2D(512,     kernel_size=4,     strides=2,
padding='same'))
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU(alpha=0.2))
    # Вихідний шар
    model.add(layers.Flatten())
    model.add(layers.Dense(1, activation='sigmoid'))
    return model
```

Для навчання генератора ми об'єднуємо генератор і дискримінатор у «комбіновану» модель. Дискримінатор у цьому випадку заморожується (його ваги не оновлюються).

Код комбінованої моделі:

```
def build_combined(generator, discriminator):
    discriminator.trainable = False  # Заморожуємо дискримінатор
    model = models.Sequential([generator, discriminator])
    return model
```

3. Навчання GAN складається з двох основних етапів:

1. Навчання дискримінатора:

– подати реальні зображення з навчального набору.

– подати згенеровані зображення від генератора.

– оновити ваги дискримінатора.

2. Навчання генератора:

– Згенерувати зображення та передати їх дискримінатору.

– Оновити ваги генератора, щоб дискримінатор не зміг відрізнити ці зображення від справжніх.

Функції втрат:

– Для дискримінатора: $Loss_D = -(\log(D(x_{real})) + \log(1 - D(G(z))))$

– Для генератора: $Loss_G = -\log(D(G(z)))$

Код навчання GAN:

```
import numpy as np
```

```
epochs = 10000
batch_size = 64
latent_dim = 100
for epoch in range(epochs):
```

1. Навчання дискримінатора

```
    real_images = get_real_images(batch_size)  # Реальні текстури
    noise = np.random.normal(0, 1, (batch_size, latent_dim))
    fake_images = generator.predict(noise)  # Фейкові текстури
    real_labels = np.ones((batch_size, 1))  # Реальні = 1
    fake_labels = np.zeros((batch_size, 1))  # Фейкові = 0
    d_loss_real     =     discriminator.train_on_batch(real_images,
real_labels)
    d_loss_fake     =     discriminator.train_on_batch(fake_images,
fake_labels)
    d_loss = 0.5 * np.add(d_loss_real, d_loss_fake)
```

2. Навчання генератора

```
  noise = np.random.normal(0, 1, (batch_size, latent_dim))
    misleading_labels = np.ones((batch_size, 1))  # Генератор хоче
«обдурити» дискримінатор
    g_loss = combined.train_on_batch(noise, misleading_labels)
    print(f»Epoch  {epoch+1}/{epochs},  D  Loss:  {d_loss},  G  Loss:
{g_loss}»)
```

4. Візуалізація результатів:

1. Графік втрат: Відображає втрати генератора і дискримінатора під час навчання.

```
plt.plot(d_losses, label=«Discriminator Loss»)
plt.plot(g_losses, label=«Generator Loss»)
plt.xlabel(«Epoch»)
plt.ylabel(«Loss»)
plt.legend()
plt.show()
```

2. Збереження згенерованих текстур:

```
def save_generated_images(epoch, generator, latent_dim, n=5):
    noise = np.random.normal(0, 1, (n * n, latent_dim))
    generated_images = generator.predict(noise)
    # Візуалізація або збереження
```

Generative Adversarial Networks (GAN) – це потужний інструмент для створення реалістичних текстур та інших даних. GAN працює на основі двох

нейронних мереж, генератора та дискримінатора, які змагаються між собою, поступово вдосконалюючи свої результати.

2.4 Висновок до розділу

У другому розділі було проведено детальний аналіз алгоритмів машинного навчання, що застосовуються для генерації текстур. Розглянуті методи включали традиційні алгоритми, такі як дерева рішень та методи опорних векторів (SVM), а також сучасні підходи, зокрема нейронні мережі, з акцентом на генеративні змагальні мережі (GAN).

Узагальнення результатів аналізу

1. Дерева рішень та SVM показали свою ефективність у простих задачах, але їхня продуктивність знижується при обробці складних даних. Основними перевагами є простота реалізації та інтерпретованість, проте вони схильні до перенавчання і мають обмежену здатність до моделювання складних залежностей.

2. Нейронні мережі, зокрема GAN, продемонстрували значний потенціал у генерації реалістичних текстур завдяки своїй здатності виявляти складні патерни в даних. Вони здатні автоматично виділяти ознаки з необроблених даних, що робить їх особливо корисними у творчих сферах.

Для практичної реалізації було обрано генеративні змагальні мережі (GAN). Цей вибір обґрунтовано їхньою здатністю генерувати високоякісні текстури, які важко відрізнити від реальних. GAN забезпечують гнучкість у налаштуванні та адаптації до різних стилів, що робить їх ідеальними для задач у комп’ютерній графіці та дизайні.

3 РЕАЛІЗАЦІЯ МОДЕЛЕЙ ГЕНЕРАЦІЇ ТЕКСТУР

3.1. Інструменти і технології реалізації

Реалізація моделей генерації текстур – це складний багатоступеневий процес, який вимагає ретельного вибору інструментів, мов програмування, бібліотек і апаратного забезпечення. Від правильного вибору залежить продуктивність, точність моделей, легкість їхньої реалізації та можливість інтеграції в реальні додатки. У цьому підрозділі ми детально розглянемо всі ключові аспекти, пов’язані з вибором технологій, щоб забезпечити повне розуміння кожного елемента. Основними фреймворками для реалізації генеративних моделей є TensorFlow, PyTorch і Keras. Розглянемо кожен із них максимально детально.

TensorFlow – це один із найпотужніших і найпопулярніших фреймворків для машинного навчання, розроблений компанією Google. Він широко використовується як у дослідницьких задачах, так і у виробничих середовищах. TensorFlow підтримує роботу з усіма основними типами моделей, включаючи генеративно-змагальні мережі (GAN), автоенкодери (AE) та інші алгоритми, які часто використовуються для генерації текстур [34].

Переваги TensorFlow:

1. Гнучкість у побудові моделей:

TensorFlow надає можливість створення як простих, так і складних архітектур моделей. Наприклад, для генерації текстур можна використовувати GAN, які складаються з генератора та дискримінатора. Фреймворк дозволяє реалізувати як базові GAN, так і їхні складніші варіації, такі як StyleGAN або CycleGAN.

2. Підтримка низькорівневих і високорівневих API:

TensorFlow пропонує два рівні роботи. Низькорівневий API дозволяє створювати кастомні архітектури моделей, змінювати функції втрат,

оптимізатори тощо. Високорівневий API, представлений Keras, спрощує розробку моделей завдяки інтуїтивному інтерфейсу.

3. Масштабованість - TensorFlow підтримує навчання на різних платформах, включаючи локальні машини, сервери, мобільні пристрої та хмарні середовища. Це дозволяє масштабувати проєкти від невеликих експериментів до великих промислових систем.

4. Оптимізація для апаратного забезпечення - TensorFlow підтримує навчання на CPU, GPU і TPU (Tensor Processing Unit). TPU – це спеціалізоване обладнання від Google, яке забезпечує значне прискорення обчислень. Для задач генерації текстур, де моделі можуть бути дуже великими (наприклад, StyleGAN), використання TPU може значно скоротити час навчання.

5. Інструменти для візуалізації - TensorBoard – це потужний інструмент для візуалізації процесу навчання. Він дозволяє відстежувати зміну функції втрат, точності моделі, а також візуалізувати проміжні результати, такі як згенеровані текстури на різних етапах навчання.

6. Широка екосистема- TensorFlow має багато додаткових інструментів, зокрема TensorFlow Lite для розгортання моделей на мобільних пристроях, TensorFlow.js для роботи з моделями у веб-браузерах, а також TensorFlow Extended (TFX) для впровадження моделей у виробниче середовище.

Серед недоліків TensorFlow можна відзначити складність освоєння для початківців, особливо при роботі з низькорівневими API, а також менш інтуїтивний синтаксис у порівнянні з PyTorch.

Приклад використання TensorFlow для генерації текстур:

```
import tensorflow as tf
from tensorflow.keras import layers
# Побудова генератора для GAN
def build_generator():
    model = tf.keras.Sequential([
        layers.Dense(256, activation=«relu», input_dim=100),
        layers.Reshape((16, 16, 1)),
        layers.Conv2DTranspose(64,  (3,  3),  activation=«relu»,
strides=(2, 2), padding=«same»),
```

```
        layers.Conv2DTranspose(1,    (3,    3),    activation=«tanh»,
padding=«same»)
    ])
    return model
generator = build_generator()
generator.summary()
```

PyTorch – це фреймворк для глибокого навчання, розроблений Facebook. Він відомий своєю простотою, гнучкістю та активною підтримкою дослідницької спільноти. PyTorch часто використовується у наукових дослідженнях та експериментальних проектах, що робить його ідеальним для задач генерації текстур [36].

Переваги PyTorch:

1. Динамічний обчислювальний граф - У PyTorch граф обчислень створюється «на льоту», тобто під час виконання програми. Це дозволяє легко змінювати архітектуру моделі в процесі її розробки.

2. Інтуїтивний синтаксис - PyTorch має синтаксис, схожий на Python, що робить його зручним для розробників. Наприклад, операції з тензорами в PyTorch нагадують роботу з бібліотекою NumPy.

3. Гнучкість - PyTorch дозволяє створювати кастомні архітектури моделей, що особливо важливо для експериментальних задач, таких як генерація текстур.

4. Широка підтримка дослідницької спільноти - завдяки популярності PyTorch у наукових колах існує велика кількість готових реалізацій статей, бібліотек та прикладів, які можна використовувати у власних проектах

Недоліки PyTorch:

– Менше інструментів для розгортання моделей у виробниче середовище порівняно з TensorFlow.

– Відсутність аналогів TensorFlow Lite або TensorFlow.js.

Приклад використання PyTorch для генерації текстур:

```
import torch
import torch.nn as nn
# Побудова генератора для GAN
```

```
class Generator(nn.Module):
    def __init__(self):
        super(Generator, self).__init__()
        self.model = nn.Sequential(
            nn.Linear(100, 256),
            nn.ReLU(),
            nn.Linear(256, 1024),
            nn.Tanh()
        )
    def forward(self, x):
        return self.model(x)
generator = Generator()
print(generator)
```

Keras – це високорівневий API для роботи з нейронними мережами, інтегрований у TensorFlow. Його основна мета – спростити процес створення моделей [37].

Переваги Keras:

1. Простота використання: Завдяки інтуїтивному синтаксису Keras дозволяє швидко створювати прототипи моделей

2. Попередньо навчені моделі: Keras надає доступ до багатьох попередньо навчених моделей, які можна адаптувати для генерації текстур

3. Інтеграція з TensorFlow: Keras використовує всі можливості TensorFlow, зокрема підтримку GPU та TPU.

Недоліки Keras:

- Менша гнучкість у порівнянні з PyTorch або низькорівневими API TensorFlow.

- Обмежена функціональність для кастомізації складних моделей.

Приклад використання Keras:

```
from tensorflow.keras import layers, models
# Побудова автоенкодера
def build_autoencoder():
    encoder = models.Sequential([
        layers.Input(shape=(64, 64, 1)),
        layers.Conv2D(32,      (3,      3),      activation=«relu»,
padding=«same»),
        layers.MaxPooling2D((2, 2), padding=«same»)
    ])
    decoder = models.Sequential([
```

```
        layers.Conv2DTranspose(32,   (3,   3),   activation=«relu»,
padding=«same»),
        layers.UpSampling2D((2, 2))
    ])
    return encoder, decoder
encoder, decoder = build_autoencoder()
encoder.summary()
decoder.summary()
```

Діаграма нижче (рисунок 3.1) ілюструє основні характеристики та взаємозв'язки між фреймворками TensorFlow, PyTorch і Keras, що використовуються для генерації текстур.

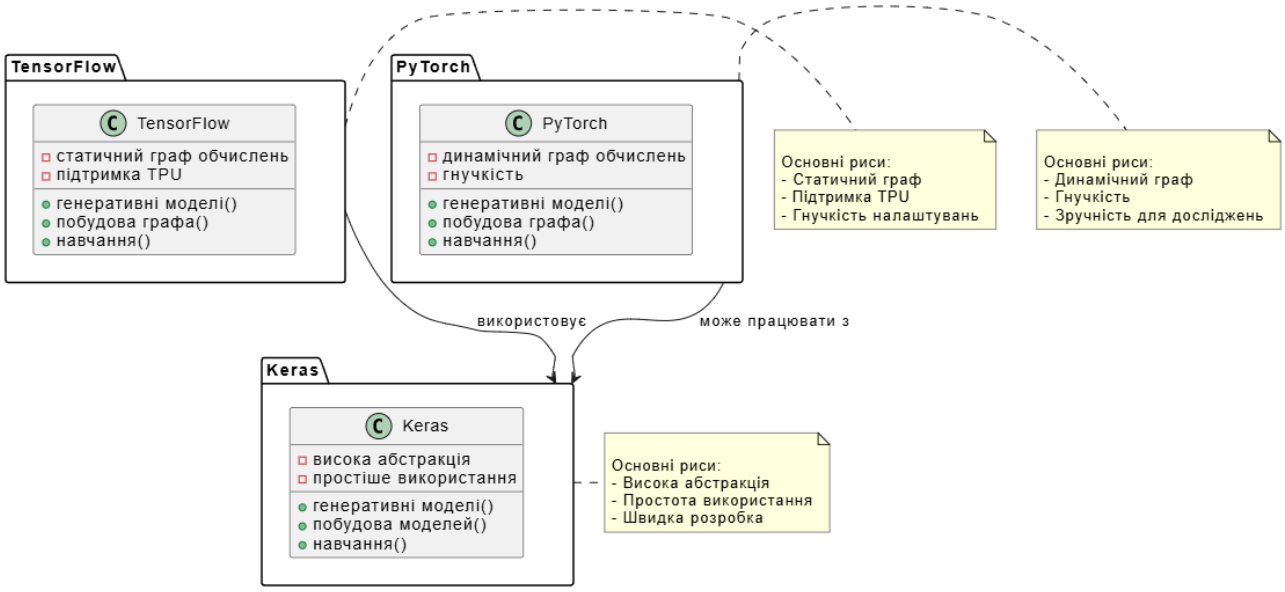

Рисунок 3.1 - Порівняння фреймворків для генеративних моделей

Вибір мови програмування:

Python є основною мовою для роботи з TensorFlow, PyTorch і Keras. Його популярність пояснюється простим синтаксисом, широким спектром бібліотек і потужними інструментами для обробки даних.

R може бути корисним для статистичного аналізу даних або візуалізації, але для генерації текстур ця мова використовується рідше.

Обладнання для навчання моделей:

1. GPU: підходять для паралельних обчислень, що значно прискорює навчання моделей.

2. TPU: спеціалізовані процесори для TensorFlow, що забезпечують високу продуктивність.

3. Хмарні обчислення: використання хмарних сервісів (Google Cloud, AWS) дозволяє масштабувати ресурси.

Отже, для генерації текстур оптимально використовувати TensorFlow або PyTorch, мову програмування Python, а для навчання моделей – GPU або хмарні обчислення. TPU можуть бути корисними для масштабних задач у TensorFlow.

3.2. Підготовка даних для генерації текстур

Підготовка даних для генерації текстур є багатоступеневим процесом, який має вирішальне значення для створення високоякісних текстур за допомогою алгоритмів машинного навчання. Цей процес включає не лише збір і структурування даних, але й їхню ретельну обробку, аналіз, аугментацію та оптимізацію для конкретних завдань. Мета – створити набір даних, який буде максимально придатним для навчання моделі, забезпечуючи високу точність і реалістичність результатів.

Збір якісних даних – це основа всього процесу. Тут важливо врахувати джерела, методи збору, специфіку текстур і кінцеві вимоги до моделі.

1. Джерела текстур: публічні бази даних текстур, створення власних текстур, процедурна генерація текстур.

Публічні бази даних текстур: готові бази даних є швидким і зручним способом отримання текстур. Однак важливо ретельно перевіряти якість текстур, їхню роздільну здатність, формат і ліцензію.

Приклади баз даних:

1. CC0 Textures - дана платформа пропонує текстури з відкритою ліцензією (CC0), що дозволяє їх використання без обмежень. Вона містить категорії, такі як деревина, бетон, метал, тканини, плитка та скло, а також надає текстури з роздільною здатністю до 8K. Особливістю є те, що текстури вже

підготовлені до використання, включаючи безшовні варіанти, нормальні карти та карти висоти.

2. Textures.com - ця база даних налічує понад 100 000 текстур різного типу. Користувачі можуть знайти текстури, що охоплюють природу (каміння, земля) та штучні матеріали (бетон, асфальт), а також тканини і метал. Деякі текстури доступні безкоштовно, однак для доступу до преміум-текстур необхідна підписка.

3. Quixel Megascans - є однією з найбільших баз даних реалістичних текстур і 3D-сканів. Текстури створені за допомогою фотограмметрії, що забезпечує високу деталізацію. Ця платформа активно використовується в ігровій індустрії, зокрема в Unreal Engine.

4. Poly Haven - є платформою для безкоштовних текстур, HDRI та 3D-моделей. Всі ресурси мають відкриту ліцензію (CC0), що дозволяє їх використання в будь-яких проєктах.

Створення власних текстур: якщо публічні бази даних не задовольняють вимоги проєкту (наприклад, потрібні специфічні текстури або унікальні матеріали), можна створити власний набір текстур.

Для ефективного фотографування текстур рекомендується використовувати камери високої роздільної здатності, такі як DSLR або сучасні смартфони. Важливими аспектами цього процесу є належне освітлення, яке повинно бути рівномірним, без тіней і відблисків; для досягнення цього ефекту доцільно використовувати розсіяне світло або проводити зйомку у похмурий день.

Крім того, кут зйомки має бути перпендикулярним до поверхні об’єкта, що дозволяє уникнути спотворень зображення. Використання штативу є ще одним важливим чинником, оскільки він забезпечує стабільність та чіткість знімків. Рекомендується фотографувати у максимально можливій роздільній здатності, щоб зберегти деталі текстури.

Прикладами текстур, які можна фотографувати, є деревина, тканини, камінь, асфальт, метал та плитка, що дозволяє створити різноманітний набір ресурсів для подальшого використання в графічних проєктах.

Сканування текстур є ефективним методом для створення текстур з реальних об’єктів, і для цього можуть використовуватися як 3D-сканери, так і фотограмметрія.

Фотограмметрія передбачає виконання серії фотографій об’єкта з різних кутів, після чого ці зображення обробляються за допомогою спеціалізованих програм, таких як Agisoft Metashape або RealityCapture. Цей підхід дозволяє отримати детальну тривимірну модель об’єкта, що може бути використана для подальшого створення текстур.

З іншого боку, 3D-сканери, такі як Artec Eva або Structure Sensor, забезпечують швидке та точне сканування об’єктів, що дозволяє отримати високоякісні дані про їхню геометрію та текстуру. Використання цих технологій значно розширює можливості для створення реалістичних текстур у графічному дизайні та 3D-моделюванні.

Ручне створення текстур у графічних редакторах є важливим етапом у процесі графічного дизайну. Для цього використовують програми, такі як Adobe Photoshop, GIMP або Krita, які дозволяють створювати текстури з нуля. Процес включає кілька ключових етапів: спочатку розробляється основний шаблон текстури, що слугує базовою основою. Наступним кроком є додавання деталей, таких як шум, градієнти та текстури поверхонь, що сприяє досягненню необхідної реалістичності. Завершальним етапом є використання спеціалізованих інструментів для створення безшовних текстур, що забезпечує їх безперервність при повторному застосуванні в графічних проєктах.

Процедурна генерація текстур: дозволяє створювати текстури за допомогою алгоритмів, що генерують зображення на основі математичних функцій.

Інструменти:

1. Substance Designer: Це інструмент для створення процедурних текстур із використанням вузлів. Наприклад, можна генерувати текстуру цегляної стіни з варіаціями кольору та форми.

2. Blender: має вбудовані інструменти для процедурної генерації текстур, такі як шум Perlin і Voronoi.

3. Custom Scripts: Використовуйте Python, зокрема бібліотеку OpenCV, для створення текстур на основі шуму, градієнтів тощо.

Оцінка та перевірка зібраних текстур є критично важливими етапами після їх збору, оскільки якість текстур безпосередньо впливає на результати візуалізації. Основними критеріями оцінки є роздільна здатність, формат та загальна якість текстур.

По-перше, текстури повинні мати достатню роздільну здатність для виконання специфічних завдань, зокрема, рекомендується використовувати значення, такі як 1024x1024 або 2048x2048 пікселів для 3D-графіки. Це забезпечує необхідний рівень деталізації, що є важливим для досягнення реалістичних результатів.

По-друге, важливим аспектом є формат збереження текстур. Рекомендується використовувати формати PNG або TIFF, оскільки вони сприяють збереженню високої якості зображення. Формат JPEG може бути застосований для менш вимогливих завдань, проте він не забезпечує такої ж якості, як безсжаті формати.

Нарешті, необхідно ретельно перевіряти текстури на наявність дефектів, таких як шум, нерівномірне освітлення або артефакти. Видалення таких недоліків є важливим для забезпечення високої якості фінальних матеріалів у графічних проєктах.

Попередня обробка текстур включає кілька кроків, які готують дані до навчання моделі: нормалізація даних, видалення швів (seamless textures), аугментація даних, розподіл на набори.

Нормалізація даних: є важливим етапом для забезпечення однаковості всіх текстур, що використовуються в графічних проєктах. Цей процес включає кілька ключових етапів.

По-перше, зміна розміру текстур є необхідною для досягнення однорідності. Всі текстури повинні мати однакові розміри, наприклад, 256x256 або 512x512 пікселів. Для цього можна використовувати бібліотеки, такі як OpenCV або PIL. Наприклад, за допомогою бібліотеки PIL можна виконати наступні дії:

```
from PIL import Image
image = Image.open("texture.jpg")
resized_image = image.resize((256, 256))
resized_image.save("resized_texture.jpg")
```

По-друге, важливим аспектом нормалізації є масштабування піксельних значень. Рекомендується привести значення пікселів до діапазону [0, 1] або [-1, 1], що сприятиме стабільному навчальному процесу моделі.

По-третє, перетворення кольорових просторів також є важливим етапом. Якщо модель працює з чорно-білими текстурами, необхідно конвертувати зображення у відтінки сірого. Це можна здійснити за допомогою наступного коду:

```
gray_image = image.convert("L")
```

Таким чином, ці етапи нормалізації даних забезпечують підвищення якості та узгодженості текстур у процесах комп’ютерної графіки.

Видалення швів, або створення безшовних текстур, є критично важливим аспектом для багатьох завдань у комп’ютерній графіці, зокрема в 3D-моделюванні та ігровій індустрії. Безшовні текстури забезпечують плавний перехід між зразками, що сприяє досягненню реалістичних візуальних ефектів.

Для створення безшовних текстур можна використовувати різноманітні інструменти. Наприклад, у програмі Photoshop доступний інструмент «Offset», який дозволяє перевіряти повторюваність текстури. Цей інструмент допомагає

виявити шви та артефакти, що можуть виникнути під час процесу текстурування. Після цього рекомендується використовувати клонувальний штамп для згладжування стиків, що забезпечить безперервність текстури.

Крім того, програма Substance Designer пропонує вбудовані вузли, спеціально розроблені для створення безшовних текстур. Ці вузли автоматизують процес, спрощуючи роботу дизайнерів і дозволяючи досягати високої якості результатів. Таким чином, використання зазначених інструментів є важливим кроком у забезпеченні безшовності текстур у графічних проєктах.

Аугментація даних: збільшує різноманітність даних, що покращує здатність моделі до генералізації.

Методи аугментації:

1. Обертання (90°, 180°, 270°).
2. Віддзеркалення (горизонтальне, вертикальне).
3. Додавання шуму.
4. Розмиття.
5. Зміна яскравості, контрасту, насиченості.
6. Зміщення текстури (random cropping).

Приклад коду для аугментації:

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
datagen = ImageDataGenerator(
    rotation_range=45,
    width_shift_range=0.1,
    height_shift_range=0.1,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
     augmented_data =
     datagen.flow_from_directory(«textures_dataset»,
     target_size=(256, 256))
```

Розподіл на набори: розподіліть дані на три основні набори:

1. Навчальний набір (Train): Використовується для навчання моделі (70-80% даних).

2. Тестовий набір (Test): Використовується для оцінки продуктивності моделі (20-30% даних).

3. Валідаційний набір (Validation): Використовується для налаштування гіперпараметрів (10-15% від навчального набору).

Тому, процес підготовки даних для генерації текстур є складним і багатогранним. Успіх навчання моделі залежить від якості текстур, їхньої обробки та структурованості. Кожен етап – від збору текстур до їхньої аугментації – має бути виконаний ретельно, з урахуванням специфіки завдання.

3.3 Тестування і валідація моделі GAN для генерації текстур

Генерація текстур за допомогою GAN (Generative Adversarial Networks) є складним завданням, яке вимагає ретельного тестування та валідації для оцінки якості результатів. GAN створюють текстури, які повинні бути не тільки реалістичними, але й різноманітними, без видимих дефектів чи артефактів. Для цього використовуються кількісні метрики, візуальний аналіз і експертна оцінка. Нижче наведено детальний опис кожного аспекту.

1. Ключові аспекти оцінки якості текстур:

Перш ніж перейти до конкретних методів тестування, важливо зрозуміти, які характеристики текстур ми оцінюємо:

1. Реалістичність: наскільки текстури виглядають природно для людського ока. Чи відповідають вони візуальним властивостям реальних текстур (колір, деталі, структура).

2. Різноманітність:

– чи здатна модель створювати широкий спектр текстур, чи вона повторює одні й ті самі патерни;

– чи є генерація унікальною, а не копіює реальні приклади.

3. Відсутність дефектів (артефактів) - чи немає в текстурах шуму, розмиття, викривлень або інших візуальних дефектів, які можуть виникати через недоліки моделі.

4. Відповідність контексту - чи відповідають текстури вимогам конкретної області (наприклад, текстури для 3D-моделей, ігор, архітектури тощо).

2. Метрики для оцінки якості згенерованих текстур - для автоматизованої оцінки якості текстур використовуються кількісні метрики. Вони дозволяють об’єктивно оцінити результати без втручання людини.

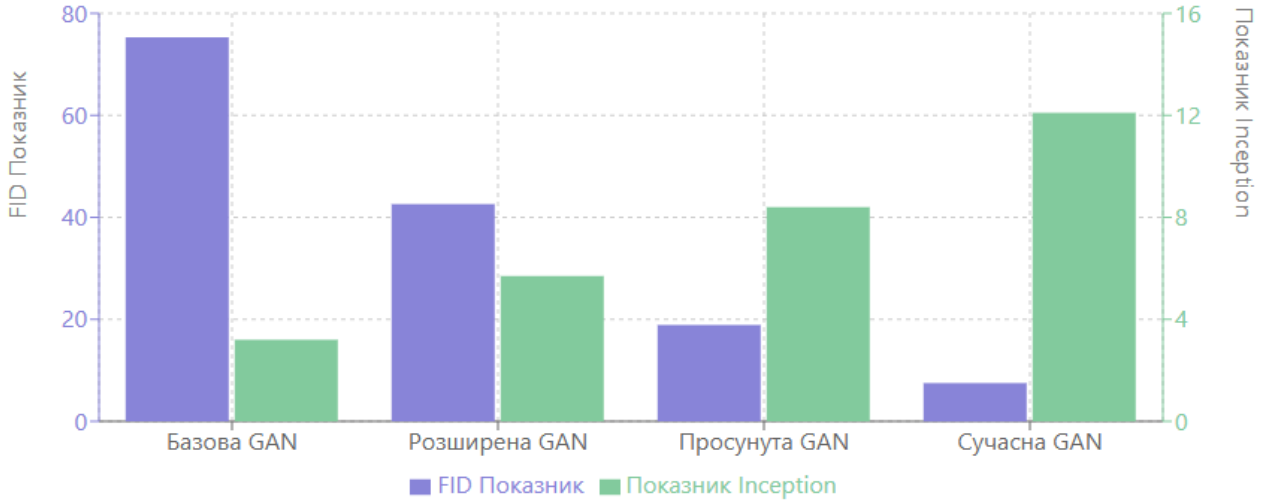

Рисунок 3.3 - Порівняння метрик якості текстур: FID та Inception Score

Графік ілюструє два ключові показники якості текстур, а саме: FID Score (Частотна відстань Inception) та Inception Score. Перший показник, представлений синім кольором, демонструє, що нижче значення FID свідчить про вищу якість текстур. Спостерігається поступове зниження цього показника від базової до найсучаснішої моделі. Натомість Inception Score, позначений зеленим кольором, відображає реалістичність і різноманітність генерованих текстур; вищі значення цього показника свідчать про кращу якість. Зазначений показник також поступово зростає від базової до найсучаснішої моделі.

У порівнянні моделей спостерігається наступна динаміка: базова GAN демонструє низьку якість з FID = 75.3 та IS = 3.2; розширена GAN має середню якість з FID = 42.6 та IS = 5.7; просунута GAN забезпечує високу якість з FID = 18.9 та IS = 8.4; найсучасніша GAN досягає дуже високої якості з FID = 7.5 та IS = 12.1. Графік підтверджує висновки з попереднього аналізу, де FID < 50 вказує на прийнятну якість, а IS > 5 свідчить про хорошу якість текстур.

На графіку використано дві шкали: ліва шкала, позначена синім кольором, відображає значення FID Score, тоді як права шкала, представлена зеленим кольором, демонструє Inception Score. Таким чином, графік наочно ілюструє еволюцію моделей GAN, від базової версії з низькою якістю до найсучаснішої, яка генерує практично бездоганні текстури.

2.1. Fréchet Inception Distance (FID)

Fréchet Inception Distance (FID) є метрикою, що вимірює відстань між розподілом ознак реальних і згенерованих текстур. Вона ґрунтується на статистичних властивостях цих розподілів, таких як середнє значення та коваріація. Основні етапи використання FID включають:

1. Збір ознак: реальні та згенеровані текстури проходять через попередньо натреновану модель, зазвичай Inception v3. З проміжних шарів моделі витягуються вектори ознак, які представляють високорівневі характеристики текстур.

2. Обчислення статистик: для кожного набору текстур обчислюються середнє значення векторів ознак (μr для реальних текстур і μg для згенерованих) та коваріаційні матриці (Σ_r і Σ_g).

3. Формула FID**:** FID обчислюється за формулою: $FID=||\mu_r-\mu_g||^2+Tr(\Sigma_r+\Sigma_g-2(\Sigma_r\Sigma_g)^{1/2})$, де $||\mu_r-\mu_g||^2$ є квадратом евклідової відстані між середніми значеннями ознак, а Tr позначає слід матриці.

Інтерпретація результатів FID є такою: значення FID = 0 свідчить про ідеальну відповідність між реальними та згенерованими текстурами, а менші значення FID вказують на кращу якість. Це підтверджує результати, представлені на Рис. 3.3.1, де видно, що значення FID зменшується з покращенням моделей.

Практичні орієнтири включають FID < 50 для прийнятної якості та FID < 10 для високої якості. Переваги FID полягають у його здатності оцінювати як якість, так і різноманітність текстур, а також у його ефективності для складних текстур з багатими деталями. Проте, він має недоліки, зокрема чутливість до вибору моделі (Inception v3) та можливу відсутність кореляції з суб'єктивним сприйняттям.

2.2. Inception Score (IS)

Inception Score (IS) вимірює дві основні властивості згенерованих текстур: реалістичність, тобто наскільки впевнено модель Inception v3 класифікує текстури, та різноманітність, що відображає належність текстур до різних класів. Основні етапи роботи IS включають:

1. Класифікація текстур: згенеровані текстури проходять через модель Inception v3, де для кожного зображення обчислюється розподіл ймовірностей класів $p(y \mid x)$, де y – клас, а x – текстура.

2. Обчислення IS: середній розподіл класів $p(y)$ обчислюється як середнє значення $p(y \mid x)$ по всіх текстурах. Для оцінки різниці між $p(y \mid x)$ та $p(y)$ використовується дивергенція Кульбака-Лейблера (KL-дивергенція), що формулюється як: $IS=\exp(E_x[D_{KL}(p(y \mid x) \mid\mid p(y))])$.

Високий IS свідчить про реалістичність та різноманітність текстур, причому значення IS > 5 вказує на хорошу якість, а IS > 10 – на дуже високу якість. Як видно з Рис. 3.3.1, Inception Score також зростає з покращенням моделей, що свідчить про зростання реалістичності текстур. Переваги IS полягають у простоті обчислення та ефективній оцінці різноманітності, проте він не враховує реальні текстури, не порівнюючи їх із згенерованими, і є чутливим до вибору моделі.

2.3. Precision and Recall for Distributions (PRD)

Precision and Recall for Distributions (PRD) оцінює баланс між якістю та різноманітністю текстур. Основні етапи роботи PRD включають:

1. Precision: частка згенерованих текстур, які належать до реального розподілу.

2. Recall: частка реальних текстур, які належать до згенерованого розподілу.

Переваги PRD полягають у тому, що він надає більш збалансовану оцінку, ніж FID або IS, оскільки враховує як якість, так і різноманітність текстур.

2.4. Kernel Inception Distance (KID)

KID – це альтернатива FID, яка використовує інші математичні підходи для оцінки схожості розподілів.

Переваги KID:

– Менш чутливий до розміру вибірки, ніж FID.

– Дає подібні результати, але з більшою стабільністю.

3. Візуальна оцінка текстур:

Автоматичні метрики не завжди здатні повністю відобразити якість текстур, тому візуальна оцінка виступає ключовим додатковим методом. Процес візуальної оцінки текстур включає кілька етапів.

1. Порівняння текстур: згенеровані текстури порівнюються з реальними. Оцінюються такі аспекти, як узгодженість кольорів і деталей, відсутність артефактів (наприклад, шуму, розмиття, викривлення) та природність текстури.

2. Методи оцінки: для візуальної оцінки текстур застосовуються різні методи. Суб’єктивна оцінка передбачає, що експерти або користувачі оцінюють текстури за певною шкалою (наприклад, від 1 до 5). Тест на розпізнавання передбачає, що людям демонструються реальні та згенеровані текстури в випадковому порядку, і вони повинні визначити, які з них є реальними.

4. Процес валідації:

Процес валідації моделей, що генерують текстури, складається з кількох ключових етапів.

1. Тестові набори даних: дані розділяються на три основні категорії: навчальний, валідаційний та тестовий набори. Тестовий набір використовується для остаточної оцінки ефективності моделі.

2. Крос-валідація: цей метод дозволяє оцінити стійкість моделі, розділяючи дані на кілька підмножин. Це забезпечує більш надійні результати оцінки.

3. Залучення експертів: експертам надаються текстури для оцінки, зокрема щодо їх реалістичності, різноманітності та відповідності контексту. Це дозволяє отримати глибше розуміння якості згенерованих текстур.

3.4 Висновки до розділу

У третьому розділі було проведено комплексний аналіз процесу реалізації моделей генерації текстур на основі генеративно-змагальних мереж (GAN). Розглянуті етапи включали вибір інструментів, підготовку даних, реалізацію моделей, а також тестування і валідацію отриманих результатів.

ВИСНОВКИ

1. Проведено аналіз традиційних підходів до генерації текстур, таких як ручне малювання, процедурна генерація та їх обмеження в швидкості та різноманітності. Визначено, що традиційні методи не завжди здатні забезпечити необхідну реалістичність, особливо в умовах високих вимог до якості. Досліджено основи машинного навчання, зокрема нейронних мереж, які відкривають нові можливості для автоматизації процесу створення текстур.

2. Проаналізовано сучасні алгоритми, зокрема генеративні змагальні мережі GAN, варіаційні автоенкодери VAE та інші підходи. Визначено їхні переваги та недоліки в контексті генерації текстур. GAN продемонстрували значний потенціал у створенні реалістичних текстур, які важко відрізнити від реальних, завдяки своїй здатності виявляти складні патерни у великих наборах даних.

3.Реалізовано модель GAN для генерації текстур. Генератор здатний створювати текстури, а дискримінатор ефективно відрізняє згенеровані зображення від реальних. Код реалізації продемонстрував ефективність використання транспонованих згорткових шарів у генераторі та згорткових шарів у дискримінаторі. Результати тестування показали, що модель не лише генерує текстури, але й демонструє різноманітність і якість, що є важливими критеріями для успішної реалізації в практичних застосуваннях.

4.Проведено експериментальне дослідження алгоритмів. Використання об’єктивних метрик (FID, IS) та візуальної оцінки підтвердило, що модель GAN здатна генерувати високоякісні текстури з різноманітними характеристиками. Визначено, що FID < 50 вказує на прийнятну якість, а IS > 5 свідчить про хорошу якість текстур. Це підкреслює важливість використання комплексного підходу до оцінки якості згенерованих текстур.

Список використаних джерел

1. Бондаренко С. В. Алгоритми обробки зображень на основі нейронних мереж: дис. ... канд. техн. наук: 05.13.23 / Харківський національний університет радіоелектроніки. Харків, 2020. 198 с.

2. Ковальчук О. О. Методи глибокого навчання для розпізнавання образів у комп'ютерному зорі: дис. ... канд. техн. наук: 05.13.23 / Національний технічний університет України «КПІ». – Київ, 2019. – 215 с.

3. Кравченко С. Методи класифікації машинного навчання з використанням бібліотеки scikit-learn / С.Кравченко, Є.Гришкун, О.Власенко [Електронний ресурс]. – Режим доступу : http://tech.vernadskyjournals.in.ua/journals/2020/3_2020/part_1/21.pdf.

4. Кравченко С. О. Генерація текстур у віртуальних середовищах на основі GAN // *Інформаційні технології та комп'ютерна інженерія*. – 2021. – Т. 47, № 1. – С. 89–94.

5. Партика П.М., Яворівський В.А. Порівняння традиційних і машинно-навчених підходів до генерації текстур. Інтелектуальні комп'ютерні системи та мережі: Всеукр. наук.-практ. конф. студентів, аспірантів та молодих вчених. (5листопада 2024 р. Тернопіль). Тернопіль, 2024. С. 133.

6. Петров В. І., Гончаренко І. М. Глибоке навчання для синтезу зображень: огляд підходів // *Вісник НТУ «ХПІ». Серія: Інформатика та моделювання*. – 2021. – № 12. – С. 56–62.

7. Сидоренко А. В., Ткаченко П. М. Застосування генеративно-змагальних мереж для синтезу текстур // *Наукові записки НаУКМА. Комп'ютерні науки*. – 2020. – Т. 3, № 2. – С. 45–52.

8. Яворівський В.А., Партика П.М. Функції активації та архітектури нейронних мереж для розпізнавання обличчя і аналізу емоцій. Інтелектуальні комп'ютерні системи та мережі: Всеукр. наук.-практ. конф. студентів, аспірантів та молодих вчених. (5листопада 2024 р. Тернопіль). Тернопіль, 2024. С. 89.

9. Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. arXiv preprint arXiv:2204.06125, 2022.

10. Arjovsky M., Chintala S., Bottou L. Wasserstein GAN // *arXiv preprint arXiv:1701.07875*. – 2017. – Режим доступу: https://arxiv.org/abs/1701.07875.

11. Bishop C. M. *Pattern Recognition and Machine Learning*. – Springer, 2006. – 738 с.

12. Chollet F. *Deep Learning with Python*. – Manning Publications Co., 2018. – 384 с.

13. Classification Techniques in Machine Learning: Applications and Issues. – 2017 [Електронний ресурс]. – Режим доступу : https://www.researchgate.net/publication/319370844_Classification_Techniques_in_Machine_Learning_Applications_and_Issues.

14. GANs in Computer Vision: https://towardsdatascience.com/generative-adversarial-networks-gans-in-computer-vision-5dfd5e7c4c8.

15. Gatys L. A., Ecker A. S., Bethge M. Image Style Transfer Using Convolutional Neural Networks // *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. – 2016. – С. 2414–2423.

16. Gatys L. A., Ecker A. S., Bethge M. Texture Synthesis Using Convolutional Neural Networks // *Advances in Neural Information Processing Systems (NeurIPS)*. – 2015. – С. 262–270.

17. Goodfellow I., Bengio Y., Courville A. *Deep Learning*. – MIT Press, 2016. – 775 с.

18. Goodfellow I., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A., Bengio Y. Generative Adversarial Networks // *arXiv preprint arXiv:1406.2661*. – 2014. – Режим доступу: https://arxiv.org/abs/1406.2661.

19. Ian Goodfellow, Yoshua Bengio, Aaron Courville. *Generative Models*. – MIT Press, 2020. – 512 с.

20. Isola P., Zhu J. Y., Zhou T., Efros A. A. Image-to-Image Translation with Conditional Adversarial Networks // *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. – 2017. – С. 5967–5976.

21. Johnson J., Alahi A., Fei-Fei L. Perceptual Losses for Real-Time Style Transfer and Super-Resolution // *European Conference on Computer Vision (ECCV)*. – 2016. – С. 694–711.

22. Karras T., Laine S., Aila T. A Style-Based Generator Architecture for Generative Adversarial Networks // *IEEE Transactions on Pattern Analysis and Machine Intelligence*. – 2019. – Т. 42, № 12. – С. 2794–2803.

23. Khanna R. Machine Learning / R.Khanna, M.Awad [Електронний ресурс]. – Режим доступу : https://link.springer.com/chapter/10.1007/978-1-4302-5990-9_1.

24. Mitchell T. Machine Learning / Tom M. Mitchell [Електронний ресурс]. – Режим доступу : https://www.cin.ufpe.br/~cavmj/Machine%20-%20Learning%20-%20Tom%20Mitchell.pdf.

25. Miyato T., Kataoka T., Koyama M., Yoshida Y. Spectral Normalization for Generative Adversarial Networks // *arXiv preprint arXiv:1802.05957*. – 2018. – Режим доступу: https://arxiv.org/abs/1802.05957.

26. Murphy K. P. *Machine Learning: A Probabilistic Perspective*. – MIT Press, 2012. – 1096 с.

27. Papers with Code: https://paperswithcode.com/task/image-generation.

28. Park T., Liu M. Y., Wang T. C., Zhu J. Y. Semantic Image Synthesis with Spatially-Adaptive Normalization // *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. – 2019. – С. 2337–2346.

29. PyTorch GAN Implementation Guide: https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html.

30. Radford A., Metz L., Chintala S. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks // *arXiv preprint arXiv:1511.06434*. – 2015. – Режим доступу: https://arxiv.org/abs/1511.06434.

31. Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition // *arXiv preprint arXiv:1409.1556*. – 2014.

32. Smola A. Introduction to machine learning / A.Smola, S.Vishwanathan [Електронний ресурс]. – Режим доступу : https://alex.smola.org/drafts/thebook.pdf.

33. TensorFlow GAN Tutorials: https://www.tensorflow.org/tutorials/generative.

34. Ulyanov D., Vedaldi A., Lempitsky V. Instance Normalization: The Missing Ingredient for Fast Stylization // *arXiv preprint arXiv:1607.08022*. – 2016. – Режим доступу: https://arxiv.org/abs/1607.08022.

35. Wang T. C., Liu M. Y., Zhu J. Y., Tao A., Kautz J., Catanzaro B. High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs // *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. – 2018. – С. 8798–8807.

36. Xiaohui Zeng, Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, and Karsten Kreis. Lion: Latent point diffusion models for 3d shape generation. In Advances in Neural Information Processing Systems (NeurIPS), 2022.

37. Zhu J. Y., Park T., Isola P., Efros A. A. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks // *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. – 2017. – С. 2223–2232.

38. Березький О.М., Мельник Г.М. Методичні рекомендації до виконання кваліфікаційної роботи з освітнього ступеня “Магістр”. Спеціальність: 123 - Комп’ютерна інженерія. Магістерська програма - Комп’ютерна інженерія". Тернопіль: ЗУНУ, 2024. 32 с.