КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет автоматизації і інформаційних технологій

Кафедра управління проєктами

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО АТЕСТАЦІЙНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

на тему:

«Розробка моделі штучного інтелекту для генерації реалістичних зображень з ескізів»

Орел Роман Васильович

Київ 2023 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет: Автоматизації і інформаційних технологій
Кафедра: Управління проєктами
Освітній рівень: Магістр за освітньо-професійною програмою
Галузь знань: 12 Інформаційні технології
Спеціальність: 126 “Інформаційні системи та технології”
Освітня програма: Штучний інтелект. Когнітивні технології

ЗАТВЕРДЖУЮ
Завідувач кафедри
Сергій БУШУЄВ
„___”_____20___ року

ЗАВДАННЯ

ДО ВИКОНАННЯ АТЕСТАЦІЙНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

Орел Роман Васильович
(прізвище, ім’я та по батькові студента)

1. Тема роботи:
Розробка моделі штучного інтелекту для генерації реалістичних зображень з ескізів

затверджена наказом ректора КНУБА № ____ від «___»_____20__ року

2. Керівник роботи:
д.т.н., проф. Терентьєв О.О.
(прізвище, ім’я та по батькові, науковий ступінь, вчене звання)

3. Строк подання студентом роботи до захисту:

4. Зміст пояснювальної записки (перелік питань, які слід розробити):

5. Графічний матеріал за розділами:
Рисунки результатів моделей штучного інтелекту, графічні зображення логіки математичних розрахунків у шарах нейронної мережі, графи опису архітектур нейронних мереж

6. Календарний план виконання роботи:

Види робіт та їх зміст	Дата виконання
Збір матеріалів обраного напряму роботи	8.10.23
Опрацювання та аналіз матеріалів роботи	15.10.23
Вступ	20.10.23
Розділ 1.	31.10.23
Розділ 2.	11.11.23
Розділ 3.	22.11.23
Висновки	28.11.23
Остаточне оформлення роботи	30.11.23
Перевірка роботи на плагіат	01.12.23
Попередній захист роботи на кафедрі	06.12.23
Направлення роботи на рецензування	08.12.23

7. Консультанти розділів атестаційної випускної роботи

Розділ	Прізвище, ініціали та посада консультанта	Перевірив	
		дата	підпис
Розділ 1.			
Розділ 2.			
Розділ 3.			

8. Дата видачі завдання _____

Зав. Кафедри _____ Сергій БУШУЄВ
(підпис) (прізвище та ініціали)

Керівник _____ _____
(підпис) (прізвище та ініціали)

Студент _____ _____
(підпис) (прізвище та ініціали)

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет автоматизації і інформаційних технологій
Кафедра Управління проєктами

ЗАТВЕРДЖУЮ
Завідувач кафедри
Бушуєв С. Д.
„___” _____20___ року

ПОЯСНЮВАЛЬНА ЗАПИСКА

ДО АТЕСТАЦІЙНОЇ РОБОТИ

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

Розробка моделі штучного інтелекту для генерації реалістичних зображень з ескізів

(назва)

Виконав студент групи: ІСТ-ШІКТ-11м

(прізвище, ім’я та по батькові повністю)

Спеціальність: 126 Інформаційні системи та технології

Освітня програма: "Штучний інтелект. Когнітивні технології

Керівник: Терентьєв О.О.
(прізвище, ініціали,)
д.т.н., проф.
науковий ступінь, вчене звання

Рецензент: _____
(прізвище, ініціали,)
_____
науковий ступінь, вчене звання

Київ 2023 р.

РЕЗЮМЕ (summary) *до атестаційної випускної роботи студента:*			
ЗВО	Київський національний університет будівництва і архітектури		
Тема	Розробка моделі штучного інтелекту для генерації реалістичних зображень з ескізів Development of AI model for generating realistic images from sketches		
Освітній ступінь	Магістр за освітньо-професійною програмою навчання		
Факультет	Автоматизації і інформаційних технологій		
Кафедра	Управління проєктами		
Спеціальність	126 “Інформаційні системи та технології”		
Освітня програма	Штучний інтелект. Когнітивні технології		
Керівник	Терентьєв Олександр Олександрович		
Обсяг роботи:	*пояснювальна записка, стор.*	*розділів*	*слайдів презентації*
	81	*3*	*15*
Розділ 1. Аналіз класичних та сучасних методів генерації зображень та вибір найбільш доцільного	Проведено аналіз класичних та сучасних методів генерації зображень, порівняно результати роботи алгоритмів комп’ютерного зору, обробки зображень та їх комбінацій з результатами роботи нейронних мереж. Оцінено важкість виконання поставленого в цій роботі завдання при використанні описаних підходів. Порівняно результати різних типів та архітектур нейронних мереж для генерації зображень. Обрано тип штучних нейронних мереж для подальшого аналізу і розробки.		
Розділ 2. Дослідження генеративних змагальних мереж, ступеня досяжної реалістичності результату	Проаналізовано можливості генеративних змагальних мереж у генерації по умові, порівняно результати створених зображень і їх відповідності поставленим умовам, Досліджено архітектурні нюанси реалізації та необхідні умови та структурні компоненти, що впливають на якість вихідного результату. Визначено найбільш доцільні технології та підходи до розробки нейромережі, здатної створювати фотореалістичні зображення з ескізів		
Розділ 3. Розробка моделі генерації зображень з ескізів та програми генерації зображень для дизайнерів на її основі	Описано необхідні критерії до набору даних, проведено попередню їх обробку для використанні у тренуванні моделі штучного інтелекту. Визначено структуру і кількість шарів, розроблено моделі генератора, дискримінатора та кодувальника; обрано стратегію переобробки набору тренувальних даних для використання моделлю. Визначено функції втрат та проведено тренування. Проведено оцінку моделі генератора, проведено його конвертацію та оптимізацію для використання у програмі для малювання ескізів. Розроблено програму малювання ескізів, що використовує генератор та дає можливість відобразити результат малювання.		

Висновки по роботі:	В цій роботі було проаналізовано найкращі підходи до генерації зображень з ескізів, описано їх позитивні та негативні сторони. Після проведення аналізу було виявлено, що генеративні змагальні мережі є не тільки найсучаснішим підходом, але й вони здатні створювати найреалістичніші результати на даний час. Розглядаючи різні архітектурні компоненти було розроблено генератор, що здатен відтворювати деталі у генерованих з ескізів зображень. Тринуючись разом з дискримінатором, ці дві нейромережі стараються випередити один одного, в такому режимі генератор вчиться все краще створювати вихідні зображення. Після тренування, генератор було підготовлено до застосування, і створено програму для малювання, що його застосовує.
Ключові слова: нейронні мережі, штучний інтелект, генеративно змагальні мережі, генерація зображень. **Keywords:**.neural networks, artificial intelligence, generative adversarial networks, image generation	

Укладач: _____

Керівник: _____

« _____ » _____ 202_____ р.

Зміст

ВСТУП

Актуальність теми.

В наш час компаніям дуже важливо встигати і пристосовуватись до сучасних тенденцій ринку та адаптувати свої продукти до нових вимог споживачів. Компанії, які можуть це зробити, отримують більше популярності та успіху. Ті, що не взмозі встигнути та пристосовуватись, згодом стають все менш і менш конкурентоздатними, та втрачають свою популярність.

Наразі, технології штучного інтелекту не в змозі замінити людину в багатьох сферах, проте вони здатні допомогти людям кратно підвищити свою продуктивність, а також зменшити виробничий цикл продукту. Саме вони показують найкращі результати в сфері генерації зображень, і майже повністю витіснили класичні підходи.

Технології генерації зображень можуть не тільки покращити продуктивність, але й забезпечити нові можливості в багатьох галузях індустрії та технологій. Вони допомагають людям прискорювати і автоматизовувати процеси, які раніше вимагали багато часу і ресурсів, а також генерувати абсолютно нові стилі. Основними прикладами використання є створення складних візуальних ефектів, візуалізація медичних даних, генерація рекламних банерів на основі тесту, реставрація та покращення зображень, поліпшення графіки у відеоіграх.

В даній роботу буде розглянуто і розроблено одну з таких технологій - створення реалістичних зображень на основі ескізів.

Визначення проблеми

Генеративні змагальні мережі (Generative adversarial networks, GANs) є інноваційним підходом у сфері штучного інтелекту та комп'ютерного зору. Особливо цікавими ці мережі є через їх здатність генерувати реалістичні зображення, що виглядають, ніби вони створені людиною, чи зняті на камеру.

Одним з найяскравіших прикладів є вебсайт this-person-does-not-exist.com, де можна генерувати реалістичні обличчя людей. Найцікавіше те, що зображення вигляжають справжніми, хоча таких людей не існує.

Рис. 1. Приклад згенерованого обличчя, this-person-does-not-exist.com

GANs застосовуються для вирішення складних завдань комп'ютерного бачення та як допомога для людей, що займаються творчими спеціальностями, особливо дизайнерів. Серед них основиними є:

1. генерація реалістичних зображень, які виглядають дуже схожими на ті, які може створити людина. Це важливо в багатьох сферах, включаючи комп'ютерні ігри, реклама, візуальні ефекти в кіноіндустрії та створення зображень для навчання машинного зору.

2. вдосконалення якості існуючих зображень, що може бути корисно для перевидання старих фільмів у покращеній якості, відновлення зіпсованих або пошкоджених зображень, а також для підвищення роздільної здатності зображень у медичних дослідженнях.

3. генерація унікальних та креативних зображень, які можуть бути використані у мистецтві, дизайні та інших творчих галузях. Вони можуть створити нові стилі, що важливо для художників та дизайнерів, які шукають незвичайні й унікальні ідеї.

4. генерація даних для навчання моделей штучного інтелекту. При створенні будь-якої моделі штучного інтелекту основним питанням є наявність даних та достатня їх кількість. Більша кількість екзмплярів може покращити якість моделі.

У випадках, коли їх недостатньо, GANs можуть бути використані для створення додаткових даних для навчання.

Великою перевагою GAN є можливість (не у всіх генеративних змагальних мережах) навчання без нагляду. Одна частина мережі - це генератор. Він намагається створити максимально реалістичні зображення. Інша частина мережі - дискримінатор, який намагається розрізнити між реальними та згенерованими зображеннями. Цей процес навчання без нагляду дозволяє GANs вивчати складні закономірності вхідних даних без потреби в парах даних типу вхід-вихід. Тобто, для навчання потрібна лише одна фотографія, яку пробує відтворити генератор.

Мета даної роботи полягає в дослідженні та розробці алгоритмів і методів для генерації зображень з ескізів; приділити основну увагу тим, які здатні генерувати зображення з масок зображень. Маски зображень відповідають за частини зображення, що повинні бути змодельовані або змінені відповідним чином. Вони вказуються які частини зображення користувач хоче змінити. Наприклад, для додавання дерева до ландшафту, ми повинні використати маску для позначення місця, де його необхідно згенерувати.

Об'єктом даної роботи є процес генерації зображень з ескізів, *предметом* - сучасні підходи в галузі генерації реалістичних зображень.

Основна *ціль* даної роботи – розробити власну модель штучного інтелекту (спираючись на кращі практики у галузі), що може створювати високоякісні реалістичні та деталізовані зображення; а також програми, в якій користувач зможе її використовувати. Програма буде корисною для сфер мистецтва, дизайну та візуальних ефектів. Основною можливістю є здатність генерувати реалістичні ландшафти на основі їх ескізів. Це буде допомогою для художників і дизайнерів при створенні візуальних ефектів у фільмах, відеоіграх, віртуальній реальності та в інших галузях. Після намалювання ескізу, є можливість змінити стиль згенерованого зображення.

Для досягнення поставленої у даній роботі цілі, основні *завдання* включають наступні пункти:

- аналіз існуючих методів та алгоритмів для генерації зображень з ескізів, що забезпечують високу якість вихідних зображень;
- аналіз та вивчення технічних аспектів генерації зображень на основі масок, включаючи вибір відповідних архітектур генеративних змагальних мереж, методів навчання та вибір оптимальних параметрів;
- проведення власних, та аналіз вже проведених експериментів для оцінки якості згенерованих зображень; порівняння отриманих результатів із стандартними методами генерації зображень для визначення переваг та недоліків кожного з підходів;
- розробка та вдосконалення моделі для генерації зображень;
- розробка програми, в якій користувачі зможуть малювати ескізи, та отримувати згенеровані зображення на їх основі. Це допоможе автоматизувати та пришвидшити роботу дизайнерів;
- розгляд можливостей використання згенерованих зображень у різних сферах.

Очікуваною фінальною ціллю є модель штучного інтелекту для генерації зображень з ескізів, та надійна програма, що здатна стати гарним помічником для дизайнерів.

РОЗДІЛ 1. АНАЛІЗ КЛАСИЧНИХ ТА СУЧАСНИХ МЕТОДІВ ГЕНЕРАЦІЇ ЗОБРАЖЕНЬ ТА ВИБІР НАЙБІЛЬШ ДОЦІЛЬНОГО

1.1. Аналіз класичних методів генерації зображень

Генерація зображень має довгу історію. Існує багато популярних класичних методів як процедурне створення картинок, метод на основі фракталів та L-систем (L-System-Based), синтез текстур. Серед них окремої уваги в рамках нашої теми варто приділити процедурному створенню та синтезу текстур, так як лише вони здатні генерувати реалістичні зображення.

Метод на основі фракталів мало підходить для наших цілей, хоча він здатен добре генерувати текстури та однотипні природні сцени. Самі по собі фрактали - це подібні один до одного частини об'єта. Якщо ми хочемо отримати реалістичне зображення, то потрібна варіативність та відмінність між об'єктами, тож фрактали можуть бути максимум маленькими частинами об’єктів, а не генерувати все зображення вцілому [1].

Рис. 1.1. Приклад згенерованих фракталів [1]

L-системи (або лінійні системи) є набором правилу росту початкової структури. Вони є популярними для моделювання рослинних структур та геометричних фракталів. Для генерації однієї картинки з ескізу потрібно буде використовувати дуже багато L-систем, прописувати для кожної з них правила, та кінцевий результат буде досить далекий від реалістичного, бо сама технологія була придумана для генерації об'єктів на зображенні, а не цілої сцени [2].

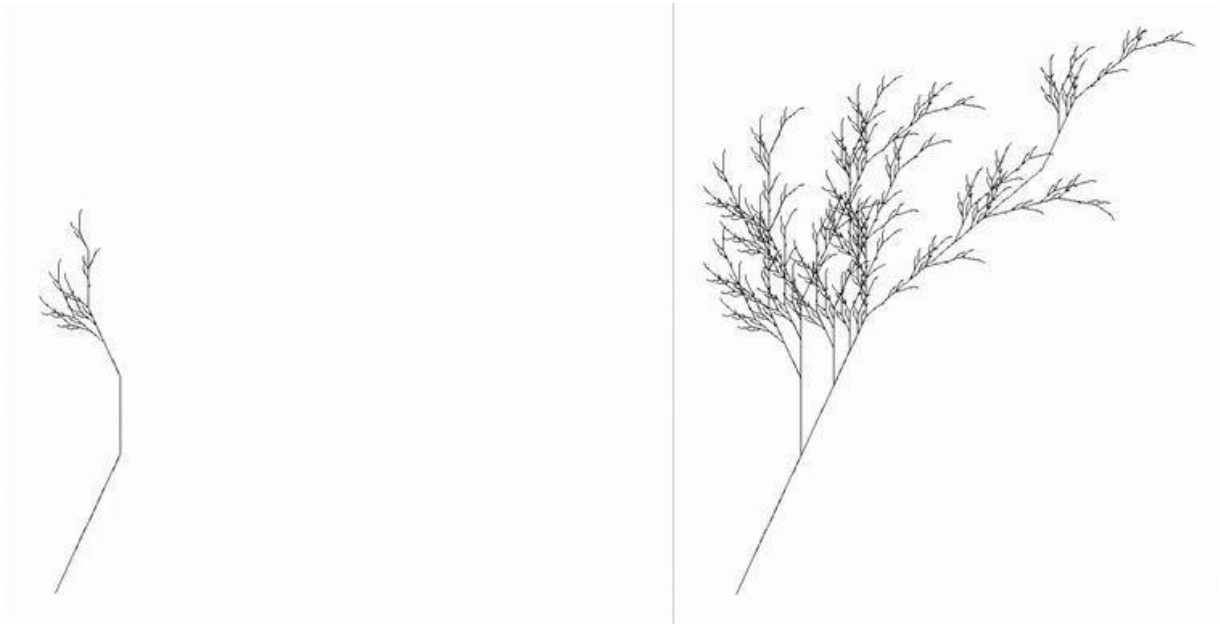


Рис. 1.2. Приклад розвитку об’єкта за допомогою L-системи

Серед класичних методів, які можна застосувати для генерації реалістичних зображень окремої уваги заслуговує процедурний.

Процедурний метод генерації зображень базується на математичних або алгоритмічних процедурах для створення зображень з нуля. Основна ідея полягає в тому, що зображення створюється за допомогою алгоритмів, які враховують параметри, правила та налаштування. Процедурні методи можуть включати в себе такі техніки, як генерація текстур, синтез ландшафтів, створення абстрактних чи природних сцен, моделювання геометричних об'єктів і багато інших аспектів графіки.

Найпростіший приклад процедурної генерації - це старі відеоігри, де персонажі генерувались шляхом простої зміни однієї з частин тіла, або кольору одягу.

Рис. 1.3. Приклад генерування персонажів процедурним стилем [3]

Для генерації фотореалістичної картинку не обов'язково полягає в накладанні зображень об'єктів на фонову картинку, хоча наявність фото об'єктів може значно полегшити процес та покращити реалістичність зображення, особливо якщо об'єкт має багато деталей.

Фотографії можуть бути використані як текстури для об'єкта, що дозволяє передавати кольори та властивості освітлення, створити реалістичні та деталізовані поверхні, відтворюючи реальні текстури, наприклад, текстури деревини, металу чи каменю. Їх можна використовувати для створення карт нормалей, які відображають деталізацію поверхні. Карти можна використовувати для відображення світла та тіней на поверхні об'єкта, надаючи йому реалістичний вигляд.

З використанням не тільки набору формул, але й достатньої бази зображень, процедурний метод створюватиме результати на основі реальних даних, що значно полегшить завдання генерації фотореалістичних зображень. А основною задачею і проблемою буде правильна обробка та використання фотографічних даних для досягнення найкращого результату.

Синтез текстур - це процес створення нових текстур, які мають схожий вигляд із вхідними прикладами текстур. Цей процес може бути застосований до генерації фотореалістичних текстур для об'єктів, поверхонь, або навіть абстрактних мистецьких творів.

Синтез текстур є досить розвиненим способом генерації зображень. Одним із більш класичних підходів є статистичний метод їх синтезу, коли програма аналізує статистику пікселів у вихідній текстурі, та створює нову текстуру зі схожою статистикою. Також популярними методами є накладання фільтрів для часткової зміни або зменшення кількості деталей та структури текстури.

Серед інших популярних підходів є шум Перліна, який спочатку створює карту шуму, а потім використовує її для визначення яскравості, інтенсивності кольорів, або висоти пікселів у текстурі (наприклад, ландшафт у Minecraft генерується з використанням шуму Перліна) [4].

Наразі синтези тестур залишається активною галуззю досліджень у сфері комп'ютерного зору та штучного інтелекту, новітні методи базуються на генеративних змагальних нейронних мережах (GANs), які будуть розглянуті в цій роботі більш детально.

1.2. Комбінація класичних методів

Жоден з зазначених вище класичних методів самостійно не здатен генерувати цілі фотографії, і для поставленої у даній роботі цілі потрібно буде комбінувати методи та застосовувати їх багаторазово. Наприклад, на першому етапі згенерувати фор, потім додати об'єкти; або комбінувати зображення фонів та об'єктів з бази данних для створення нових зображень; чи переносити стилі з одних зображень на інші.

Навіть при комбінації цих підходів велику увагу потрібно буде приділити до деталей, постобробки, авторським правам на використані зображення, а рівень реалістичності буде досить помірний без ручної обробки.

Одним з прекрасних прикладів комбінування класичних методів є Scene Completion Using Millions of Photographs, який був спільним проектом Массачусетського технологічного інституту (MIT) та Нью-Йоркського університетів, представленим на конференції SIGGRAPH 2007. Для автоматичного

заповнення пропусків у зображенні система знаходила подібні сцени у базі даних, використовувала сегментацію для виділення потрібних елементів та статистичні методи для визначення найбільш відповідних шляхів заповнення порожніх областей [5, с.1-3].

Рис. 1.4. Варіанти альтернатив заповнення пустих областей системою Scene Completion Using Millions of Photographs [5, с.4]

Продовженням цієї ідеї став проект Sketch2Photo: Internet Image Montage. Це система, що генерує зображення з ескізів, в якій користувач робить простий малюнок об'єктів та сцени за допомогою графічного інтерфейсу. На основі заданого малюнку і опису система робить запит до бази даних в інтернеті, та знаходить кандидати об'єктів для вставки. За допомогою сегментації — чиста обробка зображень у моделі кольорів HSV, без машинного навчання — зі знайдених картинок виділяються необхідні об'єкти. Перед вставкою об’єкта з бази до фінальної картинки, кандидати проходять стадії додаткової фільтрації, процес змішування стилів початкової картинки і пропонованого об’єкта, адаптацію освітлення. Далі рейтингова система підбирає найбільш підходяще до ескізу зображення, яке і додається до картинки [6, с.1,3,5].

Sketch2Photo використовує неймовірно велику базу даних, проте дає можливість досягнути досить гарні результати і високої деталізації за рахунок змішування абсолютно реальних зображень.

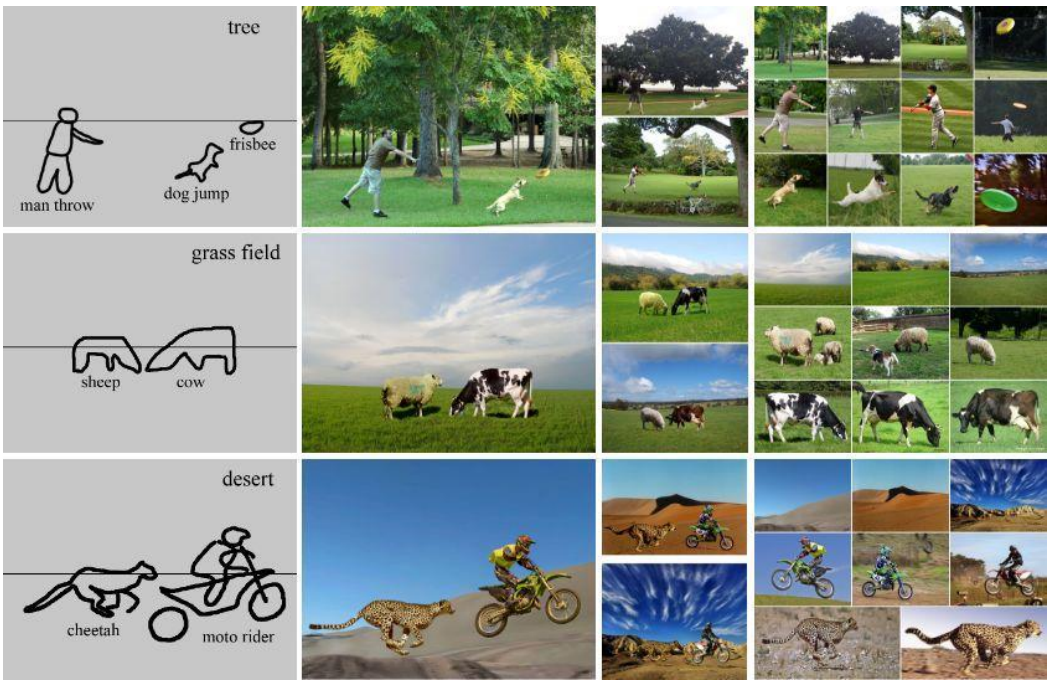


Рис. 1.5. Ескіз у Sketch2Photo, фінальне зображення та вибрані для інтеграції зображення (зліва-направо) [6, с.9]

Інші способи генерації розроблені в той час могли генерувати зображення не лише за ескізом [7, с.2], але й за текстовим описом [8, с.1], проте основна ідея була схожа: комбінація величезної бази зображень та комп'ютерного зору.

Рис. 1.6. Вибір альтернативних варіантів об'єкту в Photo Clip Art [7, с.2]

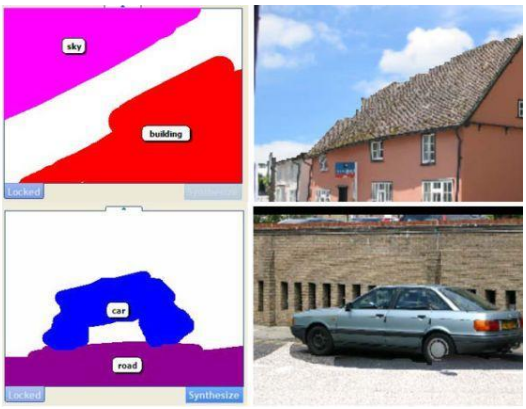


Рис. 1.7. Ескіз та опис об'єктів у Semantic Photo Synthesis [8, с.6]

Але все змінилось з появою сучасних нейронних мереж.

1.3. Генерація зображень за допомогою нейронних мереж

У наш час класичні методи генерації зображень та об’єктів все ще залишаються дуже важливими для сфери графічного дизайну, мистецтва, моделювання у відеоіграх. Проте, вони все менше і менше отримуються увагу користувачів і розробників. Це пов’язано з ростом можливостей сучасних нейронних мереж. Адже, техніки глибокого навчання — особливо генеративні змагальні мережі (GAN) — дозволяють створювати більш реалістичні та складні зображення. І їх можливості тільки зростають.

Глибокі нейронні мережі мають дуже великі переваги в порівнянні з класичними техніками. Вони здатні вивчати складні залежності в даних, що для класичних випадків потребує надважкої роботи; вивчати та переносити стилі з одного зображення на інше, виконувати постобробку та адаптацію елементів у зображенні. GANs та варіаційні автоенкодери (VAEs) здатні негерувати візуально реалістичні та деталізовані зображення високої якості. Дуже часто нейронні мережі для генерації не потребують передобробки даних, що значно спрощує і прискорює розробку продукту.

Звісно, класичні методи мають свої переваги, та все ще корисні, вони використовуються і будуть використовуватися. Проте в багатьох випадках нейронні мережі мають набагато більше ефективність для великої кількості робіт у комп’ютерній графіці, візуальних ефектах, рекламі, та все більше і більше розширюють коло своєї компетенції.

Наразі, нейронні мережі генерують майже ідеальні зображення, і стає все більш складно розпізнати які з них згенеровані, а які є справжніми.

Одним з гарних прикладів використання нейронних мереж для генерації зображень є варіаційні автоенкодери (VAEs). Вони є еволюцією звичаних автоенкодерів, які самі по собі не можуть генерувати нові дані. Автоенкодер має

можливість перетворювати дані в їх інше представлення, та декодувати нове представлення у першопочатковий формат. При кодуванні вхідні дані перетворюються у вектор в латентному просторі, який можна використовувати для зміни зображень. Кожна точка такого вектору відповідає за певний унікальний стан даних. Для модифікації або генерації нових зображень потрібно додати до вектору вхідного зображення інші вектори характеристик. Наприклад, якщо вхідним зображенням є обличчя, то після кодування ми можемо додати вектори посмішки та окулярів, і після цього декодувати цей вектор у вихідне зображення. Результатом буде фотографія того самого обличчя з посмішкою та окулярами. Цей принцип дозволяє змінювати зображення за допомогою обробки їх репрезентації у латентному просторі.

Рис. 1.8. Приклади генерації за допомогою VAE [9, с.15]

Станом на 2023 рік GANs є типом нейронних мереж, які найбільш вдало генерують реалістичні зображення. Після свого створення у 2014 році вони були використані для генерації вигаданих зображень лиць людей. Результат було важко відрізнити від справжніх фотографій. З цього часу застосування GANs у комп'ютерному баченні та мистецтві тільки зростало і розширвалось. Існує багато типів GAN для різних задач [10, с.12].

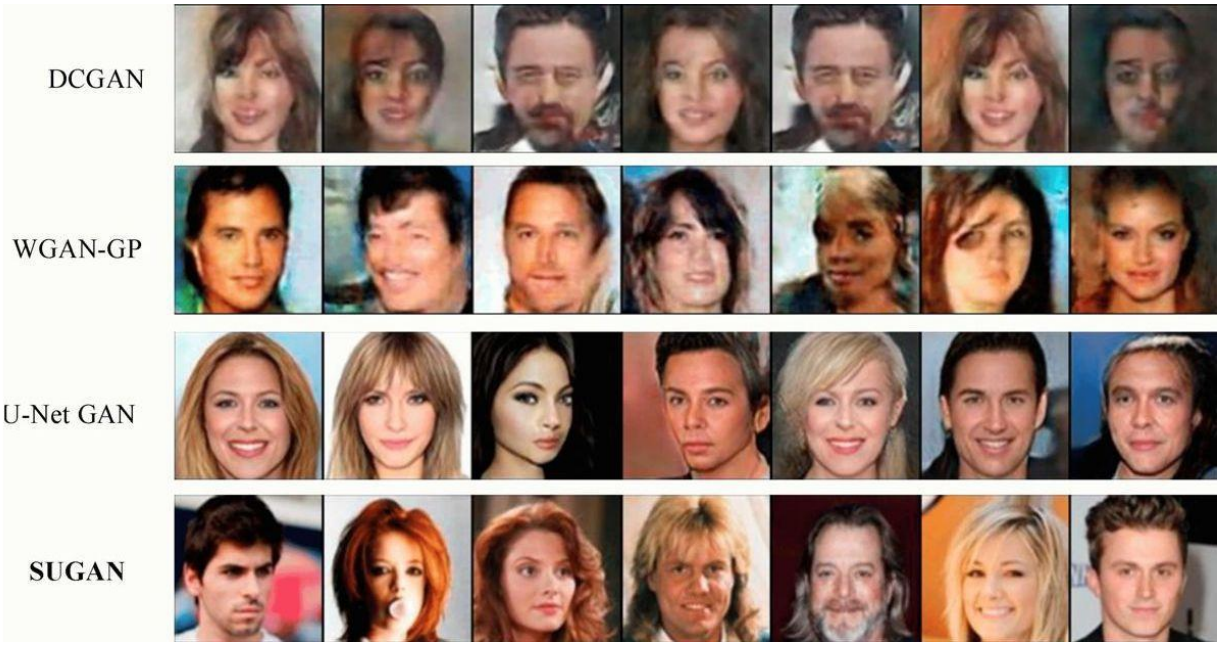


Рис. 1.9. Приклади генерації лиць людей за допомогою різних типів GAN [10, с.11]

Наразі вони здатні спокійно переносити стилі з одного зображення на інше, додавати сюрреалістичні патерни (див. рис. 1.10) [11], генерувати дорожню мапу на основі фотографій (див. рис. 12) і навпаки, фотографії з ескізу (див. рис. 1.13) [12, с.8]

Рис. 1.10. Перенос стилю та додавання патернів до вхідного зображення [11]

Рис. 1.11. Згенерована з фотографії Aierianl-мапа моделлю pix2pix [12, с.8]

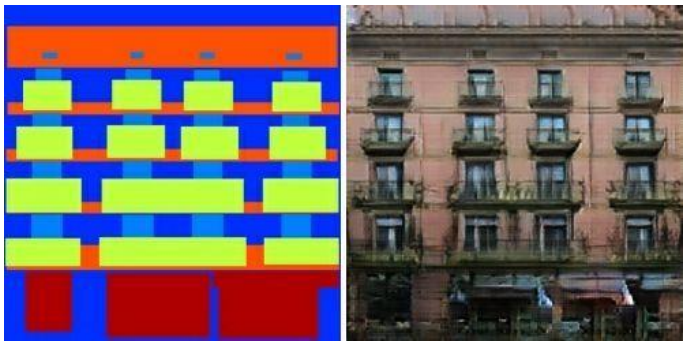


Рис. 1.12. Згенерований з ескіза фасад будинку моделлю pix2pix [12, с.5]

Найбільш вдалі реалізацій GAN, що стосується даної теми здатні не лише генерувати об’єкти з сегментаційної карти класів, але й додавати деталі до об’єктів на картинці — тіні, градієнти кольорів, вікна до будинків тощо [13, с.5-9] (див. рис. 1.12).

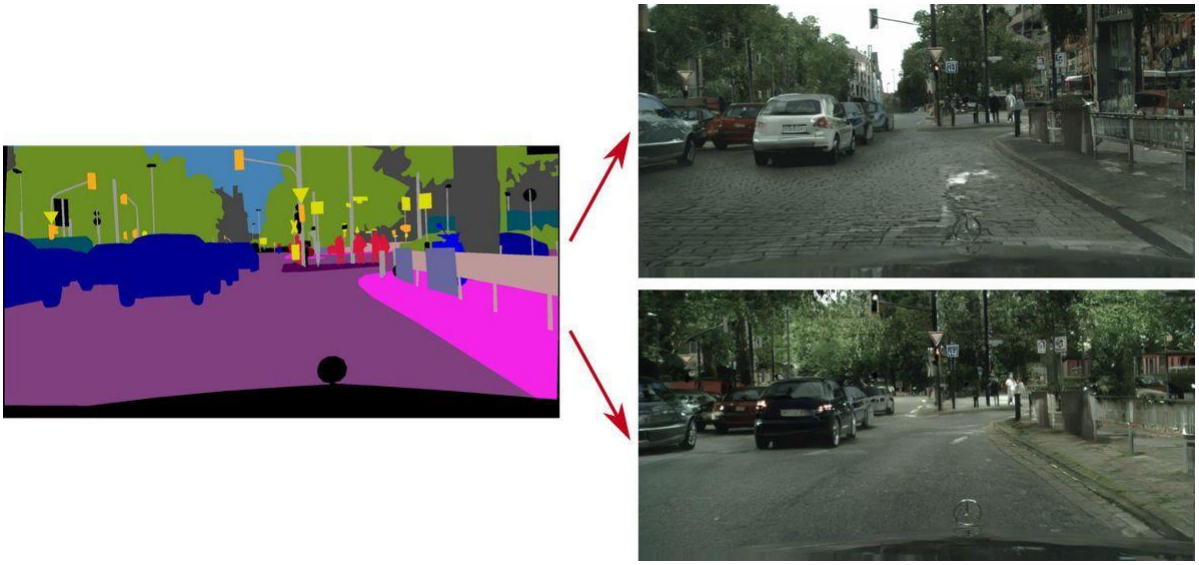


Рис. 1.13. Результат генерації зображення вулиці з лейблів за допомогою Pix2PixHD

Варто зазначити, що в наш час найбільш новітнім видом нейронних мереж є дифузійні (Diffusion) моделі, які “розповсюджують” потрібну інформацію по мережі. Найбільш типовим прикладом їх використання є генерація зображень з текстового опису. Для отримання фінального зображення модель проводить певну кількість ітерацій, в кожній з яких зображення покращується. Починаючи від, фактично, шуму, модель генерує зображення, подібні до зображених на рис. 1.14.

Рис. 1.14. Приклад зображень, згенерованих за допомогою дифузійних моделей

Висновки до розділу 1

В цьому розділі було розглянуто класичні підходи в сфері генерації реалістичних зображень, та порівняно їх з штучними нейронними мережами. На основі аналізу виявлено, що більшість підходів гарно генерують однотипні текстури, або майже однакові об’єкти з певними відмінними рисами, а для генерації цілої композиції об’єктів необхідно комбінувати класичні підходи та засоби комп’ютерного зору. Програми, що застосовують таку комбінацію дуже складні в реалізації, адже застосовують величезні набори даних та потребують детального програмування всіх деталей виділення і вставки об’єкта до зображення, пропрацювання гармонійного узгодження об’єктів між собою та фоном, щоб вони

не виглядали як посторонні наліпки. Також додатковою проблемою є ліцензовані зображення, які не можна використовувати без узгодження з власниками.

Було виявлено, що серед усіх зазначених вище методів GAN є тим, який у найбільшій мірі здатен допомогти у виконанні теми цієї роботи. Генеративні змагальні мережі є достатньо потужними, щоб створювати реалістичні зображення. Також вони потребують бази зображень лише при навчанні. При правильному підході до створення нейронної мережі, отриманий результат буде значно перевершувати результат класичних методів, адже при достатній потужності мережі та даних для тренування, об’єкти та фон будуть гармонійно виглядати один з одним. З класичних методів для досягнення співставного результату необхідна комбінація технологій комп’ютерного зору та величезна база зображень. Також, одним з необхідних елементів для досягнення реалістичності з комбінацією класичних методів є досить потужний сервер, адже необхідно проводити велику кількість обчислень при обробці всіх зображення, що містять потенційні кандидати для вставки, вирізанні їх області та гармонічному додаванні до готового малюнку.

Іншим інструментом генерації є варіаційні автоенкодери, вони теж здатні створювати реалістичні зображення, проте вони більше підходять для маніпуляції з уже готовими зображеннями, ніж для створення абсолютно нових. В той же час більш новітні інструменти як дифузійні моделі можуть допомогти отримати досить високу деталізацію, та збалансувати об’єкти, які ніколи один з одним не зустрічались. При всіх позитивних сторонах, в дифузійних моделях досить важко контролювати положення об’єктів, що є однією з головних умов створення зображень з ескізів. Ці пробдеми потребують подальшого вивчення науковими спільнотами.

Після проведеного аналізу було обрано технологію для подальшого розгляду і впровадження. В наступних розділах увага буде приділена нейронним мережам та GAN, які здатні досягти гарного результату в рамках теми цієї роботи. Особливу увагу буде приділено тим з них, які здатні контролювати місце, розміри та форму

об’єктів при створенні, адже створені картинки повинні відповідати ескізам, а не містити набір випадкових об’єктів.

РОЗДІЛ 2. ДОСЛІДЖЕННЯ ГЕНЕРАТИВНИХ ЗМАГАЛЬНИХ МЕРЕЖ, СТУПЕНЯ ДОСЯЖНОЇ РЕАЛІСТИЧНОСТІ РЕЗУЛЬТАТУ

2.1. Вступ до нейронних мереж

За більш ніж сто років до розробки комп’ютера люди почали замислюватись над програмованими обчислювальними машинами, і задаватись питанням чи зможуть вони стати розумними [14, с.7].

На сьогоднішній момент штучний інтелект (ШІ) — це одна з дисциплін, що дуже бурхливо розвиваються. Люди хочуть мати інтелектуальні програми, які здатні автоматизовувати рутинну працю, аналізувати зображення, розуміти голосові команди, ставити медичні діагнози.

На початку зародження науки про ШІ, люди працювали над вирішенням завдань, які важкі для людей, але відносно прості для комп’ютерів. Ці завдання можна описати за допомогою комбінації простих математичних правил.

Наразі перед штучним інтелектом стоїть непроста задача — навчитись розв’язувати завдання, які є легкими для людини, але важко формалізуються. Ми їх вирішуємо автоматично на інтуїтивному рівні — розпізнавання осіб на фото, або усного мовлення.

Метою цього завдання є вміння комп’ютера вчитись на досвіді та розуміти світ як сукупність ієрархій понять, кожне з яких розбите на більш прості поняття. Завдяки отриманню цих знань з досвіду, цей підхід дозволяє виключити етап формального опису всіх необхідних правил.

“Математична статистика дає багато інструментів для досягнення основної мети машинного навчання: не лише вирішити завдання на навчальному наборі, а й забезпечити можливість узагальнення. Такі фундаментальні поняття, як оцінювання параметрів, зміщення та дисперсія, корисні для формальної характеристики узагальнення, недонавчання та перенавчання” [15, с.120].

Організовуючи знання у вигляді ієрархій, комп’ютер може вчитись складним поняттям, отримуючи їх з простих. Якщо зобразити подібну ієрархію у вигляді

графу, то він буде мати багато рівнів. Саме тому даний підхід до штучного інтелекту називається глибоким навчанням [15, с.1-4]. Самі мережі називаються нейронними через те, що першопочаткова ідея їх реалізації була взята з нейробіології. В той час, як кожен прихований шар породжує вектор значень, кожен елемент вектора можна називати нейроном. Кожен нейрон отримує значення з попередніх шарів, і на їх основі робить свої власні розрахунки. Кількість нейронів кожного шару задає ширину моделі.

Досить цікавим і незвичним є той факт, що комп'ютеру дуже просто обробляти великі бази знань та обігрувати в шахи найсильніших гросмейстерів, що потребує значних розумових зусиль від людини; проте вони тільки нещодавно навчились розпізнавати прості образи та усну мову. Для людини ж навпаки - формально поставлені завдання є одними з найважчих, проте для повсякденного життя ми використовуємо величезну кількість знань на інтуїтивному рівні. Саме через представлення цих знань на інтуїтивному рівні, висловити їх важко. Для того, щоб комп'ютер став "розумним", йому потрібно отримати такі самі знання та досвід, викладені у ієрархічній, інтуїтивній, неформальній та суб'єктивній формі. Саме над цим завданням і працюють фахівці з глибокого навчання.

Отже, глибоке навчання є окремим випадком машинного навчання, що дозволяє машині вдосконалюватись шляхом представлення світу у вигляді ієрархій складних концепцій, що розбиті на більш прості та абстрактні концепції та терміни.

На рис. 2.1. зображено розташування глибокого навчання серед підходів до ШІ. Як можна побачити, глибоке навчання є одним з підходом до машинного навчання. Проте саме машинне навчання не є єдиним підходом до розробки штучного інтелекту.

Рис. 2.1. Діаграма Венна показує глибоке навчання серед ієрарії підходів до ШІ [15, с.9]

На рис. 2.2. показано різницю між підходами класичного програмування, машинного навчання, глибокого навчання до вирішення задач. Сірим кольором вказано компоненти, які навчаються на даних [15, с.9-10].

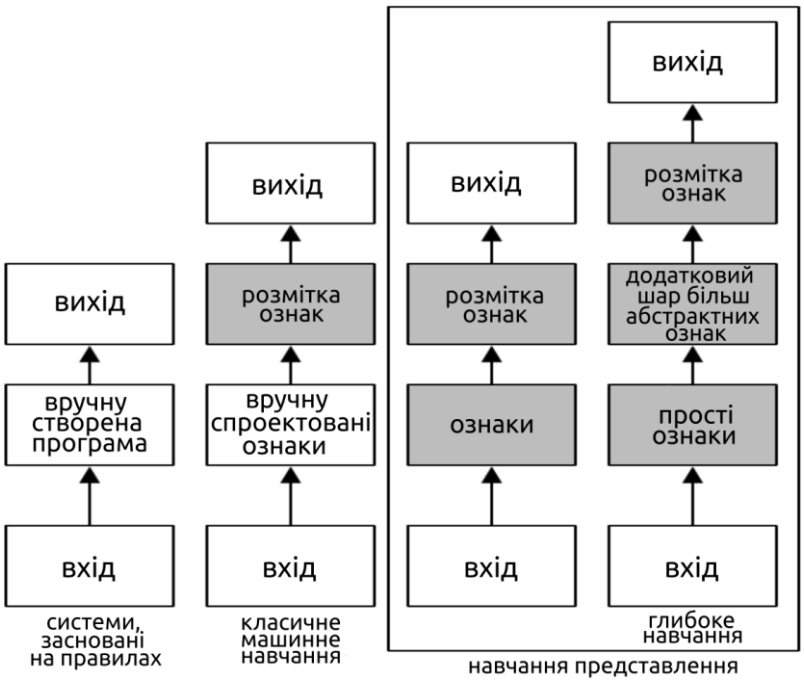


Рис. 2.2. Зв'язки різних частин системи між собою в класичному програмуванні та ШІ [15, с.10]

Історія розвитку штучного інтелекту як дисципліни триває трохи довше 80-ти років. Окремо варто виділити три основні періоди, які призвели до прогресу в області, та зробили ШІ таким, яким він є сьогодні.

Перший період розвитку пов'язаний з кібернетикою у 1940-х роках, коли були розроблені теорії біологічного навчання. На основі нейробіологічних аналогій були розроблені перші лінійні моделі, що приймали вхідні значення $x_1, \ldots, x_n$ та вираховували вихід y. Нейрон Маккаллока-Піттса був лінійною моделлю, що могла розпізнавати дві категорії виходів, перевіряючи, чи є значення $f(x, w)$ позитивним або негативним [30, с.109]. Для цього потрібно було правильно підібрати ваги. Ваги ставила людина [16]. Трохи пізніше, у 1958, вони стали основою для моделей, що дозволяли навчити один нейрон (перцептрон) — тобто знайти ваги лінійної моделі для визначення правильного результату, використовуючи вхідні дані для них [17, с. 1-5] .

Модель адаптивного лінійного елемента (ADALINE) була розширенням базової концепції перцептрону, і могла навчатися вирішувати задачі регресії по передбаченню чисел даних. У ході навчання модель знаходила ваги $w_1, \ldots, w_n$ та обчислити вихід у вигляді $f(x, w) = x_1w_1 + \ldots + x_nw_n$. Для цього був використаний окремий випадок алгоритму стохастичного градієнтного спуску [18, с.99].

Оновлення вагів відбувалось за формулою:

$$w_i = w_i + \eta * (y - \hat{y}) * x_i \qquad (2.1)$$

де w_i – вага

x_i - вхідний сигнал

η - коефіцієнт навчання

y - очікуване значення

$\hat{y}$ - вихідний сигнал, який генерується ADALINE.

Метод градієнтного спуску та його модифіковані варіанти є основними алгоритмами для навчання моделей глибокого навчання у наш час. Лінійними є моделі на основі функції *f(x, w)*, такі як перцептрон та ADALINE. Вони досі відносяться до найбільш розповсюджених моделей машинного навчання, проте методи їх навчання значно покращились та стали більш ефективними.

Другий період починається із застосуванням припадає на пов’язана з застосуванням методу зворотного поширення, що був застосований до навчання нейронної мережі з одним або двома прихованими шарами [15, с.19].

Метод зворотного поширення дозволяє мережі коригувати ваги під час тренування, враховуючи помилки передбачення. Доцільне використання інформації про те, як впливає зміна ваг на відповідь і помилку мережі дозволяє оптимізувати ваги для досягнення правильного результату.

З 2006 року починається третій період розвитку — розвиток саме глибокого навчання [15, с.13]. Розробляються методи регуляризації, такі як Dropout та L1/L2 регуляризація, що запобігають перенавчанню, та дозволяють тренувати більш глибокі нейронні мережі; виникають нові архітектурні рішення, як згорткові нейронні мережі (Convolutional Neural Networks) для обробки зображень, та рекурентні нейронні мережі (Recurrent Neural Networks) для роботи з послідовністю даних.

В цьому періоді глибоке навчання починає бути не лише теорією, а й застосовується на практиці для розпізнавання образів та голосу, машинного перекладу, ігровій індустрії. Передусім це обумовлене збільшенням обсягу даних для навчання, зростанням потужностей центральних процесорів, та графічних процесорів.

Алгоритм машинного навчання — це алгоритм, що здатен навчатись виконувати задачі, використовуючи певні даних. Сам процес навчання не є задачею. Навчання є засобом, що дозволяє виконати задачу. Наприклад, коли ми хочемо, щоб машинка рухалась, то рух — це задача. Її можна виконати написавши

програму руху вручну, або запрограмувати машинку на навчання тому, як рухатись.

В основному, задачі машинного навчання описують саме те, як потрібно обробляти вхідні дані. Даними є числове вимірювання певного об'єкта, як правило, у вигляді вектору $x \in \mathbb{R}_n$, кожен елемент якого є певною ознакою (feature). Для зображення ознаками є числові значення пікселів, для прогнозу погоди — числові значення погоди за певну кількість попередніх днів.

В залежності від досвіду, на якому буде навчатись алгоритм, алгоритми машинного навчання поділяються на два великих класи: з вчителем, без вчителя. В якості досвіду зачасту є набір даних, що складається з великої кількості прикладів, замірів показників, або точок.

Для відомого набору даних Iris ознаками є довжина та ширина чашелистика, довжина та ширина пелюстки для 150 рослин, розділених на три класи.

Алгоритми, що навчаються з вчителем спостерігають вхідний вектор ознак x та асоційованого з ним вектора y, і пробують передбачити y по x у вигляді оцінки $p(y \mid x)$.

Алгоритми, що навчаються без вчителя навчаються на досвіді у вигляді набору даних з ознак, а алгоритм повинен сам знайти закономірності та структурні залежності в даних. Так як вчитель відсутній, ніяких підказок теж немає, і алгоритм сам повинен знайти залежності.

Машинне навчання — та глибоке навчання, як його складова частина — дозволяє нам вирішувати різноманітні задачі. Найбільш розповсюдженими є наступні:

• Класифікація. В завданнях цього типу необхідно знайти відповідь, до якої з k категорій належать вхідні дані, подані у вигляді вектору x.

$$y = f(x) \tag{2.2}$$

де $f \in \mathbb{R},\ 1 \le f \le k$

Іншим варіантом класифікації є знаходження розподілу вірогідностей по класам. Прикладом може слугувати розпізнавання об'єктів, коли до вхідного

зображення вираховуються вірогідності для кожного класа, що ідентифікують їх наявність на зображенні. В даний момент найкраще працюють методи глибокого навчання [19, с.7]. Фейсбук використовує класифікацію для знаходження і автоматичного позначення друзів на фото [20].

- Регресія. В завданнях цього типу програма аналізує вхідні дані, та передбачає нове числове значення. Так само, як і для класифікації, рішенням є функція $f(x)$, $f \in \mathbb{R}$, проте відрізняється від класифікації форматом виходу. Прикладами є прогнозування погоди, або майбутнього коливання валютних курсів.
- Пошук аномалій. В подібних задачах програма аналізує дані та знаходить в них нетипові елементи. Прикладом є пошук махінацій в закупках компаній, або зловмисників по нетиповим транзакціям з банківської картки [15, с.99-101].
- Синтез та генерація. Коли алгоритм машинного навчання генерує нові приклади, схожі на вхідні дані. Це значно спрощує роботу художників, бо замість ручної обробки пікселів, програма генерує великі об'єми даних, як ландшафти, рослинність.
- Транскрипція. Програма аналізує вхідні неструктуровані дані та перетворює їх у текстову форму. Google Translate використовує глибоке навчання для сканування тексту, Google Street View обробляє таблички з адресами будинків [15, с.99]. Також технологія застосовується для розпізнавання мовлення.
- Машинний переклад. У задачах цього типу програма перетворює послідовність слів однієї мови на послідовність слів іншої мови. Наприклад, з української на німецьку.
- Структурний вихід. Це вид задач, в яких на виході породжується вектор, елементи якого мають зв'язки між собою. Для обробки тесту це може бути дерево зв'язків, що описує граматичну структуру речення. Для обробки зображень прикладом є сегментація, коли програма розподіляє пікселі по категоріям. Приклад — анотація доріг по фотографіям [21, с.2].
- підстановка відсутніх значень. Коли є дані, в яких відсутні певні значення, програма аналізує наявні дані, та заповнює пробіли.

• зменшення шуму. В цих задачах на вході є вектор $x' \in \mathbb{R}$ з даними, які були пошкоджені. Алгоритм повертає вектор $x \in \mathbb{R}$ з відновленими після пошкодження даними. Прикладом є відновлення фотографій.

Вирішення наведених вище задач дозволяє машинному навчанню (МН) застосовуватись у широкому колі програм: аналізі зображень та звуків, виявленні дефектів та ознаків захворювання, прогнозувати результати лікування, розпізнавання об'єктів, прогнозування руху об'єктів для автономного водіння, прогнозування цін та попиту, виявлення махінацій у фінансових транзакціях, оптимізації логістики, перекладу, прогнозування погоди, фільтруванні спаму. Останнім часом існує тенденція до заміни звичайних чат-ботів на аналоги, розроблені за допомогою МН. В області генерації постійно підвищуються в якості не лише мережі генерації звуку та картинок, але й реалістичних відео (діп-фейки).

Нейронні мережі здатні виконувати різноманітні задачі, для багатьох із яких вони є найкращим інструментом вирішення, доступним на даний момент. Наразі, майже в кожній області машинного навчання результати найвищої якості показують не просто нейронні мережі, а саме глибокі нейронні мережі, що складаються з великої кількості прихованих шарів. Як зазначалось раніше, генерація зображень є однією з таких задач.

Передумовами цього є насамперед поява та розвиток глибоких нейронних мереж, які почали бурно розвиватись відносно нещодавно, в основному, завдяки росту обчислювальної потужності відеокарт, та великій кількості інформації для навчання в загальному доступі. Вони відкрили нові можливості розв’язання складних завдань, завдяки своїй здатності вивчати більш абстрактні та складні залежності у даних.

Одним з найефективніших методів, що дозволяє вирішувати складні завдання генерації зображень, є генеративні змагальні мережі (GANs). Вони були винайдені у 2014 році Іаном Гудфеллоу та його колегами, і були значним кроком вперед у сфері, адже вони давали змогу генерувати реалістичні зображення. GANs

мають в своєму складі два основних компонента — генератор та дискримінатор, які змагаються один з одним під час процесу навчання.

Перші генеративні змагальні моделі не були пристосованими для генерації складних зображень [22, с.1-9], проте архітектури отримали значні модифікації та удосконалення, що допомогли не тільки стабілізувати тренування та покращити якість, а й дали змогу перейти до генерації надскладних зображень (див. рис. 1). Вони були використані у ретуші фотографій, колоризації (перетворення монохромного зображення у кольорове), генерації фотореалістичних зображень, анімації, створенні діп-фейків.

Враховуючи здатність GANs створювати фотореалістичні зображення, саме цей вид нейронних мереж було обрано для досягнення мети даної роботи.

Попри те, що дослідники зазначають, що принципи роботи мозку вплинули на саму ідею та розвиток глибокого навчання, не варто думати, що глибоке навчання є спробою імітувати мозок. Звісно, ідея використання багатьох шарів теж прийшла з нейробіології, проте сучасні нейронні мережі не прагнуть моделювати мозок, а є сукупністю математичних функцій, що використовуються ідеї та підходи статистики для статистичного узагальнення результату, а також прикладну математику. Деякі вчені вважають нейробіологію важливим джерелом ідей, інші не згадують її зовсім [15, с.15].

2.2. Вибір архітектури генеративно змагальної мережі

На даному етапі свого розвитку, нейронні мережі показують значно кращі результати, ніж стандартні підходи комп'ютерного зору.

Не дивлячись на те, що у багатьох випадках в тестах можна відрізнити реальні зображення від згенерованих, серед існуючих методів та підходів для генерації зображень на основі масок найкращі результати показують генеративно змагальних мережі. Вони складаються з генератора, що генерує реалістичні

зображення, та дискримінатора, що пробує відрізняти згенеровані зображення від реальних. Обидва цих елемента “змагаються” між собою — дискримінатор вчиться все краще й краще розрізняти яке зображення є справжнім, а генератор вчиться підвищувати якість згенерованих зображень, щоб генерувати зображення, які якомога більше схожі на справжні. Така концепція була запропонована Іаном Гудфеллоу у 2014 році [22], та до сьогодні була розширена великою кількістю підходів та архітектур націлених на різні задачі.

Подібним чином інші умовні генеративні мережі можуть створити кадр відео на основі попереднього кадру, зображення місцевості на основі зображення мапи, зображення з текстового опису та багато іншого [23, с.1].

Далі буде розглянуто та порівняно архітектури генеративно змагальних мереж, які здатні допомогти виконати цілі даної роботи.

LayoutGAN дозволяє генерувати зображення на основі розташування об'єктів на сцені.

На вхід, окрім випадкового шуму, який допомагає урізноманітнити результати, подається маска об'єктів, що задає їх розташування та розміри. Мережа вивчає взаємозв'язки між об'єктами, та їх органічне розташування відносно одне одного для підвищення реалістичності.

Маскою в даному випадку є не ескіз зображення, а набір прямокутників, які вказують в якій області буде розташовано елементи.

Виходом є згенероване зображення, що включає бажані об'єкти на бажаних позиціях. У випадку, коли об'єкти перекривають один одного (один прямокутник перетинає, або знаходиться всередині іншого), результат виглядає правдоподібно [24, с.1-7].

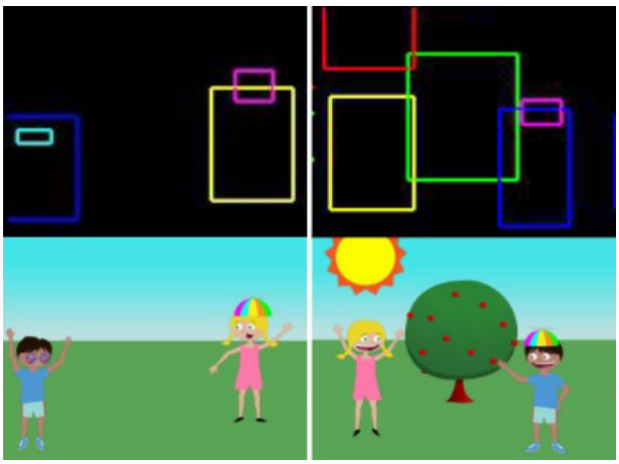


Рис. 2.3. Маска та зображення, згенероване LayoutGAN [24, с.7]

Подібна нейронна мережа може бути дуже корисною для архітектурного дизайну, або прототипування візуалізації сцен у гейм-індустрії.

Більш вдалим підходом до виконання завдання цієї роботи є *каскадна мережа уточнення* (Cascaded Refinement Network), яка спроможна генерувати зображення з високою деталізацією.

На вхід приймається маска об'єктів, яка відповідає за розташування і форму об'єктів, та вектор випадкового шуму, який урізноманітнює результат під час генерації. Згенероване зображення є досить високоякісним, та відповідає масці.

Створення зображення відбувається в три етапи. Спочатку одна генеруються грубі структурні елементи. Потім відбувається процес уточнення, коли грубе зображення проходить через кілька рівнів, які додають нові деталі. Кожен такий рівень є додатковою мережею, яка відповідає за різну ступінь деталізації. На кожному кроці якісь зображення підвищується. Фінальним кроком є проходження через мережу, яка покликана додавати високодеталізовані елементи [25, с.1,5-6]. На цьому етапі додаються тіні, змінюється яскравість елементів тощо.

Рис. 2.4. Маска та згенероване зображення за допомогою Cascaded Refinement Network [25, с.8]

Архітектура глибокої нейронної мережі, що всеціло націлена на генерацію об'єктів з масок — це MaskGAN. На вхід мережі подається двовимірна маска, яка відповідає за розташування та форму об'єктів, та вектор шуму, який забезпечує варіативність результатів [26, с.1-4].

Згенероване зображення відображає бажані об'єкти та їх деталі.

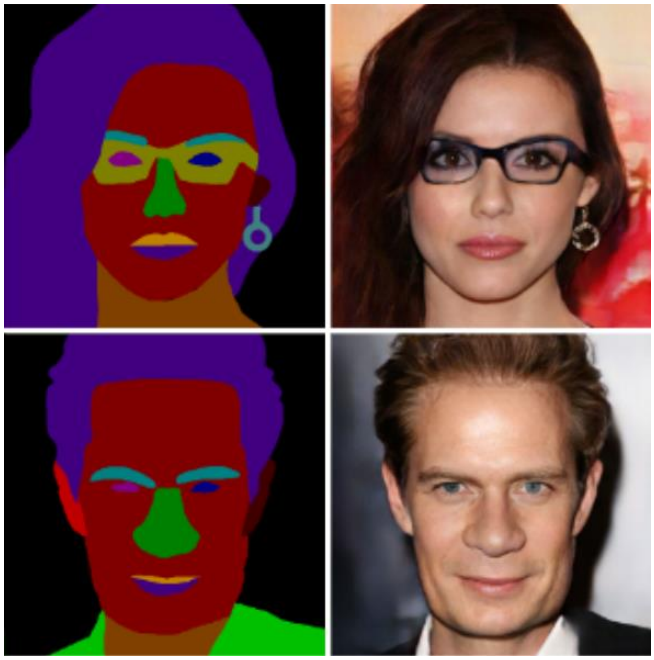


Рис. 2.5. Маска та лиця, згенеровані за допомогою MaskGAN [26, с.8]

MaskGAN може забезпечити досить добре промалювання різних об'єкти, проте певним лімітом цієї технології є низька здатність додавати деталі між різними об'єктами, порівняно з більш пізніми аналогами для подібних задач.

Мережею, яка перетворює зображення з одного домену в інший є Pix2Pix. Вона використовується в основному для колоризації чорно-білих фотографій, перетворення супутникових зображень у мапи (можна й навпаки), та генерації зображень [27, 1-13].

Входом є зображення у одному домені, наприклад, маска фасаду будинку. Виходом є вигляд вхідного зображення у іншому домені, наприклад, зображення самого фасаду будинку (див. рис. 2.6).

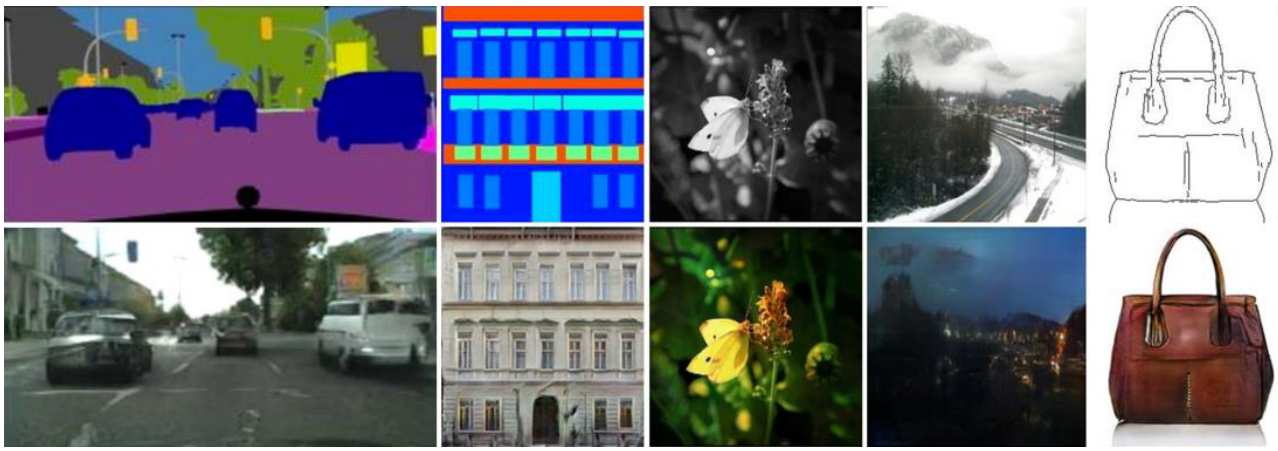


Рис. 2.6. Варіанти входів та виходів pix2pix [27, с.10-11]

Як можна бачити на рис. 2.6., перетворення зображень в новий домен включає також перетворення з контуру в зображення, та з денної фотографії в нічну.

Розширена версія архітектури Pix2Pix стала розроблена компанією NVIDIA архітектура *Pix2PixHD*. На відміну від оригінальної архітектури, яка здатна генерувати зображення розміром 256x256, вона дозволяє генерувати високороздільні зображення розміром 2048x2048.

Входи та виходи мережі є такими самими, як і у оригінальної архітектури.

Новий підхід до генерації заключається в тому, що в процесі генерації спочатку створюється низькороздільне зображення (4x4), яке поступово розширюється до бажаного розміру (8x8, ..., 64x64,...,2048x2048). На кожному етапі розширення додаються все більш дрібні деталі. Дискримінатор порівнює зображення на всіх рівнях генерації з оригінальним зображенням такого ж розміру [28, с.1-8].

Такі каскадні етапи дозволяють краще контролювати якість згенерованих зображень на різних рівнях деталізації.

Рис. 2.7. Співставлення результатів Pix2Pix та Pix2PixHD [28, с.8]

У Pix2PixHD підвищення дискретизації виконувалась за допомогою транспонованих згорткових шарів, що збільшують розміри вхідних карт для отримання більшого зображення. Подібний підхід почне втрачати популярність через те, що на зображенні вони створюють артефакти у вигляді шахової дошки (див. рис. 2.8) [29, с.9].

В пошуках шляхів вдосконалення архітектури Pix2PixHD, дослідники компанії NVIDIA створили нейромережу *GauGAN*, що спеціалізується не лише на високій роздільності зображення, але й на їх реалістичності. Це стало можливим, в основному, завдяки використанню спеціального архітектурного компоненту, названого SPADE, який аналізує вхідні дані та адаптує параметри нормалізації. Такий підхід дозволяє більш повно забезпечити реалістичність та збереження структури об'єктів [29, с.2-4].

Рис. 2.8. Порівняння зображень, згенерованих Pix2PixHD та GauGAN [29, с.14]

Вхідними даними є система масок, кожний компонент якої представляє собою зображення з чорним фоном, на якому білі області відповідають за розташування та форму об'єктів певного класу.

Виходом нейромережі є згенероване реалістичне кольорове зображення, яке відображає бажані об'єкти.

Семантична інформація — інформація, яка включає в себе деталі про об'єкти, їхні форми, розміри та інші характеристики, які роблять їх унікальними та ідентифікованими. Традиційні мережі GAN використовують шари нормалізації, які змивають частково або повністю семантичну інформацію. SPADE вирішує цю проблему. Завдяки здатності адаптувати нормалізацію до конкретних областей зображення, вона дозволяє краще зберегти семантичну та текстурну інформацію під час генерації зображень на основі масок. Тому згенеровані зображення є більш деталізованими.

Як видно з рис. 2.8, GauGAN дозволяє не лише генерувати реалістичні зображення, але й краще себе показує у запобіганні появи артефактів. Основна причина цього – використання SPADE у якості нормалізатора.

Серед зазначених вище методів для наших потреб, найбільше виділяється мережа GauGAN. На даний момент саме вона дозволяють генерувати найбільш реалістичні та якісні зображення з масок, що є корисно для генерації зображень з ескізів. Її буде вибрано як приклад для наслідування при розробці нейромережі для досягнення цілі даної магістерської роботи роботи. Тому далі ми розглянемо архітектуру і особливості GauGAN більш детально.

2.3. Аналіз та дослідження GauGAN та SPADE

Після створення Pix2PixHD компанія NVIDIA почала працювати над удосконаленням цієї моделі. Вже у 2019 році на конференції CVPR було представлено документ "Семантичний синтез зображень із просторово-адаптивною нормалізацією", який презентував новий алгоритм, названий GauGAN. Технологія GauGAN стала доповненням і покращенням pix2pixHD [29, с.3], що здатна ефективно перетворювати семантичні маски у реалістичні зображення (див. рис. 2.8). Беззаперечно, це стало одним з найбільших кроків у сфері генерації зображень.

Зазвичай для створення результату у якості вхідних даних до GAN подається вектор шуму. Тобто для генерації зображень не потрібно нічого (окрім випадкового шуму), і модель вміє сама генерувати типовий вихід у вигляді зображень лиць, звірів чи продуктів.

З іншого боку, умовні GAN зазвичай приймають певні вхідні дані, які задають початкові умови для генерації зображень. Тобто, згенеровані дані у повній мірі залежать від вхідних даних, які подаються до моделі. У GauGAN вхідними даними є карта семантичної сегментації, на основі якої нейронна мережа генерує реалістичне зображення (див. рис. 2.9).

Рис. 2.9. Генерація реалістичного зображення (знизу) на з карти сегментації (зверху) [29, с.14-15]

Звісно, як і інші генеративно змагальні мережі, GauGAN містить дискримінатор і генератор, проте додатково використовується кодувальник для синтезу інформації про стиль і кольори вхідного зображення. Одним з головних ноу-хау є представлена техніка нормалізації, що отримала назву SPADE (SPatially Adaptive (DE)normalization). У свою чергу, SPADE використовується у вигляді спеціальних блоків всередині мережі.

Входом до генератора є семантична карта сегментації та (опціонально) результат кодувальника.

Семантична карта сегментації складається з one-hot-encoded векторів для кожного класу, що об’єднані (конкатиновані) між собою (див. Рис.2.10). Тобто, при генерації 12 можливих класів, мережа откримує 12 двійкових зображень, в яких пікселі, що не належать об’єктам даного класу дорівнюють нулю, пікселі об’єкта класу дорівнюють 1.

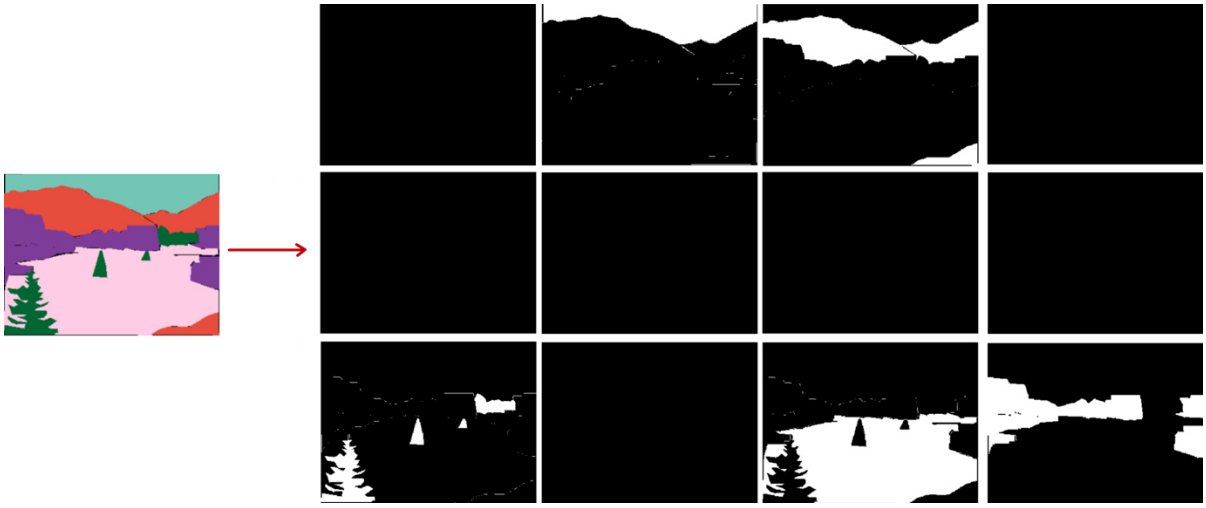


Рис. 2.10. Кодування семантичної карти сегментації з ескізу для подання генератору, розробленому у цій роботі

Звісно, що для класів, що не присутні на зображенні, вектори будуть складатись з нулів, адже мітки сегментації описують лише класи об’єктів на зображенні. При умові, коли є два об’єкта, що перекривають один одного, карта сегментації буде одною для двух і більше об’єктів.

Ситуація, коли два об’єкти накладаються, може заплутати алгоритм, зменшити якість вихідного зображення, і привести до отримання гіршого результату, ніж очікувалось.

Pix2PixHD подолав цю проблему шляхом використання карти семантичних міток, які є двійковими картами зображення зі значеннями 0 та 1. В них пікселі, чиї сусідні пікселі не належать одному сусіду мають значення 1, а всі інші — нуль.

Генератор в GauGAN є згортковим декодером (також відомий як апсемплінг-мережа — upsampling network), який складається з SPADE блоків. Він відповідає за перетворення масок (які є низькороздільними картами ознак — feature map) в

високороздільні за допомогою апсемплінг шарів, що розширюють розміри вхідних ознак. SPADE блоки використовуються для просторово-адаптивної нормалізації, що дозволяє зберігати деталі зображення. Для кожного окремого SPADE-блоку карта сегментації змінюється, щоб відповідати розмірам карти ознак**.**

Це не показано на малюнку (див. рис. 2.11), але вихід кожного згорткового шару нормалізується спектральною нормою, що обмежує постійну Лейпшиця. Це є необхідним кроком, адже занадто великий спектральний радіус ваг (максимальне за модулем число у матриці вагів) у шарах нейромережі призводить до вибухання градієнтів, а занадто маленький спектральний радіус спричиняє затухання градієнтів, що майже унеможливлює навчання. Як результат використання такого підходу, підвищується стабільність та швидкість навчання.

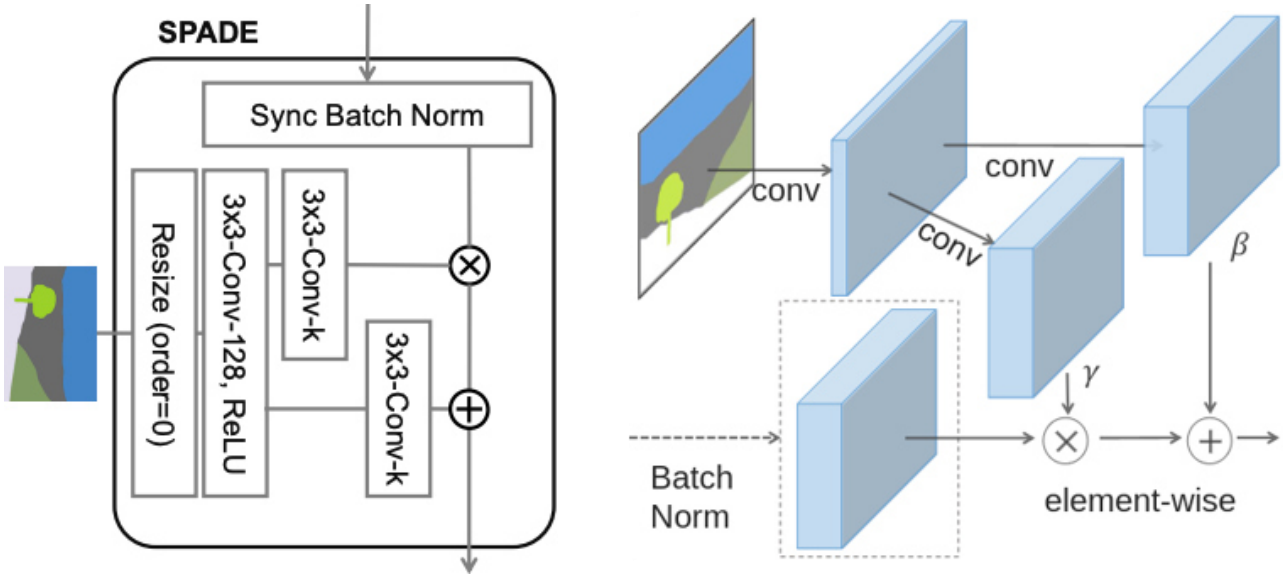


Рис. 2.11. Конструкція SPADE [29, с.3,11]

Генератор складається з серії SPADE блоків зі зворотним зв’язком (SPADE Residual Block), за кожним з яких стоїть апсемплінг (див. рис. 2.12) у вигляді, білінійної або бікубічної інтерполяції.

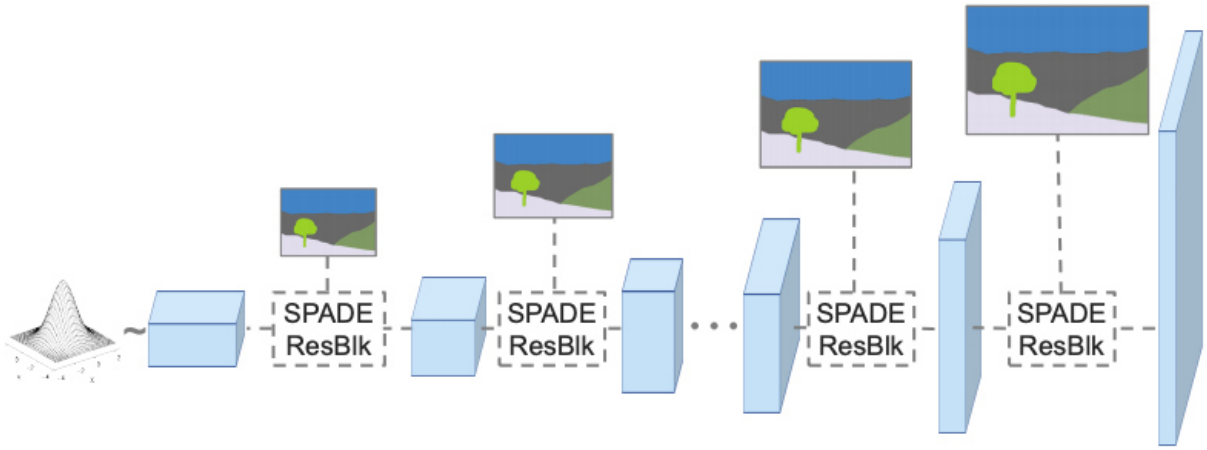


Рис. 2.12. Генератор спейд приймає сегментаційну маску як вхід до кожного шару нормалізації [29, с.4]

В GauGAN карта семантичної сегментації подається безпосередньо подається на вхід кожного SPADE блоку, на кожному рівні нейронної мережі. Підвищення дискретизації виконується зміною розміру, але вже не за рахунок інтерполяції, а методом найближчого сусіда.

Нормалізація без умови може призвести до втрати семантичної інформації. Це включає як нормалізацію міні-пакетів (mini-batch), так і нормалізацію по екземпляру (Instance Norm) [29, с.3].

Нормалізація у глибокому навчанні включає певні кроки, які допомагають досягти стабілізувати та пришвидшити процес навчання:

1. обчислення статистичних показників зображення, як середнє значення та стандартне відхилення

2. нормалізація даних шляхом віднімання середнього значення

3. ділення отриманого числа на стандартне відхилення

4. масштабування входу $y=\gamma * x + \beta$ за допомогою тренованих параметрів γ *(гамма)* та β *(бета)*.

Нормалізація міні-пакету та нормалізація екземпляру відрізняються один від одного на обчисленням статистичних показників на кроці 1. Для міні-пакету обчислюються показники для групи зображень, а в екземплярі - для одного зображення.

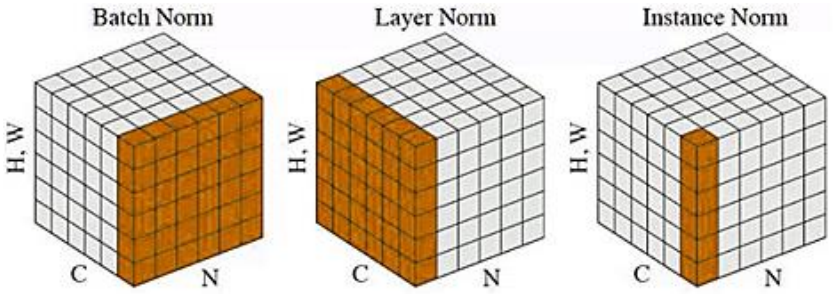


Рис. 2.13. Різні види нормалізації

Серед загальновикористованих нормалізацій є чотири. Кожна з них має свої певні особливості, і відрізняються лише на перших етапах процесу нормалізації.

Нормалізація батчу (Batch Normalization): середнє та дисперсія обчислюються для кожного каналу у батчі (наприклад, для батчу з 16 RGB-картинок буде вирахувано три середніх і три дисперсії: для R-каналу, G-каналу та B-каналу).

Нормалізація шару (Layer Normalization): середнє та дисперсія обчислюються для кожного нейрона по всіх зразках у батчі.

Нормалізація екземпляру (Instance Normalization): середнє та дисперсія обчислюються для кожного зразка (екземпляра) по кожному каналу.

Подальші кроки для всіх нормалізацій однакові:

- центрування та масштабування активацій на основі цих статистик;
- (опціонально) застосування афінного перетворення за допомогою параметрів, що тренуються.

В SPADE теж обраховуються статистичні дані у міні-батчі, проте вивчаються не набір параметрів на канал, а різні набори параметрів для кожного пікселю на карті ознак. Кількість параметрів норми міні-пакету збільшується, щоб відповідати кількості пікселів.

Модуль, з яких складається генератор є невеликим блок із залишковим зв'язком (Residual block), що створює дві карти функцій, кожен піксель яких є

параметром для масштабування значень відповідних пікселів на карті ознак: для пікселів β′, та пікселів γ′. Подібні блоки залишають зв'язок між своїм входом та виходом за рахунок додавання. Це ефективно протидіє проблемі знакання градієнтів, та дозволяє будувати більш глибокі мережі.

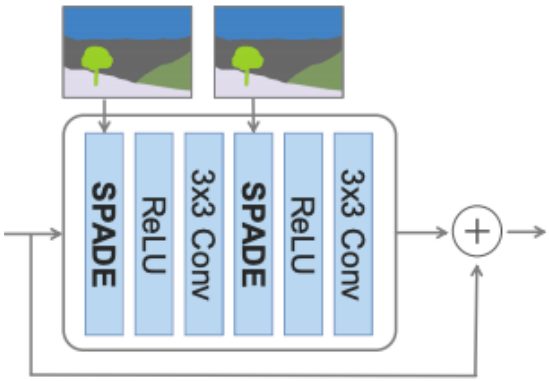


Рис. 2.14. Структура SPADE Residual block [29, с.4]

На відміну від звичайного BatchNorm, який обчислює статистичні дані міні-пакету (mini-batch), обчислення статистичних даних у SPADE є синхронізованими між декількома графічними процесорами в одній системі. Основна причина цього — розмір мережі та градієнта, що розраховується, адже для пакета з 4 зображень це може складати 32 гігабайти. Масштабування на основі розрахованих статистичних даних відбувається пізніше — у звичайному не синхронізованому режимі. Необхідно зазначити, що синхронізована між багатьма відеокартами нормалізація корисна, коли пакет складається з декількох зображень. У разі обчислення статистичних даних для великих (наприклад, з 64 зображень) пакетів призводить до нестабільності процесу навчання [29, с.8].

Після розрахунків, обчислена у модулі SPADE карта ознак, поелементно множиться та додається до нормалізованої вхідної карти ознак.

Основні рівняння, що описують SPADE можна побачити нижче.

$$\hat{x}_i = (x_i - \mu_i) / \sigma_i \qquad (2.3)$$

де $\hat{x}_i$ — значення нейрону після батч-нормалізації

x_i — вхідні значення нормалізаційного шару

μ_i — середнє значення

σ_i — дисперсія

$$\hat{y}_i = \gamma_i * \hat{x}_i + \beta_i \quad (2.4)$$

де $\hat{y}$ — вихідне значення шару нормалізації SPADE

γ_i, β_i— параметри, отриманий після згортки

$\hat{x}_i$ - значення нейрону після батч-нормалізації

Гарні результати SPADE насамперед обумовлені тим, що вхідні дані є картою семантичної сегментації, де кожен канал зображення має дані лише про один єдиний клас. GAN бере області однорідних значень (значення 1.0 у масці) та створює пікселі, які б виглядали як справжній об’єкт. Наявність іншого пакету параметрів міні-пакету допомагає урізноманітнити результат краще, ніж один набір параметрів для одного зображення. Також SPADE в більш повній мірі ніж інші GANs зберігає різницю між пікселями зображення, згенерованого на основі семантичної карти. Одна з найбільш вдалих GAN для генерації зображень з масок на час створення GauGAN була Pix2PixHD. Не маючи різноманіття міток — лише одна мітка на зображенні — GauGAN здатна створювати зображення з різними значеннями, тоді як Pix2PixHD генерує однорідний сірий фон. Це обумовлено тим, що застосування нормалізації до однорідних значень перетворює вихідні дані в нулі (так як вихідні дані GAN лежать в межах [-1, 1], після зміни масштабу до меж [0,1] близькі до нуля значення стають близькими до 0.5 для трьох каналів, що відповідає сірому фону), і семантична інформація втрачається.

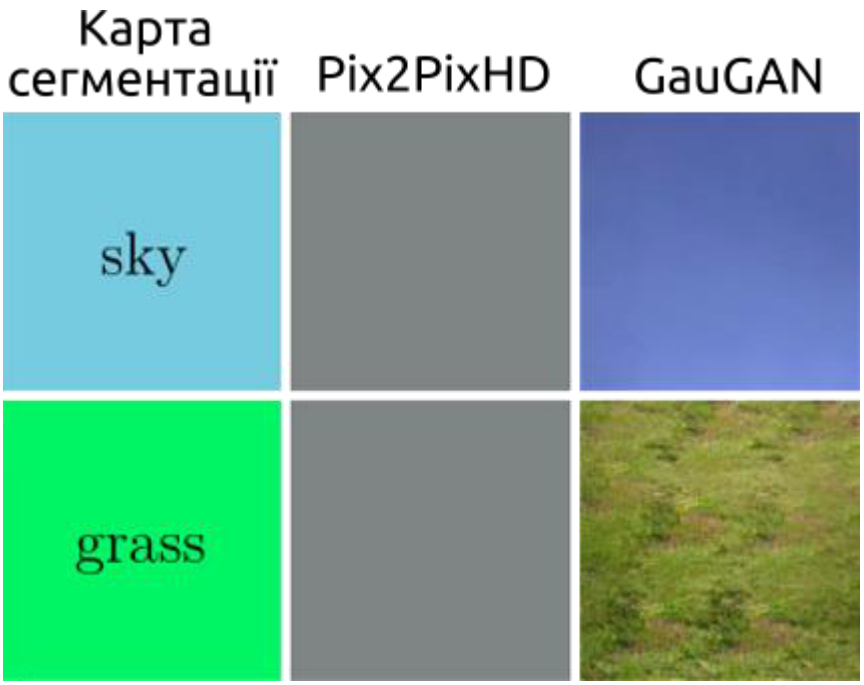


Рис. 2.15. Порівняння результатів генерації Pix2PixHD та GauGAN для карти сегментації, що складається лише з одного класу [29, с.3]

Потрібно зазначити, що SPADE та Instance Norm відрізняються не на етапі нормалізації, а на етапі масштабування. Також, в GauGAN параметри нормалізації тренуються під час навчання, тоді як Instance Norm — ні. Так само, як і у нормалізації міні-пакету, статистичні дані розраховуються для декількох зображень одразу, що призводить до більш стабільних розрахунків цих даних.

В загальному випадку, дискримінатор є мережею, в якій останні шари є повністю зв’язаними (Dense), і результатом є оцінка в межах від 0 до 1, яка показує наскільки згенероване зображення є реалістичним. В GauGAN дискримінатор (див. рис. 2.16) є повністю згортковою мережею, яка виводить карту ознак, яка після усереднення значень оцінює реалістичність зображення.

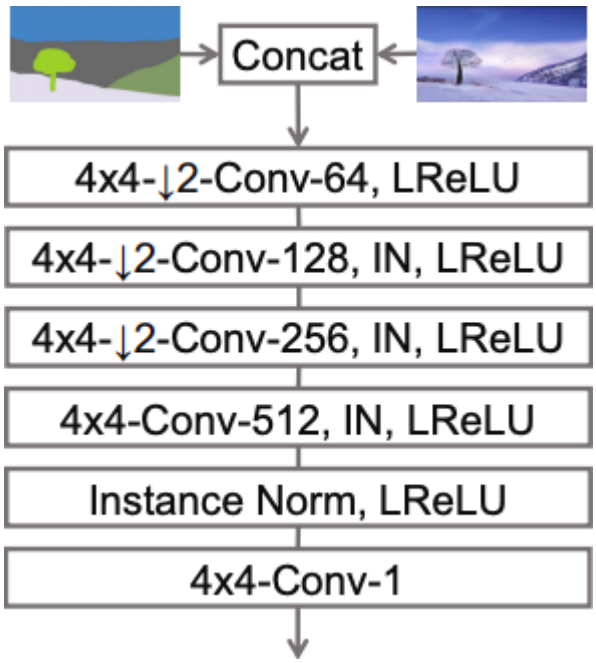


Рис. 2.16. Архітектура дискримінатора в GauGAN [29, с.12]

Зазвичай GANs приймають вектор випадкового шуму для генерації зображень. В свою чергу, GauGAN приймає лише карту сегментації, що призводить до однакових результатів генерації. В такому випадку мережа вчиться генерувати виключно зображення, що відповідають даним для тренування. Можна сказати, що мережа реконструює дані замість того, щоб створювати різноманітні результати поза межами даних для тренування, як це роблять інші GANs. Але ще одним елементом є кодувальник.

Кодувальник приймає зображення, і кодує його у два вектори, які потім використовуються як середнє значення та стандартне відхилення (див. рис. 2.17), які будуть об’єднані з вхідною сегментаційною картою і подані на вхід генератора.

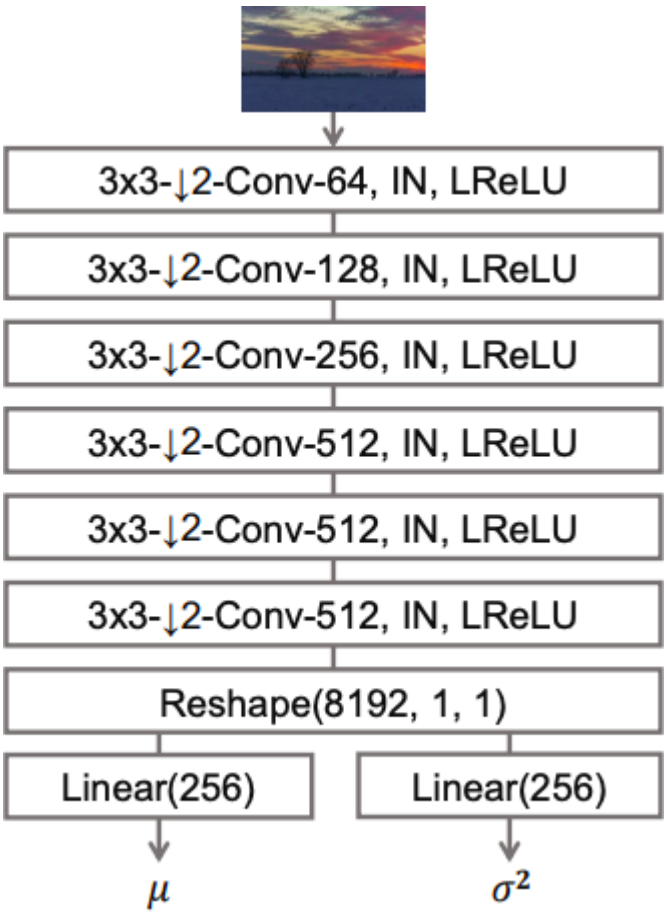


Рис. 2.17. Кодувальник GauGAN [29, с.12]

Різні вектри сприяють отриманню різних результатів генератора. Під час генерації вони відповідають за стиль вихідного зображення.

Рис. 2.18. Генерація зображення з одного ескізу з різними стилями [29, с.19]

Кожен вектор буде сприяти створенню зображення з різними характеристиками, як кольорова гамма, яскравість тощо.

Одним з важливих і складних елементів, які заставляють GauGAN працювати є функція втрат. Вона є комплексною, та складається з декількох елементів.

Під час процесу генерації, генератор створює вихідне зобюраження у кількох масштабах. Кожне з них використовується для оцінки реалістичності за допомогою дискримінатора. Це є багатомасштабною змагальною витратою (Multiscale Adversarial Loss), що розраховується за формулою:

$$L_D = -\mathbb{E}_{(x,y)\sim p_{data}}[\min(0, -1 + D(x,y))] - \mathbb{E}_{z\sim p_z, y\sim p_{data}}[\min(0, -1 - D(G(z), y))], \quad (2.5)$$

$$L_G = -\mathbb{E}_{z\sim p_z, y\sim p_{data}} D(G(z), y), \quad (2.6)$$

де *D(...)* - це вихідна матриця дискримінатора, розрахована для реального (*x*) та згенерованого (*G(s)*) зображень.

Для того, щоб вихідне зображення було не тільки реалістичним, а ще й мало ті є самі статистичні показники, що й реальне, розраховується Feature Matching Loss. Ця втрата розраховується за допомогою L1 штрафу відстані між картою ознак дискримінатора та реального зображення. Обчислюється така втрата для всіх масштабів згенерованого зображення.

$$L_{FM}(G, D_k) = \mathbb{E}_{s,x} \sum_{i=1}^{T} \frac{1}{N_i}[||D_k^{(i)}(s,x) - D_k^{(i)}(s, G(s))||_1] \quad (2.7)$$

де $D_k(...)$ — це карта ознак, яку розраховує дискримінатор (*x*) для реального та згенерованого (*G(s)*) зображень;

k — масштаб зображення;

T — кількість карт ознак дискримінатора;

N_i — нормалізуюча константа для кожної карти ознак, щоб L1 різниця між картами мала однаковий масштаб, незважаючи на кількість показників.

Втрата VGG має той самий принцип, що і Feature Matching Loss, однак замість дискримінатора використовується вже тренована мережа VGG19. Справжнє і згенероване зображення проходять через мережу, і ми обчислюємо штраф L1 для карти ознак після проходження основних шарів, зазначених у формулі нижче:

$$L_{VGG}(G, D_k) = \mathbb{E}_{s,x} \sum_{i=1}^{5} \frac{1}{2^i} [||VGG(x, M_i) - VGG(G(s), M_i)||_1] \quad (2.8)$$

де *VGG(...)* – матриця параметрів, розрахована на і-му шарі моделі VGG19 для реального (*x*) та згенерованого (*G(s)*) зображення.

Втрата розбіжності KL використовується для кодера.

$$L_{KLD} = D_{kl}(q(z|x)||p(z)) \quad (2.9)$$

де *q (z|x)* — варіаційний розподіл, з якого за допомогою реального зображення *x* береться випадковий вектор *z*;

p(z) — стандартний розподіл Гауса.

З кодувальником процес навчання нагадує варіаційний автоенкодер, де декодером є сам GauGAN.

Ця втрата штрафує розбіжність KL між розподілом, отриманим за допомогою кодувальника, та гаусівським розподілом з нульовим середнім.

Без розрахунку цієї втрати кодер би не навчався, і фактично давав би абсолютно випадкові вектори для будь-якого зображення.

Висновки до розділу 2

В сучасному світі обсяг даних росте з кожним днем. Необхідно обробляти все більше інформації не лише для ефективної роботи компаній, а й для повсякденного життя. Для цих цілей використовуються як програми, створені на основі математичних алгоритмів, так і нейронні мережі, логіка яких є чорною скринькою, адже людина задає не кроки самого алгоритма, а лише правила обробки інформації, за допомогою яких нейромережі навчаються самостійно вирішувати певні проблеми, самостійно вивчаючи залежності в даних; це є необхідним компонентом вирішення багатьох завдань, для рукописного оформлення всіх залежностей яких вручну знадобились би десятиліття. Такі сфери як редагування

фотографій, створення реклами, соціальні мережі, медична діагностика, мистецтво зазнали значних змін в зв'язку з впровадженням моделей штучного інтелекту, адже відкрились можливості не лише прискорювати роботу, але й швидко отримувати раніше недосяжну якість результатів.

Перехід від генерації зображень класичними методами до генерації на основі генеративних змагальних мереж (GANs) був значним кроком вперед у світі машинного навчання та комп'ютерного бачення. Збільшення потужності комп'ютерів, доступність великих наборів даних і вдосконалення методів навчання більш глибоких мереж стали основними причинами, що призвели до такого результату. А майбутнє обіцяє нові проблеми та можливості подальшого вдосконалення глибокого навчання з виходом за свої рамки [15, с.26].

Наразі GANs є потужним інструментом генерації, який щороку застосовується у все більшій кількості програм і застосунків для різноманітних сфер нашого життя. Велика кількість архітектур GAN дозволяють обрати найбільш підходящі варіанти не лише для генерації фото, але й музики, тексту та цілих відео; вони широко застосовуються для покращення зображень при їх збільшенні, комп’ютерних іграх, редагуванні фотографій, мають перспективи у сфері доповненої реальності. А основним обмеженням реалізації є наявність набору даних для тренування.

Генеративно-змагальні нейромережі беруть багато ідей з представлених для генерації варіаційних автоенкодерів (VAE). VAE складається з кодувальника (енкодера), який перетворює вхідні дані в латентний простір, та декодера, який відтворює першопочаткові дані з даних в латентному просторі. Латентний простір представляє собою абстрактне представлення даних, яке дозволяє модифікувати першопочаткові дані для отримання нових результатів. Подібний підхід прямо не використовується в GANs, проте використовується його модифікована версія, що дозволяє генерувати зображення з певними ознаками.

В цьому розділі було розглянуто і порівняно різні архітектури та підходи до генерації зображень по масці. Найреалістичніші результати здатна

продемонструвати глибока нейромережа GauGAN, розроблена компанією NVIDIA. Для її тренування необхідне обладнання з потужністю, яку майже неможливо знайти в домашніх умовах. Проте, при розробці власної генеративної змагальної мережі можна застосовувати ідеї та прийоми, що містяться в GauGAN. Для розробки моделі штучного інтелекту в рамках цієх теми буде застосовано модифіковані нормалізаційні шари SPADE без синхронізованої нормалізації, що здатні зберігати деталі елементів та їх текстур навіть у дуже глибоких мережах. Також, функції втрат та принцип роботи кодувальника допоможуть покращити результат. Тож, ці ідеї будуть взяті для застосування при розробці і тренуванні генеративної моделі. Нажаль, для тренування GauGAN необхідне досить потужне обладнання, тому в даній роботі буде розроблено генеративну змагальну мережу, яка є більш легкою для навчання в домашніх умовах.

Наразі генеративно змагальні моделі є найкращим інструментом для досягнення мети даної роботи. Серед великої кількості архітекту GAN є ті, що здатні створювати зображення по масці, по ескізу, по виділеній області, кольорові або чорно-білі зображення. Використовуючи найкращі практики у сфері генерації реалістичних зображень, в наступному розділі буде розроблено нейронну мережу та програму для її застосування.

РОЗДІЛ 3. РОЗРОБКА МОДЕЛІ ГЕНЕРАЦІЇ ЗОБРАЖЕНЬ З ЕСКІЗІВ ТА ПРОГРАМИ ГЕНЕРАЦІЇ ЗОБРАЖЕНЬ ДЛЯ ДИЗАЙНЕРІВ НА ЇЇ ОСНОВІ

3.1. Підготовка даних для тренування

Як вже зазначалось раніше, для досягнення цілі даної роботи було обрано використовувати архітектуру GAN, беручи певні структурні елементи з GauGAN.

Оригінальний документ, що описує GauGAN використовує набори даних COCO-stuff, Cityscapes, ADE20K та Flickr Landscapes. Спочатку мережа тренується 200 епох на даних Cityscapes та ADE20K, потім 100 епох на COCO-stuff, і 50 епох з набором даних Flickr Landscapes [29, с.12].

В рамках цієї роботи найбільш підходить набір даних ADE20K, який є набором даних для сегментації, і включає зображення різноманітних фото ландшафтів, фотографій вулиць та домашніх приміщень. Основна ідея цього набору – надати можливість нейромережі для вирішення завдань сегментації не лише вчитись на різноманітних даних, але й мати можливість оцінити спроможності та потужність мережі.

Набір складається із 25574 зображень, що включають в себе об’єкти 150 категорій. Кожна пара даних зображень у ADE20K має в собі саме 3-канальне RGB зображення та його маску сегментації. У масці сегментації кожен колір відповідає за певний клас.

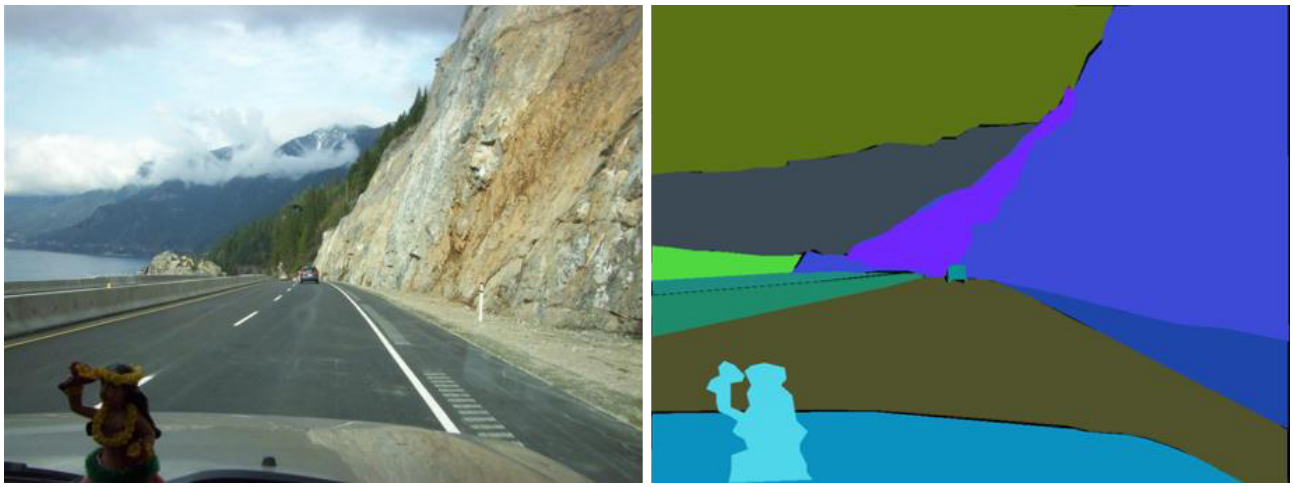


Рис. 3.1. Приклад даних з набору ADE20K

Оригінальний набір даних має 150 класів, які досить добре урізноманітнюють дані. Для наших цілей необхідно модифікувати набір даних, та дати змінивши класи анотації таким чином, щоб об’єднати суміжні класи (наприклад, озеро, річка та водоспад).

Нажаль, оригінальний набір даних не має в собі файлу, що описував би анотації, проте методом аналізу було виявлено які кольори відповідають за які категорії. Залежність проявляється, в основному, в R та G каналах, пари яких відповідають за певний клас. B-канал для одного й того самого класу може відрізнятись.

Використовуючи цю інформацію, був написаний скрипт, що шукає та об’єднує класи, і на їх основі готує маску сегментації, необхідну для тренування мережі. Кожна карта семантичної сегментації ADE20K була пропущена через скрипт, що перевіряє її пікселі, та відносить до одного з необхідних класів, необхідних для тренування.

Класи, які буде генерувати мережа: фон, небо, гори, камінь, зелень/трава, земля, пісок, вода, дерево, дорога, сніг, будівля.

Звісно, ADE20K, є досить великим набором даних, який включає об’єкти, що не потрібні для малювання ескізів пейзажів. Такі об’єкти в масці сегментації були замальовані чорним кольором, тобто віднесені до класу фону (чорний колір відповідає за відсутність класу). Самі зображення не модифікувались. У випадках, коли оригінальні зображення мають розміри менші, ніж вихідне зображення мережі, або кількість пікселів без класу становила більше 14**%,** зображення було видалено з даних для навчання.

Також, деякі зображення є занадто великими. Для покращення деталізації, вони були зменшені до розмірів, щоб кожна сторона була в рамках 1200 пікселів.

Хоча для деяких нейронних мереж використання RGB-карти сегментації та карти сегментації, в яких кожен піксель має значення цілого числа, що відповідає певному класу, дозволяють отримати прийнятні результати, на основі попереднього аналізу було виявлено, що карта сегментації з окремим каналом для

кожного класу є одним із основних чинників, що відповідають за якість згенерованого зображення. Цю стратегію використовують такі мережі, які генерують найбільш реалістичні зображення з масок, наприклад, розглянута раніше мережа GauGAN.

Як результат, набір даних складається з трьох видів даних:

1) зображення, яке генератор буде старатись згенерувати. Це 3-канальне (RGB) зображення з мінімальним розмірм у 300x300 пікселів.

2) 8-бітного зображення індексів класів для створення карти сегментації, у якому кожен піксель відповідає за один з 12 класів. Розмірами зображення відповідають розмірам RGB-зображення. На основі цього зображення індексів при тренуванні буде згенеровано 12-канальну карту сегментації, кожен канал якої відповідає за об’єкти певного класу.

3) маска, у якій 8-бітне зображення для карти сегментації має RGB-відображення для кращої візуалізації даних при тренуванні і аналізі. Ця маска ніяк не буде використовуватись у тренуванні нейромережі, проте є корисною для візуалізації результату при створенні набору даних.

Розмір отриманого набору даних складається з 4572 пар зображень та маск індексів для тренування, а також кольорових варіантів масок для візуалізації.

Додатково було зібрано високодеталізовані зображення природи та підготовлено вручну їх сегментаційні маски, які будуть застосовані на фінальних стадіях тренування для підвищення деталізації текстур. Кількість подібних зображень — 100. Вони будуть змішані з зображеннями набору ADE20K великого розміру. Загальна кількість таких високодеталізованих зображень — 1300.

3.2. Архітектура мережі

Відштовхуючись від найкращих практик у сфері генерації зображень з масок, розроблена у рамках цієї магістерської роботи нейромережа буде використовувати багато ідей з GauGAN.

Для досягнення мети цієї роботи необхідно розробити генератор, який зможе створювати зображення на основі вхідних карт сегментації. Проте для тренування подібного генератора необхідний також дискримінатор, а для урізноманітнення результатів — кодувальник. Отже, буде розроблено три нейромережі: генератор, дискримінатор, кодувальник. Кожну з них буде описано більш детально.

Нейронні мережі прямого поширення, зазвичай, складаються з композиції різних функцій. Саме через це вони й називаються мережами. Граф, що описує цю композицію є моделлю. В разі послідовного зв'язку трьох функцій $f^{(1)}$, $f^{(2)}$, $f^{(3)}$ в ланцюг $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$, $f^{(1)}$ є першим шаром мережі, $f^{(2)}$ — другим, $f^{(3)}$ — третім. Загальна довжина поджібного ланцюга відповідає глибині мережі, і термін "глибоке навчання" пов'язано саме з цією термінологією.

Останній шар мережі називається вихідним. Під час навчання ми прагнемо наблизити вихід пережі $f(x)$ та бажаного значення y [15, с.164-166]. Ми стараємось наблизити лише вихід останнього шару мережі до бажаної відповіді, при цьому поведінка інших шарів мережі вирішується самим алгоритмом. Так як для отримання бажаного результату вихідні дані чи пряме кодування не задають поведінки інших шарів, ці шари називаються прихованими.

Мережа для *генерації* буде згортковою, що використовує модифіковані SPADE блоки для досягнення кращих результатів деталізації. Складатись вона буде з серії SPADE блоків із залишковим зв'язком (SPADE Residual Block), після кожного з яких буде йти апсемплінг для збільшення розміру генерованого зображення.

Вхідними даними є сегментаційні карти класів для відтворення об'єктів, та латентний вектор, що відповідає за стиль готового зображення.

Як можна бачити на рис. 3.2, генератор використовує карту сегментації не лише на початку генерації, а й під час проходження через кожний окремий SPADE Residual блок. Це дозволяє мережі в повній мірі зажди "пам'ятати" необіжні форму і положення бажаних об'єктів, та зберігати градієнт у нормі, тобто вони не будуть

вибухати (підніматись вище потрібного для тренування) чи знакати (ставати ближче до нуля). На своєму виході, кожен блок створює зображення.

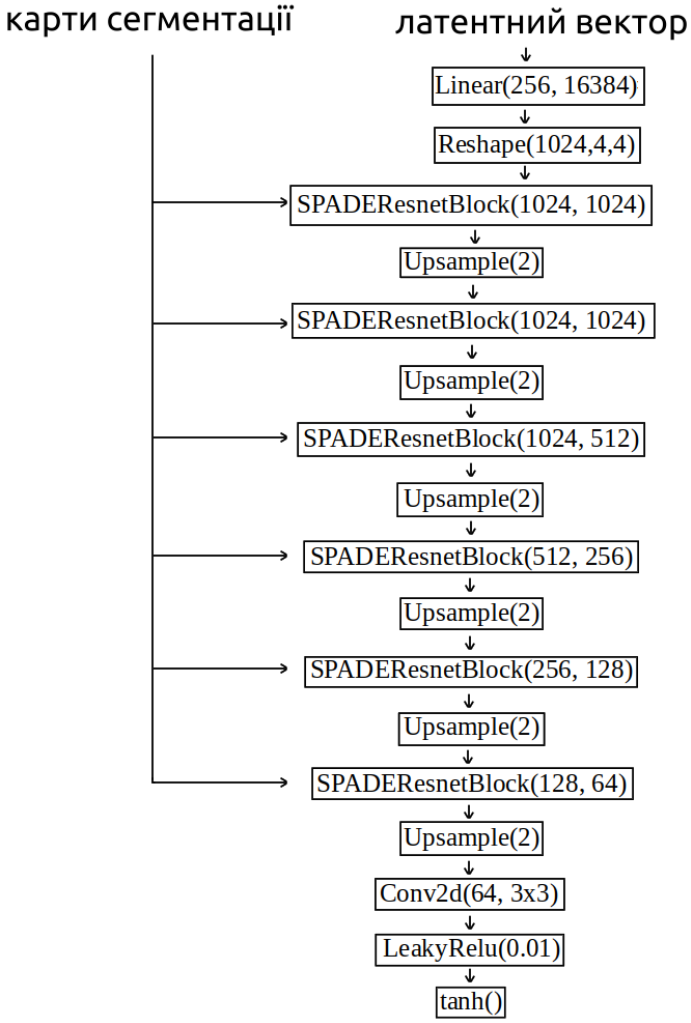


Рис. 3.2. Архітектура генератора

Кожен SPADE блок з залишковим зв’язком складається з пари (інколи трьох) послідовностей SPADE блоку (див. рис. 3.3), Leaky ReLU активації та згорткового шару.

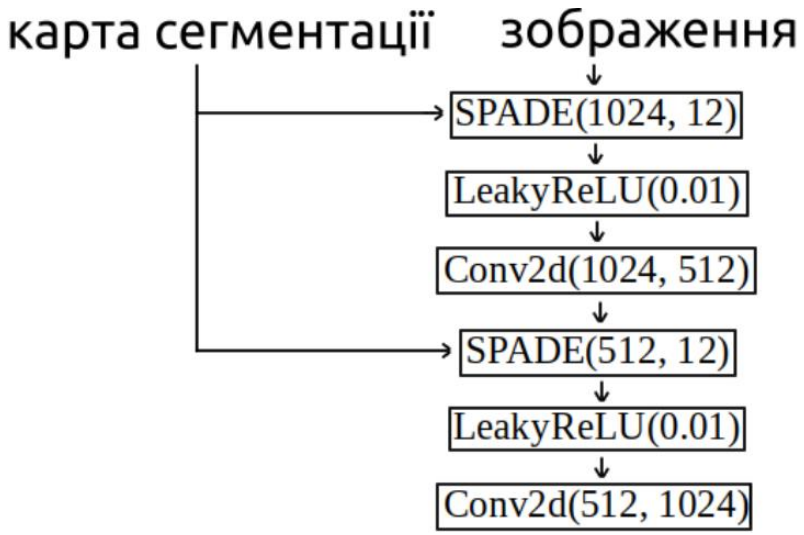


Рис. 3.3. Структура SPADE Residual блоку

SPADE блоки використовуються як шари “розумної” нормалізації, і включають комбінацію батч-нормалізації зображення та згорток карт сегментації (див. рис. 3.4), що дозволяє зберігати деталі зображення.

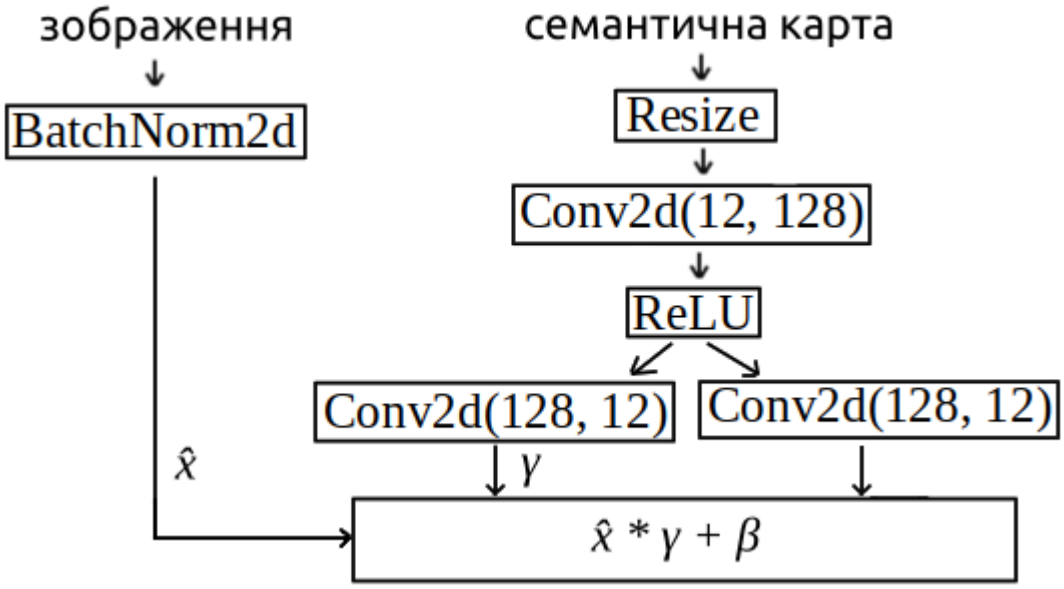


Рис. 3.4. SPADE блок з нормалізацією батчу

Шари нормалізації є однією з ключових концепцій, що забезпечують якість результату даної генеративної мережі. Вони допомагають стабілізувати та пришвидшити процес навчання, нормалізуючи входи до шару, та зменшують чутливість мережі до статистики вхідних даних. Це є необхідним і через проблеми вибухання та зникнення градієнтів, коли під час зворотного поширення помилки

значення стають дуже малими, або просто гігантськими. Подібні шари прийнято використовувати в глибоких нейронних мережах, оскільки кожна з зазначених проблем стає більш гострою зі збільшенням кількості прихованих шарів. Дана реалізація нормалізаційного шару дає ще одну перевагу — здатність передавати деталі реального зображення, що допомагає кожному шару мережі вивчати не лише загальні тенденції даних, але й незначні закономірності.

В даній реалізації SPADE буде використано батч-нормалізацію з увімкненим параметром affine для виконання афінної трансформації. Без цього параметру вихідні дані шару нормалізації нормалізуються з використанням параметрів середнього та дисперсії вхідних даних. Параметр affine дозволяє шару нормалізації навчитись виконувати операції масштабування (scale) та зсуву (shift) для кожного каналу над своїми вихідними даними. Тобто, спочатку проходить звичний процес нормалізації батча, а потім її результат проходить через афінну трансформацію. В такому випадку параметрами, що навчаються є γ (гамма) — для масштабування — та β (бета) — для зсуву. Вони дозволяють моделі більш гнучко адаптуватись до вхідних даних.

Як і багато інших GAN, останній шар має функцію активації — гіперболічний тангенс. Це рішення має на меті нормалізацію даних, та утримання значень у діапазоні від -1 до 1, що сприяє однорідності згенерованих даних та стабілізує процес навчання.

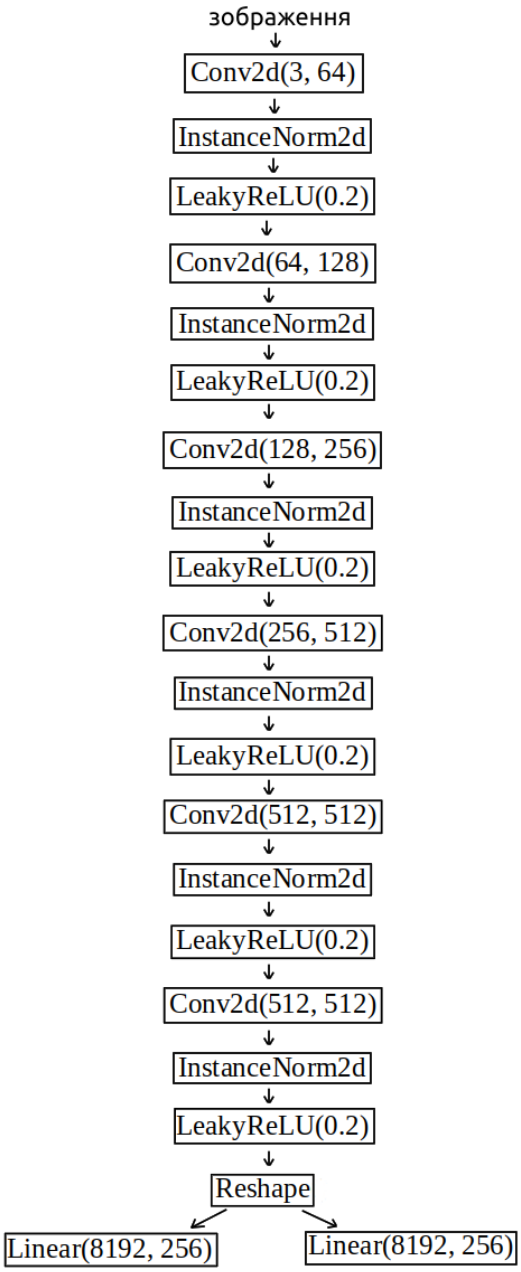
$$tanh(x) = (e^x - e^{-x})/(e^x + e^{-x}) \qquad (3.1)$$

де e — число Ейлера (e ≈ 2,71828)

x — вхідні дані шару

Перед поданням даних до генератора кодувальник (див. рис. 3.5) перетворює реальне зображення у латентний вектор з 256 елементів. Варто зазначити, що цього замало для перетворення картинки з 196608 (3 канали 256x256) елементів. Він є згортковою нейронною мережею, яка покликана не так мати змогу перетворювати

зображення в певний числовий вектор, як забезпечити можливість генератору вивчати стилі.



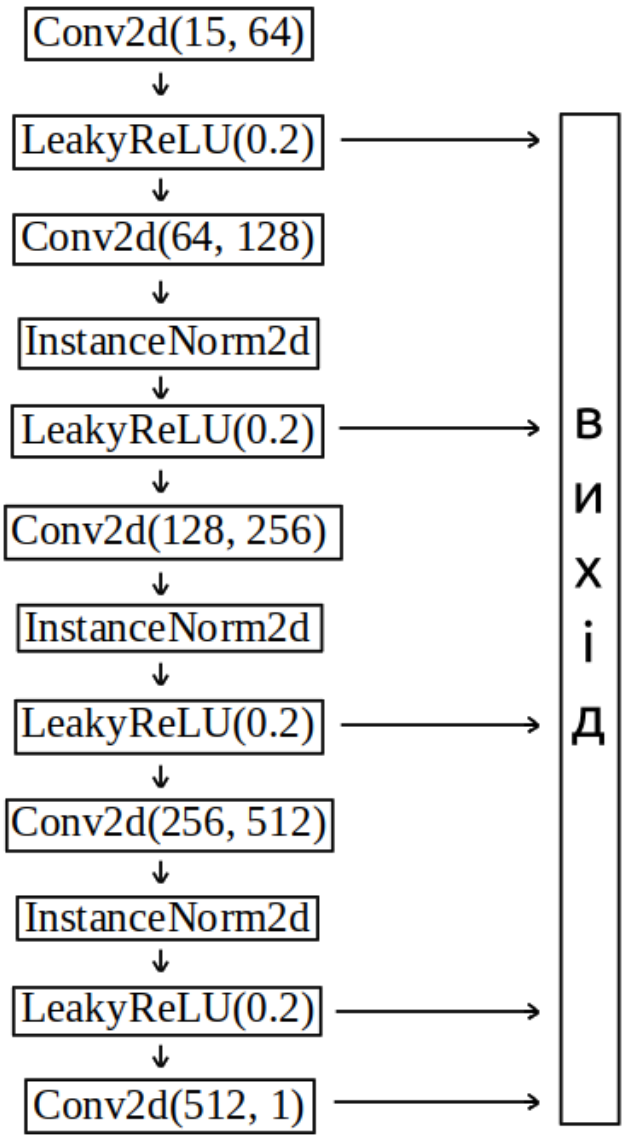


Рис. 3.5. Архітектура кодувальника *Рис. 3.6. Архітектура дискримінатора*

Після генерації реальне та створене зображення потрапляють на перевірку до дискримінатора. Він теж є згортковою мерею, що вираховує п’ять різних матриць для кожного з вхідних зображень. На основі розрахованих матриць для обох

зображень розраховується наскільки відмінними є реальне і згенероване зображення.

3.3. Тренування мережі

Для створення мережі та її тренування буде використано бібліотеку Pytorch версії 2.0.1. Це відкрита бібліотека машинного навчання, яку розробляє група дослідження штучного інтелекту компанії Facebook. Вона застосовується для розробки нейронних мереж, в основному для задач комп'ютерного зору та обробки природної мови у вигляді тексту та звуків. Інтерфейс і стиль цієї бібліотеки повністю відповідають усім кращим практикам кодування на мові Python, втілених у бібліотеці numpy. Тож з нею досить легко почати працювати тим, хто має досвід професійного і напівпрофесійного програмування на Python.

Також бібліотека включає прискорення через графічний процесор (відеокарту), що значно пришвидшує тренування нейронних мереж.

Наразі, усі зображення, що є у тренувальних даних мають розміри більші, ніж вхідні розміри мережі. Підходами до "підлаштування" даних є зміна розміру картинки, або вирізання областей необхідного розміру. Для урізноманітнення генерованих зображень будуть застосовані обидва цих методи у рівних пропорціях. Тобто, в половині випадків випадковим чином із зображення буде вирізатися область необхідного розміру (вибір такої області відбувається під час кожної ітерації); а в іншій половині випадків буде змінюватись розмір зображення. Така ж стратегія буде застосована і до сегментаційної карти.

Для *ініціалізації вагів мережі* було використано функцію ініціалізації Каймінга (Kaiming Uniform Initialization, також відома як He Initialization). Вона є рекомендованим вибором для ініціалізації вагів зготкових шарів.

$$W \sim U[-\sqrt{6/n}, \sqrt{6/n}] \quad (3.2)$$

$$W \leftarrow W * \sqrt{2/n} \quad (3.3)$$

де n – кількість входів шару;

W – ваги шару;

U – розподіл.

Тобто, матриця *W* розміром *n***m* (*n* – кількість входів, *m* – кількість виходів) буде створена наступним чином: випадкова величина буде розподілена за рівномірним розподілом в межах інтервалу [-sqrt(6/*n*), sqrt(6/*n*)]; після цього матрицю ваг *W* буде масштабовано на sqrt(2/*n*).

Стратегія щодо *швидкості тренування (Learning Rate)* буде взята з GauGAN. Для першої половини тренування коефіцієнт швидкості навчання (*Learning Rate*) для дискримінатора і генератора буде задана, як 0.00023. У другій половині тренування швидкості тренування будуть різні. Це обумовлено тим, що дискримінатор навчається швидше і показує гарні результати раніше, ніж генератор. Для стабілізації цього дизбалансу, для генератора темп навчання буде постійно зменшуватись від 0.0001 до 0, що призведе до повільнішого, проте більш стабільного навчання, та дозволить краще вивчати деталі. Дискримінатор буде теж зменшувати свої значення швидкості лінійно, проте, починаючи з 0.0004.

Оригінальний GauGAN генерує зображення розміром 768x576 пікселів, та потребує величезної кількості пам’яті. Для тренування міні-батчу розміром один необхідно близько 12Гб пам’яті на відеокарті. Тому для тренування дослідники з компанії NVIDIA використовували машину з 8 відеокартами NVIDIA V100, щоб розпаралелити обчислення для міні-батчу розміром в 32, яке займає близько 128Гб. Нажаль, в домашніх умовах важко знайти подібне обладнання, тому розмір вихідного зображення для розробки у цій роботі було зменшено до 256x256. Не лишнім буде зазначити, що для створення більших зображень в генератор потрібно додати додаткові SPADE Residual блоки, кожен з яких буде збільшувати вдвічі кожну грань вихідного зображення. Тож додавання одного такого блоку дасть можливість генерувати картинки 512x512, двох — 1024x1024 і т.д.

В моєму випадку розмір вихідного зображення складає 256x256 пікселів, і характеристики відеокарти дозволяють тренувати нейромережу з міні-батчами розміром 12, що займає 9.5Гб відеопам’яті. Саме це значення буде застосоване.

Тренування мережі лише на центральному процесорі можливе в теорії, проте буде потребувати занадто багато часу — близько одного місяця безперервної роботи.

Алгоритм Адам (Adam — Adaptive Moment Estimation) буде застосовано для оптимізації навчання генератора і дискримінатора, що допоможе підтримувати напрямок оновлення. Він є одним із найефективніших та найпопулярніших оптимізаторів для навчання нейронних мереж.

Адам — це адаптивний оптимізатор градієнтного спуску. Він об'єднує в собі ідеї методу RMSProp та Momentum, дозволяючи адаптувати швидкість навчання. Швидкість навчання для кожного параметра обчислюється на основі експоненційно зглаженого мементу та квадрату градієнту, що мають два гіперпараметра - β_1 (згладжування моменту) та β_2 (згладжування квадрату градієнту). Для цього використовується інформація про попередні градієнти [33,с.2-4].

І генератор, і дискримінатор у нашому випадку використовують β_1=0.5 та β_2=0.999.

Необхідно зазначити, що Адам використовує bias-коригування для стримування росту швидкості навчання на ранніх етапах навчання. Експоненційне згладжування моменту та квадрату градієнту допомагають утрмувати параметри, що являється свого роду регуляризацією.

Момент першого порядку (first moment) обчислює експоненційно згладжений момент за допомогою формули:

$$m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * g_t \quad (3.4)$$

де m_t - момент градієнту на кроці t;

g_t - градієнт на кроці t.

Момент другого порядку (second moment) обчислює експоненційно згладжений квадрат градієнту за формулою:

$$v_t = \beta_2 * v_{t-1} + (1 - \beta_2) * g_t^2 \quad (3.5)$$

де v_t - квадрат градієнту на кроці t.

Bias-коригування (коригування зсуву), яка важлива у випадках (особливо на початку навчання), коли значення моменту та квадрату градієнту близькі до нуля обчислюється за формулою:

$$\hat{m}_t = m_t / (1 - \beta_1^t),\ \hat{v}_t = v_t / (1 — \beta_2^t) \quad (3.6)$$

де t - поточна ітерація.

Швидкість навчання кля кожного параметра обчислюється за формулою:

$$\alpha_t = \alpha * sqrt(1 - \beta_2^t) / (1 — \beta_1^t) \quad (3.7)$$

де α - початкова швидкість навчання.

І нарешті, оновлення параметрів обчислюється:

$$\theta_t = \theta_{t-1} - \alpha_t * \hat{m}_t / (sqrt(\hat{v}_t) + \varepsilon) \quad (3.8)$$

де θ — параметри мережі;

ε - маленьке число для запобігання ділення на нуль.

У навчанні нейронних мереж часто виникає проблема з повільними оновленнями вагів мережі. Моменти — які є свого роду історією градієнтів - необхідні для вирішення цієї проблеми. Експоненційний згладжування момент допомагає допомагають визначити і коригувати напрямок оновлення в сторону, в якій градієнти є більш стабільними. Момент другого порядку забезпечує стабільність навчання за рахунок обчислення розмірів градієнтів. Як результат, це стабілізує та прискорює оновлення у потрібному напрямку, що призводить до підвищення ефективності навчання.

Під час тренування програма на основі маски з класами створює семантичну карту сегментації, і перетворює її у Pytorch тензор. У половині випадків готове зображення та карта сегментації просто змінюють свій розмір, щоб задовольнити потреби мережі, в іншій половині на вхід мережі подається виріз випадкової області зображення і її карта з необхідними розмірами. Це робиться для того, щоб

мережа мала змогу вчитися на більш різноманітних даних. Для тієї ж цілі, у половині випадків зображення перевертається по горизонталі.

Під час навчання використані функції витрат, запропоновані у мережі GauGAN (див. Розділ 2).

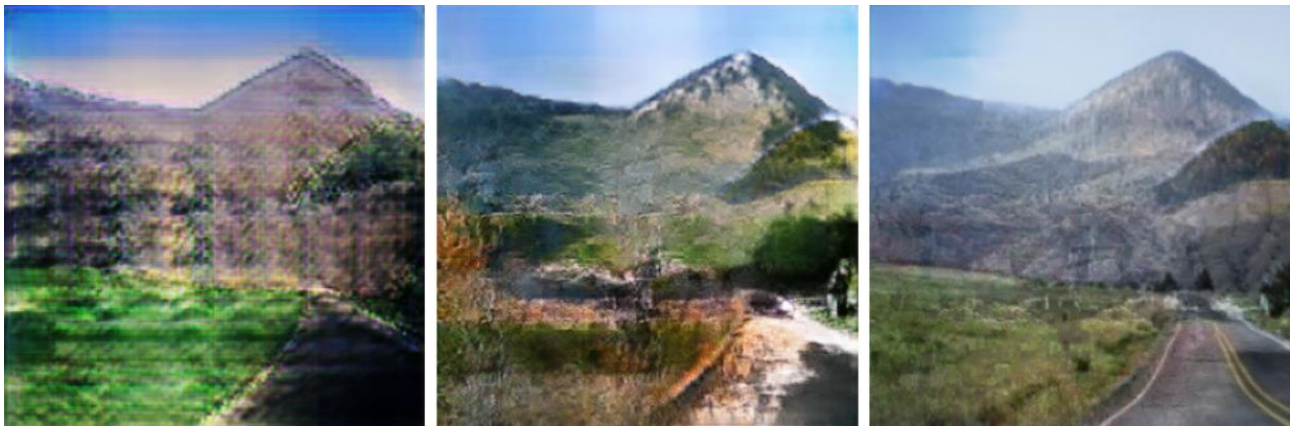


Рис. 3.7. Прогрес тренування мережі: пілся 3-ї, 60-ї та 240-ї епохи

Говорячи про навчання штучних нейронних мереж, прийнято згадувати такі поняття як точність та зниження втрат, і оцінювати прогрес за їх допомогою. Проте, як зазначалось раніше, в генеративно змагальних мережах генератор та дискримінатор стараються перевершити один одного, а розрахована втрата генератора повністю залежить від результатів дискримінатора, а не від порівняння з справжнім зображенням. Тож, прогресу падіння втрат не стається, і при правильному навчанні показники втрат обох частин мережі повинні бути стабільними. У цьому випадку, втрата генератора завжди була в межах [12.2, 15.4], а дискримінатора — [0.95, 1.1].

3.4. Оцінка моделі

Для того, щоб дати оцінку алгоритму машинного навчання, ми повинні мати кількісну міру його якості. Задачі класифікації часто вимірюються правильністю (accuracy), що показує відношення відповідей, на які система відповіла правильно, до загальної кількості запитів. В такий самий спосіб можна визначити і долю неправильних відповідей. Часто для таких задач використовується бінарна функція

втрат, в якій 0 означає неправильний результат, а 1 — правильний [15, с.102]. Для задачі цієї роботи вимірювання accuracy не має сенсу, і потрібно по іншому вимірювати якість моделі.

Зазвичай нас цікавить наскільки добре алгоритм машинного навчання працює з даними, які не використовувались в навчанні, щоб мати змогу оцінити наскільки якісним буде результат після релізу алгоритма/системи. Для цього ми виділяємо певну кількість даних від тих, на яких модель навчалась, і створюємо тестовий набір даних для перевірки фінального варіанту моделі. Для цієї цілі з отриманого набору даних було вибрано близько 5% пар, які будуть використані для оцінки моделі.

Звісно, якість можна оцінити на око, проте для об’єктивності оцінки необхідно вибрати кількісну міру, яка в повній мірі дозволить оцінити чи відповідає поведінка і результати системи бажаним. Хоча це необхідно для оцінки фінального результату і порівняння альтернативних алгоритмів, визначення кількісної міри є досить непростою задачею, адже часто досить важко вирішити, що саме потрібно вимірювати.

Наприклад, в задачах регресії необхідно вирівити чи потрібно штрафувати систему за помилки середньої тяжкості, чи за досить рідкі, але дуже серйозні помилки. Інколи потрібно створити критерій, який добре узагальнює бажаний результат.

У роботі з генеративними мережами їх оцінка є однією з найважчих задач. В задачах розпізнавання об’єктів чи класифікації на виході модель надає певні числові результати, як оцінки знайдених класів чи область розташування об’єкта. Подібні дані легко співставляти з очікуваними результатами, і оцінювати якість моделі. Для прикладу, в задачах розпізнавання автомобілів, ідеальним результатом є координати прямокутника, що описують об’єкт. Маючи подібний набір бажаних і отриманих результатів не важко зрозуміти, як коригувати модель для досягнення високої якості виходу.

У випадку з розробленою у цій роботі моделі, як і в GauGAN та Pix2PixHD, на вхід мережі подається латентний вектор та карта сегментації. Для карти сегментації ідеальним результатом є повне відтворення реального зображення. Проте латентний вектор є набором цілком випадкових чисел, що майже унеможливлює визначення результату, адже є проблема з описом залежності між даними, бо немає ніяких закономірностей між випадковими числами та кольоровим зображенням [31, с.23].

У випадку використання квадратичної функції витрат між згенерованим зображенням та реальним результат розрахунку буде завжди незадовільно великим. В залежності від вхідного латентного вектора модель генерує схожі зображення, які відрізняються кольоровою гамою, або стилем. Тож просте віднімання числових значень між згенерованим зображенням та реальним буде завжди давати досить великі значення.

Проте кольори є різними для дня й ночі (див. рис. 3.8). Тож звичайні метрики порівняння результатів просто вкажуть на значні розбіжності в даних, і цей результат буде мати значно гірші результати оцінки ніж будь-яке інше зображення денного пейзажу, навіть просто розмитий силует.

Рис. 3.8. Обидва зображення мають одну й ту саму сегментаційну карту

Фактично, різноманітність згенерованих даних є необхідністю в задачах генерації, бо в протилежному випадку задачею моделі буде відтворення даних, а не створення нових.

Враховуючи те, що звичайні математичні підходи не можуть бути застосовані для оцінки подібних нейронних мереж, дуже часто GANs оцінюються на рівні людини. Це є абсолютно нормальним процесом у питанні оцінювання генеративних мереж. В рамках цього підходу людям показують реальні зображення, та згенеровані різними нейромережами, після цього опитувані визначають, які саме результати генерації з поданих їм більше імпонують. Саме на основі подібних опитувань робиться висновок, що нова нейромережа краща за інші.

Останнім часом для оцінки GANs популярним є використання Inception дистанції Фреше (FID, Frechet Inception Distance). Це не є ідеальним варіантом оцінювання, проте на даний момент подібний підхід є одним з найкращих.

InceptionV3 — це мережа для класифікації, результати роботи якої не залежать від кольору чи стилю об’єктів. В денних або нічних сценах, зелені або білі автомобілі сфотографовані з будь-якого ракурсу розпізнаються однаково.

В рамках цього підходу до оцінювання генеративних мереж через нейромережу InceptionV3 почергово пропускаються реальне і фейкове зображення. Нас не цікавить фінальний результат обчислення, проте на одному з останніх шарів рівнів InceptionV3 утворюється латентний вектор з 2048 елементів, що описує вхідне зображення. Далі ми використовуємо два утровених вектора для кожного зображення, щоб порівняти їх за формулою:

$$FID = \|\mu_r - \mu_g\|^2 + T_r(\Sigma_r + \Sigma_g - 2(\Sigma_r \Sigma_g)^{1/2}) \quad (3.9)$$

де μ_r та μ_g — вектори ознак реального та згенерованого зображень;

$\sum_r$ та $\sum_g$ — дисперсії векторів реального та згенерованого зображень

Тобто, порівнюються не пікселі, а статистичні репрезентації обох зображень у латентному просторі.

Потрібно зазначити, що подібну метрика має певні обмеження, пов’язані з самою моделлю InceptionV3, і її можна використовувати лише для моделей, що генерують об’єкти подібні до тих, на розпізнавання яких була тренована InceptionV3. Всі інші категорії об’єктів є непомітними для мережі, і будуть розпізнані нею, як фон. В разі використання цієї метрики для розпізнавання, наприклад, результатів медичних знімків результати будуть незадовільними, і непередбачуваними, адже весь вміст медичного знімку не має нічого, що було б корисним для моделі.

Цей метод повністю підходить для нашої генеративної змагальної моделі, бо вихідні зображення є подібними до тих, що включає набір даних ImageNet, на яких тренувалась InceptionV3.

Сукупний набраний бал FID склав 71.2, що вказує на досить гарні результати генерації.

Іншим обмеженням даного підходу необхідно зазначити специфічність даних, що були використані для тренування InceptionV3 (див. рис. 3.9). Так як модель є моделлю класифікації, дані тренування складаються з зображень, які завжди мають основний об’єкт, що займає досить велику частину зображення, тоді як в нашому випадку зображення включає багато різноманітних об’єктів. Це призведе до того, що не всі об’єкти будуть представлені у латентному просторі в повній мірі.

Рис. 3.9. Приклад зображень в наборі даних ImageNet

Рис. 3.10. Приклад зображень в наборі даних ADE20K

В даній роботі для обчислення оцінки моделі буде використано офіційну реалізацію FID. Для цієї цілі підготовлені оригінальні зображення та ті, що згенеровані на основі їх сегментаційних карт. Всі зображення мають однакові розміри.

3.5. Розробка програми для застосування нейромережі

Звісно, створена нейромережа є потужним інструментом для художників, що дозволяє генерувати зображення з семантичних карт сегментації. Наразі потрібно вручну готувати семантичні карти, щоб передати їх нейромережі для генерації. І без розробки спеціальної програми користування нейромережею є занадто складним для звичайного дизайнера. Тому далі буде розроблена програма для застосування нейромережі.

Для розробки графічного інтерфейсу користувача обрано кросплатформову графічну бібліотеку GTK 3. Сама бібліотека написана на мові C, і весь код потрібно писати на цій мові. Проте у цієї бібліотеки є офіційний C++ інтерфекс, gtkmm, що є обгорткою GTK, яка дозволяє не лише нативно інтегрувати C++ код в програми на GTK, але й потребує написання меншої кількості коду для досягнення того самого результату. Подальше розробку інтерфейсу користувача буде виконано з використанням gtkmm версії 3.24.33.

Для спрощення створення графічного інтерфейсу для цієї бібліотеки є програма Glade, що дозволяє проектувати графічний інтерфейс з візуальним відображенням. Як результат, буде створено XML-файл, який можна динамічно завантажувати з C/C++-коду.

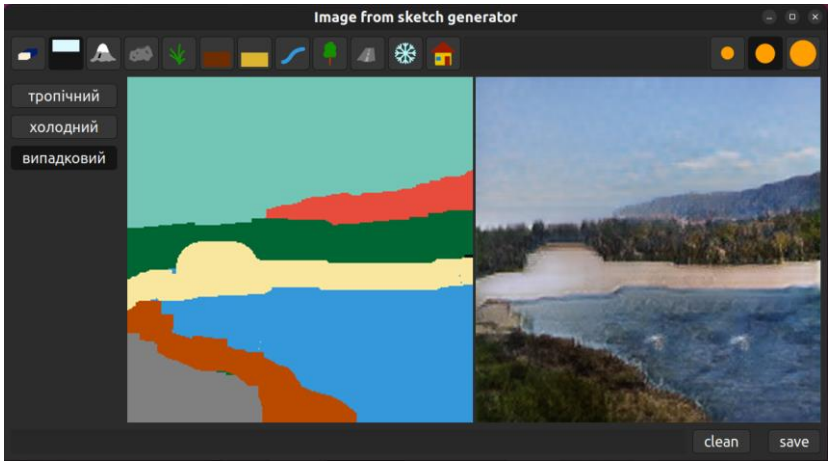


Рис. 3.11. Користувацький інтерфейс готової програми

Також необхідно підготувати Pytorch-модель для застосування. Для цього розроблену модель буде конвертовано. Для цих цілей обрано екосистему ONNX (Open Neural Network Exchange) Runtime. Перші версії ONNX було розроблено Pytorch командою Facebook в кооперації з Microsoft, а з 2019 року вся розробка ONNX ведеться Linux Foundation AI, а ONNX Runtime розробляється Microsoft. Бібліотека дозволяє не лише підготовлювати готові моделі для розгортки рішень ШІ в реальних проектах, але й надає широкі можливості до оптимізації нейромереж. Сам процес конвертації є необхідним кроком, адже Pytorch-модель оптимізована для тренування , і містить багато інформації для цих цілей.

В більшості випадків параметри нейромереж зберігаються в форматі float32 (32-бітне число з рухомою комою). Всі розрахунки при тренуванні і готова мережа використовують цей формат. ONNX Runtime дозволяє конвертувати параметри мережі в такі формати, як float16, і навіть у int8 та unsigned int8. Це є дуже корисним для застосування нейромереж на мікрокомп'ютерах, адже це призводить до значного зростання швидкість розрахунків, навіть у 10 разів [32]. Варто зазначити, що при конвертації моделі в подібні формати всі розрахунки будуть проводитись з меншою точністю, і якість мережі буде падати. В багатьох випадках можна жертвувати 3% падінням точності заради підвищення швидкості у рази, проте інколи падіння точності може бути катастрофічним. Про це необхідно пам’ятати обираючи подібні формати даних.

Вихідний ONNX-формат є досить популярним, і широко застосовується при розгортанні ШІ рішень в реальних проектах. Версія ONNX Runtime: 1.16.3

Кодування і збірка проекту будуть зроблені в середовищі для розробки програмного забезпечення Eclipse версії 4.29.0.

Розроблена програма дозволяє малювати ескіз з лівої сторони, та бачити результат генерації справа. Користувач може обирати для малювання один з 12 класів, кожен з яких має свій унікальний колір. Ширину пензля для малювання можна також міняти.

Дана програма перетворює знімок полотна для малювання в сегментаційну карту для кожного класу і подає на вхід нейромережі, яка, в свою чергу, генерує нове зображення. Результат генерації відмальовується з правої сторони користувацького інтерфейсу. Цей процес відбувається близько 7 разів на секунду.

Висновки до розділу 3

В розділі 3 було описано виконання практичних кроків до генерації зображень з ескізів. Як вже зазначалось раніше, для впровадження сучасних методів досягнення цілі було обрано використовувати генеративну змагальну модель.

Для початку було проаналізовано які саме дані потрібні, в якому обсязі. На основі переліку вимог до даних було знайдено набір даних, який може бути використано для навчання подібної моделі — ADE20K. Даний набір даних прямо не пристосований до подібних задач — адже моделі необхідні карти семантичної сегментації, тому для використання його в навчанні подібних мереж дані було видозмінено відповідно до задачі: створені маски індексів класів, та відібрано ті з них, які містять в собі об’єкти, що відповідають темі даної роботи.

Після цього було створено генеративну змагальну нейромережу (GAN), здатну навчитись створювати зображення з масок зображень. Вона складається з трьох частин — кодувальника, генератора та дискримінатора. Кодувальник відповідає за перетворення зображення у вектор в латентному просторі, який буде використано для навчанню моделі генерувати дані певного стилю.

Генератор і дискримінатор є двома окремими мережами, які вчаться одночасно і “змагаються” один з одним. Дискримінатор вчиться відрізняти реальні зображення від згенерованих, а генератор вчиться генерувати більш реалістичні зображення, які дискримінатор не зможе відрізнити від реальних.

Генератор є глибокою мережею, що приймає на вхід карту семантичної сегментації та вектор, створений кодувальником, та складається з послідовності нормалізаційних шарів SPADE, згорткових шарів, та апсемпліну, між якими

присутній зворотній зв’язок. Саме шари SPADE та наявність зворотного зв’язку забезпечують реалістичність вихідного зображення та стабільність результату в нейронній мережі такої глибини. Загальна кількість шарів генератора складає 99 — це згорткові та нормалізаційні шари, шари активації та шари апсемплінгу.

Під час тренування з підготовлених раніше індексних масок створюються карти сегментації, що відповідають кожному класу бажано картинки, і задають розмір об’єкта, його форму та розташування. Для урізноманітнення результатів, інколи зображення зжимаються до потрібного мережі розміру, інколи вирізається певна область потрібного розміру, також в половині випадків зображення і його карта семантичної сегментації перевертаються по горизонталі. Потім на основі цих карт генератор генерує реалістичні зображення, яке дискримінатор порівнює з реальними.

Загальний час тренування подібної мережі для генерації зображень 256x256 пікселів на сучасній відеокарті зайняв близько двух діб.

Генератор був оцінений за допомогою метрики Inception відстань Фреше, в якій набрав 71.2 бали, що є досить гарним результатом для генеративної мережі, яка здатна урізноманітнювати результати. Оскільки через згадані в 3-му розділі причини з тренуванням моделі InceptionV3, для використання в оцінці були відібрані зображення, які схожі по вмісту як в наборі ADE20K, так і в наборі ImageNet (подібних зображень досить багато). Оцінка проводилась з використанням програми з офіційного репозиторію на github (github.com/bioinf-jku/TTUR). Потрібно додати, що гарним результатом для звичайних генеративниї змагальних мереж вважається бал нижче 50, проте розроблена у цій магістерській роботі нейромережа здатна створювати зображення різних стилів, що призводить до значного підвищення балу FID, наприклад, мережа Pix2PixHD набирає більше 80 балів зі складними наборами даних [29, с.5]. Потенційними шляхами пониження балу FID є: використання набору даних з дуже деталізованих зображень великого розміру, набагато довше і повільніше тренування.

Після тренування і оцінки, модель генератора була конвертована і оптимізована для роботи на центральному (не графічному) процесорі за допомогою бібліотеки ONNX Runtime для застосування у програмі для малювання. В такому режимі, час створення зображення складає близько 140 мілісекунд, чого достатньо для генерації 7 кадрів у секунду.

Після конвертації, використовуючи бібліотеку GTK, було створено інтерфейс користувача, в якому можна малювати ескізи, та бачити згенеровані нейромережею на їх основі зображення. Інтерфейс програми є досить мінімалістичним і легким в освоєнні. В програмі користувач може обрати один з 12 класів для малювання і почати малювати. Для пришвидшення малювання можна вибрати більш вирокі пензлі, а для відмальовки деталей та гострих кутів — вужчі. Для урізноманітнення результату можна вибрати “тропічний”, “холодний” або “випадковий” стилі. Для “тропічного” та “холодного” застосовуються заздалегідь збережені вектори, створені кодувальником для фотогрфій тропічних пляжів та пувничних гірських пейзажів, відповідно. В разі застосування “випадкового” стилю, програма генерує вектор випадкових чисел самостійно. Під час малювання результат генерації поновлюється декілька разів на секунду, тож користувач майже одразу може бачити результат. Є можливість зберігати згенеровані картинки на диск.

Висновок до магістерської роботи

Деякі сфери нашого життя вже зараз переповнені машинним навчанням, яке все більше відображається не в своїх класичних формах, а у вигляді нейронних мереж. Це сфери розпізнавання зображень і аудіо, фільтрації, рекомендаційних систем. Проте нейромережі мають значні можливості росту й у інших сферах, таких як створення контенту.

В ході роботи було продемонстровано переваги нейромереж над класичними підходами у сфері генерації зображень та візуально відображено зростання точності та правдоподібності результатів. Вже зараз подібні технології допомагають у науці, а вміле використання наявних можливостей здатне оптимізувати багато процесів життєдіяльності, виробництва, прискорити інноваційний прогрес суспільства. Тож цей потенціал необхідно розкрити у повній мірі.

В цій магістерській роботі було розглянуто традиційні і сучасні способи досягнення цілі генерації реалістичних зображень з ескізів. Порівняно їх результати, можливості і необхідні ресурси. Коротко переказано історію розвитку і впровадження штучного інтелекту.

Порівняльний аналіз класичних методів генерації зображень, побудованих на основі алгоритмів, та генерацій на основі нейромереж показав, що класичні методи самостійно не здатні генерувати зображення, що схожі на справжні, проте подібної цілі можна досягти комбінацією таких методів. В той же час класичні методи потребують великої кількості людино-годин на обробку, кодування та дебагінг. На противагу таким методам генерації були розглянуті нейромережі. Вони здатні більш якісно передавати відношення між об’єктами, кольорову гаму, та потребують кодування, яке б показало мережі яким саме способом обробляти дані, замість прямого кодування всіх деталей алгоритму. Основним недоліком є необхідність набору даних для тренування. Також, для прискорення процесу навчання необхідною є наявність сучасного графічного процесору з якомога більшим об’ємом відеопам’яті — бажано більше 8Гб.

Порівнюючи результати, легко помітити, що моделі штучного інтелекту генерують набагато кращі картинки. Зважаючи на якість результату та те, що для створення нейромережі потребується менше часу і ресурсів, абсолютно зрозумілими є сучасні тенденції збільшення ролі генеративних мереж в сфері генерації візуального та аудіо контенту.

Серед нейронних мереж для реалізації цілі даної роботи було розглянуто варіаційні автоенкодери. Ці мережі здатні модифікувати зміст першопочаткового зображення. Вони показують гарний результат, проте не підходять для візуалізації ескізів, адже важко контролювати розташування бажаного об’єкта та його форму.

Не дивлячись на те, що дифузійні моделі абсолютно неперевершено здатні реалістично комбінувати будь-які раніше не комбіновані поняття, присутня та сама проблема розташування об’єктів та їх форми, що і у варіаційних автоенкодерів.

Найкращим типом нейронних мереж для даної магістерської роботи є генеративні змагальні мережі. Вони мають багато підтипів та архітектур, спрямованих на досягнення різних цілей. Найкраще з подібними задачами справляються моделі Pix2Pix, LayoutGAN, MaskGAN, Cascaded Refinement Network, Pix2PixHD, GauGAN. Порівняльний аналіз показав переваги GauGAN у передачі деталей зображення. Саме тому, створена в межах цієї роботи модель штучного інтелекту, взяла для використання багато архітектурних рішень з GauGAN, хоча й не повторювала її повністю.

В 2023 році технології генерації деталізованих зображень не є ідеальними, проте стрімко розвиваються. В той час як їх впровадження у готові продукти є буденністю, основною проблемою у сфері є необхідність дуже потужного та дорогого обладнання, необхідного для тренування нейронних мереж. Проте розвиток триває; на ринку з'являються все потужніші відеокарти, а дослідники винаходять шляхи оптимізації наявних алгоритмів, та шляхи для досягнення кращих результатів. Зважаючи на явні обмеження в реалізації технології, у цій роботі було обрано стратегію розробки нейромережі, яку можливо тренувати не

лише у престижних дослідницьких лабораторіях, але й на сучасних домашніх комп'ютерах. Тож розроблена модель штучного інтелекту використовує оптимізовані алгоритми та є доволі компактною, проте має можливості розширення для отримання більших за розміром вихідних зображень.

Для генерації зображень з ескізів існують два підходи до подання даних на вхід глибокої нейронної моделі: подання маски сегментації, або карт семантичної сегментації. У масках сегментації кожен колір (в 1-канальному або 3-канальному зображенні) відповідає за бажаний клас. В картах семантичної сегментації для кожного класу створюється окреме чорне 1-канальне зображення, на якому білим кольором вимальовується зона присутності об’єктів цього класу. Карти семантичної сегментації, зрозуміло, займають більше пам’яті адже за наявності 12 класів, вхід моделі буде складатись не з трьохканального зображення, а з 12-канального, що займає в чотири рази більше пам’яті. Проте такий підхід допомагає створювати більш якісні результати.

Використовуючи найкращі практики у області генерації зображень за умови, та дані, отримані під час попередніх аналізів нейромереж, було створено три моделі: генератор, дискримінатор та кодувальник.

Кодувальник допомагає дістати з зображення дані про стилі, як пора року, день чи ніч, тропічна чи холодна гама.

Генератор приймає дані кодувальника, та семантичні карти сегментації, з яких старається відтворити бажане реальне зображення. Потрібно наголосити що після тренування, модель зможе створювати не лише зображення, які є подібними до використовуваного набору даних, але й перемішувати стилі, та створювати нові. Наприклад, при правильному підборі параметрів ми зможемо побачити крижані айсберги в пустелі, або новітній хмарочос посеред середньовічного міста.

Генератор використовує цілу серію згорткових шарів та SPADE блоків, які допомагають передавати інформацію про текстури та деталі аж до останніх шарів. Подібна задача не є легкою, оскільки нормалізації та кожен додатковий шар нейронної мережі все більше і більше стирають подібну інформацію. Використані

архітектурні підходи здатні не приблизно, а точно вказати де саме на зображенні, та який саме об’єкт буде відображено.

Генератор є основною мережею, яка розроблялась в даній роботі, яку не можливо було б навчити створювати подібні зображення без наявності дискримінатора та кодувальника, що є допоміжними для досягнення основної цілі.

Використовуючи метрику Inception відстані Фреше оцінено спроможності моделі, та порівняно її з моделями для подібних задач, розробленими великими компаніями та університетами.

Саме генератор в подальшому було підготовлено для використання у розробці програми для візуалізації ескізів, в якій користувач може малювати силуети об’єктів, на основі яких нейромережа згенерує реалістичне зображення. Подібна програма для малювання була теж розроблена у даній роботі з використанням бібліотеки GTK 3. В програмі можна малювати різнокольорові ескізи, з яких буде створено карти семантичної сегментації, що будуть подані генератору. В свою чергу, спираючись на них, генератор створює зображення. Є можливість вибирати стиль згенерованого зображення для урізноманітнення результатів.

Особливу увагу варто приділити факту, що є можливість модифікування генератора, розробленого у цій магістерській роботі для генерації більших за розміром зображень. Також, є можливість навчити працювати з іншими даними для реалізації малювання не лише ландшафтів, але й зображень міст, тварин і людей, дезайнів гаджетів, генерувати персонажів мультфільмів та відеоігор. Для цього необхідно ще раз пройти через кроки розділу 3 з іншими наборами тренувальних даних.

Основними сферами застосування є дизайн настільних та комп'ютерних ігор, реклама, створення фонів у кіноіндустрії та віртуальній реальності, генерація даних для тренування алгоритмів комп'ютерного зору.

Програма з використанням розробленої мережі дасть можливість дизайнерам значно прискорити свою роботу, швидко намалювавши нариси ескізу, та

випробувавши різноманітні стилі ландшафту. Це дозволить у короткий час продивитись можливі варіанти, та доопрацювати їх вручну. Завдяки такому підходу значно скорочується час створення роботи, що в рази підвищує продуктивність праці, що в свою чергу, дасть можливість компаніям встигати до змінних умов ринку, майже миттєво перебираючи варіанти прототипів дизайну, зменшити тривалість виробничого циклу та виділити час на інші етапи проекту. Подібний підхід до роботи здатен не лише автоматизувати процеси, але й дає можливість знайти абсолютно новий неочікуваний стиль.

Джерела

1. Fractal-generating software [Електронний ресурс] / Wikipedia.org

Режим доступу: https://en.wikipedia.org/wiki/Fractal-generating_software

2. L-system [Електронний ресурс] / Wikipedia.org

Режим доступу: https://en.wikipedia.org/wiki/L-system

3. Introduction to Images and Procedural Generation in Python [Електронний ресурс] / FC Python

Режим доступу: https://fcpython.com/blog/introduction-to-images-and-procedural-generation-in-python

4. Noise generator [Електронний ресурс] / Minecraft Wiki

Режим доступу: https://minecraft.fandom.com/wiki/Noise_generator

5. James Hays. Scene Completion Using Millions of Photographs / James Hays, Alexei A. Efros. // Computer Graphics Proceedings, Annual Conference Series. - 2007. - С. 1-4

6. Tao Chen. Sketch2Photo: Internet Image Montage / Ming-Ming Cheng, Ping Tan [та ін.]. // Siggraph Asia. - 2009. - С. 1,3,5,9

7. Jean-Francois Lalonde. Photo Clip Art / Jean-Francois Lalonde., Derek Hoiem [та ін.]. // Carnegie Mellon University. - 2007. - С.2

8. M. Johnson. Semantic Photo Synthesis / M. Johnson, G.J. Brostow [та ін.]. // University of Cambridge. - 2006. - С. 1,6

9. Ruoqi Wei. Optimizing Few-Shot Learning Based on Variational Autoencoders / Ruoqi Wei, Ausif Mahmood. // University of Bridgeport. - 2021. - С. 15

10. Shijie Cheng. SUGAN: A Stable U-Net Based Generative Adversarial Network / Shijie Cheng, Lingfeng Wang [та ін.] // School of Artificial Intelligence, Hubei University. - 2023. - С.11,12

11. Neural style transfer // Tensorflow Core

Режим доступу: https://www.tensorflow.org/tutorials/generative/style_transfer

12. Phillip Isola. Image-to-Image Translation with Conditional Adversarial Networks / Phillip Isola, Jun-Yan Zhu [та ін.] // CVPR. - 2017. - С. 5,8

13. Ting-Chun Wang. High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs / Ting-Chun Wang, Ming-Yu Liu [та ін.] // NVIDIA Corporation. - 2018. - С.5-9

14. Ada Lovelace. Notes on the Analytical Engine of Charles Babbage. - 1843. - С.7

15. Ian Goodfellow. Deep Learning / Ian Goodfellow, Yoshua Bengio [та ін.]. // The MIT Press. - 2016. - С.1-4, 9-10, 13, 15, 19, 26, 99-102, 120, 164-166

16. History of the Perceptron // California State University, Long Beach

Режим доступу: https://home.csulb.edu/~cwallis/artificialn/History.htm

17. F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain / F. Rosenblatt // Psychological Review, 1958, С. 1-5

18. B. Widrow. Adaptive Switching Circuits / B. Widrow, M. E. Hoff. / IRE Transactions on Circuit Theory. - 1960. - С. 99

19. Alex Krizhevsky. ImageNet Classification with Deep Convolutional / Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton. // University of Toronto. - 2012. - С. 7

20. Doug Ridley. Facebook’s New Facial Recognition Photo Tagging: How to Use (Or Not Use)

Режим доступу: https://vitaldesign.com/digital-marketing/social-media/nh-facebook-marketing/how-to-disable-facebook-facial-recognition-photo-tagging-nhmarketing/

21. Volodymyr Mnih. Learning to Detect Roads in High-Resolution Aerial Images / Volodymyr Mnih, Geoffrey E. Hinton. // ECCV. - 2010. - С.2

22. Ian Goodfellow. Generative Adversarial Networks / Ian Goodfellow, Jean Pouget-Abadie [та ін.] // NIPS (Conference on Neural Information Processing Systems). - 2014. - С.1-9

23. David Donahue. Label-Conditioned Next-Frame Video Generation with Neural Flows / David Donahue // University of Massachusetts Lowell. - 2016. - C.1

24. Jianan Li. LAYOUTGAN: GENERATING GRAPHIC LAYOUTS WITH WIREFRAME DISCRIMINATORS / Jianan Li, Jimei Yang [та ін.] // Beijing Institute of Technology, Adobe Research. - 2019. - C.1-7

25. Qifeng Chen. Photographic Image Synthesis with Cascaded Refinement Networks / Qifeng Chen, Vladlen Koltun // IEEE International Conference on Computer Vision. - 2017. - C.1,5-6,8

26. Cheng-Han Lee. MaskGAN: Towards Diverse and Interactive Facial Image Manipulation / Cheng-Han Lee, Ziwei Liu [та ін.] // SenseTime Research, The Chinese University of Hong Kong, The University of Hong Kong. - 2020. - C.1-4,8

27. Phillip Isola. Image-to-Image Translation with Conditional Adversarial Networks / Phillip Isola, Jun-Yan Zhu [та ін.] // Berkeley AI Research (BAIR) Laboratory, UC Berkeley. - 2016. - C. 1-13

28. Ting-Chun Wang. High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs / Ting-Chun Wang, Ming-Yu Liu [та ін.] // NVIDIA Corporation, UC Berkeley. - 2018. - C.1-8

29. Taesung Park. Semantic Image Synthesis with Spatially-Adaptive Normalization / Taesung Park, Ming-Yu Liu [та ін.] // UC Berkeley, NVIDIA, MIT CSAIL. - 2019. - C. 1-19

30. Warren S. McCulloch. A logical calculus of the ideas immanent in nervous activity, / Warren S. McCulloch, Walter Pitts // Bulletin of Mothemnticnl Biology Vol. 52. - 1943. - C.109

31. Martin Heusel. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium / Martin Heusel, Hubert Ramsauer [та ін.] // LIT AI Lab & Institute of Bioinformatics. - 2018. - C.23

32. YOLOv8 Detection 10x Faster With DeepSparse—Over 500 FPS on a CPU // neuralmagic. - 2023

Режим доступу: https://neuralmagic.com/blog/yolov8-detection-10x-faster-with-deepsparse-500-fps-on-a-cpu/

33. Diederik P. Kingma. ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION / Diederik P. Kingma, Jimmy Lei Ba // University of Amsterdam, OpenAI, University of Toronto. - 2015. - С.2-4